SITUATION D'APPRENTISSAGE ET D'ÉVALUATION 22

2022

PERRE MATTHIEU
FARETIE BENJAMIN

BUT RÉSEAUX & TÉLÉCOMMUNICATION

Sommaire

Séance 1 : Jupyter Notebook :

1. Travail Préliminaire :

Nous devions réaliser des scripts permettant de :

- Lire un fichier audio .flac
- D'afficher ses principales caractéristiques (Fe, Nombre de Bits, durée totale, nombre de piste)
- D'afficher la représentation temporelle du signal ainsi que son spectre d'amplitude

Cette fonction nécessite plusieurs librairies, soundfile, _soundfile, IPythonDisplay, os et le fichier python Util_TP206.

```
def LaTotale(Son):

    x, sr = read(Son)                       #Lecture brut du fichier
    piste = x.shape[1]                      #Nombre de piste
    d = len(x) / sr                         #Durée totale du fichier audio

    print(info(Son)) # Informations global sur le son
    print(info(Son).format,"\n") # Format du fichier utiliser ici .flac
    print("Le nombre de bits par échantillon est de type ", info(Son).subtype,) # type de son
    print("La fréquence d'échantillonage est de", sr,"Hz" ) # On affiche la Fréquence d'échantillonage
    print("Le nombre de piste est de :", x.shape[1],) #On rleve le nombre de piste
    print("Le nombre d'echantillon est de",len(x),) #Affichage du nombre d'echantillon de l'enregistrement
    print("Le fichier a donc une durée de :",d,"secondes") #Calcul de la durée et affichage en seconde

    T = linspace(0, d, len(x))              #Création d'un vecteur de la durée du signal
    figure(figsize = (17,8))                #Taille des graphique
    subplots_adjust(hspace = 0.38)          #Gère l'espacement entre les grahps
    i = 0                                   #Une variable qui permet de faire les action sur tout les pistes
    g = 1                                   #Une variable qui permet d'afficher tout les graphs
    for i in range (0, piste):              #Boucle pour réaliser des action sur chacune des pistes

        P = x[:,i]                          #Lecture de la  piste i, pour isoler les pistes
        Titre = "Signal d'origine, piste N° " + str(i) # Création du titre des graphiques
        subplot(piste,2,g)                  #Fonction utile au bon déroulement de l'affichage des graphs
        g= g + 1                            #ajout de 1 a cette variable pour le bon déroulement du subplot
        plot(T, P, 'b.--')                  #Graphique 1(Representation Temporelle)
        grid()                              #Affiche la grille des graphs
        title(Titre)                        #Le titre
        xlabel('Temps(secondes)')
        ylabel('Amplitude')

        subplot(piste,2, g)
        g = g + 1
        plotSpectreAmplitude(P,sr)          #Graphique 2(Spectre d'Amplitude)
        xlim([-2500, 2500])
```

Cette fonction nous permet de réaliser toutes les exigences de la phase préliminaire,

Nous lisons le fichier et stockons les valeurs dans un tableau et la Fréquence d'échantillonnage dans une variable. On détermine le nombre de piste en regardant la 2^{e} colonne du tableau x.

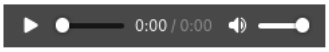
Puis il nous reste plus qu'à afficher les informations relevé par la fonction info ou nos calcul

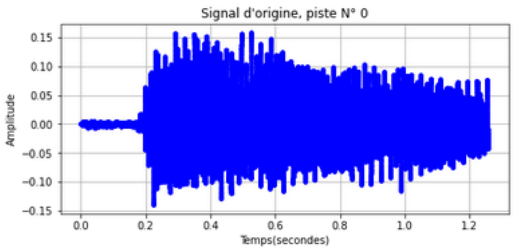
Pour l'affiche, nous avons utilisé une boucle prenant en paramètre le nombre de piste pour pouvoir afficher à la suite les figures correspondantes à ce nombre ici avec 2 pistes nous avons un affiche de la forme :

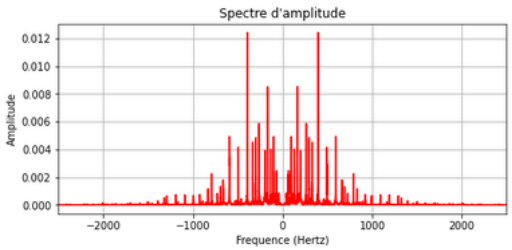
Entrée [5]:
```
LaTotale(Son)
ipd.Audio(Son)           #Création d'un lecteur audio
```

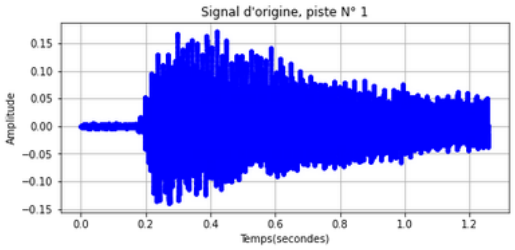
```
NY_HDsample_mini.flac
samplerate: 96000 Hz
channels: 2
duration: 1.259 s
format: FLAC (Free Lossless Audio Codec) [FLAC]
subtype: Signed 24 bit PCM [PCM_24]
FLAC

Le nombre de bits par échantillon est de type  PCM_24
La fréquence d'échantillonage est de 96000 Hz
Le nombre de piste est de : 2
Le nombre d'echantillon est de 120843
Le fichier a donc une durée de : 1.25878125 secondes
```

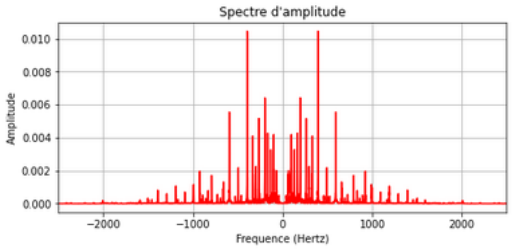
Out[5]:











le script de la phase préliminaire étant fini nous pouvons nous attaquer la suite

