

# Optimisation du traitement du signal sonore sous forme numérique

Où comment utiliser la transformée de Fourier pour créer un identifiant unique et reconnaître efficacement une musique

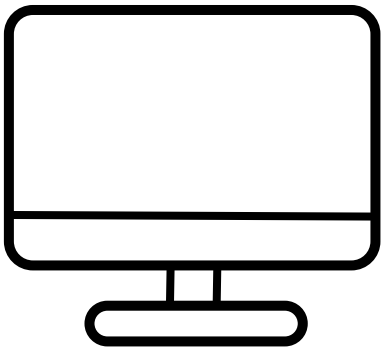
---

PETIT Robin      MPI

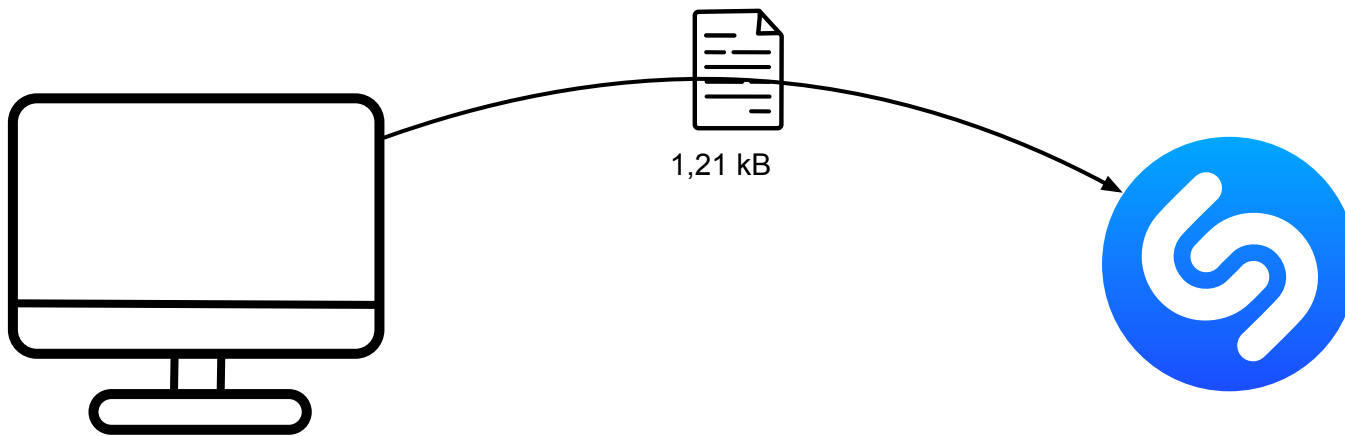
1. Introduction
2. Problématique
3. 1ère Approche - La Fast Fourier Transform
4. Les limites de la FFT
5. 2ème Approche - La STFT
6. Les limites de la STFT
7. L'algorithme Shazam

1. **Introduction**
2. Problématique
3. 1ère Approche - La Fast Fourier Transform
4. Les limites de la FFT
5. 2ème Approche - La STFT
6. Les limites de la STFT
7. L'algorithme Shazam

