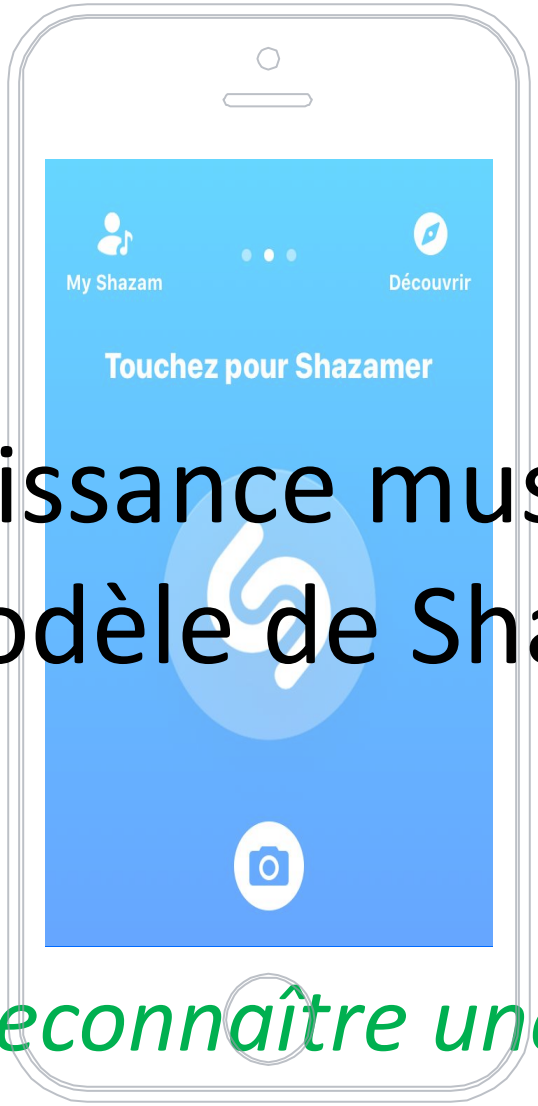
PLORMEL
Allan
33198



La reconnaissance musicale selon le modèle de Shazam

Comment reconnaître une musique ?

Démarche

I) Introduction

1- Reconnaissance

2- Nécessité d'utiliser la FFT

II) Gestion de flux

1- Fichier audio

2- Spectre 2D

III) Empreinte (Fingerprinting)

1- Principe du seuillage

2- Création d'une empreinte

IV) Comparaison des empreintes

1- Utilisation du produit scalaire matriciel

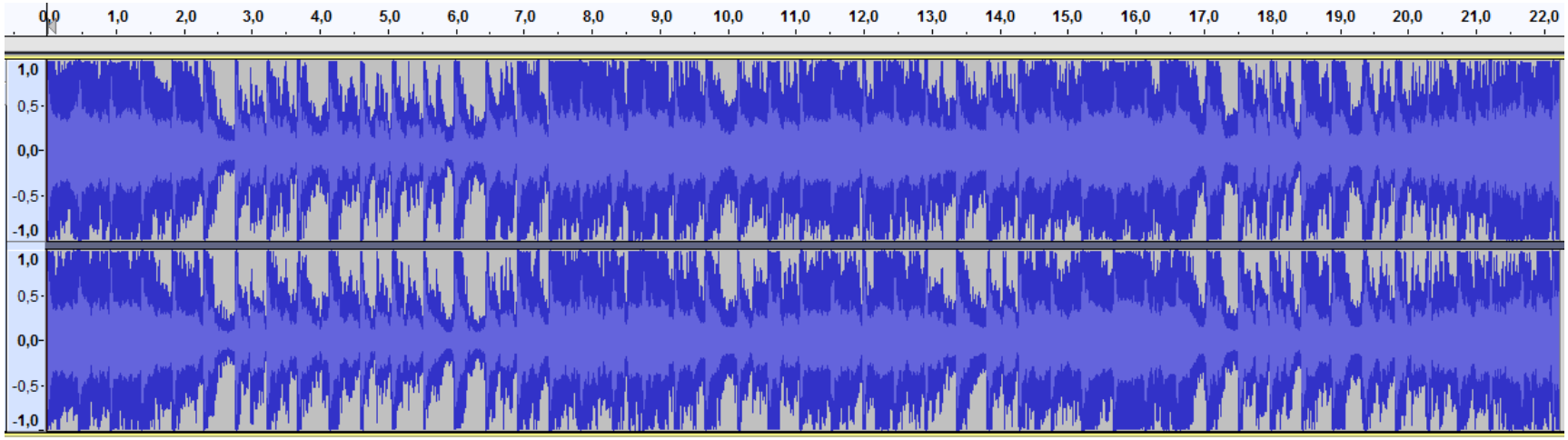
2- Identification à partir des empreintes

V) Conclusion et ouverture

I. 1- Reconnaissance

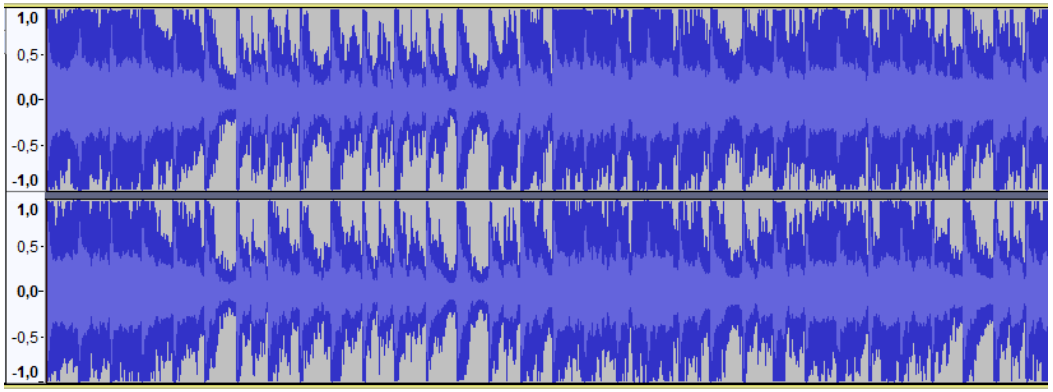
Utilisation d'extraits ou flux musicaux privilégiée