2. Travail encadré

Nous devions maintenant réaliser des scripts pouvant :

- Extraire X secondes d'un fichier audio
- Echantillonner cet extrait à une fréquence inférieure
  Quantifier les échantillons à un nombre inférieur de bits

Extraction :

```
def extract(Son, debut, fin): # Fonction poure extraire une durée determinée du fichier audio

    Nom_extrait = input("Nom du fichier qui sera crée (Extrait): \n" )
    v,fs = read(Son)                        #Lire le Son, return les signal des piste et la freq échant
    Se = v[debut*fs:fin*fs]                 #Permet d'extraire la partie du Son de 'debut' sec à 'fin' sec

    piste = Se.shape[1]                     #Nombre de piste
    d = len(Se) / fs                        #Durée totale du fichier audio

    T = linspace(0, d, len(Se))             #Création d'un vecteur de la durée du signal
    figure(figsize = (17,8))                #Taille des graphique
    subplots_adjust(hspace = 0.38)          #Gère l'espacement entre les grahps
    i = 0                                   #Une variable qui permet de faire les action sur tout les pistes
    g = 1                                   #Une variable qui permet d'afficher tout les graphs
    for i in range (0, piste):              #Boucle pour réaliser des action sur chacune des pistes

        Titre = "Signal de l'extrait, piste N° " + str(i)
        P = Se[:,i]                         #Lecture de la  piste i, pour isoler les pistes
        subplot(piste,2,g)
        g = g + 1
        title(Titre)
        plot(T, P, 'b.--')                  #Graphique 1(Representation Temporelle)
        grid()
        xlabel('Temps(secondes)')
        ylabel('Amplitude')

        subplot(piste,2, g)
        g = g + 1
        plotSpectreAmplitude(P,fs)          #Graphique 2(Spectre d'Amplitude)
        xlim([-2500, 2500])

    write(Nom_extrait+'.flac', Se, fs)                      #Créer un Son
    print('\nLe fichier extrait se nomme :',Nom_extrait+'.flac')
    NvSon = Nom_extrait+'.flac'                             # on stock le nom du nouveau son dans une variable
    print(info(NvSon))                                      # information du nouveau son

    return NvSon                                            # retourne le nouveau son
```

Nous avons appelé ce premier script « extract » il prends en paramètre, le fichier audio, le début de l'extrait et sa fin.

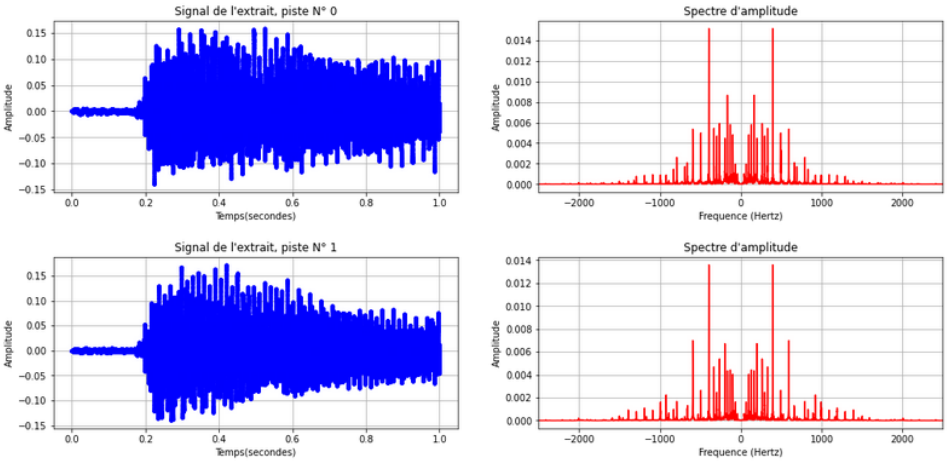
Nous devons donc lire le fichier puis extraire le nombre d'échantillon à utiliser pour obtenir l'extrait de durée X.

Après avoir fait cela nous pouvons afficher et lire le nouveau fichier crée

Entrée [7]:
```
NvSon = extract(Son, 0, 1)
print(NvSon)
ipd.Audio(NvSon)            #Création d'un lecteur audio
```

```
Nom du fichier qui sera crée (Extrait):
TEST

Le fichier extrait se nomme : TEST.flac
TEST.flac
samplerate: 96000 Hz
channels: 2
duration: 96000 samples
format: FLAC (Free Lossless Audio Codec) [FLAC]
subtype: Signed 16 bit PCM [PCM_16]
TEST.flac
```

Out[7]:



Echantillonnage :

Cette fonction nous permet de réduire le nombre d'échantillon utilisé dans la vidéo. Elle prend en paramètre le son et le nombre de fois que l'on souhaite diviser le nombre d'échantillon.

