Filtrage_eleve

October 9, 2019

1 Librairie

[1]:
```
import wave
import math
import binascii
import winsound
import struct
import os
import numpy as np
import scipy.io.wavfile
import scipy
import matplotlib.pyplot as plt
```

2 Signal d'entrée

2.1 définition des différents paramètres

$T_a >$ temps de réaction humain
$\$f_e \overline{2>\$frquencemaximaleaudible}$
f_a et f_b deux sons composant le signal

2.1.1 A vous choisissez T_a, f_e, f_a, et f_b

[ ]:
```
Ta = #durée d'acquisition en seconde du signal
Fe = #fréquence d'échantillonnage en Hz pour signaux audio
f_a = # fréquence en Hz de l'harmonique a présente dans le signal
f_b = # fréquence en Hz de l'harmonique b présente dans le signal
niveau = 1 # niveau sonore des hauts-parleur
nbCanal = 2    # stéreo
nbOctet = 1    # taille d'un échantillon : 1 octet = 8 bits
```

2.2 calculs du nombre d'échantillon, période d'échantillonnage, résolution spectrale

$N = T_a \times F_e$ relation nombre d'échantillon, durée d'acquisition, fréquence d'échantillonnage
$T_e = \frac{1}{f_e}$ définition période d'échantillonnage

$\Delta f = \frac{1}{T_a} = \frac{f_e}{N}$ définition résolution spectrale

2.2.1 A vous: écrire un code ici qui calcule N, T_e, et Δf

2.3 calcul trace temporelle et spectre du signal d'entrée

t est une liste de date telle que $t[i] = i \times T_e$
$v_e = A\sin(2\pi f_a t) + A\sin(2\pi f_b t)$

2.3.1 A vous: écrire un code ici qui calcule v_e

la série de Fourier de v_e se calcule avec l'algorithme FFT de numpy np.fft.fft

2.3.2 A vous: écrire un code ici qui calcule la transformée de Fourier de v_e

affichage du signal temporel entre T_{min} et T_{max}, et entre f_{min} et f_{max}

2.3.3 A vous: écrire un code ici qui affiche l'un en dessous de l'autre $v_e(t)$ et son spectre

2.4 encodage dans un fichier .wav

écriture dans le fichier son.wav

[ ]:
```
NomFichier = 'son.wav'
Monson = wave.open(NomFichier,'w')
parametres = (nbCanal,nbOctet,Fe,N,'NONE','not compressed')# tuple
Monson.setparams(parametres)    # création de l'en-tête (44 octets)
for i in range(0,N):
    # canal gauche
    # 127.5 + 0.5 pour arrondir à l'entier le plus proche
    valG = wave.struct.pack('B',int(128.0 + 127.5*0.5*ve[i]))
    # canal droit
    valD = wave.struct.pack('B',int(128.0 + 127.5*0.5*ve[i]))
    Monson.writeframes(valG + valD) # écriture frame

Monson.close()
```

visualisation de l'effet de quantification sur 1 octet = 8 bits

2.4.1 A vous: choisissez T_{min} et T_{max} pour voir la solution

[ ]:
```
Tmin =
Tmax =

Se = np.zeros(N)
for i in range(0,N):
    Se[i] = int(128.0 + 127.5*0.5*ve[i])
vT = np.arange(N)*Te
plt.plot(vT[int(Tmin/Te):int(Tmax/Te)],Se[int(Tmin/Te):int(Tmax/Te)],'.')
plt.xlabel('t (s)')
```

```
plt.ylabel('$v_{e}$')
```

2.5 lecture du fichier son

[ ]:
```
Fichier = open(NomFichier,'rb')
data = Fichier.read()
tailleFichier = len(data)
print('\nTaille du fichier',NomFichier, ':', tailleFichier,'octets')
print("Lecture du contenu de l'en-tête (44 octets) :")
print(binascii.hexlify(data[0:44]))
print("Nombre d'octets de données :",tailleFichier - 44)
Fichier.close()
winsound.PlaySound('son.wav',winsound.SND_FILENAME)
```

3 Filtrage numérique passe-bas

3.1 paramètre d'un filtre passe-bas

f_c fréquence de coupure en Hz

$\omega_c = 2\pi f_c$ pulsation de coupure en rad.s^{-1}

3.1.1 A vous: définir f_c et calculer ω_c

3.2 établissement de la relation de récurrence du filtre numérique

fonction de transfert filtre RC: $H = \frac{v_s}{v_e} = \frac{1}{1+j\frac{\omega}{\omega_c}}$

