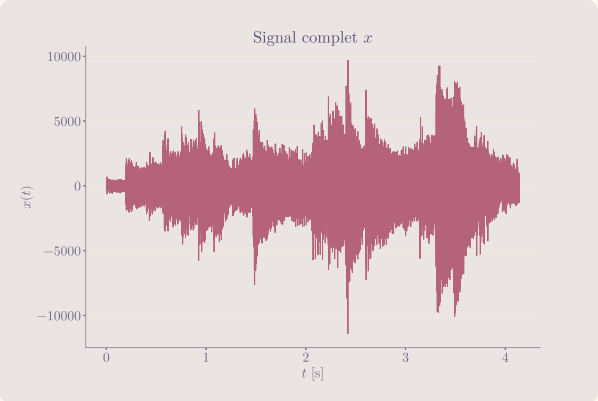
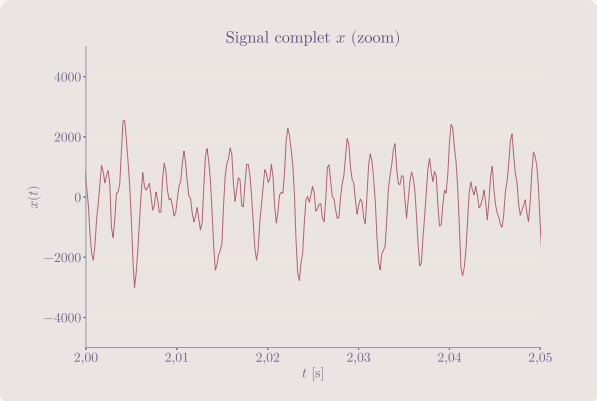
I. Récupération et visualisation de données

En Python, l'équivalent de `audioread` semble être `scipy.io.wavfile.read`. La commande `soundsc` envoie le son dans les haut-parleurs, un équivalent Python semble être possible avec le module `sounddevice` (et notamment la fonction `play` qui permet de lire un tableau `numpy`). Pour la question 4, on s'attend à répondre « non », et qu'il faut privilégier une analyse *fréquentielle* plutôt qu'une analyse *temporelle*.

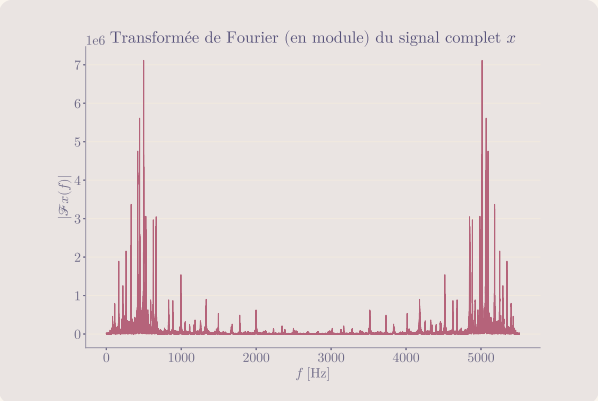


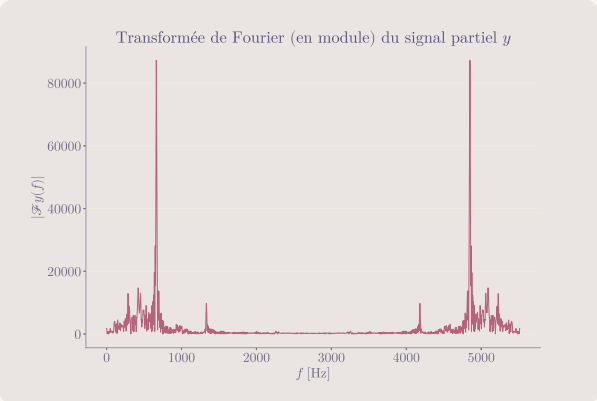


II. Analyse spectrale du signal

Pour la question 2, on peut reconnaitre les notes jouées, mais on manque l'information temporelle (quand est jouée la note), on ne peut donc pas remonter à la partition.

En se concentrant sur 100 ms, on sait quelles notes sont jouées pendant ce court instant car on peut identifier les « pics » de la transformée de Fourier (en module).