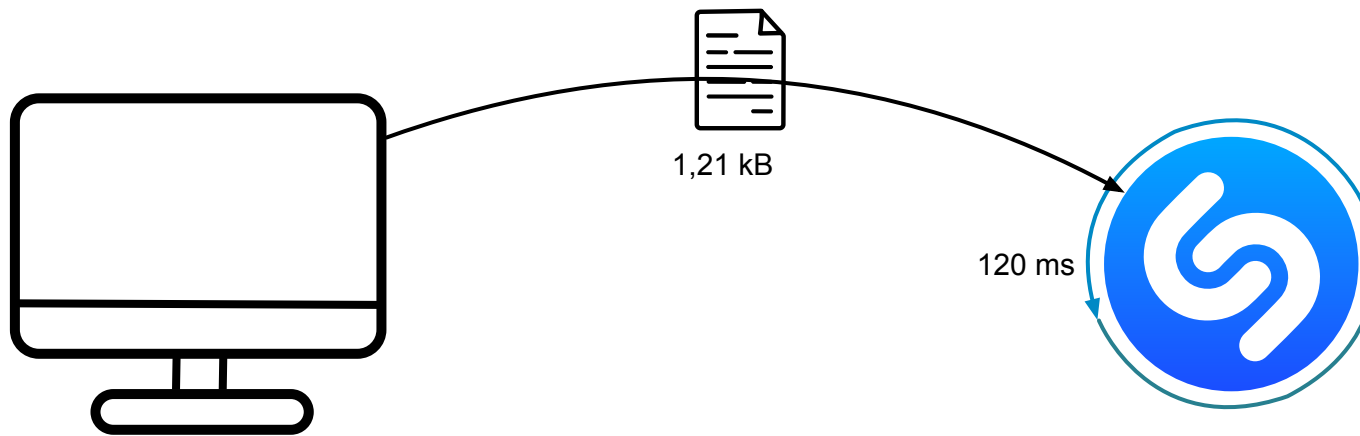
# Les exigences croissantes



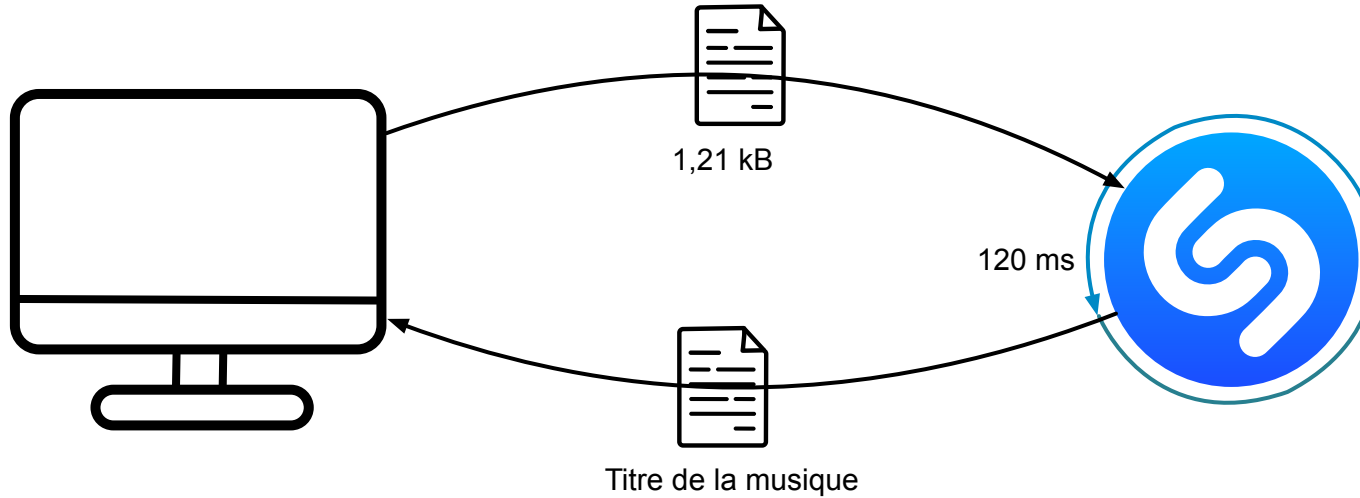
# Les exigences croissantes



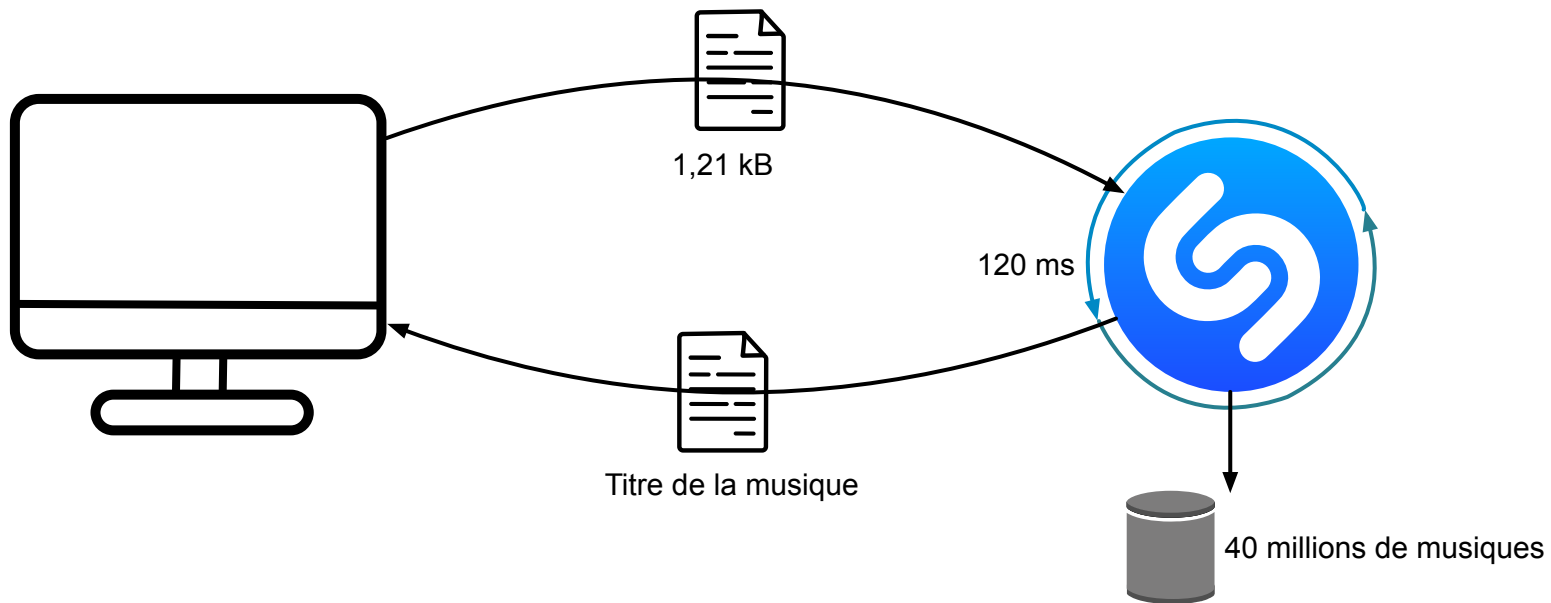
# Les exigences croissantes



# Les exigences croissantes



# Les exigences croissantes



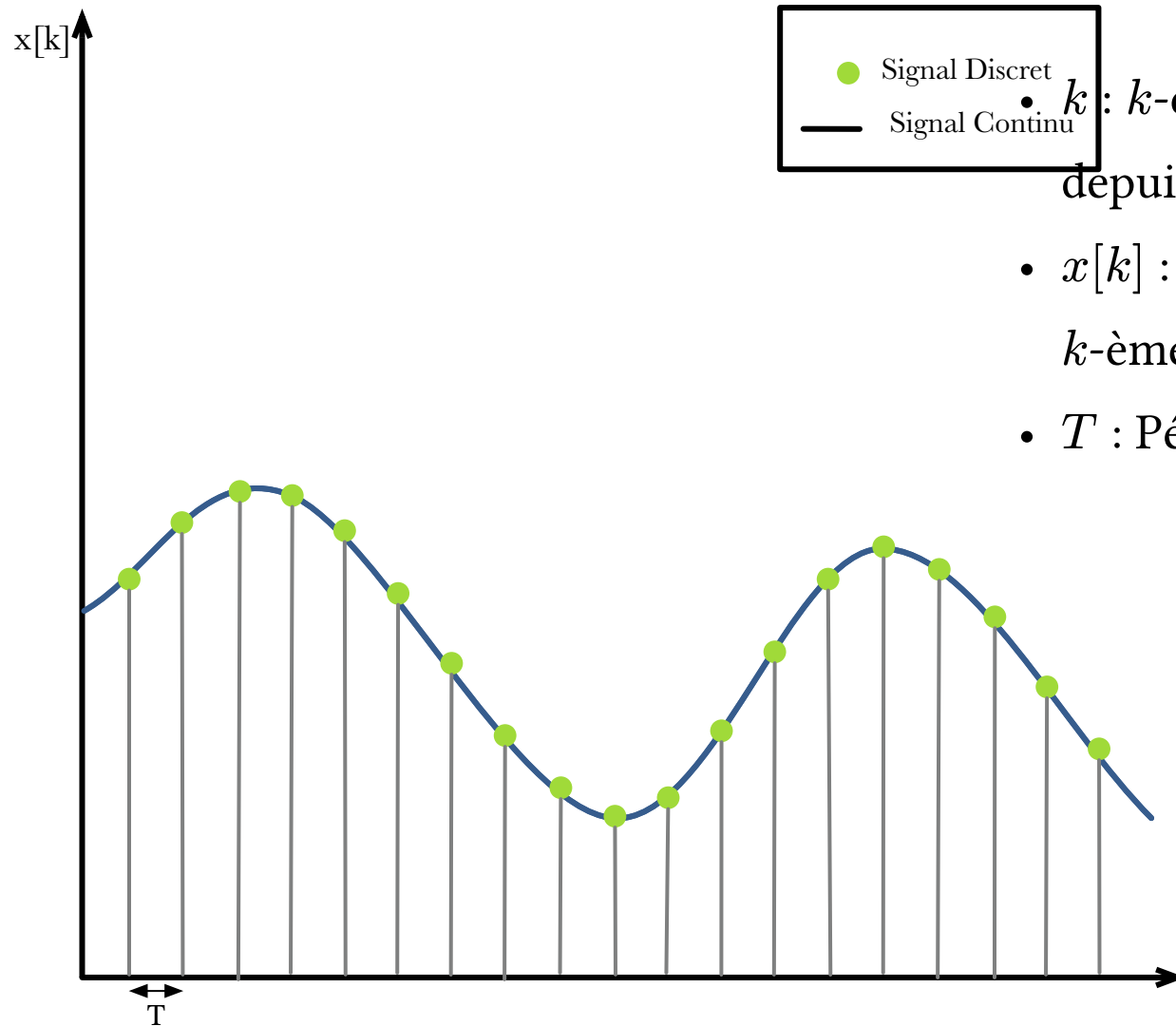


1. Introduction
2. **Problématique**
3. 1ère Approche - La Fast Fourier Transform
4. Les limites de la FFT
5. 2ème Approche - La STFT
6. Les limites de la STFT
7. L'algorithme Shazam

# Problématique

Comment identifier un morceau de musique complet de manière unique et efficace à partir d'un échantillon de celle-ci ?

# Modélisation du problème



- $k$  :  $k$ -ème donnée acquise numériquement depuis le début de l'acquisition.
- $x[k]$  : Amplitude correspondante à la  $k$ -ème donnéé.
- $T$  : Période d'acquisition

1. Introduction
2. Problématique
3. **1ère Approche - La Fast Fourier Transform**
4. Les limites de la FFT
5. 2ème Approche - La STFT
6. Les limites de la STFT
7. L'algorithme Shazam

# L'algorithme de Cooley-Tukey – Principales caractéristiques

# L'algorithme de Cooley-Tukey – Principales caractéristiques

- Récursif

# L'algorithme de Cooley-Tukey – Principales caractéristiques

- Récursif
- Basé sur le paradigme « diviser pour régner »