Fichier sonore complet

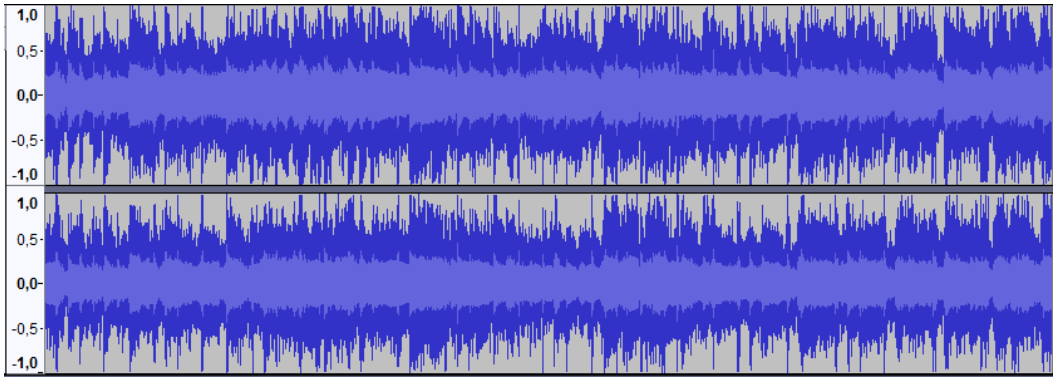


Extrait de *A kind of magic* de Queen

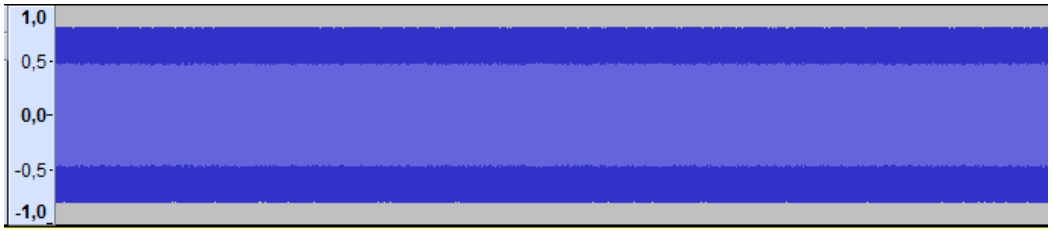
I. 2- Nécessité d'utiliser la FFT



A kind of magic de Queen



Good night song de Tears for fears



Good night song de Tears for fears bruité

II. 1-Fichier audio

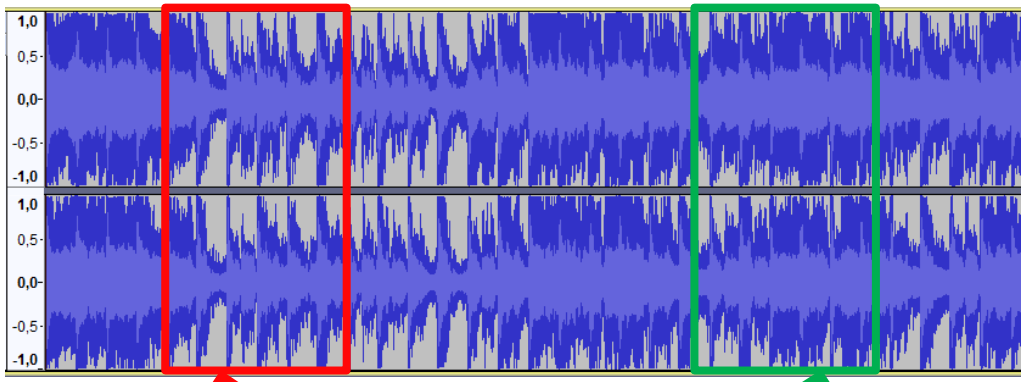
Utilisation de fichiers wav

Ce format de fichiers audio permet un encodage sans aucune perte de qualité en 16 bits–44khz.

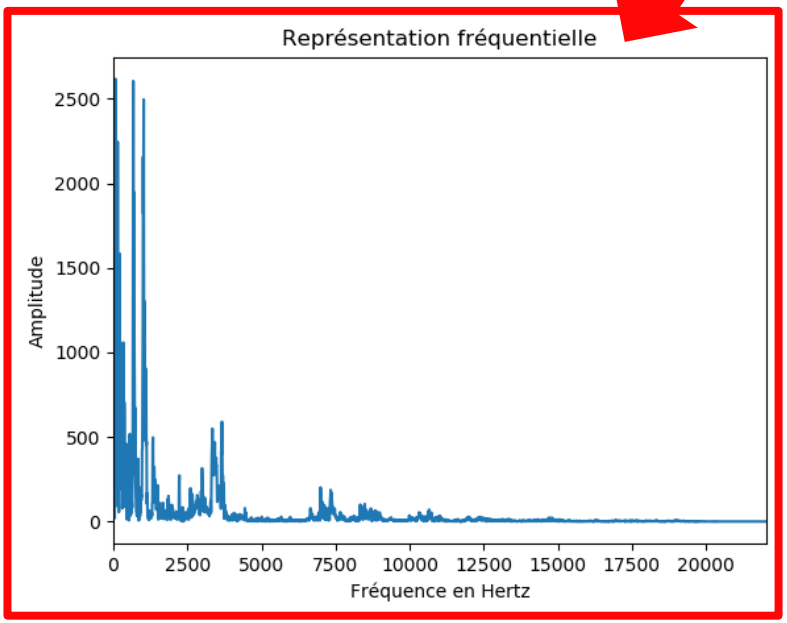
Avantages: lu par quasiment tous les types d'appareils
Inconvénients: format volumineux

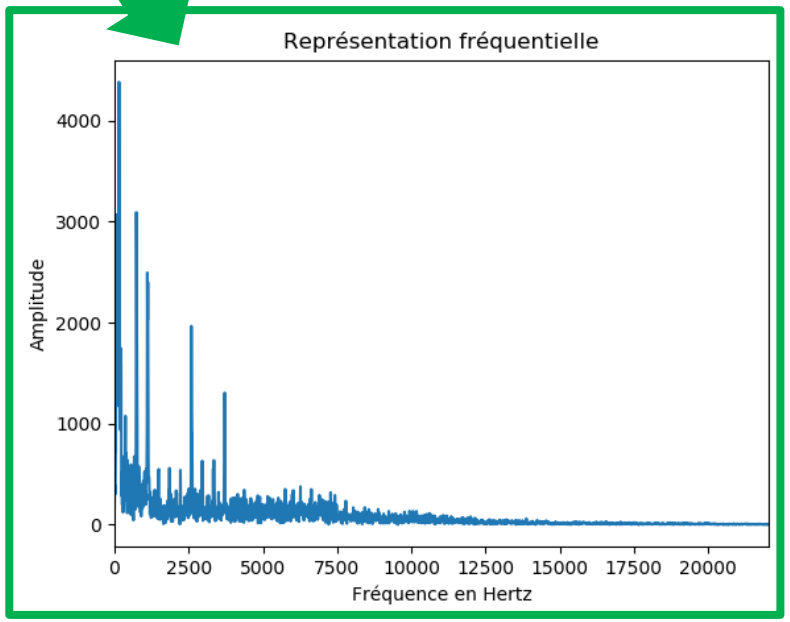
II. 2- Spectre 2D

a) Illustration du fenêtrage (séquençage)



Après *FFT*





Fenêtrage pour *A kind of magic* de Queen

II. 2- Spectre 2D

b) Mise sous forme matricielle

Fichier sonore

Fonction « *Séquence* »