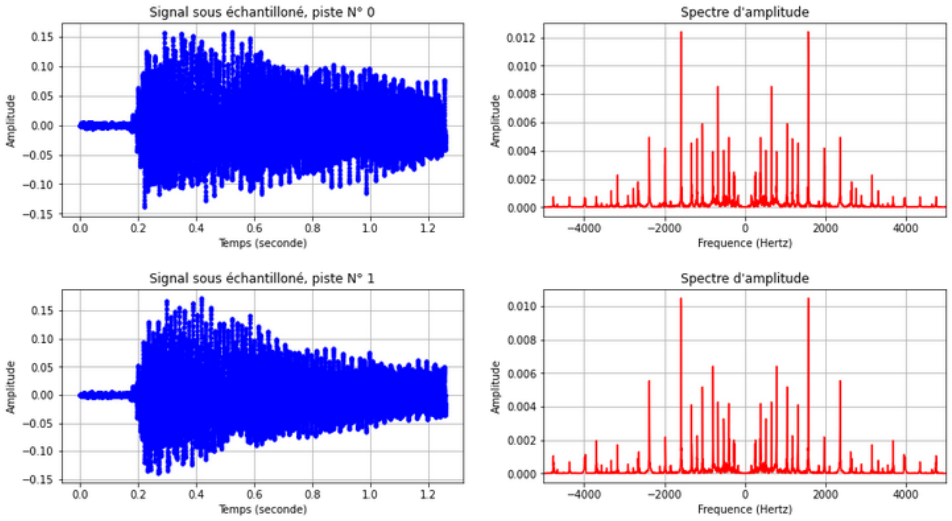
```
def EchantMoins(Son,Fmoins): #Fonction pour echantilloner

    Nom_extrait = input("Nom du fichier qui sera crée (Sous-échantilllonage) : \n" )

    infos = info(Son)                           #Informations sur le Son
    v, fs = read(Son)                           #Lire le Son, return les signal des piste et la freq échant
    Nbmoins = int(ceil(len(v)/Fmoins))          #Calcule du nombre d'échantillons à utilisés (arrondie au dessus)
    T = linspace(0,infos.duration,Nbmoins)      #Création d'un vecteur de la durée du signal
    piste = v.shape[1]                          #Nombre de piste
    FS = int(Nbmoins / infos.duration)          #Nouvelle fréquence d'échantillonage

    s = np.array([[0, 0]])                      #Création du nouveau signal, ne faites pas attention à
                                                #l'initialisation je n'ai pas trouver d'autre méthode qui marche
                                                #que d'initialiser en mettant une valeur dans l'np.array

    for p in range(0,len(v),Fmoins):            #Boucle permettant de récupérer un certains nbr d'échantillons
                                                #du signal de base
        s = np.append(s, [[v[p][0], v[p][1]]], 0)  #Ajout des éléments pour créer le signal sous échantillonné
    s = s[1:]                                   #Elimination du première élément à cause de l'initialisation


    figure(figsize = (15,8))                    #Taille des graphiques
    subplots_adjust(hspace = 0.38)              #Gère l'espacement entre les grahps
    i = 0                                       #Une variable qui permet de faire les action sur tout les pistes
    g = 1                                       #Une variable qui permet d'afficher tout les graphs

    print('Echantillons utilisés : ', Nbmoins)  #Affiche le nombre d'échantillons utilisés

    for i in range (0, piste):                  #Boucle pour réaliser des action sur chacune des pistes

        P = s[:,i]                              #Lecture de la  piste i, pour isoler les pistes
        Titre = "Signal sous échantilloné, piste N° " + str(i)
        subplot(piste,2,g)
        g = g + 1
        plot(T, P, 'b.--')                      #Graphique 1(Representation Temporelle)
        title(Titre)
        grid()
        xlabel('Temps (seconde)')
        ylabel('Amplitude')

        subplot(piste,2, g)
        g = g + 1
        plotSpectreAmplitude(P,fs)              #Graphique 2(Spectre d'Amplitude)
        xlim([-5000, 5000])

    write(Nom_extrait+'.flac', s, FS)
    print('Le fichier sous echantilloné se nomme :',Nom_extrait+'.flac')
    NvSon = Nom_extrait + '.flac'
    print(info(NvSon))
    return NvSon
```

Et elle produit l'affichage suivant

```
Entrée [12]: Sous_Ech = EchantMoins(Son,4) #A essayer avec une valeur plutot grande pour éviter de faire durée l'éxécution
             print(Sous_Ech)
             ipd.Audio(Sous_Ech)
```

```
Nom du fichier qui sera crée (Sous-échantilllonage) :
TEST
Echantillons utilisés :  30211
Le fichier sous echantilloné se nomme : TEST.flac
TEST.flac
samplerate: 24000 Hz
channels: 2
duration: 1.259 s
format: FLAC (Free Lossless Audio Codec) [FLAC]
subtype: Signed 16 bit PCM [PCM_16]
TEST.flac
```

Out[12]: 0:00 / 0:00



<u>Quantification :</u>

Dans celle-ci, nous devons réduire le nombre d'échantillon pour cela nous devons donc diminuer le nombre de bits/échantillon. Pour cela nous allons quantifier avec la loi uniforme et non uniforme (crée 2 fichiers que l'on peut ensuite comparé).

```python
def QuantifMoins(Son,Bits): # Fonction pour quantifier le son en diminuant le nombre de bits
    Nom_extrait = input("Nom du fichier qui sera crée (Quantification): " )
    v, fs = read(Son)
    niv = 2**Bits        #Definition du nombre de niveau du quantificateur
    piste = v.shape[1]   #Stockage du nombre de piste
    t = len(v)/fs        #durée de l'entrée "Son"
    T = arange(0,t,1/fs) #Création du vecteur Temps
    A = 87.6             #Valeur fixée de la loi A
    g = 1                #definition de la variable pour incrementer a partir de 1
    Max = 0              #definition de la variable pour eviter tout type de conflit
    Min = 0              #definition de la variable pour eviter tout type de conflit
    SNU = np.array([[0, 0]])
    SU = np.array([[0, 0]])

    for i in range(0, piste):
        Max = max(v[:,i]) #On prends la valeur maximale de la piste i
        Min = min(v[:,i]) #On prends la valeur minimale de la piste i

    ########################SIGNAL NON UNIFORME#############################

    uni1 = A_law(v[:,0],A) #On applique une quantification non-uniforme pour permettre une meilleure conversion
    uni1, pasNU1 = uniform_quantizer(uni1, niv, Min, Max)
    SNU1 = inverse_A_law(uni1,A)

    uni2 = A_law(v[:,1],A) #On applique une quantification non-uniforme pour permettre une meilleure conversion
    uni2, pasNU1 = uniform_quantizer(uni2, niv, Min, Max)
    SNU2 = inverse_A_law(uni2,A)

    for p in range(0,len(v)):
        SNU = np.append(SNU, [[SNU1[p], SNU2[p]]] , 0)
    SNU = SNU[1:]
```

