APP

Traitement numérique du signal

Semaine 1 – Signaux échantillonnés

Groupe G4E

Grégoire Debray-Genty

Alexandre Lalau

Ambroise Malinvaud

Karim Ouarti

Alexandre Perbet

Mis à jour le 22/04/2020

Table des matières

I – Signaux échantillonnés

Problème I-A

Introduction

Le premier problème consiste à créer un algorithme capable de **détecter la présence ou l'absence d'un signal audio**. L'idée est de fournir un fichier .wav à l'algorithme MATLAB. Un signal échantillonné est extrait du fichier audio. Ce signal va alors être traité par l'algorithme qui va renvoyer un graphique indiquant les moments où un signal audio est détecté, c'est-à-dire la présence d'autre chose que du bruit de fond. Nous allons vous détailler le fonctionnement de notre algorithme.

Méthode proposée

Le début du programme fonctionne de manière classique : on réinitialise les variables, on charge les fichiers audios à analyser et on convertit les sons stéréos en sons mono pour pouvoir les traiter.

On va ensuite créer un vecteur rempli de zéros de même longueur que le vecteur qui contient les échantillons du signal audio. Celui-ci servira à stocker la puissance de chaque échantillon et ainsi de savoir s'il y a un signal sonore ou non.

L'une des questions que nous nous sommes posées pour déterminer la présence d'un signal a été de déterminer sur quelle caractéristique du signal nous allions nous baser. Deux choix se présentaient à nous : calculer la puissance instantanée ou calculer la puissance moyenne de chaque échantillon. Nous avons choisi d'utiliser la puissance instantanée car elle est plus simple à calculer et donc à implémenter. Le nombre de calculs à faire pour traiter un signal est donc plus faible et par conséquent le programme est moins lourd à exécuter pour l'ordinateur.

La formule du calcul de la puissance instantanée à l'échantillon n est :

$$p(n) = x(n)^2$$

Afin de différencier un signal audio du bruit de fond nous allons d'abord calculer la puissance instantanée de chaque échantillon sur les 50 premières millisecondes du fichier audio. En effet, l'énoncé indique que ces 50 premières millisecondes ne contiennent que du bruit de fond. Ainsi, nous obtenons une valeur seuil. Pour calculer ce seuil nous devons déterminer le rang de l'échantillon qui correspond à 50 ms. Ce rang varie en fonction de la fréquence d'échantillonnage du signal. Par exemple pour un signal échantillonné à 96 kHz :

$$f_e = 96000\ Hz$$

Pour obtenir le rang n de l'échantillon qui correspond à 50 ms on fait le calcul suivant :

$$n = 0{,}05 f_e = 4800$$

Ainsi, on sait qu'à l'échantillon 4800 il s'est écoulé 50 ms. On va alors calculer la puissance instantanée des 4800 premiers échantillons pour déterminer qu'elle est la puissance maximale relevée sur les 50 premières millisecondes. Cette valeur maximum sera notre seuil. En dessous de cette valeur on considèrera qu'il s'agit de bruit de fond, au-dessus de son.

Ensuite nous allons balayer les échantillons restants. Pour chaque échantillon nous allons aussi calculer la puissance instantanée et la comparer à la valeur seuil calculée précédemment. Si la puissance instantanée est inférieure au seuil on laisse la valeur 0 dans notre vecteur créé au début du programme.

En revanche, si la puissance instantanée est supérieure au seuil deux choix s'offrent à nous : soit mettre un 1 à l'index de l'échantillon dans notre vecteur, soit mettre la valeur de puissance calculée. Nous avons décidé de garder la puissance instantanée calculée pour plusieurs raisons : tout d'abord l'affichage graphique de la puissance est beaucoup plus clair et lisible (voir figure 1) et ensuite la puissance instantanée permet de se rendre compte de l'intensité du signal sonore à un moment donné (voir figure 2).

