UE MU4RBI08-audio
Compte-rendu de TP 4 :
Compression d'un Signal Audio

KADI - koussaila
DA ROCHA MARTINS - Mickael
Mention : Automatique Roborique (ISI)

Introduction

La compression de données est un domaine clef dans l'ère du numérique : pouvoir stocker,partager, et lire des données avec le minimum d'espace de stockage, le minimum de mémoire vive, et dans un minimum de temps. La compression audio avec perte (MPEG-1 layer I-II-III) repose sur une idée simple : on ne code pas ce qui ne s'entend pas. Ainsi, l'audio compressé est sensé être identique perceptivement à l'audio original.

4.1 Codage audio perceptif

1. Calculer la TFCT X du signal audio x (.wav de votre choix, avec une taille de fenêtre et un pas d'avancement judicieusement choisis en fonction de la fréquence d'échantillonage du signal audio, et tracer le spectrogramme correspondant en fonction du temps en secondes et des fréquences en Hertz :

   - Le choix : on choisit une taillede trame de 1024 (2**10) avec une taille temporelle <24ms on constate qu'on a une densité d'enrgie fréquentielle dans la zone des fréquence [0,1000 HZ] très élevée

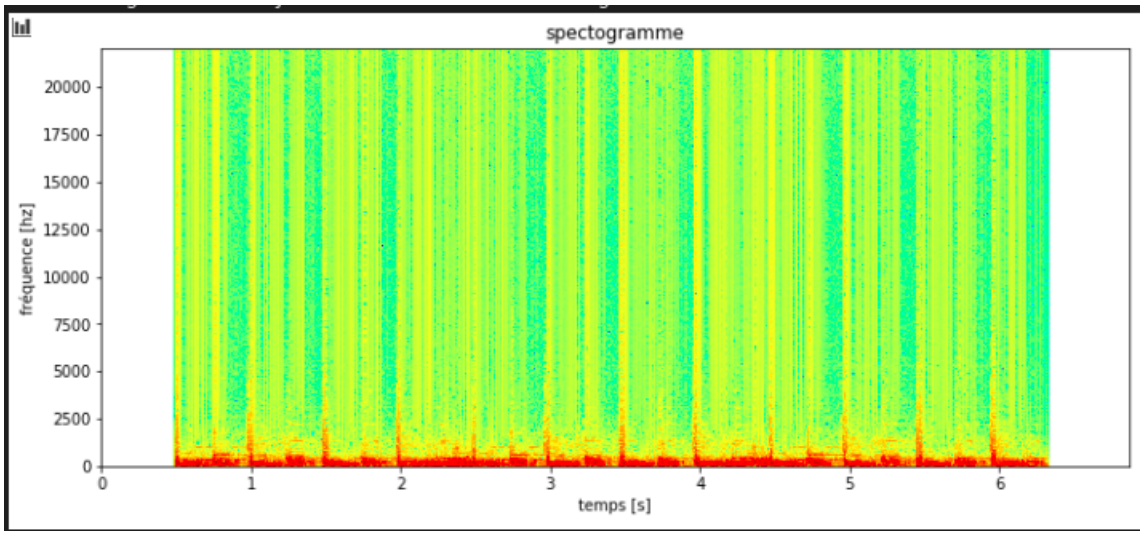

FIGURE 1 – spectogramme du signal

2. Pour chaque trame temporelle n, calculer le facteur de gain A n permettant la normalisation unitaire du spectre d'amplitude :

   ---

   le facteur de gain est la valeur maximale en valeur absolue pour chaque trame du signal X.