# L'algorithme de Cooley-Tukey – Principales caractéristiques

- Récursif
- Basé sur le paradigme « diviser pour régner »
- Complexité temporel en  $O(N \log(N))$  avec  $N$  le nombre de point acquis.



# L'algorithme de Cooley-Tukey – Principales caractéristiques

- Récursif
- Basé sur le paradigme « diviser pour régner »
- Complexité temporelle en  $O(N \log(N))$  avec  $N$  le nombre de point acquis.
- **Entrée** : Un signal sous forme de décomposition de fourrier.

# L'algorithme de Cooley-Tukey – Principales caractéristiques

- Récursif
- Basé sur le paradigme « diviser pour régner »
- Complexité temporel en  $O(N \log(N))$  avec  $N$  le nombre de point acquis.
- **Entrée** : Un signal sous forme de décomposition de fourrier.
- **Sortie** : La liste des coefficients de fourrier.

# L'algorithme de Cooley-Tukey – Principe

Soit :

- $N$  le nombre de points acquis.
- $x : \llbracket 0, N - 1 \rrbracket \rightarrow \mathbb{R}$  l'application qui pour tout point  $k \in \llbracket 0, N - 1 \rrbracket$  acquis associe son amplitude  $x[k]$ .
- $X : \begin{matrix} n \mapsto X[n] = \sum_{k=0}^{N-1} x[k] e^{-2\pi i \frac{kn}{N}} \\ \llbracket 0, N - 1 \rrbracket \rightarrow \mathbb{R} \end{matrix}$  où  $X[n]$  est appelé « coefficient de Fourier ».

