Rapport TP

TL Traitement Du Signal

Réalisé par:
Nabil ABDELOUAHED
Yann MARTIN

May 20, 2025

Contents

1 Partie 1: Quelques opérations de base sur les signaux

1.1 Signal numérique de synthèse

1.1.1 Génération du signal

```
def sinus_echant(f0, fe, N):
    t = np.arange(N) / fe
    signal = np.sin(2 * np.pi * f0 * t)

    plt.figure(figsize=(10, 4))
    plt.plot(t, signal)
    plt.title(f'Signal sinusoidal : f0 = {f0} Hz, fe = {fe} Hz, N = {N}')
    plt.xlabel('Temps (s)')
    plt.ylabel('Amplitude')
    plt.grid(True)
    plt.show()

return signal
```

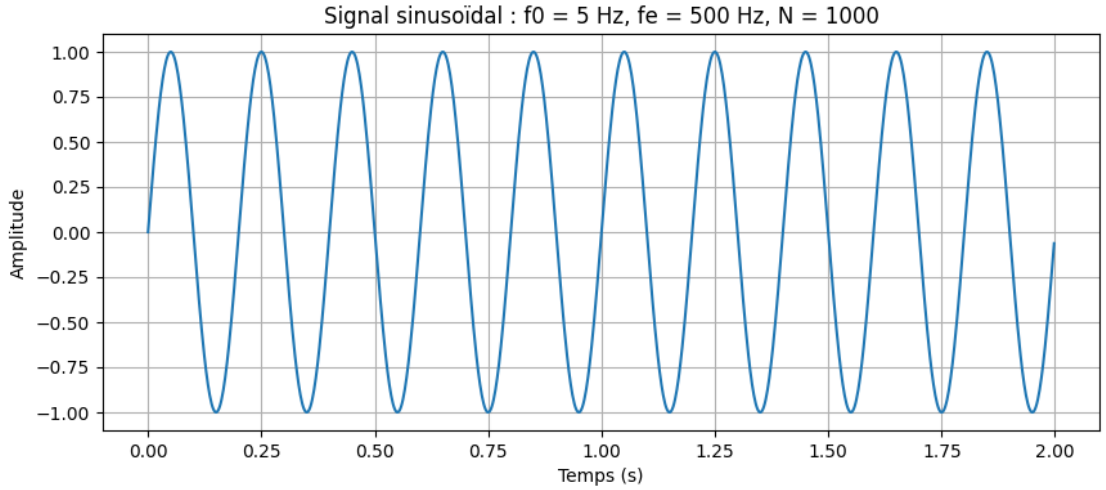


Figure 1: Signal sinusoïdal de fréquence f0 échantillonné à fe points par seconde et comportant en tout N points

1.1.2 Energie et puissance

Afin de calculer l'énergie et la puissance du signal sans utiliser de boucle for directement, on utilise:

```
def energie_signal(signal):
    return np.sum(signal ** 2)

def puissance_signal(signal):
    return np.mean(signal ** 2)
```

```
puissance moyenne du signal echantillonné :  0.5
puissance moyenne theorique : 0.5
```

Figure 2: Comparaison entre la puissance moyenne théorique et la puissance moyenne calculée du signal

1.1.3 Quantification

```
def quantifier(signal, N_bits):
    """Quantifie un signal sur N bits"""
    min_val, max_val = np.min(signal), np.max(signal)
    levels = 2 ** N_bits
    step = (max_val - min_val) / (levels - 1)
    quantized_signal = np.round((signal - min_val) / step) * step + min_val
    return quantized_signal
```