3. Calculer, à partir d'un débit souhaité de 392 kbits/s, le nombre de bits alloué à chaque trame du module de la TFCT, noté |X norm |, en fonction des paramètres d'analyses utilisés et de la fréquence d'échantillonage. En déduire le nombre de bits alloués en moyenne à chaque point fréquentiel :

   ---

   debit = 392000 (392kb/S) nombre de bits par trame = debit/(fe/Nwin) bit par trame=arrondi du (nombre de bits par trame)
   Avec :
   fe= Fréquence d'échantillonnage
   Nwin : largeur de la trame
   on obtient :
   le nombre bit par chaque trame : 9102.0
   avec une moynne de bit par point fréquentiel est de 8 bits

4. Calculer la répartition des bits en fréquence pour chaque trame, en utilisant l'algorithme d'allocation de bits perceptif qui minimise la différence entre le rapport signal à masque SM R calculé à partir d'un modèle psycho-acoustique et le rapport SN R calculé en fonction du nombre de bits utilisés pour la quantification. Pour rappel :
NMR(dB) = SMR(dB) SNR(dB)

On considère que le modèle acoustique est une simple droite (constate pour tout les points fréquentiels, on considère pas le car réel où le seuil dépond des fréquences) Adb= -96 dB. chaque point fréquentiel avec une intensité spéctrale > -96 dB , on prend ce point en considération, et il va être désigné dans la zone audible (au dessus du mask) et donc on essaye de le coder sur le plus de bit possible pour avoir une bonne résolution, et pour tout les autres points fréquentiels au dessous de -96dB ils vont pas être pris en compte car tout simplement ils vont pas être audibles.

La matrice Q de même taille que xmat (la TFCT) du signal audio, elle contient le nombre de bits perceptifs alloués pour chaque point fréquentiel de chaque trame.

Remarque :

On constate que les points fréquentiels en intensité en dB sont preque pour les trames représentées ci-dessous suppérieurs à -96dB (seuil acoustique) donc ces fréquences la ne vont pas être supprimées.