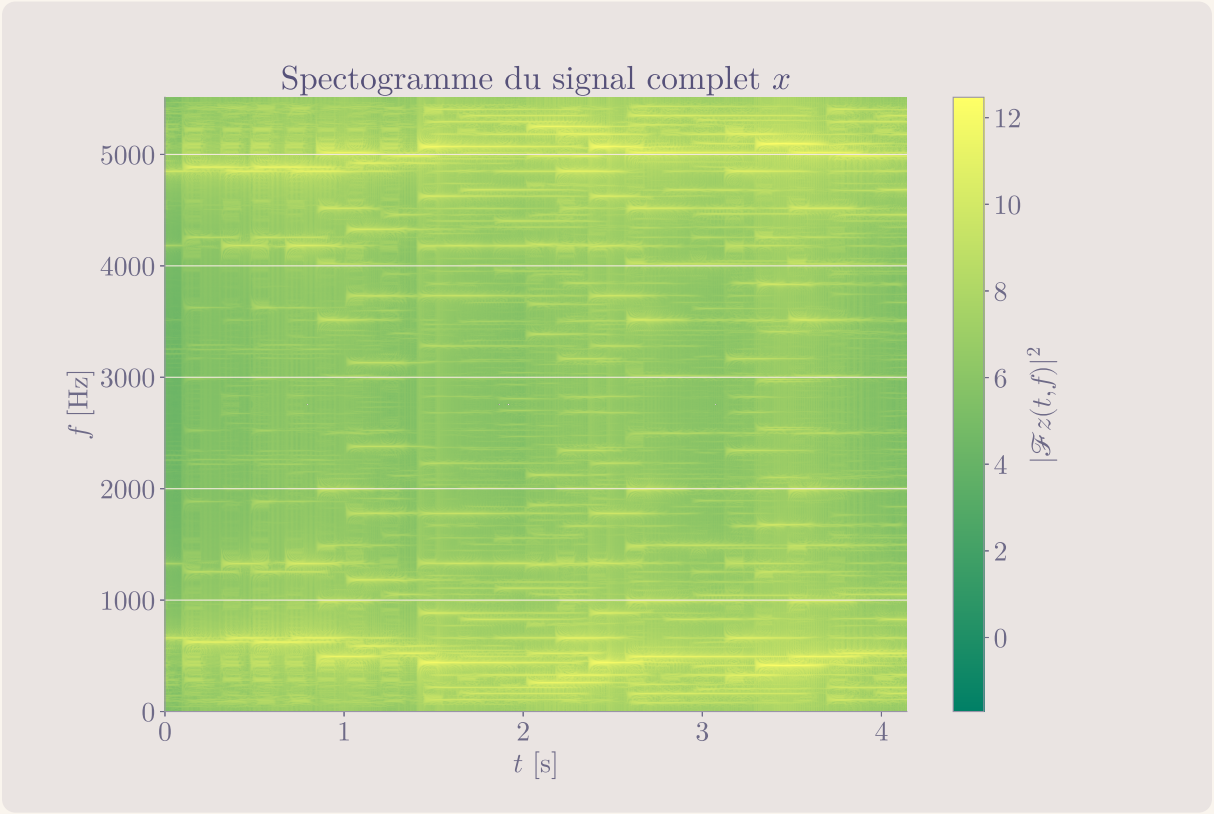
III. Analyse temps-fréquence

Lorsqu'on analyse le signal temporellement uniquement, on perd l'information de fréquence, on ne sait pas répondre à « quelles notes sont jouées ? ». Lorsqu'on analyse le signal en fréquence uniquement, on perd l'information de temps, on ne sait pas répondre à « quand est jouée la note ? ». On analyse donc le signal en temps et en fréquence en même temps.

On isole des fenêtres de 100 ms tout au long du signal, et ses fenêtres se recouvrent (on n'isole pas une partie du signal toutes les 100 ms mais à chaque échantillon). La matrice `S` est de la forme suivante :

$$\texttt{S} = \begin{bmatrix} \cdots & \boxed{\begin{array}{c}\text{FFT}\\ \text{de la}\\ \text{fenêtre}\\ [t_k, t_k+\tau]\end{array}} & \cdots \end{bmatrix}.$$

Avec les notations de l'énoncé la colonne rouge correspond au vecteur `|fft(z)|^2` transposé.