Amplitudes pour chaque échantillon

Ne

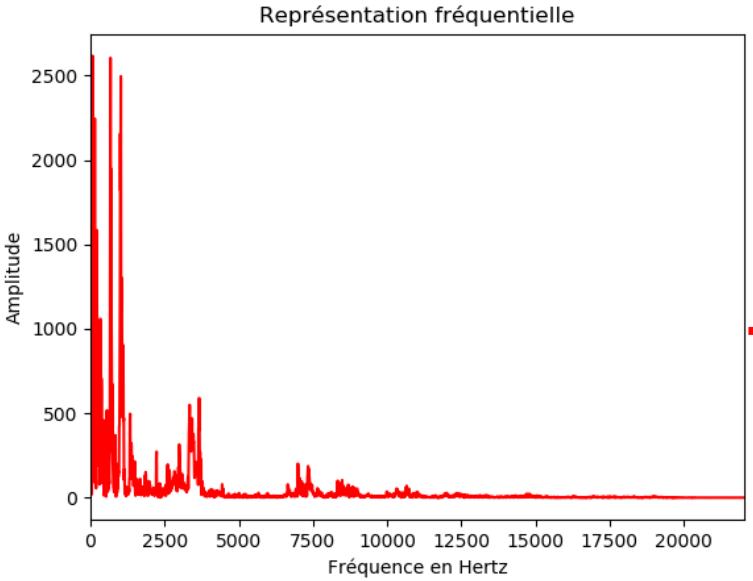
3.63	6.07	5.87 ... 3.87 4.64 4.40
5.74	5.11	5.27 ... 3.09 3.43 4.05
3.32	4.09	4.13 ... 4.38 5.04 5.57
...		
5.26	4.71	4.74 ... 4.97 4.75 5.05
3.41	4.75	4.28 ... 4.27 4.42 4.39
6.06	6.49	5.90 ... 4.78 4.42 4.88

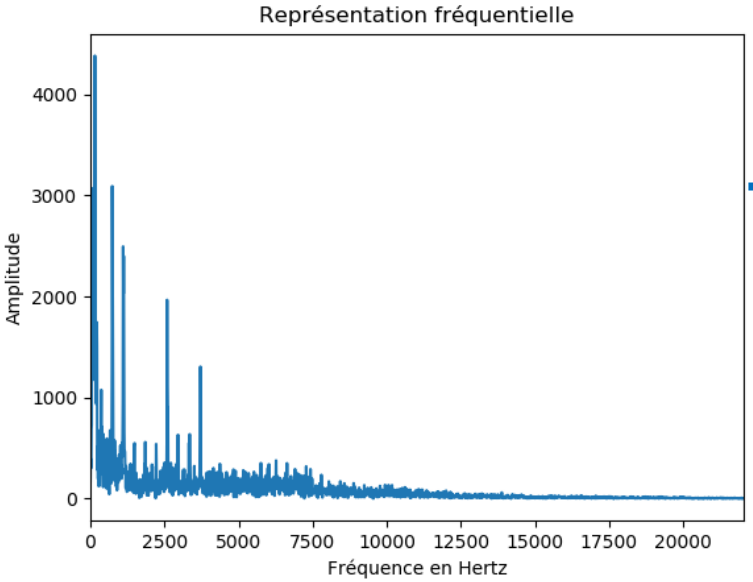
Fréquence

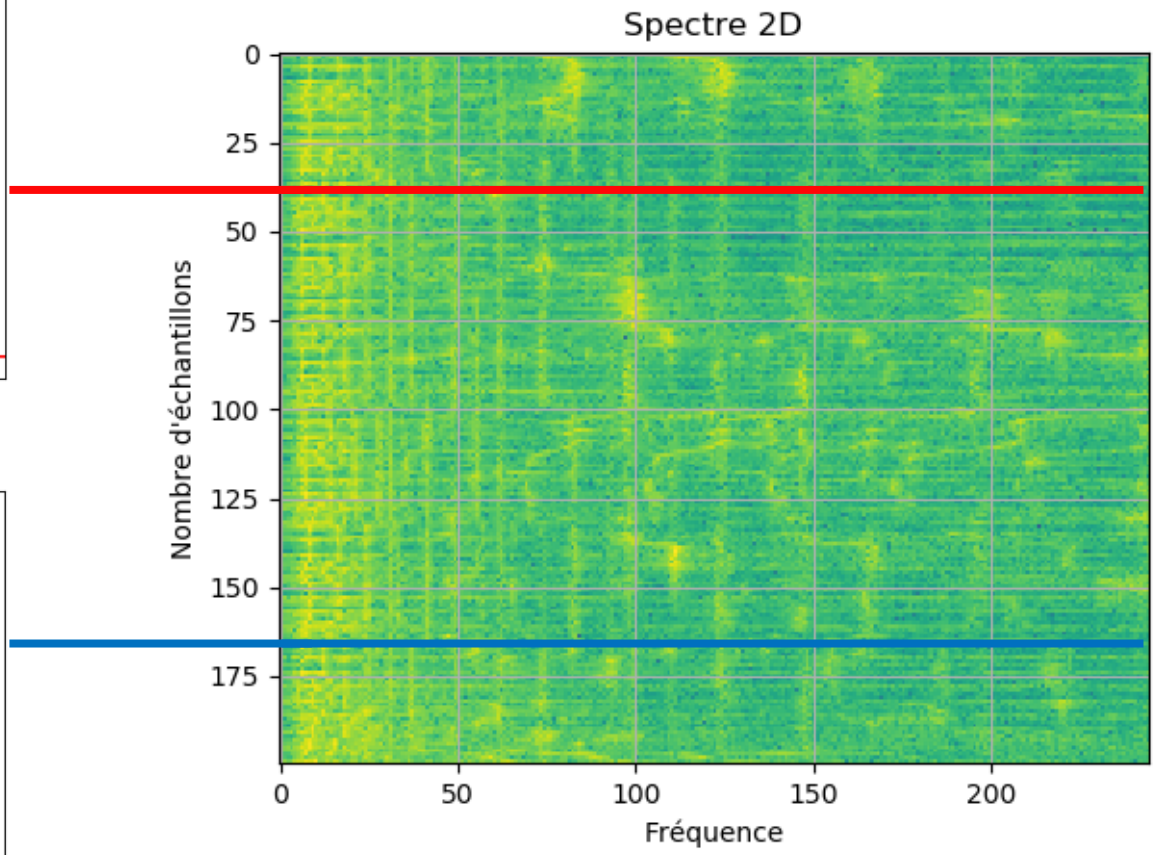
Matrice obtenue pour *A kind of magic* de Queen dont les coefficients sont arrondis au centième