la trame 5 représente un silence
la droite en rouge est le seuil acoustique

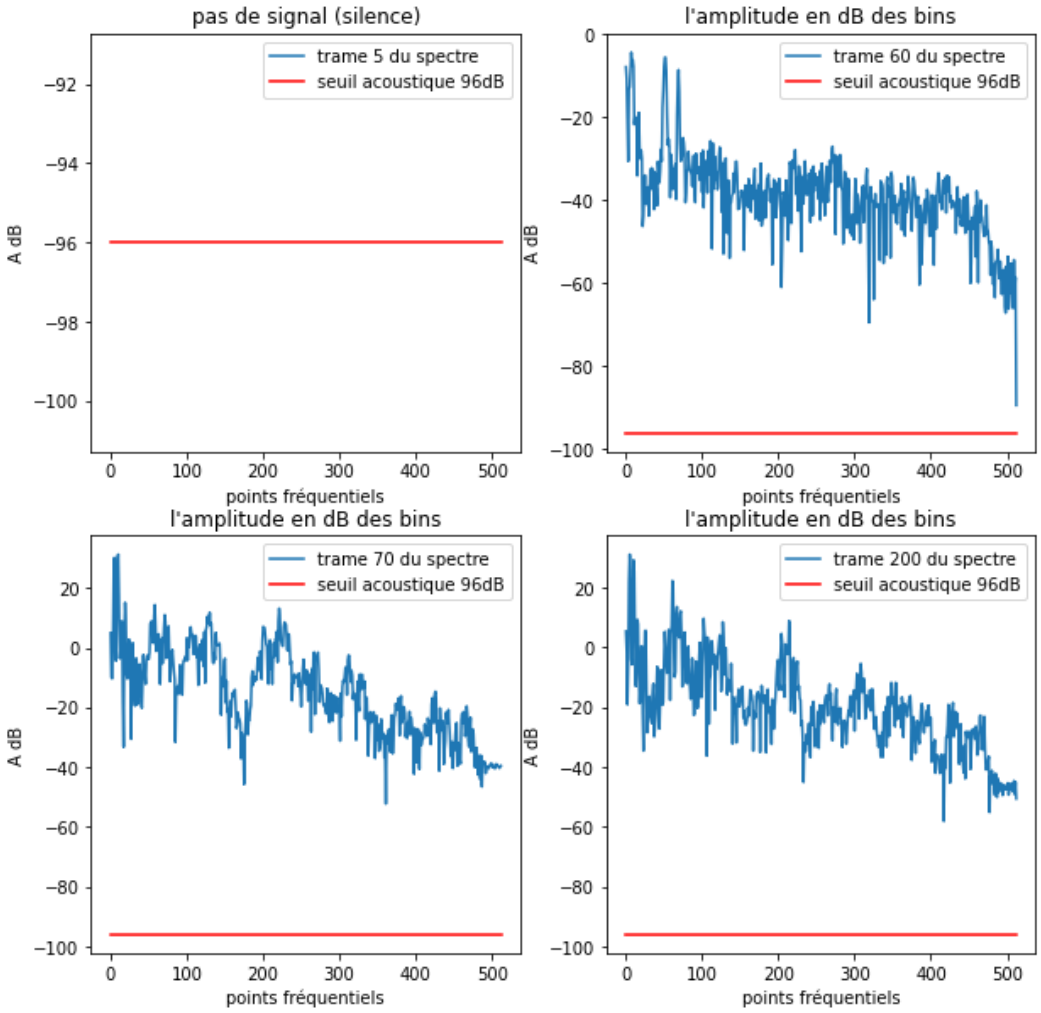

FIGURE 2 – Amplitude en dB des points fréquentiels de quelques trames

5. Expliciter les conditions d'arrêts à utiliser pour l'arrêt de l'allocation.

Les condistions d'arrêt :
Le nombre de bits alloués à chaque trame : 9102 bits,chaque trame doit utiliser ce nombre de bit (au maximum) donc il faut utiliser sur les points fréquentiels qui respectent la condition du seuil acoustique.
l'algorithme s'arrête une fois qu'on a fini de répartir tout les bits, où qu'on finit de parcourir tout les points fréquentiels de la trame avant de passer à la trame suivante.

6. En déduire le spectrogramme quantifié |X qnorm | à partir du nombre de bits donné par les éléments de la matrice Q. On utilisera pour cela la fonction Fuquant donnée dans le matériel du TP4. Tracer le spectrogramme et comparer avec le spectrogramme original en fonction du nombre de bits alloués. Quelles fréquences disparaissent majoritairement ? Expliquer.