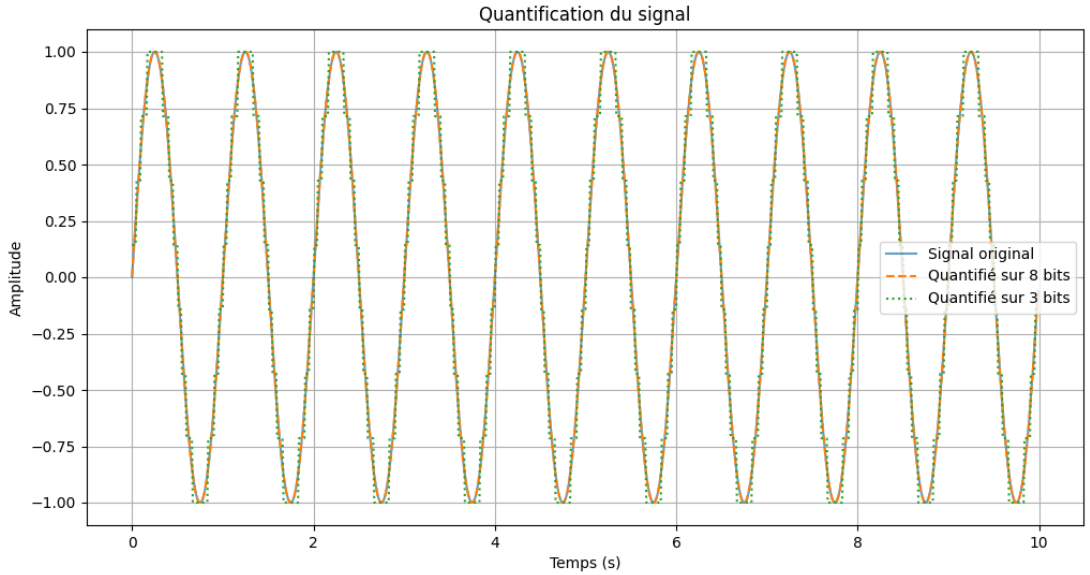


Figure 3: Quantification du signal à 3 et 8 bits

On remarque que le signal quantifié à 8 bits colle presque parfaitement à la courbe comparé au signal quantifié à 3 bits qui lui présente des paliers légérements visibles et qui donc colle moin au signal d'origine.

Ensuite on calcule SNR du signal pour les deux quantifications:

```
bruit_q8 = signal - signal_q8
bruit_q3 = signal - signal_q3

energie_bruit_q8 = energie_signal(bruit_q8)
energie_bruit_q3 = energie_signal(bruit_q3)

energie_signal = energie_signal(signal)

snr_q8 = 10 * np.log10(energie_signal / energie_bruit_q8) #en dB
snr_q3 = 10 * np.log10(energie_signal / energie_bruit_q3) #en dB
```

```
Énergie du bruit de quantification (8 bits): 0.005038
SNR (8 bits): 49.97 dB

Énergie du bruit de quantification (3 bits): 5.899996
SNR (3 bits): 19.28 dB
```

Figure 4: SNR pour chaque quantification

On remarque que l'energie du bruit est plus élevée pour le signal quantifié à 3 bits. Le résultat est logique car on voit sur le graphe que ce signal est plus éloigné du signal d'origine comparé au signal quantifié à 8 bits. On a la méme conclusion en raisonant sur le SNR: $SNR_{q8} > SNR_{q3}$.

1.2 Signal audio

1.2.1 Enregistrement

Les deux mots enregistrés sont "Bonjour" et "ChatGpt".