II. 2- Spectre 2D

c) Création d'un spectre



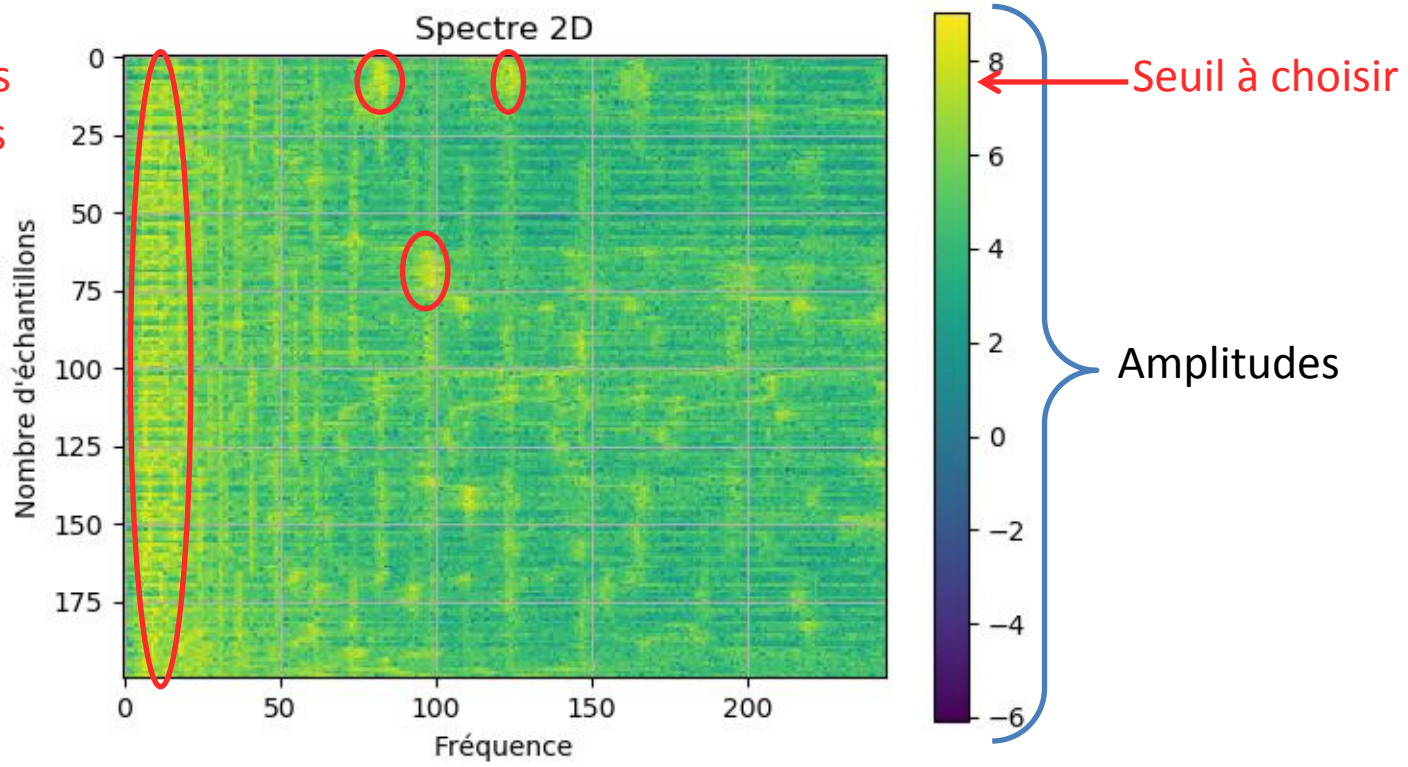




III.1- Principe du seuillage

a) Pourquoi utiliser un seuillage ?

Zones de pics d'amplitudes



Spectre 2D pour *A kind of magic* de Queen

III.1- Principe du seuillage

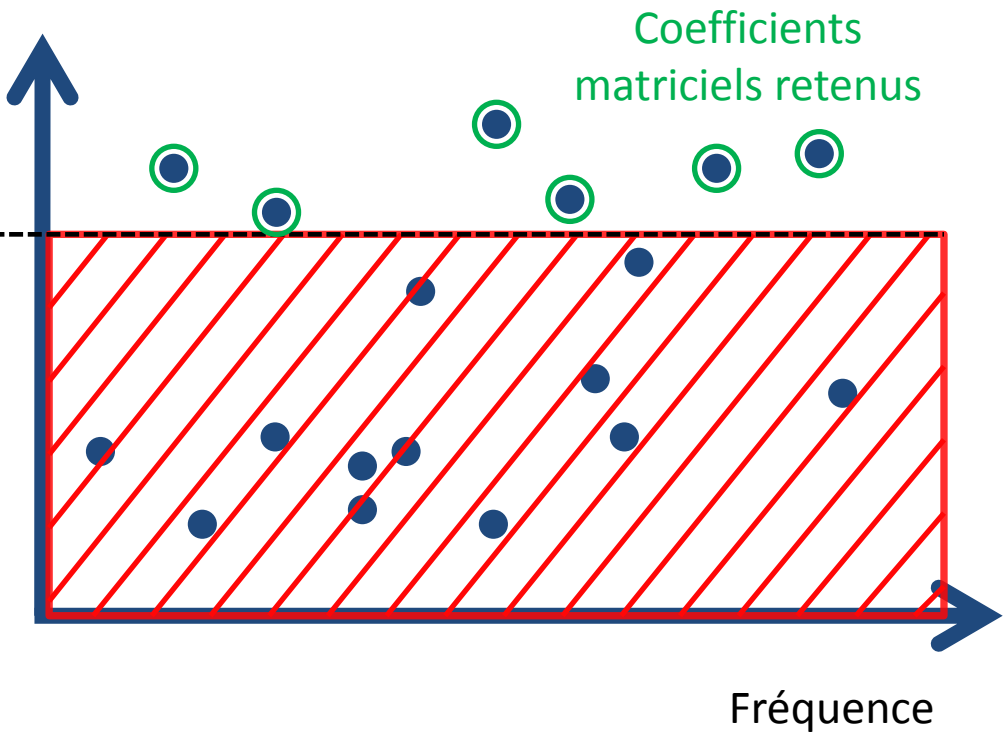
b) Utilisation du seuillage (méthode de dichotomie à partir du spectre 2D)

Pour obtenir 6 coefficients:

3.63 6.07 5.87 … 3.87 4.64 4.40
5.74 5.11 5.27 … 3.09 3.43 4.05
3.32 4.09 4.13 … 4.38 5.04 5.57
…
5.26 4.71 4.74 … 4.97 4.75 5.05
3.41 4.75 4.28 … 4.27 4.42 4.39
6.06 6.49 5.90 … 4.78 4.42 4.88