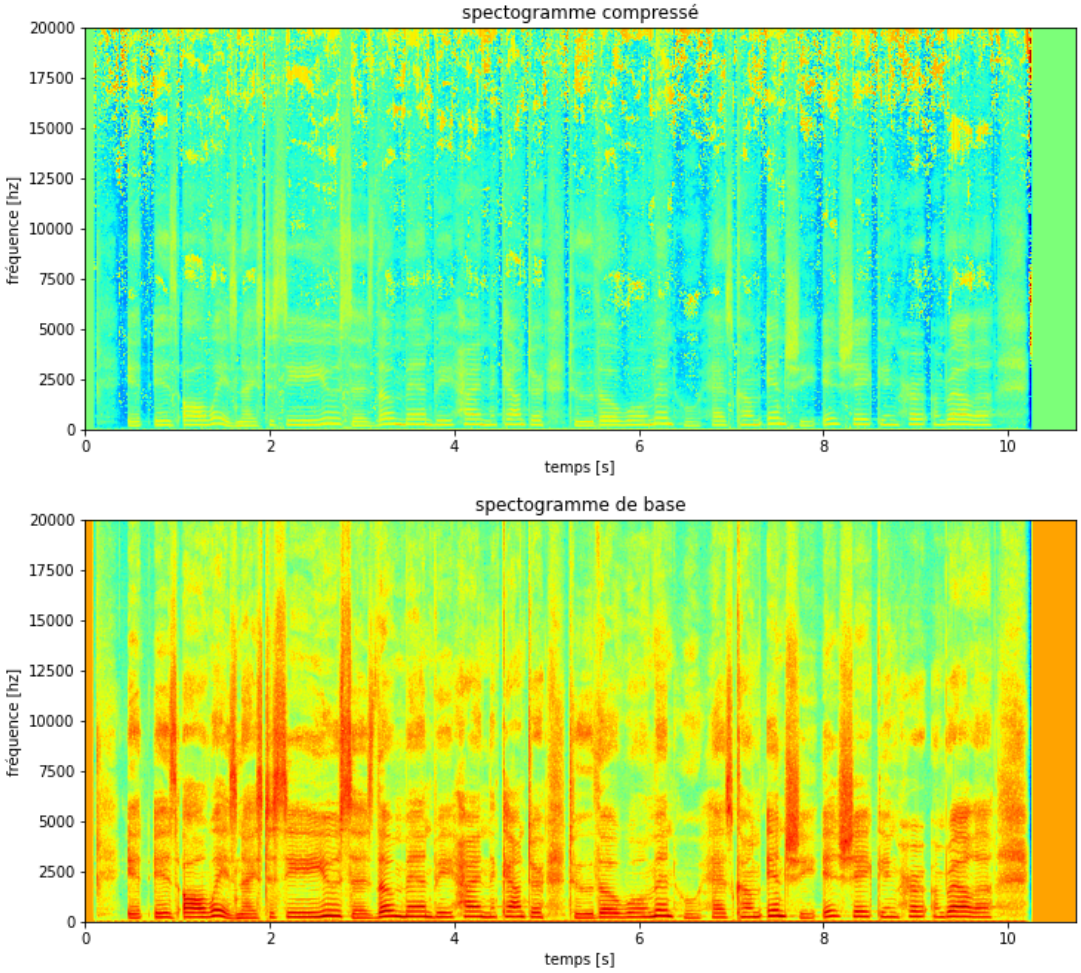

FIGURE 3 – comparaison de spectogramme compressé et non compressé

On fait de la compression avec perte, donc pour cela on constate qu'on a une perte en intesité sur les points fréquentiels d'amplitudes les plus faibles, car ils vont êtres quantifier sur un nombre de bit inférieur au points fréquentiels avec des intensité élevées, on générale on pert de l'intesité spectrale dans les fréquence qui sont au-dessous du seuil acoustique, car dans tout les cas ces points fréquentiels ne vont pas être audibles.

7. A partir de la matrice de codage |X q norm | et du nombre de bits alloués donné dans la matrice Q, restituer la valeur de chaque point fréquentiel. On utilisera la fonction de déquantification Fuquantinv donnée dans le matériel du TP4. On pensera également à

remettre à l'échelle les spectres d'amplitude des trames avec les facteurs de gains An.On stockera le résultat dans une matrice de codage déquantifiée et dénormalisée notée |Xuq|

8. 11. Conclure sur l'intérêt de l'allocation perceptive de bits par rapport à une allocation uniforme. Que faudrait-il faire pour rendre le codage plus efficace ?

- Allocation perceptive : ça permert d'allouer d'avantage de bits pour les points fréquentiels qui ont une intensité élevé que le seuil de la perception de l'oreil, et d'allouer moins de bits au points fréquentiels et voire même supprimer des points au dessous du seuil du mask.
- Allocation uniforme, on alloue le même nombre de bits pour tout les points fréquentiels, ce n'est pas une bonne compression vu que l'humain n'entend pas quelques fréquences, et donc pour permettre d'avoir accès à la musique (SPOTIFY) exemple dans des endroits lointain, exemplemontagne où le débit ne soit pas trop fort, donc on aura pas accès à cette musique en ligne.

