Activité 1 : Analyse de la parole (Energie, Passage par zéro, Autocorrélation, F0)

[durée en séance ~3h]

Objectifs : Effectuer quelques analyses dans le domaine temporel sur un signal de parole et visualiser les résultats. Ces analyses seront utilisées pour réaliser une première ébauche de calcul de F0.

Outils : Python (Librairies utilisées numpy, scipy et matplotlib)

Pour ce TP vous utiliserez principalement le fichier test_seg.wav (même si les programmes seront écrits pour n'importe quel fichier wav)

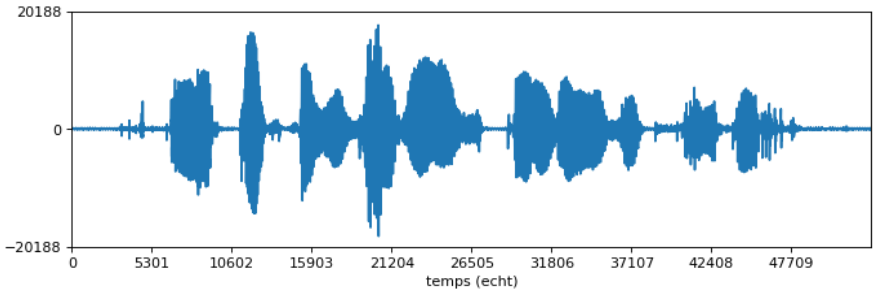
Le signal de parole

Ouvrez un fichier wav (et afficher l'information de la fréquence d'échantillonnage, et la taille du fichier en échantillons et en millisecondes)

Pour lire un fichier wav : **from scipy.io.wavfile import read** La fonction read renvoie la fréquence d'échantillonnage et le tableau des valeurs du signal

Réalisez une fonction d'affichage du signal.

Pour afficher des points : **import matplotlib.pyplot as plt** avec la fonction plot(signal) et show() Il est possible de modifier les éléments du graphique : xticks pour les étiquettes de l'axe des x et xlabel pour le titre de l'axe de x (idem pour y).



*[Rappel : si le signal est échantillonné à 16000 Hz : 4ms correspond à 4*16000/1000 échantillons]*

Energie d'un signal :

Réalisez une fonction qui donne le (logarithme de l') énergie d'un signal toutes les *m* échantillons (pour tester on prendra l'équivalent de 4ms) sur des fenêtres de taille *N* (pour tester on prendra l'équivalent en échantillons de 10 ms).

Cette fonction retournera un tableau de valeurs

```
def computeenergy(signal, m, N)
```

Cette fonction parcourra le tableau *signal*. [la taille du tableau retourné correspond au nombre de fois où la fenêtre de calcul a pu être déplacée (pour vérifier : len(signal)-N)/m +1]
Cette fonction fera appel à une autre fonction qui calcule effectivement le (logarithme de l') énergie du morceau de signal passé en paramètre.

```
def windowenergy(sig)
```

Énergie d'un signal (à court terme) : $E(n) = \sum_{k=0}^{N} s^2(n+k)$

avec *s(n)* le signal de parole à l'instant n
Il faut *caster* les valeurs du tableau avec `numpy.float32(sig)`
[Normalement la valeur de l'énergie correspond à l'énergie au milieu de la fenêtre à *n*+*N*/2]

Pour représenter l'énergie, il est préférable d'utiliser une échelle en dB :

$$10 \log_{10}(E(n))$$

Réalisez une fonction d'affichage de la courbe.

L'énergie est généralement plus élevée dans les parties voisées du signal de parole. Comparez vos résultats avec le résultat donné par Winsnoori.

Le taux de passage par zéro

Ce calcul peut donner une information sur le signal de parole : En effet, le taux est plus élevé pour les sons non-voisés (ou bruits) que pour les sons voisés.

Mais avant de poursuivre **:** La moyenne d'un signal doit être nulle sinon on parle d'une *composante continue* qu'il faut enlever pour ne pas fausser l'analyse du signal de parole et notamment le taux de passage par zéro qui est très sensible à cela. Calculez la moyenne du signal et si sa valeur est positive (ou négative) enlevez-la du signal. (Il faut donc le faire après l'ouverture du signal)

Réalisez une fonction qui calcule le taux de passage par 0 toutes les *m* échantillons sur des fenêtres de taille *N*.(on prendra aussi 4ms et 10 ms). Cette fonction retournera un tableau de valeurs. Procédez comme pour le calcul de l'énergie.