Matrice

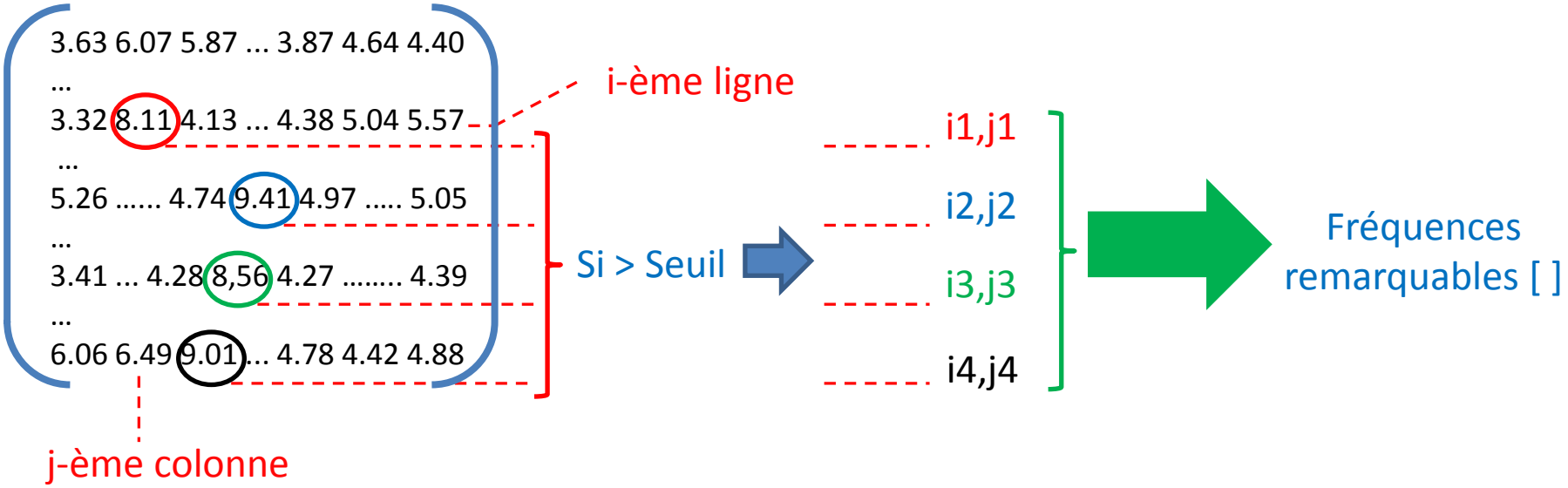
Fonction
« *Seuil_adapté* »



Vue en « coupe » du spectre pour un échantillon fixé

III. 2- Création d'une empreinte

a) Identification des coefficients matriciels caractéristiques



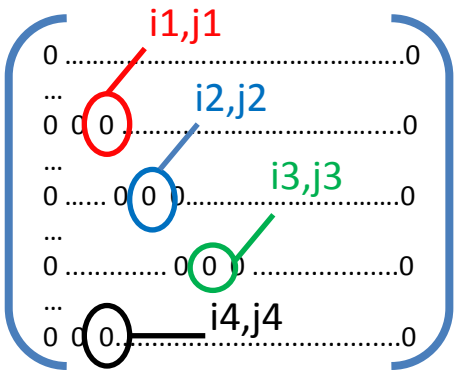
Fonction « *Liste_fréq_remarquables* »

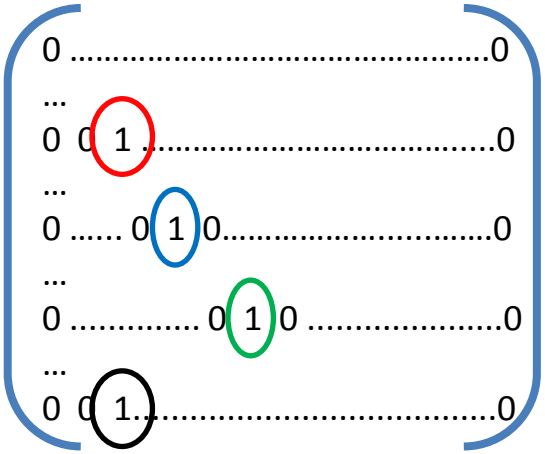
III. 2- Création d'une empreinte

b) Elaboration de l'empreinte

3.63 6.07 5.87 … 3.87 4.64 4.40
…
3.32 8.11 4.13 … 4.38 5.04 5.57
…
5.26 …… 4.74 9.41 4.97 ….. 5.05
…
3.41 … 4.28 8,56 4.27 …….. 4.39
…
6.06 6.49 9.01 … 4.78 4.42 4.88

;