# L'algorithme de Cooley-Tukey – Principe

$$\begin{aligned}
 X[n] &= \sum_{m=0}^{\lfloor \frac{N}{2}-1 \rfloor} x[2m] e^{-2\pi i \frac{2mn}{N}} + \sum_{m=0}^{\lfloor \frac{N}{2}-1 \rfloor} x[2m+1] e^{-2\pi i \frac{(2m+1)n}{N}} \\
 &= \underbrace{\sum_{m=0}^{\lfloor \frac{N}{2}-1 \rfloor} x[2m] e^{-2\pi i \frac{mn}{N}}}_{E[n]_{\frac{N}{2}}} + e^{-2\pi i \frac{nn}{N}} \underbrace{\sum_{m=0}^{\lfloor \frac{N}{2}-1 \rfloor} x[2m+1] e^{-2\pi i \frac{mn}{N}}}_{O[n]_{\frac{N}{2}}} \\
 &= E[n]_{\frac{N}{2}} + e^{-\frac{2\pi i}{N}} O[n]_{\frac{N}{2}}
 \end{aligned}$$

On en déduit que

$$\begin{cases} X[n] = E[n]_{\frac{N}{2}} + e^{-\frac{2\pi i}{N}} O[n]_{\frac{N}{2}} \\ X[n + \frac{N}{2}] = E[n]_{\frac{N}{2}} - e^{-\frac{2\pi i}{N}} O[n]_{\frac{N}{2}} \end{cases}$$

# L'algorithme de Cooley-Tukey – Pseudo-Code

```

1 Fonction FFT ( $X$ )
2   Soit  $N \leftarrow$  Taille  $X$ 
3   Si  $N = 1$  alors
4     | renvoyer  $X[0]$ 
5   Fin Si
6    $\text{partie\_paire} \leftarrow \text{FFT}(X[0:N:2])$ 
7    $\text{partie\_impaire} \leftarrow \text{FFT}(X[1:N:2])$ 
8   Pour  $k$  de 0 à  $N/2$  :
9     |  $t \leftarrow e^{-2i\pi \frac{k}{N}} \times \text{partie\_impaire}[k]$ 
10    |  $X[k] \leftarrow \text{partie\_paire}[k] + t$ 
11    |  $X[k + N/2] \leftarrow \text{partie\_paire}[k] - t$ 
12  Fin Pour
13  renvoyer  $X$ 

```

Initialement,  $X = \left[ x[k] * e^{2\pi i \frac{k}{N}} \right]_{k \in \llbracket 0, N \rrbracket}$

1. Introduction
2. Problématique
3. 1ère Approche - La Fast Fourier Transform
4. **Les limites de la FFT**
5. 2ème Approche - La STFT
6. Les limites de la STFT
7. L'algorithme Shazam

# Les performance de Shazam

# Les performance de Shazam

- Reconnaissance  $\approx 4$  s



# Les performance de Shazam

- Reconnaissance  $\approx 4$  s
- Fréquence d'échantillonnage = 44000 Hz

# Les performance de Shazam

- Reconnaissance  $\approx 4$  s
- Fréquence d'échantillonnage = 44000 Hz
- Nombre de point :  $N = 176400$

# Le résultat d'une simple FFT – Les échantillons



$\approx 4\text{ s}$

176400 points

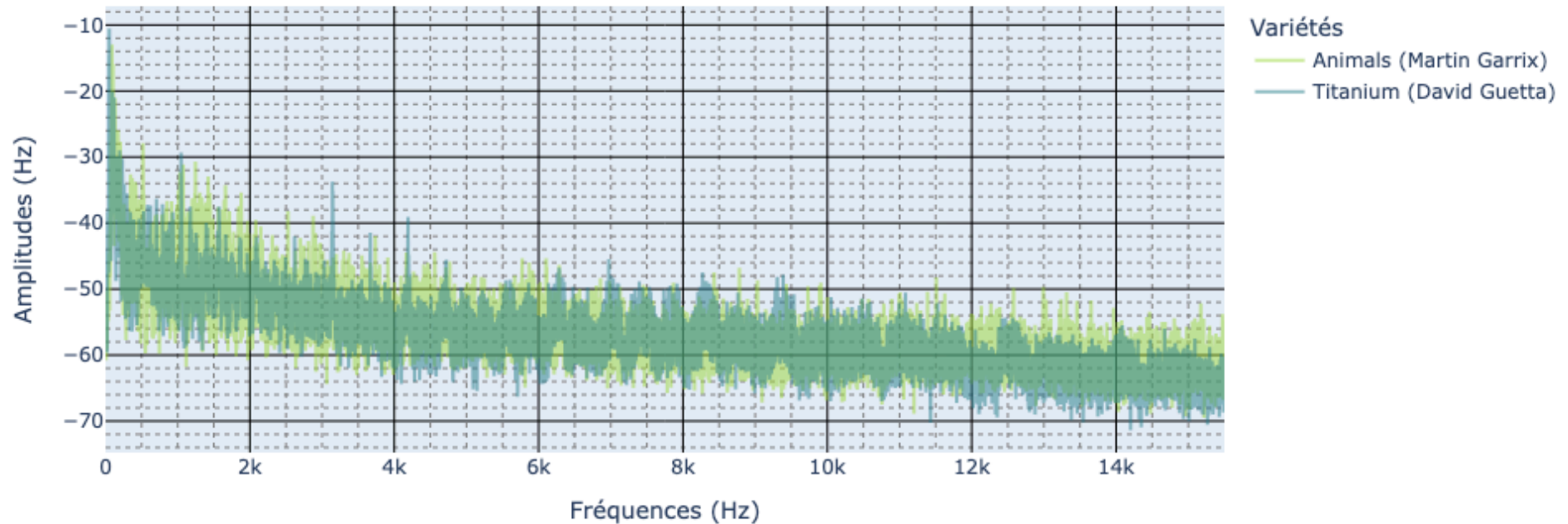


$\approx 4\text{ s}$

176400 points

# Le résultat d'une simple FFT – Le Résultat

## Spectre



# Le résultat d'une simple FFT – Bilan

- Graphe de pourcentage de même point (pour tout les points)
- Graphe de ce qui se passe avec un spectre différent