Taux de passage par zéro (à court terme):

$$Z(n) = \frac{1}{2N} \sum_{k=0}^{N} |\operatorname{sgn} s(n+k) - \operatorname{sgn} s(n+k-1)\ |$$

avec sgn s(n) le signe de l'échantillon s(n) (1 si s(n)>0, -1 sinon).

(Il n'existe de fonction "signe" toute faite en python)

La fonction d'autocorrélation

Elle joue un rôle majeur dans le traitement d'un signal, en particulier de parole (calcul du pitch, LPC, … etc.).
Réalisez une fonction qui calcule l'évolution du coefficient d'autocorrélation sur *L* ms toutes les *m* ms sur des fenêtres de taille *N*. Pour le test on prendra *L, m et N* équivalent respectivement à *25ms, 4ms et 25ms* (en échantillons). Cette fonction retournera un tableau à deux dimensions : pour chaque instant *m* on a l'évolution sur *L* ms.

```
1. def computeautocorrelation(signal, m, N, L)
2. def coeffautocorrelation(sig,L)
3. def windowcorrelation(sig1, sig2)
```

Dans la 3. on a le calcul effectif :

$$R_n(l) = \frac{1}{N}\sum_{k=0}^{N-1} s(n+k).s(n+k+l)$$

avec *s(n)* le signal de parole à l'instant n, *l* le pas de décalage (compris entre [0,L]).

Il existe une formule pour estimer le coefficient d'autocorrélation[1] si l'on ne peut calculer le coefficient que sur les N coefficients :

$$R_n(l) = \frac{1}{N-l}\sum_{k=0}^{N-l-1} s(n+k).s(n+k+l)$$

Mais nous ne l'utiliserons pas.

Dans 2.(qui appelle 3.), le coefficient d'autocorrélation sera :

$$CR_n(l) = \frac{R_n(l)}{R_n(0)}$$

Avec $R_n(l)$ la fonction d'autocorrélation à l'instant *n* et pour un pas de *l*.
La valeur obtenue est comprise entre -1 et 1. 1 est obtenu à l'instant *n* (*l=0*) et le pic maximum est normalement obtenu lorsque l'on arrive à l'instant de la période.

Comme il est difficile d'afficher toutes les évolutions du coefficient d'autocorrélation pour chaque instant du signal, prenez pour le test deux instants précis pour en faire une sortie écran : un instant dans une partie voisée du signal puis dans une partie non voisée. (Regardez avec Winsnoori le fichier wav pour trouver ces instants et les entrer dans votre programme)

Réalisez une fonction qui recherche l'instant du (premier) pic maximum pour une évolution des coefficients d'autocorrélation (c'est-à-dire à un instant donné *n* : une ligne de la matrice des coefficients d'autocorrélation).

[1] Tirée du livre : « Traitement de la parole » R. Boite, H. Bourlard, T. Dutoit, J. Hancq et H. Leich, Presses polytechniques et universitaires romandes, 2000, p9

Ebauche de calcul de F0

Le calcul va s'appuyer sur les fonctions précédentes.

1. La F0 est l'inverse de la période (= l'instant (en seconde) du pic maximum dans les coefficients d'autocorrélation).
2. Mais la F0 ne doit être cherchée ou trouvée que sur des parties voisées : utilisez la fonction d'énergie et du taux de passage par zéro pour prendre une décision (avec des seuils par exemple).

Ne cherchez pas à obtenir un résultat parfait, généralement les algorithmes de calcul de F0 s'accompagnent ensuite de fonction de lissage des résultats (pour éviter les données aberrantes). Vous pouvez néanmoins comparer votre résultat avec le résultat obtenu avec Winsnoori.
Pour aller plus loin : il est possible de filtrer le signal (filtre passe-bas) au préalable pour ne garder que les fréquences inférieures à 1000Hz (pour éviter de capturer les variations des harmoniques qui pourraient introduire des doublements ou divisions par 2) ;