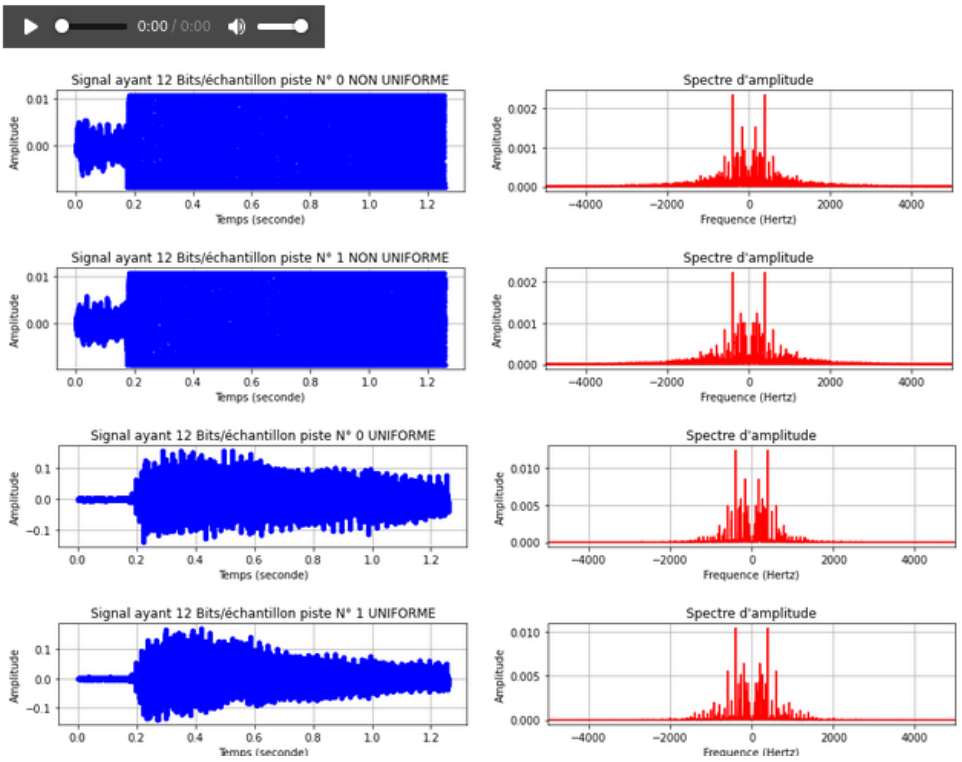
```python
    for i in range(0, piste):

        Titre = "Signal ayant " + str(Bits) +" Bits/échantillon piste N° " + str(i) + "NON UNIFORME" #Titre du graphique
        figure(figsize = (15,8))
        subplot(piste*2,2,g)
        g= g + 1
        plot(T, SNU[:,i], 'b.')
        #plot(T, abs(Sq-v[:,i]), 'g-') #Affichage de l'erreur de quantification (Désactiver car rends la lecture du graphe difficile)
        grid()
        title(Titre)
        xlabel('Temps (seconde)')
        ylabel('Amplitude')

        subplot(piste*2,2, g)
        g = g + 1
        #Graphique 2(Spectre d'Amplitude)
        plotSpectreAmplitude(SNU[:,i],fs)
        xlim([-5000, 5000])

    write(Nom_extrait + '_non_uniform.flac', SNU, fs) #On ecrit le fichier au nom donnée en entrée (affiche une output du fichier créé)
    NvSon1 = Nom_extrait + '_non_uniform.flac' #On crée une variable contenant le nom du fichier

    ########################SIGNAL UNIFORME#############################

    SU1, pasU1 = uniform_quantizer(v[:,0], niv, Min, Max)
    SU2, pasU2 = uniform_quantizer(v[:,1], niv, Min, Max)

    for p in range(0,len(v)):
        SU = np.append(SU, [[SU1[p], SU2[p]]] , 0)
    SU = SU[1:]
```

```
for i in range(0, piste):

    Titre = "Signal ayant " + str(Bits) +" Bits/échantillon piste N° " + str(i) + "UNIFORME" #Titre du graphique
    figure(figsize = (15,8))
    subplot(piste*2,2,g)
    g= g + 1
    plot(T, SU[:,i], 'b.')
    #plot(T, abs(Sq-v[:,i]), 'g-') #Affichage de l'erreur de quantification (Désactiver car rends la lecture du graphe difficile)
    grid()
    title(Titre)
    xlabel('Temps (seconde)')
    ylabel('Amplitude')

    subplot(piste*2,2, g)
    g = g + 1
    #Graphique 2(Spectre d'Amplitude)
    plotSpectreAmplitude(SU[:,i],fs)
    xlim([-5000, 5000])

write(Nom_extrait + '_uniform.flac', SU, fs) #On ecrit le fichier au nom donnée en entrée (affiche une output du fichier créé)
NvSon2 = Nom_extrait + '_uniform.flac' #On crée une variable contenant le nom du fichier

return NvSon1, NvSon2 # On recupere le nom de la variable
```