1. Introduction
2. Problématique
3. 1ère Approche - La Fast Fourier Transform
4. Les limites de la FFT
5. **2ème Approche - La STFT**
6. Les limites de la STFT
7. L'algorithme Shazam

# La STFT – Définition

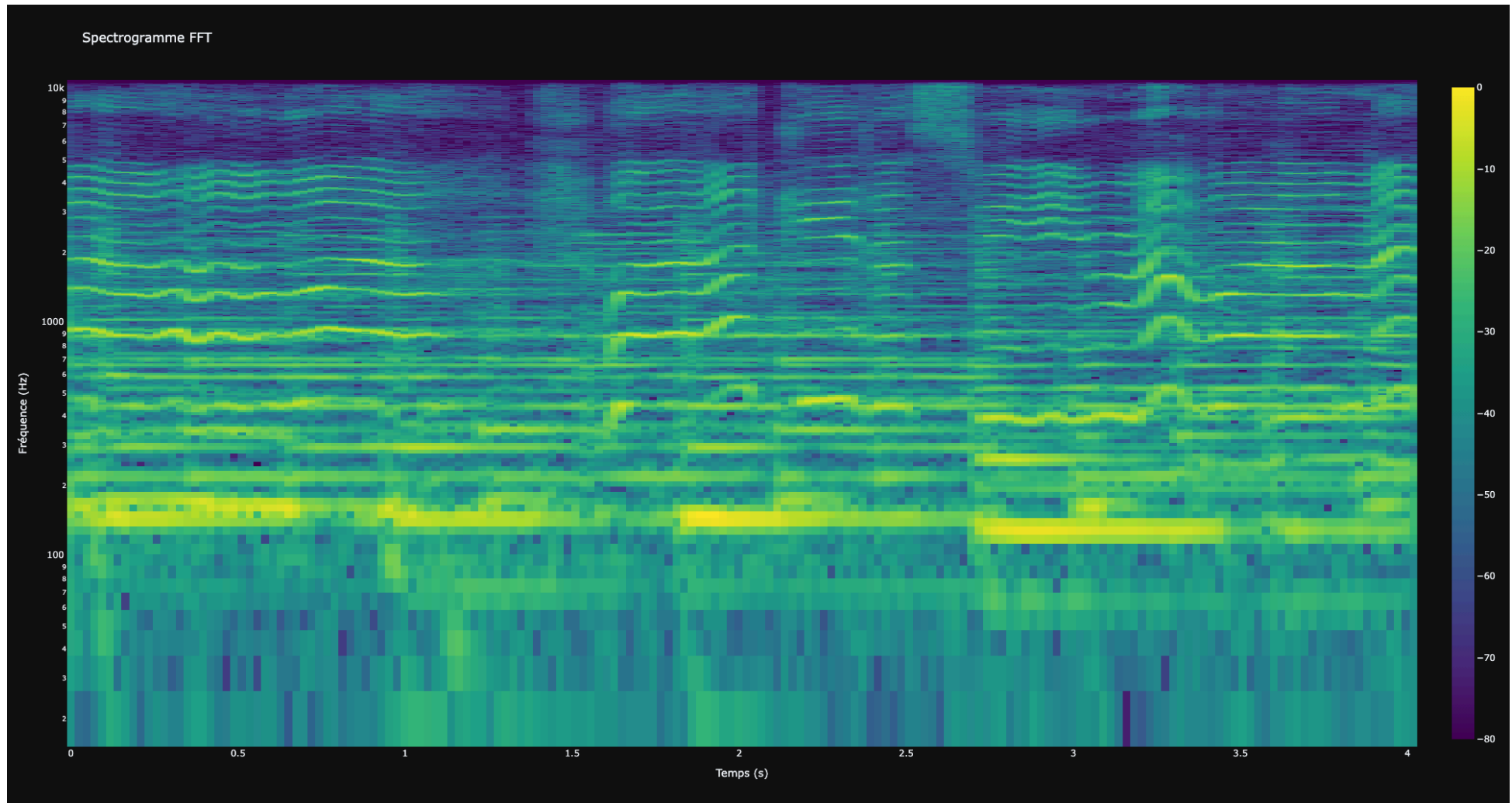
Algorithme permettant de représenter le spectre de fourrier d'un signal numérique en fonction du Temps

# Application – L'échantillon





# Application – Résultats



1. Introduction
2. Problématique
3. 1ère Approche - La Fast Fourier Transform
4. Les limites de la FFT
5. 2ème Approche - La STFT
6. **Les limites de la STFT**
7. L'algorithme Shazam

# Les limites – La taille de la clé

- Problème 1 : La taille de la clé

# La STFT – Définition


- Problème 1 : La taille de la clé




:  $\approx 2$  Mo pour 4 secondes

# La STFT – Définition


- Problème 1 : La taille de la clé


 :  $\approx 2$  Mo pour 4 secondes

 :  $\approx 3$  min 30 s

# La STFT – Définition

- Problème 1 : La taille de la clé


 :  $\approx 2$  Mo pour 4 secondes


 :  $\approx 3$  min 30 s

 :  $\approx 20$  M de son.

# La STFT – Problème

- La taille de la clé

 :  $\approx 2$  Mo pour 4 secondes

 :  $\approx 3$  min 30 s

 :  $\approx 20$  M de son.

Résultat : 206 565 To de donné à parcourir.