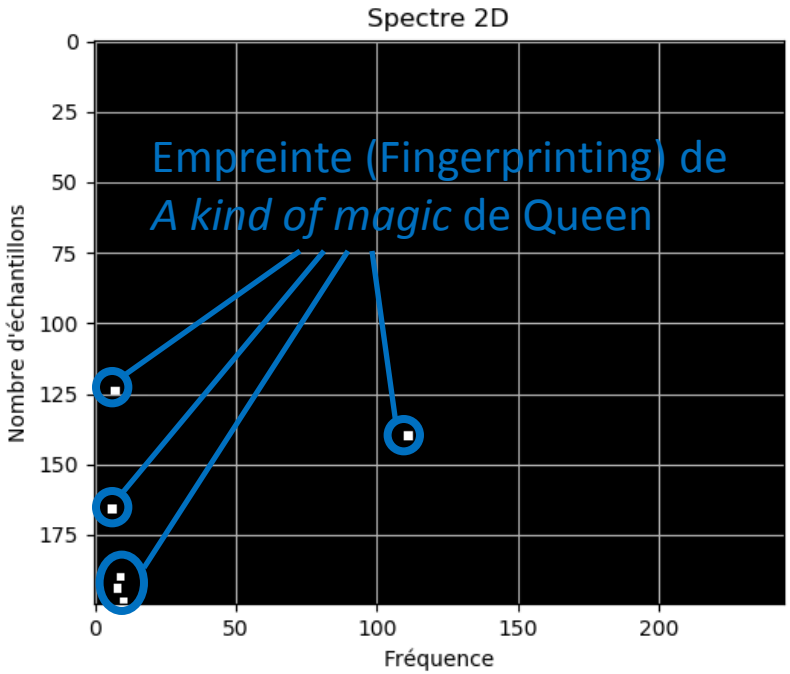




Empreinte
sous forme matricielle

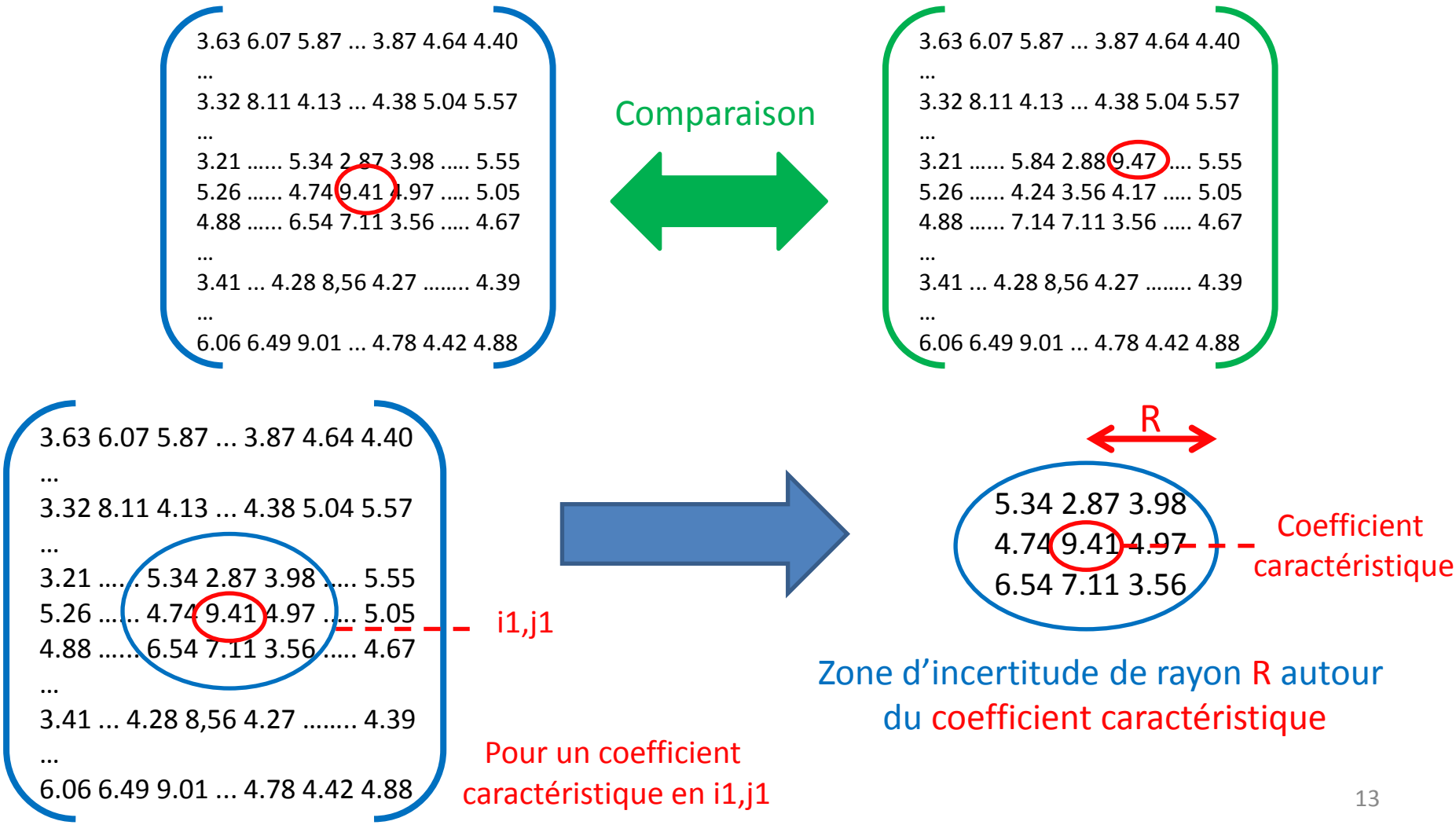
Empreinte
sous forme matricielle

Affichage du spectre
2D de la matrice



III. 2- Création d'une empreinte

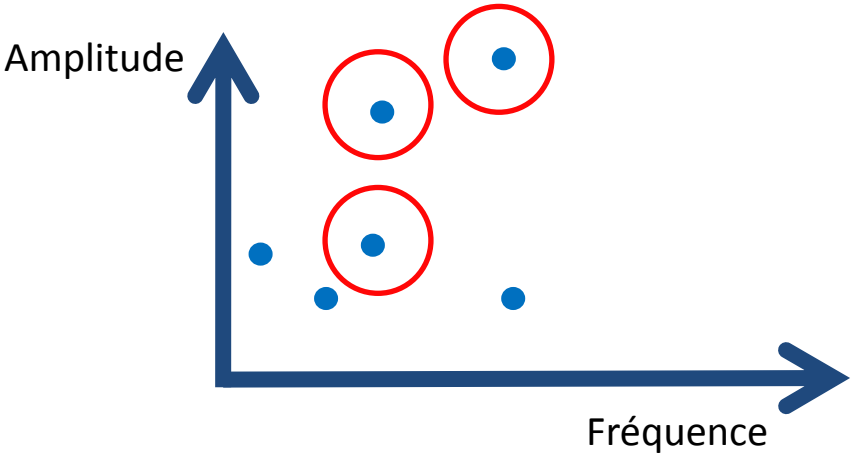
c) Création d'une zone d'incertitude autour des coefficients caractéristiques



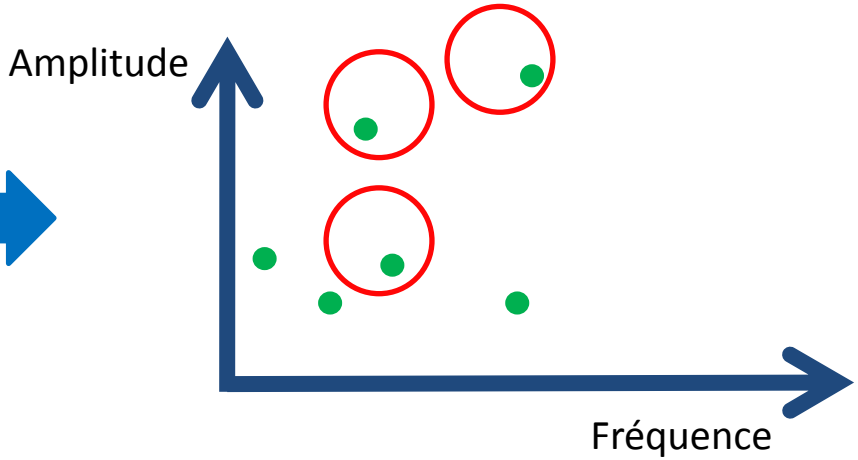
III. 2- Création d'une empreinte

d) Application du principe d'incertitude

Pour un certain R fixé:  R



A kind of magic de Queen
en bibliothèque



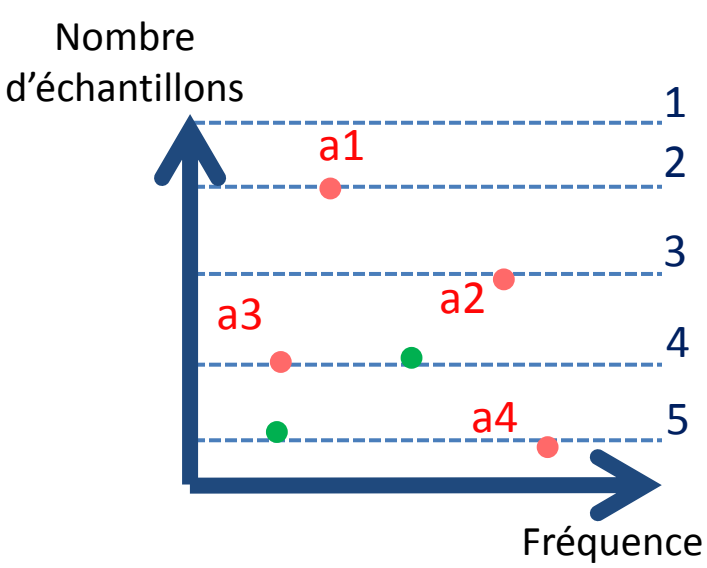
A kind of magic de Queen
bruité

Pour un échantillon fixé commun aux deux matrices

IV. 1- Utilisation du produit scalaire matriciel

Soient $A = (a_{i,j})_{1 \leqslant i,j \leqslant n}$ et $B = (b_{i,j})_{1 \leqslant i,j \leqslant n}$ deux matrices carrées.