1.2.2 Restitution

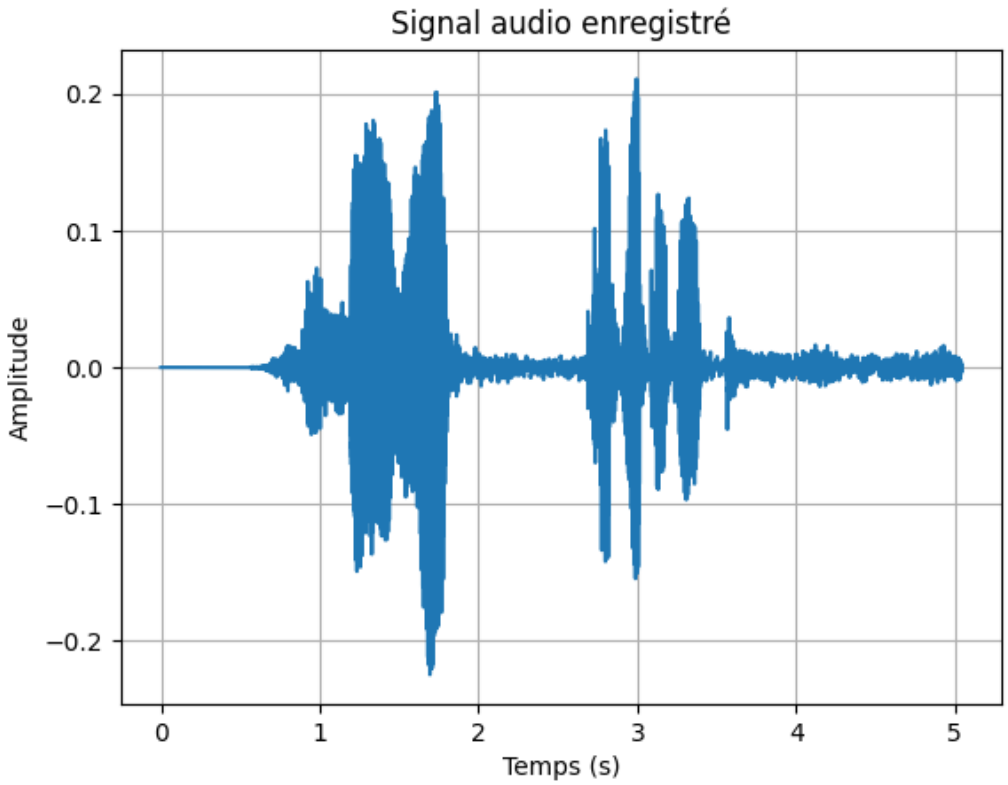


Figure 5: Signal enregistré avec Audacity

On écoute ensuite le signal à des fréquences de restitution différentes : fe, fe * 2 et fe // 2.

Analyse du résultat:

- **Durée:**
  - Restituer à 2× fe → le son est plus rapide, la durée est divisée par 2.
  - Restituer à 0.5× fe → le son est ralenti, la durée est doublée.
- **Spectre:**
  - À fréquence de restitution plus haute → son plus aigu (fréquences multipliées).
  - À fréquence plus basse → son plus grave (fréquences divisées).

1.2.3 Quantification

Analyse du résultat:

- Moins de bits → moins de niveaux → plus d'erreurs d'arrondi → plus de bruit de quantification.
- À 3 bits, le signal est très "en escalier", ce qui altère fortement la qualité audio. L'audio est très bruité.
- À 8 bits, le son est encore reconnaissable, mais moins naturel. Cependant, il reste trés proche du signal enregistré qui lui est à 16 bits PCM.

1.2.4 Extraction de mots

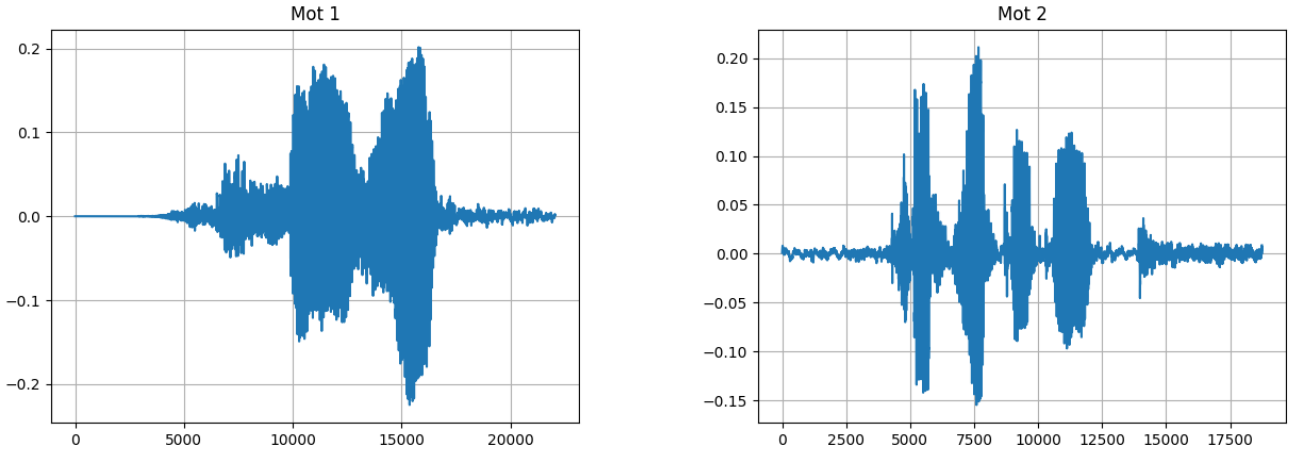
Aprés avoir identifié l'intervalle de temps pour chaque mot dans le signal enregistré on utilise ce code pour séparer les deux mots:

```
n1 = int(0.3 * fe)      #0.3s
n2 = int(2.3 * fe)      #2.3s
n3 = int(4 * fe)        #4s

mot1 = y[n1:n2]
mot2 = y[n2+1:n3]

print("Mot 1 :")
sd.play(mot1, fe)
sd.wait()

print("Mot 2 :")
sd.play(mot2, fe)
sd.wait()
```



(a) Mot 1

(b) Mot 2

Figure 6: Séparation des deux mots

Puis on enregistre les deux mots séparément dans des fichiers .wav :

```
sf.write("mot1.wav", mot1, fe)
sf.write("mot2.wav", mot2, fe)
```