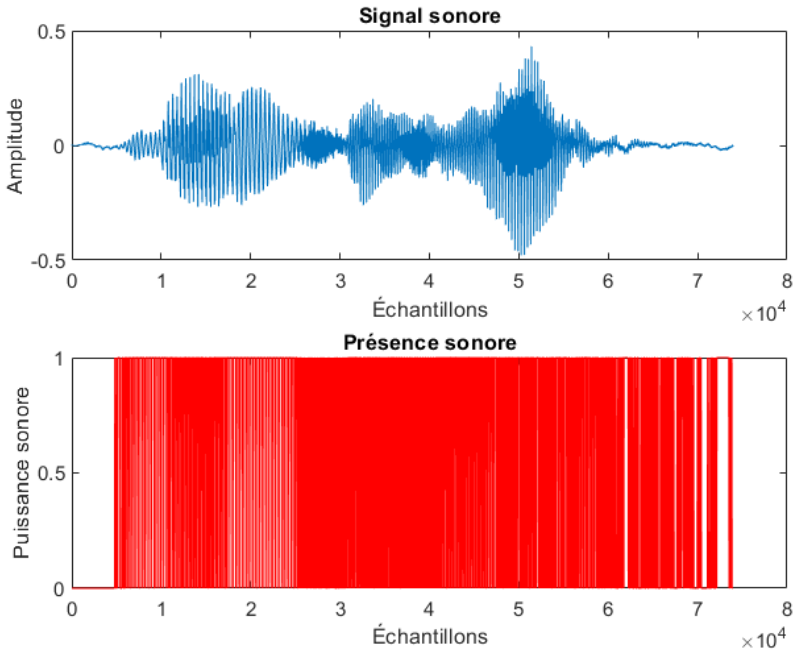


Figure 1 - Graphique avec la puissance indiquée avec 1 ou 0

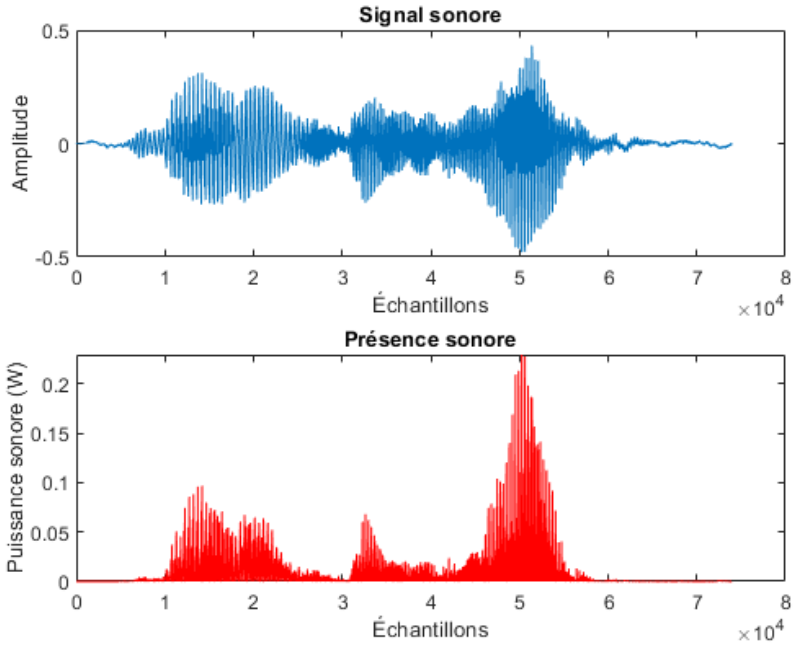


Figure 2 - Graphique avec la puissance instantanée réelle en W

Concernant l'unité à utiliser pour la puissance instantanée nous avons essayé de la mettre en dBm mais les valeurs devenant alors négatives l'affichage n'était pas lisible non plus, nous les avons donc laissées en watts.