Produit scalaire p_s = $\mathrm{tr}({}^tAB) = \sum_{j=1}^{n} \left(\sum_{i=1}^{n} a_{i,j} b_{i,j} \right) = \sum_{1 \leqslant i,j \leqslant n} a_{i,j} b_{i,j}.$



Empreinte matrice A

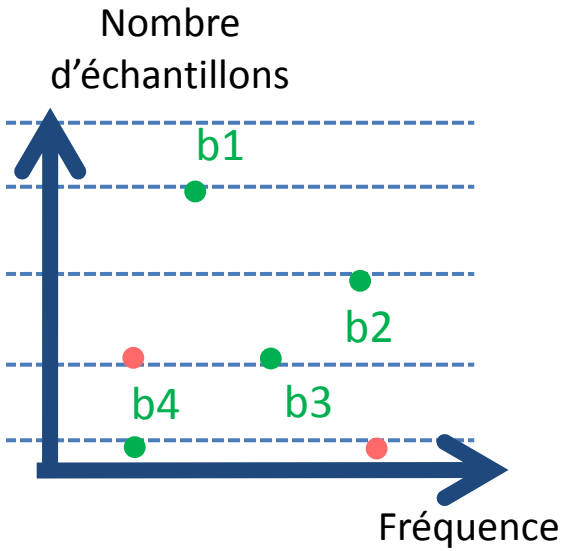
P_s = 0

P_s = a1*b1

P_s = a1*b1 + a2*b2

P_s = a1*b1 + a2*b2

P_s = a1*b1 + a2*b2



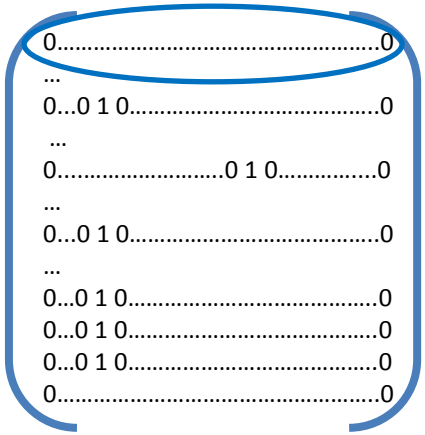
Empreinte matrice B

IV. 2- Identification à partir des empreintes

a) L'algorithme d'identification à partir du produit scalaire

Produit scalaire empreinte 1: p_s1= 0

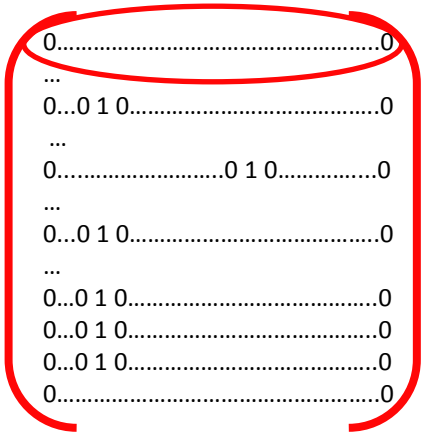
Produit scalaire empreinte 2: p_s2 = 0



A kind of magic de Queen en bibliothèque

P_s1+ = M[i , j] x M[i , j]

P_s2 += M[i , j] x M[i , j]



Extrait d'une musique captée par un micro

Si max (p_s1,p_s2) == p_s2:

➡ Les deux empreintes sont identiques

➡ Les musiques semblent être identiques

A kind of magic de Queen ⟺ Extrait capté

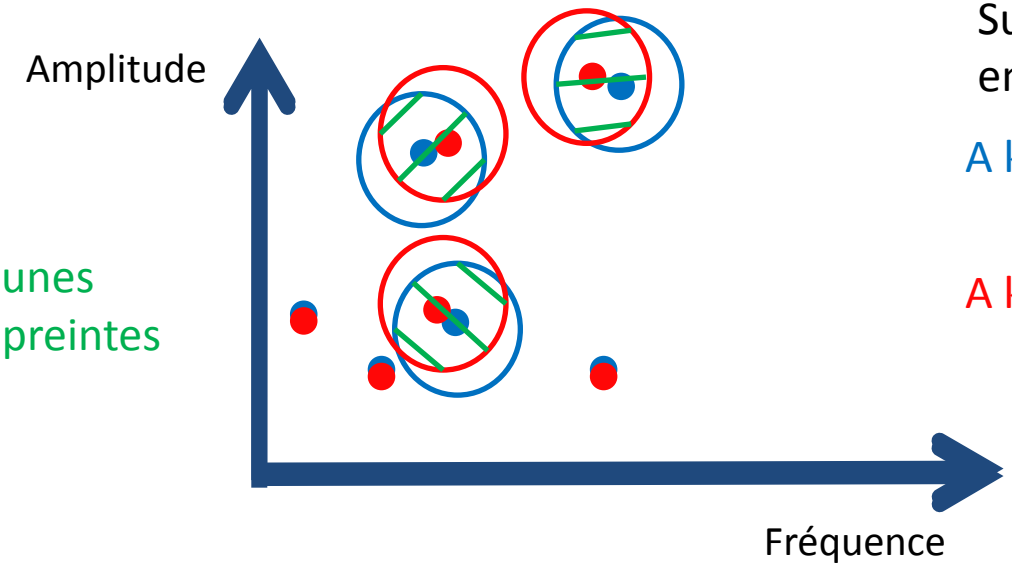
Si max (p_s1,p_s2) == p_s1:

Il faut regarder le nombre de points de l'empreinte de l'Extrait capté en commun avec ceux de *A kind of magic* de Queen pour déterminer s'ils correspondent ou non

IV. 2- Identification à partir des empreintes

b) Schématisation de la superposition des deux empreintes

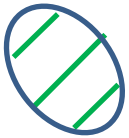
Pour un certain R fixé:



Superposition des empreintes de:

A kind of magic de Queen bruité

A kind of magic de Queen en bibliothèque



Zones communes aux deux empreintes

Le nombre de points à l'intérieur de ces zones correspond à l'ensemble des points communs aux deux empreintes

Pour un échantillon fixé commun aux deux matrices

IV. 2- Identification à partir des empreintes

c) Résultats obtenus à partir de différents échantillons

Rayon R de la zone d'incertitude utilisée	Fichier audio comparé à A kind of magic de Queen (version studio)	Type de bruit	Amplitude du Bruit (Ajouté sur Audacity) (1 correspond à l'amplitude maximale de A kind of magic)	Nombre de points en commun entre les empreintes du fichier et de A kind of magic de Queen
R = 2	AKOMQB1	Blanc	0,1	42
R = 2	AKOMQB2	Blanc	0,5	42
R = 2	AKOMQB3	Blanc	0,9	33
R = 2	AKOMQB4	Blanc	1	33
R = 2	GNSTFF	Aucun	0	0

V. CONCLUSION ET OUVERTURE

1- Ajout manuel de bruit dont l'amplitude est limitée à 1 avec Audacity

2- Les fichiers audio de la bibliothèque sont eux même des extraits de musiques

3- Ce procédé algorithmique dans son principe à de nombreuses applications dans divers domaines scientifiques