Cela produit l'affichage suivant :

```
Entrée [31]: S_non_uniform, S_uniform = QuantifMoins(Son,12)
             print('########################SIGNAL NON UNIFORME##############################\n', info(S_non_uniform))
             print('########################SIGNAL UNIFORME##################################\n', info(S_uniform))
             ipd.Audio(S_non_uniform) # A MODIFIER SELON VOTRE BESOIN D'ÉCOUTE

Nom du fichier qui sera crée (Quantification): TEST
########################SIGNAL NON UNIFORME##############################
 TEST_non_uniform.flac
samplerate: 96000 Hz
channels: 2
duration: 1.259 s
format: FLAC (Free Lossless Audio Codec) [FLAC]
subtype: Signed 16 bit PCM [PCM_16]
########################SIGNAL UNIFORME##############################
 TEST_uniform.flac
samplerate: 96000 Hz
channels: 2
duration: 1.259 s
format: FLAC (Free Lossless Audio Codec) [FLAC]
subtype: Signed 16 bit PCM [PCM_16]
```

Out[31]:



Maintenant que tout les scripts pythons sont fonctionnel nous devons effectuer extraire d'un fichier HD un fichier SD.

Pour cela nous devons donc utiliser les scripts précédents :

```
def HDtoSD(Son,debut,fin,Fmoins,Bits):
    print('\n###################### INFORMATION DU SON ######################\n')
    LaTotale(Son)
    Extrait = extract(Son,debut,fin)
    print('\n################### INFORMATION DE L\'EXTRAIT ###################\n')
    print(Extrait)
    Sous_Ech_Extrait = EchantsMoins(Extrait,Fmoins)
    print('\n######### INFORMATION DE L\'EXTRAIT SOUS ÉCHANTILLONNER ########\n')
    print(Sous_Ech_Extrait)
    Extrait_final = QuantifMoins(Sous_Ech_Extrait,Bits)
    print('\n################### NOM DES FICHIER FINALES ####################\n')
    print('le fichier HD passée en SD se nomme :', Extrait_final[0] , ' OU ', Extrait_final[1])
    print('\n########## INFORMATION DU FICHIER FINALE NON UNIFORME ##########\n')
    print(info(Extrait_final[0]))
    print('\n############ INFORMATION DU FICHIER FINALE UNIFORME ############\n')
    print(info(Extrait_final[1]))
    return Extrait_final[0], Extrait_final[1]
```

L'enchainement de toutes les fonctions mise en place nous permettent donc de réaliser la mission finale.

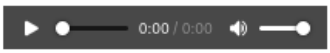
Nous avons donc en sortie, un fichier audio SD provenant d'un fichier HD.

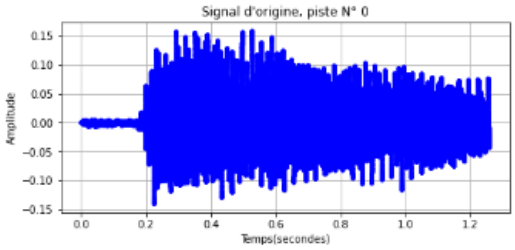
Affichage :

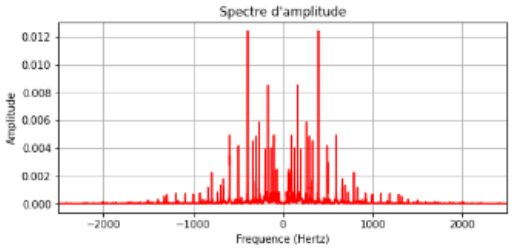
```
###################### INFORMATION DU SON ######################

NY_HDsample_mini.flac
samplerate: 96000 Hz
channels: 2
duration: 1.259 s
format: FLAC (Free Lossless Audio Codec) [FLAC]
subtype: Signed 24 bit PCM [PCM_24]
FLAC

Le nombre de bits par échantillon est de type  PCM_24
La fréquence d'échantillonage est de 96000 Hz
Le nombre de piste est de : 2
Le nombre d'echantillon est de 120843
Le fichier a donc une durée de : 1.25878125 secondes
Nom du fichier qui sera crée (Extrait):
TEST

Le fichier extrait se nomme : TEST.flac
TEST.flac
samplerate: 96000 Hz
channels: 2
duration: 96000 samples
format: FLAC (Free Lossless Audio Codec) [FLAC]
subtype: Signed 16 bit PCM [PCM_16]

################## INFORMATION DE L'EXTRAIT ##################

TEST.flac
Nom du fichier qui sera crée (Sous-échantilllonage) :
TEST
Echantillons utilisés :  48000
Le fichier sous echantilloné se nomme : TEST.flac
TEST.flac
samplerate: 48000 Hz
channels: 2
duration: 48000 samples
format: FLAC (Free Lossless Audio Codec) [FLAC]
subtype: Signed 16 bit PCM [PCM_16]

######### INFORMATION DE L'EXTRAIT SOUS ÉCHANTILLONNER ########

TEST.flac
Nom du fichier qui sera crée (Quantification): TEST

################## NOM DES FICHIER FINALES ###################

le fichier HD passée en SD se nomme : TEST_non_uniform.flac  OU  TEST_uniform.flac

########## INFORMATION DU FICHIER FINALE NON UNIFORME ##########

TEST_non_uniform.flac
samplerate: 48000 Hz
channels: 2
duration: 48000 samples
format: FLAC (Free Lossless Audio Codec) [FLAC]
subtype: Signed 16 bit PCM [PCM_16]
```

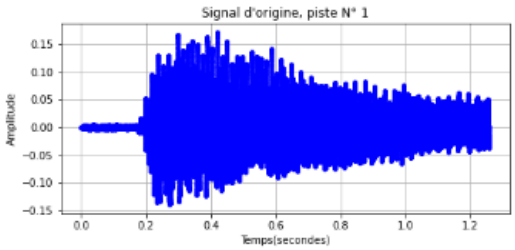
```
############ INFORMATION DU FICHIER FINALE UNIFORME ############

TEST_uniform.flac
samplerate: 48000 Hz
channels: 2
duration: 48000 samples
format: FLAC (Free Lossless Audio Codec) [FLAC]
subtype: Signed 16 bit PCM [PCM_16]
TEST_non_uniform.flac
TEST_uniform.flac
```