Résultats expérimentaux

Nous avons appliqué notre algorithme aux quatre signaux demandés. Comme vous pouvez le voir le programme fournit un résultat satisfaisant et lisible pour chacun des signaux. On identifie clairement les moments du signal où du son est présent ainsi que son intensité.

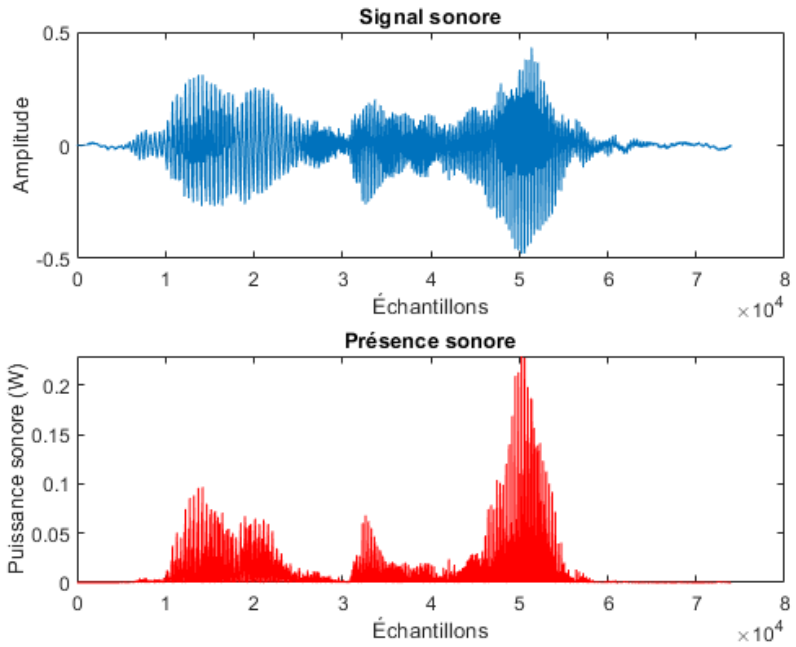


Figure 3 - Signal "Bonne journée"

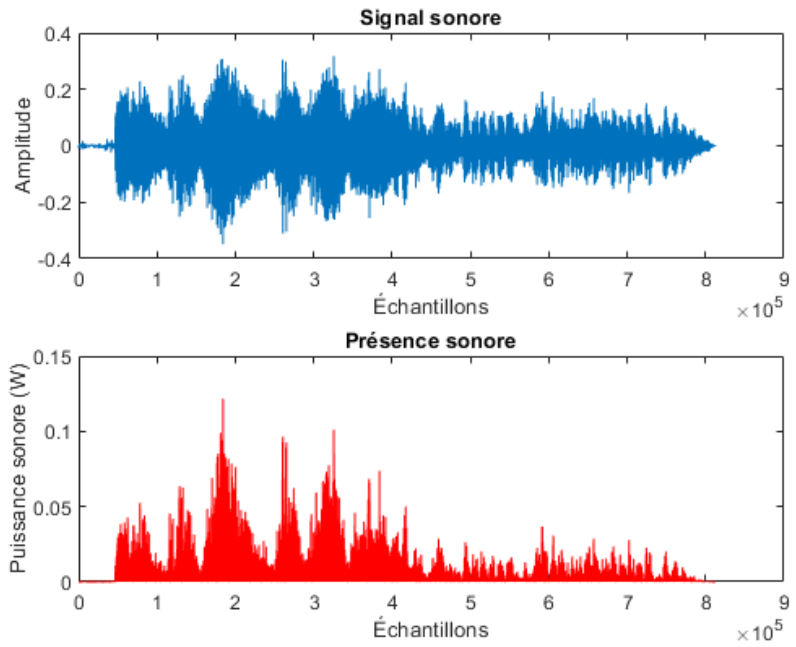


Figure 4 - Signal "James Bond Theme"