1. Introduction
2. Problématique
3. 1ère Approche - La Fast Fourier Transform
4. Les limites de la FFT
5. 2ème Approche - La STFT
6. Les limites de la STFT
7. **L'algorithme Shazam**

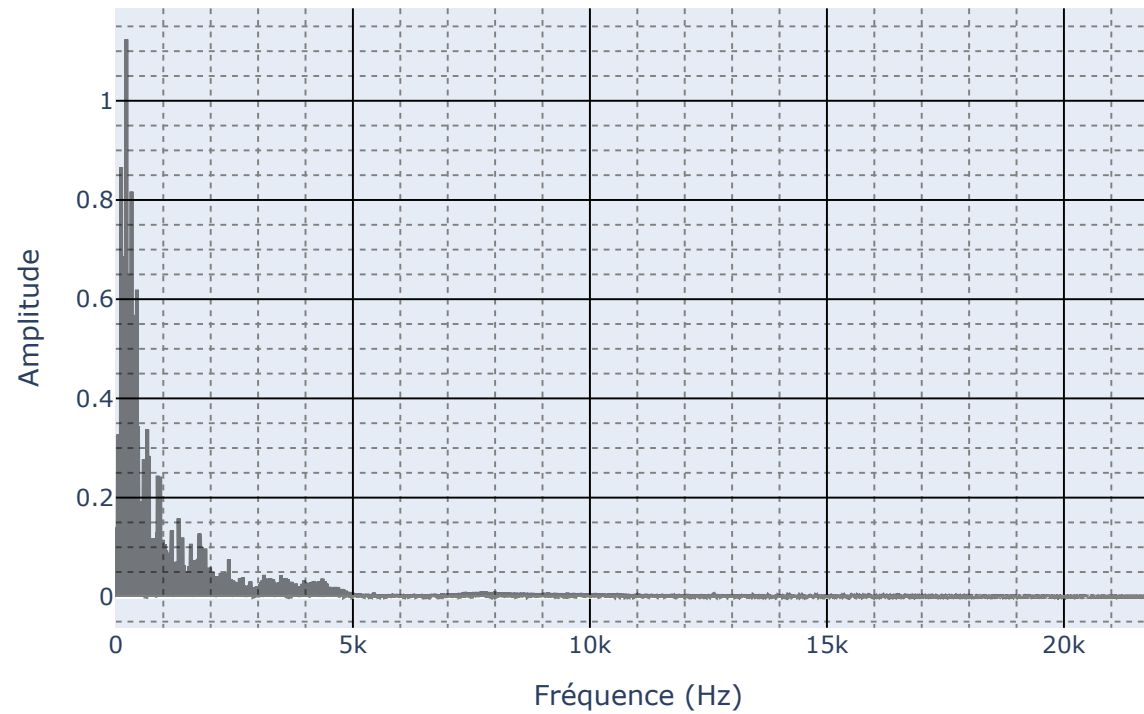


# Principe – L'échantillon utilisé



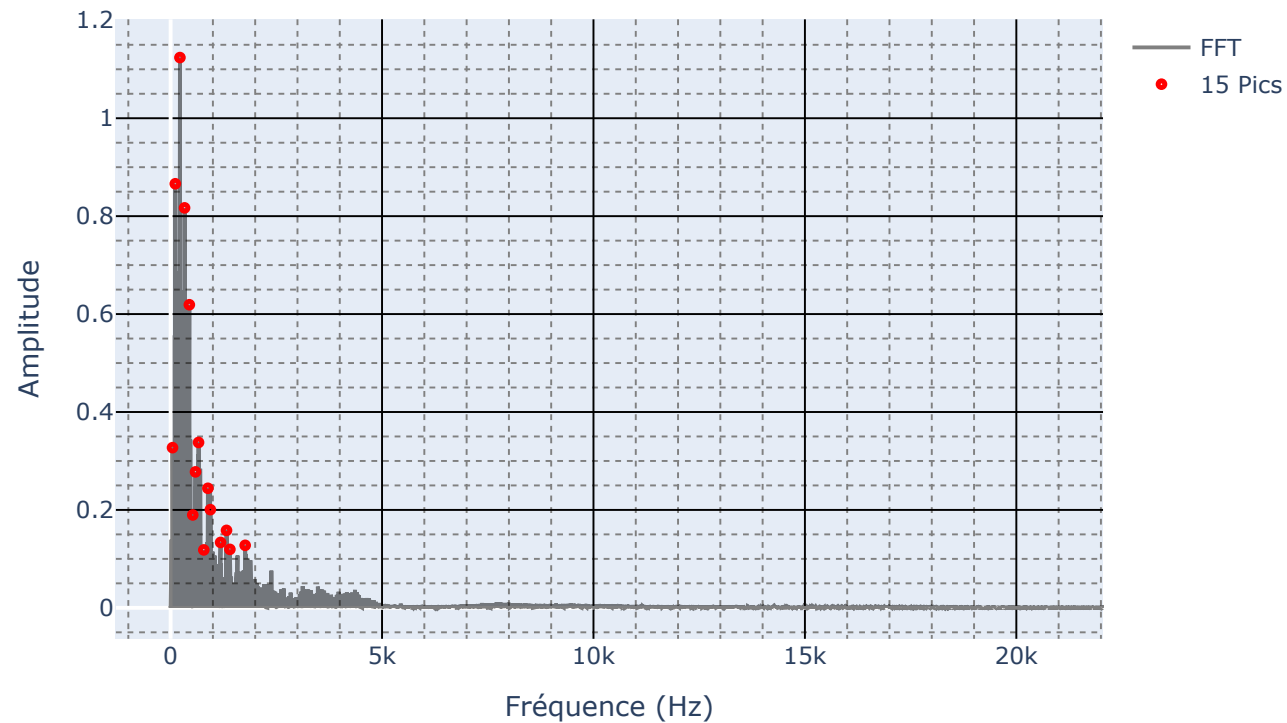
# 1ère étape : La FFT – Simple FFT

FFT

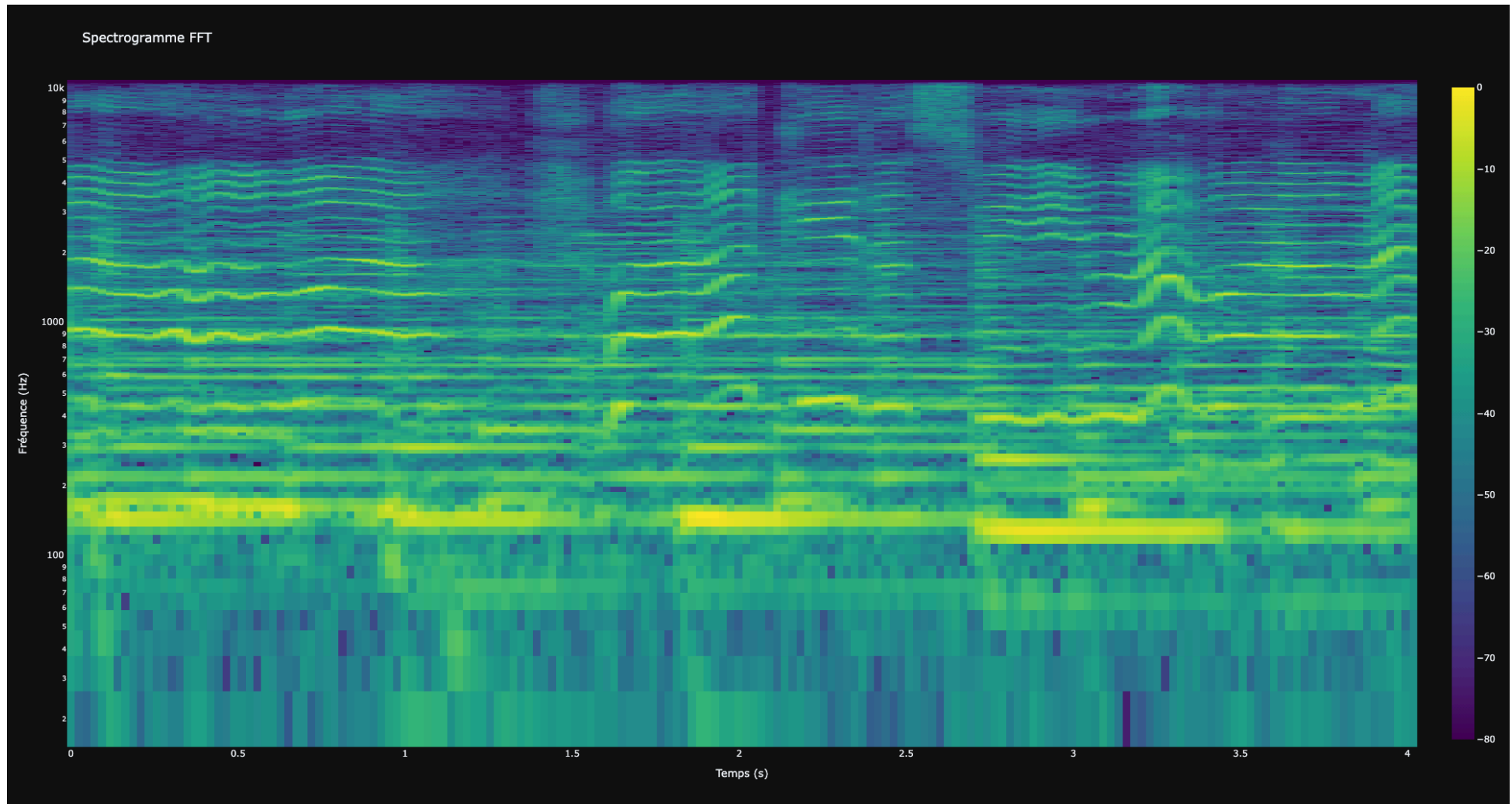