Partie 2 Bonus

1. 4. écouter les sons sur le site : sur certain sons on a réussi à distinguer la meilleure qualité pour certaines musiques, il n'est pas facile reconnaitre la meilleure qualité pour cetrains sons

A Annexes

A.1 Code

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io.wavfile as wav
import scipy.signal as signal
from numpy.fft import fft, ifft, fftshift
from IPython.display import display, Audio
```

```
#Question 3

def tfct(x, Nwin, Nhop, Nfft, fe):

    L=int((len(x)-Nwin)/Nhop) +1 ;
    M=Nfft
    xmat=np.zeros((M,L))
    sigMat = []
    colonne=0
    i=0
    hamming=np.hamming(Nwin)
    while (int(i+Nwin)<=int(len(x))):
        for j in range(int(i),int(i+Nwin)):
            sigMat.append(x[j])
        TFD=np.fft.fft(sigMat*hamming)
        xmat[:,colonne]=abs(TFD)
        colonne+=1
        i=i+Nhop
        sigMat=[]
    f=np.linspace(0,fe,Nfft)
```

```
    t=np.linspace(0,len(x)/fe,Nfft)

    print(M,L)
    return xmat[0:int(Nfft/2+1),:],t,f

def xmat2dB(xmat):
    xmat_dB=np.zeros(xmat.shape)
    for i in range(xmat.shape[1]):
      for j in range(xmat.shape[0]):
        if xmat[j,i]==0:
          xmat_dB[j,i]= -1
        if xmat[j,i]>0:
          xmat_dB[j,i]=20*np.log10(xmat[j,i])

    return xmat_dB
```

QST1:

```
Nwin = 1024
Nfft = Nwin
Nhop = Nwin//2 # pas d'avancement de chaque trame.

fe,xvect=wav.read("suzanneVega_tomsDiner.wav")
print("la fr quence d'  chantillonnage   du signal: ",fe)

display(Audio(xvect,rate=fe))

#----trac  du spectogramme:
# normaliser le signal xvect:
x=xvect/max(abs(xvect))

xmat,temps,freq=tfct(x,Nwin,Nhop,Nfft,fe)
xmat_dB=xmat2dB(xmat)

plt.figure(figsize=(12,5))
plt.title("spectogramme")
plt.imshow(xmat_dB,aspect ='auto', origin='lower',cmap='jet',extent=(0,temps[-1]
plt.xlabel('temps [s]')
plt.ylabel("fr quence [hz]")
plt.show()
```

QST2:

```
A= np.zeros(xmat.shape[1]) # le facteur de gain
for i in range (xmat.shape[1]):
    A[i]=max(abs(xmat[:,i]))


# Normalisation unitaire du spectre :
xmat_Norm=np.zeros(xmat.shape)
for i in range(xmat.shape[1]):
  for j in range(xmat.shape[0]):
    xmat_Norm[j,i]=xmat[j,i]/A[i]
```

QST 3:

```
debit = 392000
nb_par_trame= debit/(fe/Nwin)
bit_par_trame=np.floor(nb_par_trame)
print("le nombre bit par chaque trame : ", bit_par_trame)

nb_par_point_freq=bit_par_trame/Nwin
bit_par_point=np.floor(nb_par_point_freq)
print("le nombre de bit par point frequenciel : ", bit_par_point)
```

QST 4:

```
# convertir xmat en dB:
xmat_dB=xmat2dB(xmat_Norm)

for i in range(xmat.shape[0]):          # parcourir les trames
    for j in range(xmat.shape[1]):      # parcourir les points fr q
        if xmat[i,j]==0:
            xmat_dB[i,j]=1               # ces points ne vont pas  tre
        elif xmat[i,j]>0:
            xmat_dB[i,j]=20*np.log(xmat[i,j])   #en dB