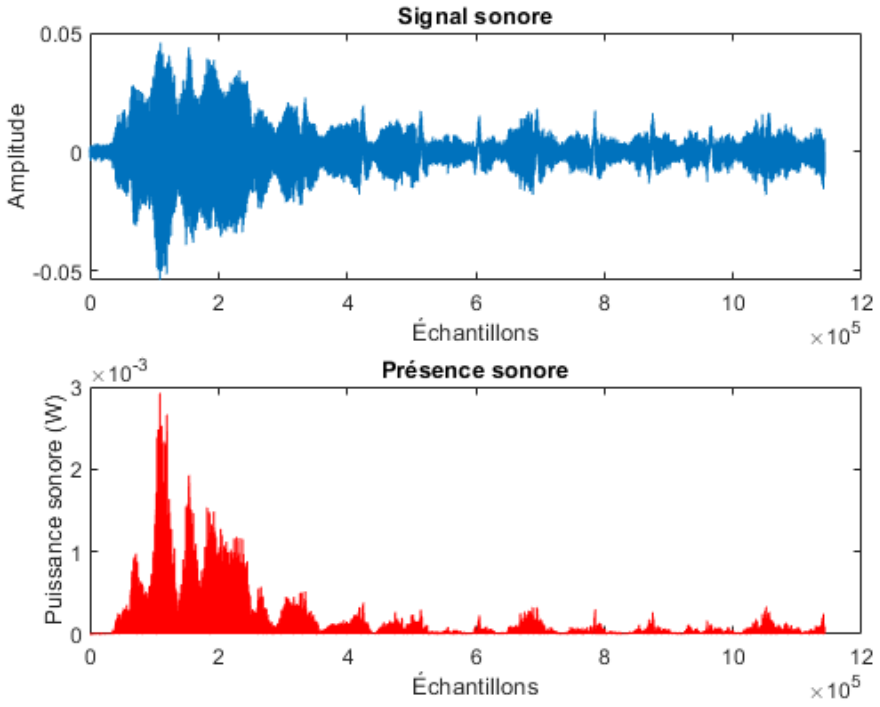


Figure 5 - Signal "Allegretto"

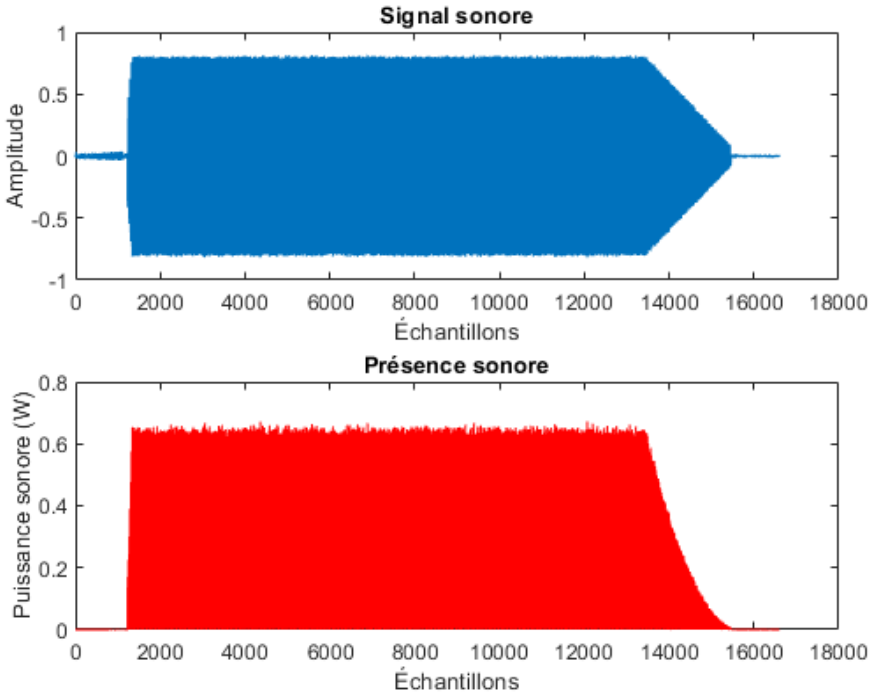


Figure 6 - Signal "Atone"