# 1ère étape : La FFT – Sélection des n extremum FFT

FFT

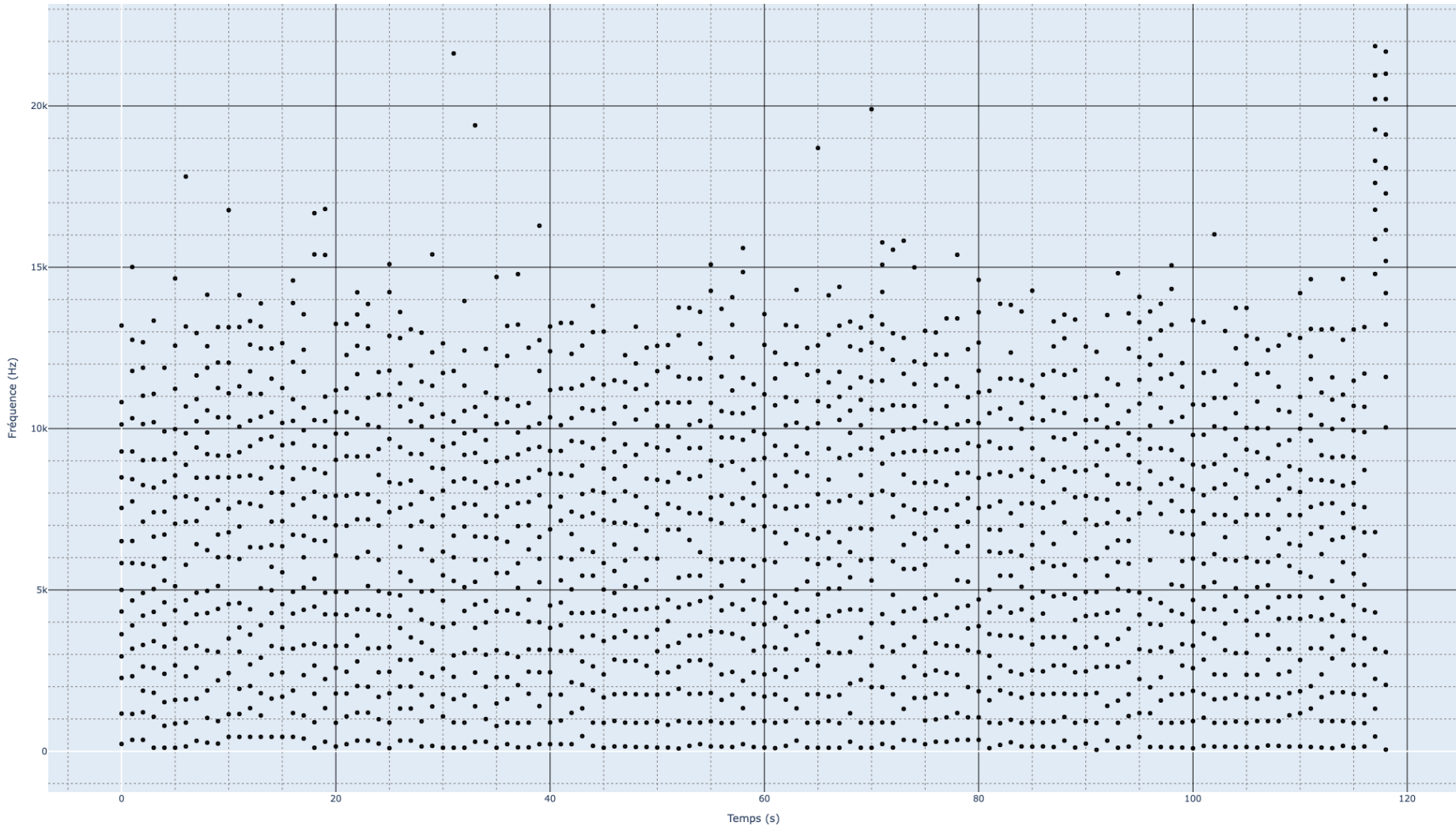


# 1ère étape : La FFT – Application du principe à la STFT



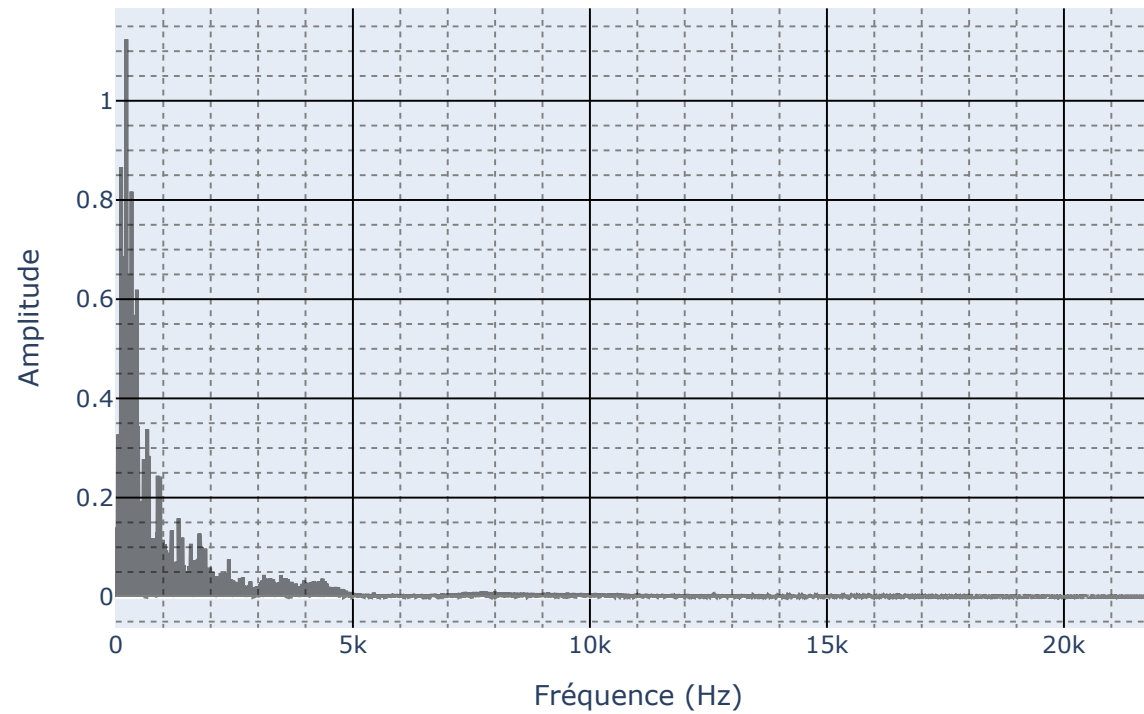
# 1ère étape : La FFT – Application du principe à la STFT

Constellation



# 1ère étape : La FFT – Simple FFT

FFT



## **Principe**

Explication du principe grâce au blog

## **Pseudo-Code**

Pseudo-Code

## **Test sur des exemples**

- Calcul du taux de collision
- Calcul de la taille de la clé
- Calcul du temps de génération de la clé (hors les 15s)

## **Conclusion**

## **Principe**

Explication du principe grâce au blog

## **Pseudo-Code**

Pseudo-Code

## **Test sur des exemples**

- Calcul du taux de collision
- Calcul de la taille de la clé
- Calcul du temps de génération de la clé (hors les 15s)

# Conclusion