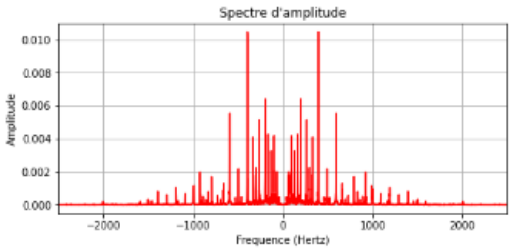
Out[24]:

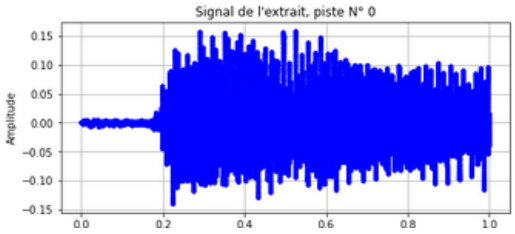


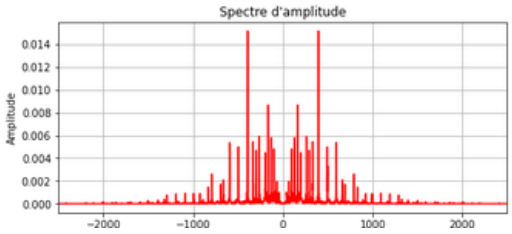

Signal d'origine, piste N° 0
Amplitude
Temps(secondes)

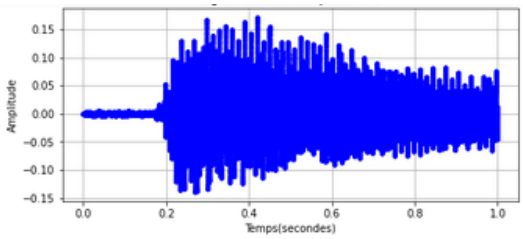

Spectre d'amplitude
Amplitude
Frequence (Hertz)

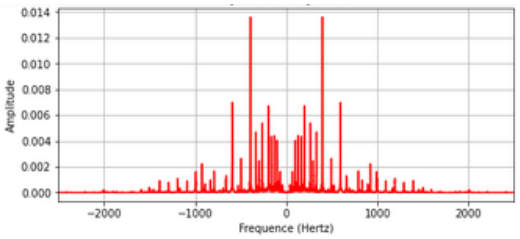

Signal d'origine, piste N° 1
Amplitude
Temps(secondes)

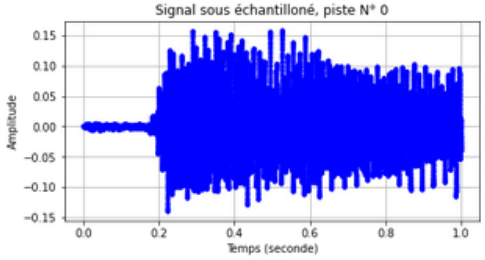

Spectre d'amplitude
Amplitude
Frequence (Hertz)

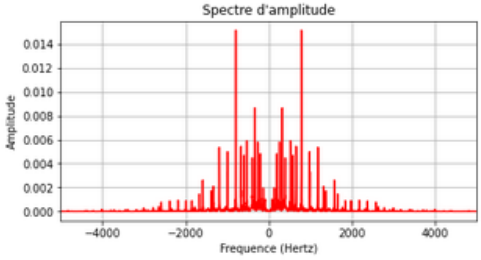

Signal de l'extrait, piste N° 0
Amplitude

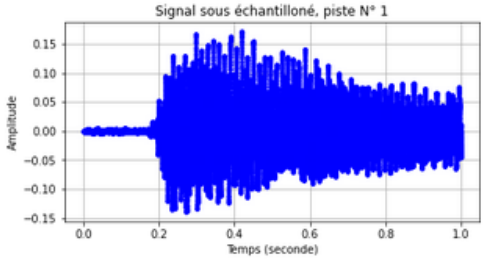

Spectre d'amplitude
Amplitude

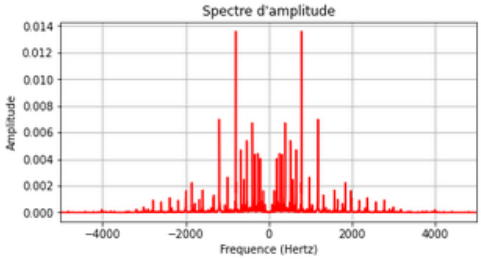

Amplitude
Temps(secondes)

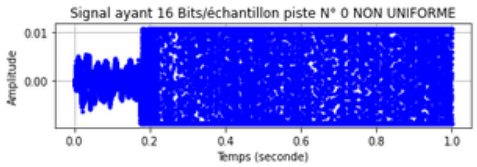

Amplitude
Frequence (Hertz)

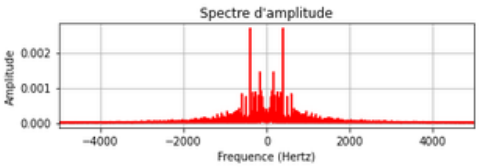

Signal sous échantilloné, piste N° 0
Amplitude
Temps (seconde)

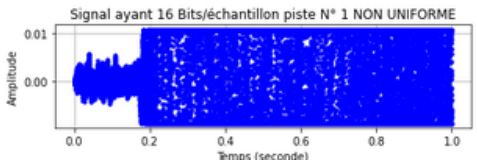

Spectre d'amplitude
Amplitude
Frequence (Hertz)