Conclusion

Le but de cet algorithme était de déterminer la présence d'un signal sonore ou non. Nous avons pu nous familiariser avec la notion de bruit de fond et la technique que nous avons développée nous a permis d'apprendre à supprimer en partie ce bruit de fond afin d'isoler le signal utile. Ces connaissances nous serons probablement très utiles pour le développement du programme de la fonction « reconnaissance de tonalité ».

Problème I-B

Introduction

Le but de ce second problème est de générer deux signaux purs avec des caractéristiques aléatoires et de trouver une méthode pour comparer leurs fréquences et ainsi déterminer s'ils se ressemblent ou non. Pour ce faire nous avons utilisé une méthode vue au cours du chapitre 1 : l'intercorrélation de deux signaux.

Méthode proposée

La première partie du programme consiste à générer deux signaux aléatoires. Les caractéristiques à générer sont les suivantes : amplitude, fréquence et phase.

Pour la génération de l'amplitude nous avons fixé les valeurs possibles entre 1 et 5 compris. Cela génère des signaux ayant des amplitudes variables ce qui nous a semblé cohérent.

Concernant la génération de fréquence aléatoire nous avons généré, comme l'indiquait l'énoncé, des fréquences comprises entre 130 Hz et 4000 Hz.

Et enfin pour la phase nous générons un nombre compris entre 0 et π radians, c'est-à-dire un décalage de phase compris entre 0° et 180°.

La fréquence d'échantillonnage est quant à elle fixée à 16 000 Hz par l'énoncé. Cependant, nous l'avons augmenté à 64 000 Hz pour obtenir un signal plus propre lors de l'affichage.

Nous obtenons ainsi nos deux signaux audio purs de la forme :

$$Acos(2\pi ft + \varphi)$$

Nous affichons ces deux signaux superposés sur un graphique pour que l'utilisateur se rende compte visuellement de la ressemblance, ou non, des deux signaux.

L'idée est ensuite de calculer l'intercorrélation entre les deux signaux. Cette méthode de comparaison vue en cours se calcule grâce à la formule suivante :

$$C_{x,y}(\tau) = \int_{-\infty}^{+\infty} x(t)y(t+\tau)dt$$

Pour les signaux échantillonnés :

$$C_{x,y}(kT_e) = \sum_{n=-\infty}^{+\infty} x(nT_e)y((n+k)T_e)$$

Le logiciel MATLAB propose directement une fonction qui permet de calculer l'intercorrélation entre deux signaux. Nous l'avons donc utilisée.

Afin de pouvoir comparer les résultats entre plusieurs signaux et pouvoir analyser l'intercorrélation plus facilement nous avons normalisé l'intercorrélation. Autrement dit, tous les résultats sont ramenés entre 0 et 1 ce qui est beaucoup plus facile à interpréter par la suite.

En dessous du graphique sur lequel sont représentés nos deux signaux, nous avons tracé l'intercorrélation normalisée.

Il nous faut maintenant exploiter cette intercorrélation afin de déterminer la ressemblance entre les deux signaux. Pour cela nous avons déterminé la valeur absolue maximale de l'intercorrélation. En effet, l'intercorrélation étant normalisée, si deux signaux ne se

ressemblent presque pas, l'amplitude de leur intercorrélation sera proche de zéro. À l'inverse, si deux signaux sont très proches, la valeur absolue maximale de l'amplitude sera proche de 1. Ainsi, cette valeur absolue maximale représente le taux de ressemblance entre nos deux signaux. Par exemple, si cette valeur vaut 0,7543 cela signifie que les signaux se ressemblent à 75,43 %.