# initialisatin:
nbits_trame=0
xmat_compress=np.zeros(xmat.shape)

# SMR :
SMR=np.zeros(xmat.shape)
for i in range(xmat.shape[0]):          # parcourir les trames
    for j in range(xmat.shape[1]):      # parcourir les points fr q
        if xmat[i,j]==0:
            SMR[i,j]=1                 # ces points ne vont pas  tre  cod  N=0
        elif xmat[i,j]>0:
            SMR[i,j]=20*np.log(abs(xmat[i,j]))-96  #en dB


R=bit_par_trame
Q=np.zeros(xmat.shape)

xmat_compress=xmat_dB
```

```
# algorithme d'allocation de bits :
```

```
for i in range (xmat.shape[1]):          # parcourir les trames
    for j in range(xmat.shape[0]):       # parcourir les points fr qentiels
        N=0
        SNR=0
        NMR=1
        if xmat[j,i]==0:
            Q[j,i]=0
            continue
        else:
```

```
            while (R>0 or NMR>0 ):
                NMR=xmat_dB_q[j,i]-96-SNR                    #NMR
                kmax=max(abs(xmat_compress[:,i]))                    #K arg
                index=np.where(abs(xmat_compress[:,i])==kmax) #
                xmat_compress[index,i]= kmax-6
                NMR=SMR[j,i]-SNR
                Q[index,i]+=1
                SNR+=6
                R-=1
                N+=1


acoustique=np.zeros(xmat.shape[0])
acoustique=acoustique-96

plt.figure(figsize=(10,10))
plt.subplot(2,2,1)
plt.plot(20*np.log10(xmat[:,5]),label='trame 5 du spectre')
plt.plot(acoustique,c='r',label="seuil acoustique -96dB")
plt.xlabel("points fr quentiels")
plt.title("pas de signal (silence)")
plt.ylabel("A dB")
plt.legend()

plt.subplot(2,2,2)
plt.plot(20*np.log10(xmat[:,60]),label='trame 60 du spectre')
plt.plot(acoustique,c='r',label="seuil acoustique -96dB")
plt.xlabel("points fr quentiels")
plt.title("l'amplitude en dB des bins")
plt.ylabel("A dB")
plt.legend()

plt.subplot(2,2,3)
plt.plot(20*np.log10(xmat[:,70]),label='trame 70 du spectre')
plt.plot(acoustique,c='r',label="seuil acoustique -96dB")
plt.xlabel("points fr quentiels")
plt.title("l'amplitude en dB des bins")
plt.ylabel("A dB")
plt.legend()

plt.subplot(2,2,4)
plt.plot(20*np.log10(xmat[:,200]),label='trame 200 du spectre')
plt.plot(acoustique,c='r',label="seuil acoustique -96dB")
plt.xlabel("points fr quentiels")
plt.title("l'amplitude en dB des bins")
plt.ylabel("A dB")
plt.legend()

plt.show()
```

QST 5: juste des commentaires

QST 6:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Apr  2 11:10:49 2021

@author: boutin
"""
import numpy as np

def Fuquant(x, N):
# quantificateur uniforme sur n bits
# saturation    +-1
# entr e: vecteur de reels
# sortie: vecteur d'entiers positifs

    if N>0:
        pown = 2**(N-1)
        xq = pown*(x<0) + np.floor( pown *np.min([np.abs(x),1-1e-10]) )
    else:
        # xq = []
        xq = 0
    return xq

plt.figure(figsize=(12,5))
plt.title("spectogramme compress ")
plt.imshow(xmat_compress,aspect ='auto', origin='lower',cmap='jet',extent=(0,temp
plt.xlabel('temps [s]')
plt.ylabel("fr quence [hz]")
plt.ylim([0,20000])
plt.show()

plt.figure(figsize=(12,5))
plt.title("spectogramme de base")
plt.imshow(xmat_dB,aspect ='auto', origin='lower',cmap='jet',extent=(0,temps[-1]
plt.xlabel('temps [s]')
plt.ylabel("fr quence [hz]")
plt.ylim([0,20000])
plt.show()
```

**********fin**********