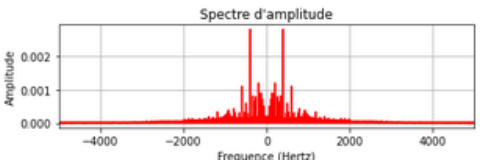

Signal sous échantilloné, piste N° 1
Amplitude
Temps (seconde)

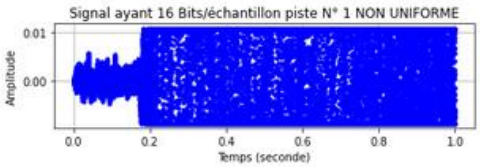

Spectre d'amplitude
Amplitude
Frequence (Hertz)

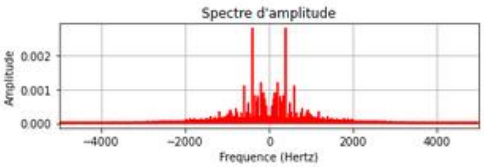

Signal ayant 16 Bits/échantillon piste N° 0 NON UNIFORME
Amplitude
Temps (seconde)

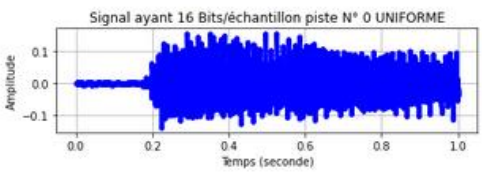

Spectre d'amplitude
Amplitude
Frequence (Hertz)

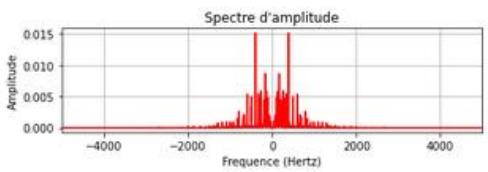

Signal ayant 16 Bits/échantillon piste N° 1 NON UNIFORME
Amplitude
Temps (seconde)

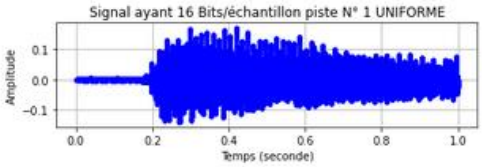

Spectre d'amplitude
Amplitude
Frequence (Hertz)

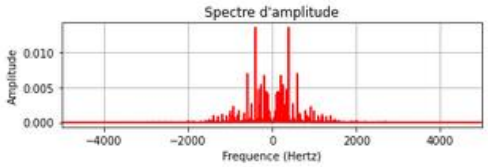

Signal ayant 16 Bits/échantillon piste N° 1 NON UNIFORME
Amplitude
Temps (seconde)


Spectre d'amplitude
Amplitude
Frequence (Hertz)


Signal ayant 16 Bits/échantillon piste N° 0 UNIFORME
Amplitude
Temps (seconde)


Spectre d'amplitude
Amplitude
Frequence (Hertz)


Signal ayant 16 Bits/échantillon piste N° 1 UNIFORME
Amplitude
Temps (seconde)


Spectre d'amplitude
Amplitude
Frequence (Hertz)

c. Conclusion :

Grace à cette SAE, nous avons appris qu'un fichier audio est en HD lorsque sa quantification est de 24bits, un fichier audio SD quant à lui est quantifié à 16 bits ce qui démontre une différence de niveau élevé est donc de qualité en même temps dans la théorie.

Nous devions donc réaliser 3 principales fonctions nous permettant de comparer ces 2 qualités.

Ces 3 fonctions sont donc :

- L'extraction de donnée (Récupérer un échantillon entre 2 valeurs choisis A et B)
- Le sous-échantillonnage (Permet de recréer un échantillonnage de qualité SD car SD = 48000Hz et HD = 96000Hz)
- La sous-quantification (Diminue le nombre de niveau du son et par la même occasion sa précision)

Lorsque l'on quantifie, 2 options s'offre à nous :

- La quantification uniforme
- La quantification non-uniforme

Ces 2 solutions ont chacune leurs avantage mais dans notre cas, selon les spectres que l'on peut apercevoir lorsque l'on fait appel à ces quantifieurs, la quantification uniforme maintient un niveau de qualité plus intéressant que la quantification non-uniforme. On peut donc s'apercevoir que lorsque les valeurs en amplitude sont assez proches il faut prioriser une quantification uniforme et une non-uniforme si le signal possède des amplitudes très éloigné les unes des autres.

On peut également apercevoir que l'extension ".flac" ne permet pas d'identifier le nombre exact de bits utilisé dans un fichier audio. Il n'affiche que 16bits si nous sommes en dessous de cette valeur.

Pour finir, on ne peut pas facilement différencier à l'oreille un fichier de qualité SD et HD et lorsque l'on effectue cette manipulation, la qualité du son diminue quand même comme le démontre les spectres temporels et d'amplitudes.

Séance 2 : Mesure de signaux périodiques à l'oscilloscope numérique

1. Travail Préliminaire

1 – Recherches sur l'oscilloscope numérique

Contrairement aux modèles analogiques, le signal à visualiser est préalablement numérisé par un convertisseur analogique-numérique. La capacité de l'appareil à afficher un signal de fréquence élevée sans distorsion dépend de la qualité de cette interface.