équation différentielle: $\frac{1}{\omega_c}\frac{dv_s}{dt} + v_s = v_e$

intégration entre iT_e et $(i+1)T_e$: $\frac{1}{\omega_c}\int_{iT_e}^{(i+1)T_e}\frac{1}{\omega_c}\frac{dv_s}{dt}dt + \int_{iT_e}^{(i+1)T_e} v_s dt = \int_{iT_e}^{(i+1)T_e} v_e dt$

donc $\frac{v_s((i+1)T_e)-v_s(iT_e)}{\omega_c} + \frac{v_s((i+1)T_e)+v_s(iT_e)}{2T_e} = \frac{v_e((i+1)T_e)+v_e(iT_e)}{2T_e}$

donc $v_s((i+1)T_e) = Av_s(iT_e) + Bv_e(iT_e)$ avec $A = \frac{2-\omega_c T_e}{2+\omega_c T_e}$ et $B = \frac{\omega_c T_e}{2+\omega_c T_e}$

3.2.1 A vous: définir A et B et calculer v_s

4 Signal de sortie

4.1 trace temporelle et spectre du signal de sortie

4.1.1 A vous: tracer $v_s(t)$ entre T_{min} et T_{max} et son spectre entre f_{min} et f_{max}

4.2 encodage dans un fichier son.wav

écriture dans le fichier son.wav

[ ]:
```
NomFichier = 'son.wav'
Monson = wave.open(NomFichier,'w')
parametres = (nbCanal,nbOctet,Fe,N,'NONE','not compressed')# tuple
Monson.setparams(parametres)    # création de l'en-tête (44 octets)
for i in range(0,N):
```

```
    # canal gauche
    # 127.5 + 0.5 pour arrondir à l'entier le plus proche
    valG = wave.struct.pack('B',int(128.0 + 127.5*0.5*vs[i]))
    # canal droit
    valD = wave.struct.pack('B',int(128.0 + 127.5*0.5*vs[i]))
    Monson.writeframes(valG + valD) # écriture frame

Monson.close()
```

visualisation de l'effet de quantification sur 1 octet = 8 bits

4.2.1 A vous: choisissez T_{min} et T_{max} pour voir la solution

[ ]:
```
Tmin =
Tmax =

Se = np.zeros(N)
for i in range(0,N):
    Se[i] = int(128.0 + 127.5*0.5*vs[i])
vT = np.arange(N)*Te
plt.plot(vT[int(Tmin/Te):int(Tmax/Te)],Se[int(Tmin/Te):int(Tmax/Te)],'.')
plt.xlabel('t (s)')
plt.ylabel('$v_{e}$')
```

4.3 lecture du fichier son

[ ]:
```
Fichier = open(NomFichier,'rb')
data = Fichier.read()
tailleFichier = len(data)
print('\nTaille du fichier',NomFichier, ':', tailleFichier,'octets')
print("Lecture du contenu de l'en-tête (44 octets) :")
print(binascii.hexlify(data[0:44]))
print("Nombre d'octets de données :",tailleFichier - 44)
Fichier.close()
winsound.PlaySound('son.wav',winsound.SND_FILENAME)
```

5 Application à un enregistrement

5.0.1 A vous: télécharger sur internet (par exemple sonothèque) un fichier son .wav de quelques secondes (une ou deux) encodé sur 16 bits en mode stéréo et filtrons-le

[ ]:
```
Fe, data = scipy.io.wavfile.read('mon_fichier_avant.wav')
print('fréquence d échantillonage', Fe, 'Hz')
v_e = data[:,1]
```

5.0.2 A vous: avec l'attribut size des objets numpy array calculer N puis en déduire T_e et Δf

5.0.3 A vous: calculer la fft du signal d'entrée avec la fonction numpy fft.fft

5.0.4 A vous: afficher le signal temporel et la transformée de Fourier du signal d'entré

5.0.5 A vous: définir la fréquence de coupure de votre filtre numérique et calculer le signal de sortie

5.0.6 A vous: afficher le signal temporel et la transformée de Fourier du signal de sortie

5.0.7 A vous: encoder le signal de sortie dans un fichier .wav avec le code ci-dessous et écoutez-le avec le lecteur de votre ordinateur

[ ]:
```
NomFichier = 'mon_fichier_apres.wav'
Monson = wave.open(NomFichier,'w')
parametres = (nbCanal,nbOctet,Fe,N,'NONE','not compressed')# tuple
Monson.setparams(parametres)    # création de l'en-tête (44 octets)
for i in range(0,N):
    # canal gauche
    # 127.5 + 0.5 pour arrondir à l'entier le plus proche
    valGbis = wave.struct.pack('B',int(128.0 + 127.5*(2**(-16))*v_s[i]))
    # canal droit
    valDbis = valGbis
    Monson.writeframes(valGbis + valDbis) # écriture frame

Monson.close()
```