IV. Détection de notes

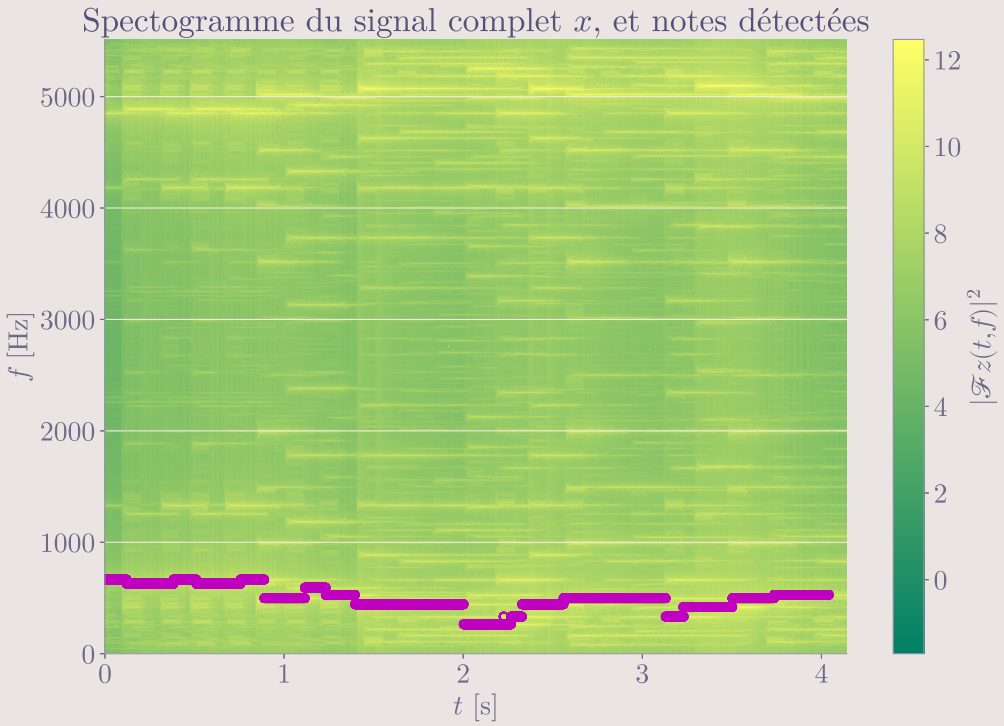
L'équivalent de ce que fait la fonction `max` en Matlab peut se faire avec la fonction `np.argmax` du module `numpy`.[1] Pour trouver la fréquence la plus proche parmi celles du tableau, on trouve l'index i minimisant $|f - f_i|$ où f est la fréquence qu'on a détecté, et f_i correspond à la i-ème fréquence du piano. Le tableau `liste_notes` ne contient pas les indices, mais bien les fréquences à jouer.

Pour commenter le tracé, on peut écouter la musique et comparer la hauteur des notes et les fréquences détectées.

La musique obtenue est très « électronique ». On peut y remédier :
- on peut inclure plus d'harmoniques (pour le moment, une seule est incluse),
- on peut permettre à deux notes d'être jouées en même temps,

[1] Attention, un problème que j'ai eu en testant le programme en Python est qu'il détecte parfois le second pic (dû au repliement), ce qui n'était pas le cas du programme Matlab (peut-être dû à des précisions différentes sur les nombres). J'ai donc dû limiter la « zone » de la FFT analysée à $[0, F_s/2]$, pour ne pas avoir de problèmes.

- on peut changer l'amplitude à laquelle on joue pour qu'elle corresponde plus à ce que l'on voit sur le spectrogramme : par exemple, jouer $A\exp(-t/\tau')\sin(2\pi ft)$ au lieu de simplement $A\sin(2\pi ft)$.



V. Autres questions posées, pas dans le sujet de TP

- *À quoi correspond l'autre fréquence détectée dans la question 2.3 ?* Ça n'est pas une note jouée : on peut le voir en comparant les fréquences, ou sinon en sachant que pour jouer *Für Elise* on ne joue qu'une note à la fois. Il y a deux solutions. Ça peut être une harmonique, mais ce n'est pas possible car ce n'est pas un multiple de l'autre fréquence détectée. C'est un phénomène de repliement, le pic a lieu en $f_s - f_0$ où f_0 est l'autre fréquence détectée (celle jouée au piano), et f_s est la fréquence d'échantillonnage.
- *Certaines notes n'apparaissent que pour quelques « points » dans le graphe de la partition, comment s'en débarasser ?* On peut les filtrer facilement, mais on peut aussi jouer plusieurs notes en même temps.

$\dot{i}$