Nous avons arbitrairement fixé le seuil de ressemblance à 90 %. C'est-à-dire que si deux signaux se ressemblent à plus de 90 % nous considérons qu'ils sont presque similaires.

Ainsi, à la fin du programme, en fonction du pourcentage de ressemblance obtenu nous affichons dans la fenêtre de commande le résultat, par exemple :

```
Les signaux ne sont pas identiques.
Pourcentage de ressemblance : 0.7182 %.
f1 = 3208 Hz et f2 = 853 Hz.
0
```

Dans cet exemple les deux signaux sont considérés comme différents car ils ont un taux de ressemblance de 0,7182 %. L'affichage des deux fréquences nous le confirme. Un zéro est renvoyé si les signaux sont différents, sinon 1, comme indiqué dans l'énoncé.

Résultats expérimentaux

Voici les résultats obtenus pour deux signaux qui ne se ressemblent pas :

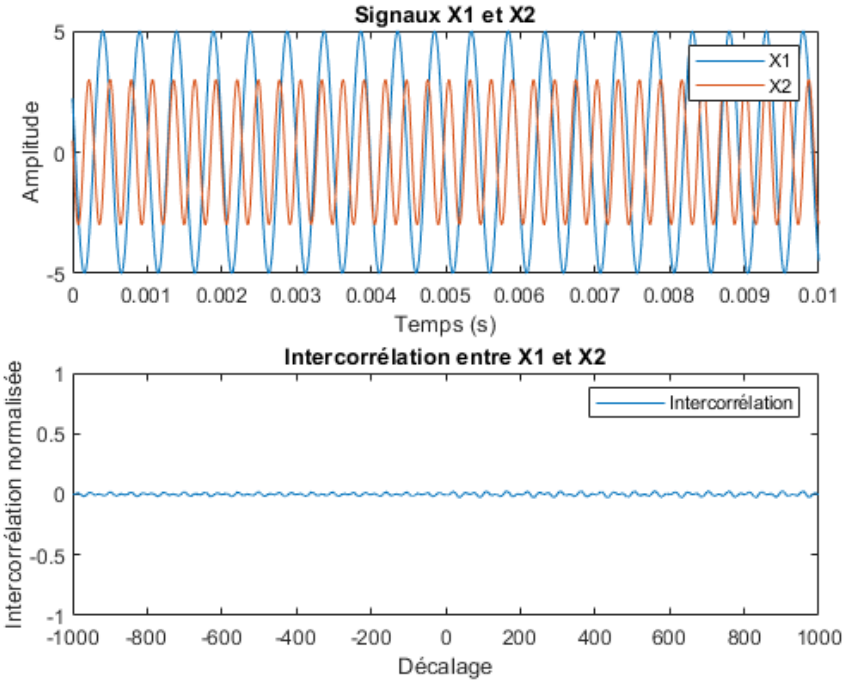
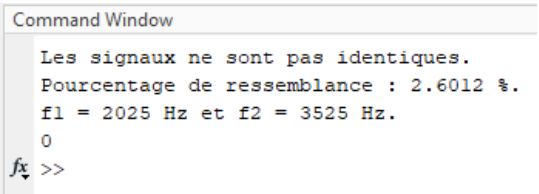


Figure 7 - Signaux ne se ressemblant pas



Plus les signaux se ressemblent, plus l'amplitude de l'intercorrélation est grande :