ANNEXE

```
from pylab import *
from scipy.io import wavfile
from scipy import fftpack
import matplotlib.pyplot as plt
from math import *


def Extraction(NomFichier):
    Fe, source = wavfile.read(NomFichier)
    Te=1./Fe
    tps=[k*Te for k in range(len(source))]
    source=source[:,0]
    return tps,source,Fe

def FFT(tps,source,Fe):
    TF_source = fftpack.fft(source)/np.size(source)
    N=len(source)+1
    axe_f = arange(0.,N-1)*Fe/N
    return axe_f,TF_source

def Affiche_spectre(axe_f,TF_source):
    figure(1)
    amplitude = np.abs(TF_source)
    Fe = axe_f[-1]
    plot(axe_f,amplitude,'-')
    title(u'Représentation fréquentielle')
    xlim(0,Fe/2)
    xlabel('Fréquence en Hertz')
    ylabel('Amplitude')
    show()

def Affiche_mapping(matrice):
    figure(1)
    imshow(matrice,"Greys_r",interpolation='nearest')
    grid(True)
    title(u'Spectre 2D')
    xlabel('Fréquence')
    ylabel("Nombre d'échantillons")
    colorbar()
    show()
```

Extraction des données du fichier wav sous forme de listes

Transformée de Fourier rapide à partir des listes obtenues précédemment

Représentation spectrale

Affichage du spectre 2D

```python
def Séquence(NomFichier):
    tps,source,Fe = Extraction(NomFichier)
    Ne=200
    longueur=len(tps)
    M = []
    for k in range(Ne):
        axe_f,TF_source = FFT(tps[k*(longueur//Ne):(k+1)*(longueur//Ne)],source[k*(longueur//Ne):(k+1)*(longueur//Ne)],Fe)
        N=len(TF_source)
        M.append(TF_source[0:N//20])
        #Affiche_spectre(axe_f,TF_source)
    M = np.log(abs(np.array(M)))
    #Affiche_mapping(M)
    return M

def Liste_fréq_remarquables(M,seuil):
    n_e,n_f = np.shape(M)
    L=[]
    for i in range(n_e):
        for j in range(n_f):
            if M[i,j]>seuil:
                L.append([i,j])
    return L

def Seuil_adapté(M,n_max):
    seuil1 = 10
    seuil2 = 0
    L = Liste_fréq_remarquables(M,seuil1)
    taille = len(L)
    while taille < n_max:
        seuil = (seuil1+seuil2)/2
        L = Liste_fréq_remarquables(M,seuil)
        print(len(L),seuil,taille)
        if len(L)>n_max:
            seuil2 = seuil
            taille = n_max-1
        else:
            seuil1 = seuil
            taille = len(L)
    return seuil
```

Création d'une matrice correspondant à la superposition de tous les échantillons de l'extrait

Liste contenant les coordonnées de tous les coefficients caractéristiques de la matrice

Elaboration par dichotomie d'un seuil à utiliser pour isoler les pics d'amplitude du spectre 2D

```
def Remplissage_Matrice(Matrice1,nbr_lig1,nbr_col1):
    M = np.zeros([nbr_lig1,nbr_col1])
    for i in range(nbr_lig1):
        for j in range(nbr_col1):
            M[i,j] = Matrice1[i,j]
    return M

def Test_Len_Matrices(Matrice1,Matrice2):
    nbr_lig1,nbr_col1 = np.shape(Matrice1)
    nbr_lig2,nbr_col2 = np.shape(Matrice2)
    if nbr_col1 != nbr_col2:
        if min(nbr_col1,nbr_col2) == nbr_col1:
            Matrice2 = Remplissage_Matrice(Matrice2,nbr_lig2,nbr_col1)
            return Matrice1,Matrice2
        else:
            Matrice1 = Remplissage_Matrice(Matrice1,nbr_lig2,nbr_col2)
            return Matrice1,Matrice2
    else:
        return Matrice1,Matrice2


def Valeurs_fermées(M,seuil):
    L_f_r = Liste_fréq_remarquables(M,seuil)
    nbr_lig,nbr_col = np.shape(M)
    R = 2
    n = len(L_f_r)
    A = np.zeros([nbr_lig,nbr_col])
    for i in range(nbr_lig):
        for j in range(nbr_col):
            for c in L_f_r:
                ic,jc = c
                d = sqrt(((i-ic)**2)+((j-jc)**2))
                if d < R:
                    A[i,j] = 1
    Affiche_mapping(A)
    return A
```

Remplissage d'une matrice

Modification (ou non) des matrices pour qu'elles soient de même taille

Création des empreintes avec une zone d'incertitude

```
def Produit_scalaire_matriciel(Matrice1,Matrice2):
    M1,M2 = Test_Len_Matrices(Matrice1,Matrice2)
    nbr_lig,nbr_col = np.shape(M1)
    p_s1 = 0
    p_s2 = 0
    nombre_de_points_commun = 0
    for i in range(nbr_lig):
        for j in range(nbr_col):
            p_s1+= M1[i,j]*M1[i,j]
            p_s2+= M1[i,j]*M2[i,j]
    if max(p_s1,p_s2) == p_s2:
        return True
    else:
        return False
##    return num/den

def Points_communs(NomFichier1,NomFichier2,Nombredepoints):
    Matrice1 = Séquence(NomFichier1)
    Matrice2 = Séquence(NomFichier2)
    seuil1 = Seuil_adapté2(Matrice1,Nombredepoints)
    seuil2 = Seuil_adapté2(Matrice2,Nombredepoints)
    V_f1 = Valeurs_fermées(Matrice1,seuil1)
    V_f2 = Valeurs_fermées(Matrice2,seuil2)
    M1,M2 = Test_Len_Matrices(V_f1,V_f2)
    nbr_lig,nbr_col = np.shape(M1)
    nombre_de_points_commun = 0
    for i in range(nbr_lig):
        for j in range(nbr_col):
            if M1[i,j]*M2[i,j] != 0:
                nombre_de_points_commun+=1
    print(nombre_de_points_commun)
```

Comparaison du produit scalaire de « référence » et « d'application »

Affichage du nombre de points communs à deux empreintes

```
def Points_communs2(Empreinte1,Empreinte2):
    M1,M2 = Test_Len_Matrices(Empreinte1,Empreinte2)
    nbr_lig,nbr_col = np.shape(M1)
    nombre_de_points_commun = 0
    for i in range(nbr_lig):
        for j in range(nbr_col):
            if M1[i,j]*M2[i,j] != 0:
                nombre_de_points_commun+=1
    return nombre_de_points_commun

def Identification_musique(NomFichier1,NomFichier2,Nombredepoints):
    Matrice1 = Séquence(NomFichier1)
    Matrice2 = Séquence(NomFichier2)
    seuil1 = Seuil_adapté2(Matrice1,Nombredepoints)
    seuil2 = Seuil_adapté2(Matrice2,Nombredepoints)
    V_f1 = Valeurs_fermées(Matrice1,seuil1)
    V_f2 = Valeurs_fermées(Matrice2,seuil2)
    if Produit_scalaire_matriciel(V_f1,V_f2)==True or Points_communs2(V_f1,V_f2) >= 30:
        print('Les musiques semblent être identiques !')
    else:
        print('Les musiques semblent être différentes !')
```

Affichage du nombre de points communs à deux empreintes

Identification de l'extrait inconnu en le comparant à un extrait connu