Les oscilloscopes numériques possèdent tous à l'heure actuelle un module de calcul de FFT pour effectuer l'analyse fréquentielle des signaux. Contrairement aux véritables analyseurs FFT, ces oscilloscopes n'ont pas de filtre anti-repliement

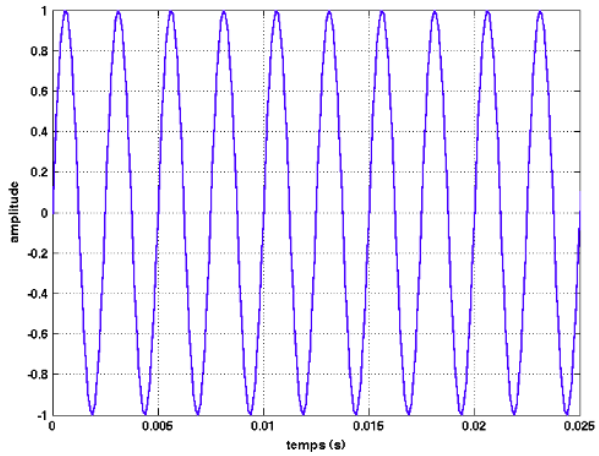
L'appareil est couplé à des mémoires permettant de stocker ces signaux et à un certain nombre d'organes d'analyse et de traitement qui permettent d'obtenir de nombreuses caractéristiques du signal observé :

- Mesure des caractéristiques du signal : valeur de crête, valeur efficace, période, fréquence
- Transformation rapide de Fourier qui permet d'obtenir le spectre du signal
- Filtres perfectionnés qui, appliqués à ce signal numérique, permettent d'accroître la visibilité de détails
- Décodage de signaux numériques : LIN, CAN, USB

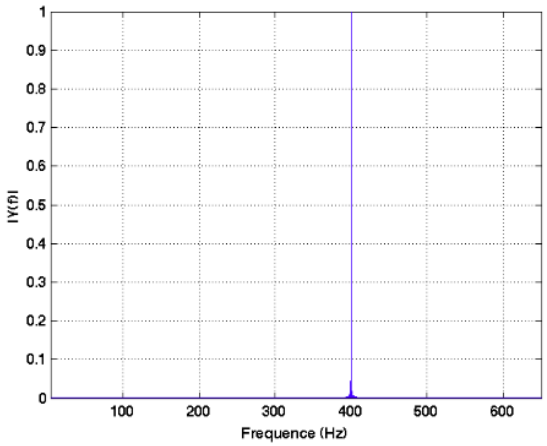
L'affichage du résultat s'effectue de plus en plus souvent sur un écran à cristaux liquides, ce qui rend ces appareils faciles à déplacer et beaucoup moins gourmands en énergie.

2 – Rappels des spectres théorique

Signal Sinusoïdal :

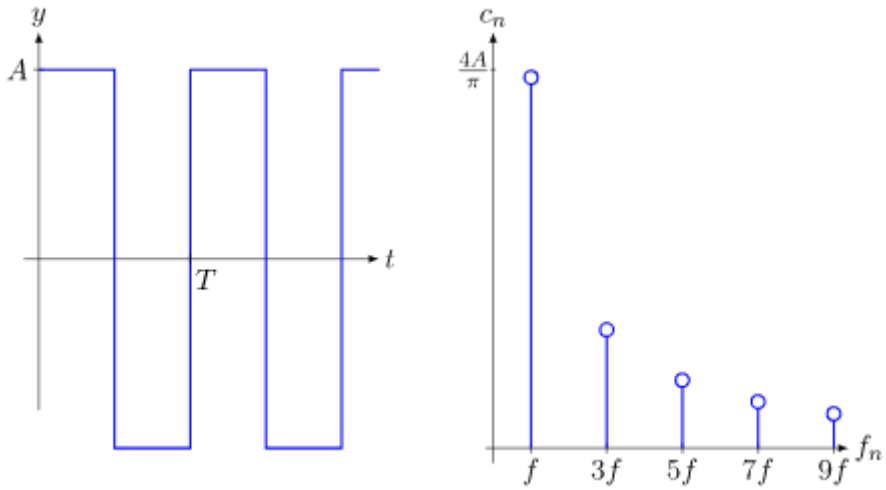


(a)

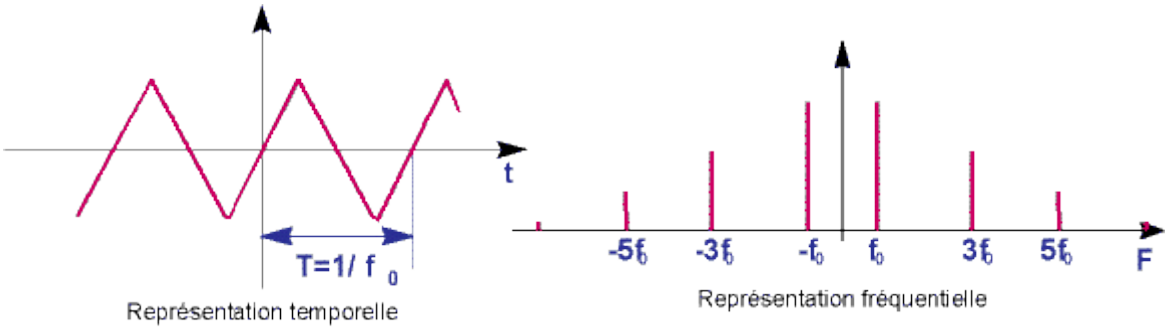


(b)

Signal Carré :



Signal Triangulaire :



3- Documentation sur le phénomène d'aliasing

Le repliement de spectre (aliasing en anglais) est un phénomène qui introduit, dans un signal, des fréquences qui ne devraient pas s'y trouver, lorsque la fréquence porteuse ou la fréquence d'échantillonnage sont inférieures à deux fois la fréquence maximale contenue dans le signal.

Pour éviter cela, il suffit de respecter le théorème de Nyquist-Shannon disant que Fe > 2Fmax.