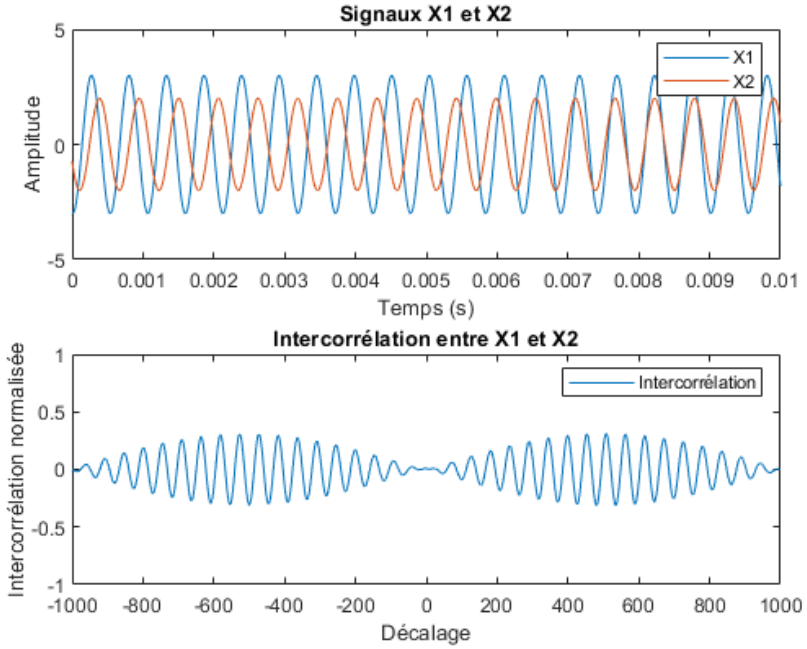


Figure 8 - Signaux se ressemblant très légèrement

```
Command Window
  Les signaux ne sont pas identiques.
  Pourcentage de ressemblance : 31.1497 %.
  f1 = 1887 Hz et f2 = 1786 Hz.
  0
fx >>
```

Et enfin, quand deux signaux sont pratiquement identiques, l'intercorrélation a une amplitude très proche de 1 :

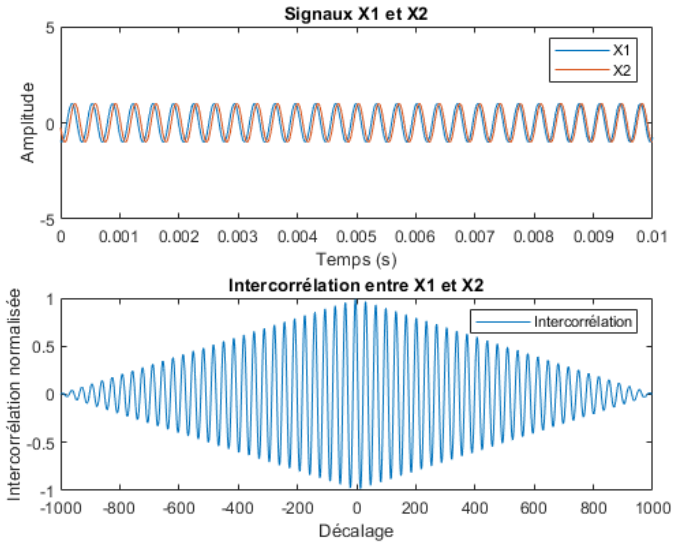
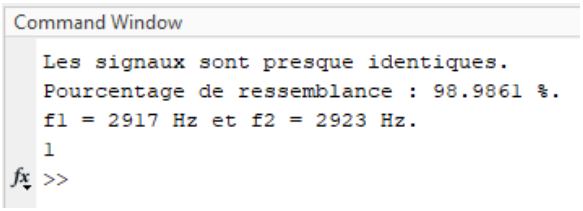


Figure 9 - Signaux pratiquement identiques



Les valeurs des deux signaux sont en effet quasiment les mêmes : 2917 Hz et 2923 Hz. Le pourcentage de ressemblance entre les deux signaux est alors de 98,9861 %.

Conclusion

Ce problème nous a permis de bien saisir l'utilité de l'intercorrélation entre deux signaux et la manière de la mettre en place. Cette méthode sera probablement utile pour la fonction « reconnaissance de tonalité » afin de comparer la fréquence produite par l'utilisateur et celle générée par le programme.