Filtrage numérique

Capacité numérique : simuler l'action d'un filtre sur un signal périodique dont le spectre est fourni. Mettre en évidence l'influence des caractéristiques du filtre sur l'opération de filtrage.

1) Outils, exemples

Calcul d'un signal à partir de ses harmoniques

Le signal périodique, de fréquence fondamentale f , est défini par les amplitudes de ses harmoniques (A_k pour $k = 0, 1, ...P$) et par leur phase à l'origine (φ_k) :

$e(t) = P\sum k=0 A_k \cos(2\pi kft + \varphi_k)$

où P est le rang de l'harmonique de plus haute fréquence, s'il existe, ou bien le rang auquel on décide de stopper la somme pour en donner une approximation.

La fonction signal définie ci-dessous effectue le calcul de cette somme et renvoie sa valeur :

```
import numpy as np
import math
import matplotlib.pyplot as plt
```

```
def signal(t,f,A,phi):
    '''
        Paramètres :
            t : temps
            f : fréquence
            A : liste des amplitudes des harmoniques
            phi : liste des phases à l'origine des harmoniques
        Objets renvoyés :
            y : somme des harmoniques
    '''
    y = 0.0
    for k in range(len(A)):
        y += A[k]*np.cos(2*np.pi*k*f*t+phi[k])
    return y
```

On prendra pour simplifier la fréquence du fondamental f = 1 ; cela revient à tracer en fonction d'une fréquence réduite f/f_0.

Tracé d'un signal

Voici un exemple de signal, comportant 5 harmoniques :

```
A = [0,1,0.0,0.4,0.0,0.1] #liste des amplitudes des harmoniques
phi = [0,0,0,-0.7,0,0.4] # liste des phases à l'origine des harmoniques
P = len(A)+1 # nombre d'harmoniques
f = 1.0 # fréquence réduite
(a,b)=(0,2/f) # intervalle du tracé (2 périodes)
N = 30*P #  (tracé précis, plus de points si plus d'harmoniques)
t = np.linspace(a,b,N)
plt.figure() # ouverture d'une nouvelle figure
plt.plot(t,signal(t,f,A,phi),'b',label='e(t)') #ajout d'une courbe (en bleu:'b') sur la figure
plt.grid()
plt.xlabel('t',fontsize=16)
plt.ylabel('u',fontsize=16)
plt.legend(loc='upper right')
plt.show()
```

Tracé d'un spectre en amplitude

Pour cela, on utilise la fonction vlines() de matplotlib.pyplot qui permet de tracer des lignes verticales à travers les axes. Sa synthaxe est : vlines(x, ymin, ymax, 'colors', 'linestyles'). Exemple à deux composantes spectrales :

```
plt.figure(2)
plt.vlines(0,0,4,'b',label='e(t)')# Tracé de la composante continue
plt.vlines(100,0,2,'b') # Tracé de la composante fondamentale
```

```
plt.xlabel('f(Hz)')
plt.title ('spectres')
plt.ylabel('amplitude(V)')
plt.show()
```

Définition d'un filtre

L'action du filtre sur un signal périodique est déterminée par la fonction de transfert harmonique. Par exemple, pour un filtre passe-bas du premier ordre, on définit la fonction suivante :

```
def H(f):
    fc=1 # fréquence de coupure réduite
    return 1/(1+1j*(f/fc))
```

Opération de filtrage

La fonction suivante applique une fonction de transfert à un signal défini par la liste des amplitudes et des phases de ses harmoniques :

```
def filtrage(H,A,phi):
    '''
        Paramètres :
            H : fonction de transfert (premier paramètre = fréquence)
            A : liste des amplitudes des harmoniques
            phi : liste des phases à l'origine des harmoniques

        Objets renvoyés :
            A_f : liste des amplitudes du signal filtré
            phi_f : liste des phases à l'origine du signal filtré
    '''
    A_f = A.copy() # on doit faire une copie des listes pour ne pas les modifier
    phi_f = phi.copy()
    for k in range(len(A)):
        h = H(k*f) # harmonique de fréquence k*f
        A_f[k] *= np.absolute(h) # calcul du module
        phi_f[k] += np.angle(h) #calcul de la phase
    return (A_f,phi_f)
```

Voici par exemple le filtrage passe-bas du signal défini plus haut, avec une fréquence de coupure égale à sa fréquence. La courbe du signal filtré est tracée sur la même figure que la courbe du signal d'entrée.

```
(A_f,phi_f) = filtrage(H,A,phi)
plt.figure(1)
t = np.linspace(a,b,N)
plt.plot(t,signal(t,f,A_f,phi),'r',label='s(t)')
plt.legend(loc='upper right')
plt.show()
```

Travail à faire

En s'inspirant de ce qui est au dessus,dans plusieurs cellules de code (menu Cellule, insérer) :

1. Ecrire une fonction coeffs(n) renvoyant les deux tableaux des n premières amplitudes et phases de la décomposition d'un signal carré symétrique d'amplitude A=1, qui s'écrit

$$e(t) = \frac{4A}{\pi}\left(\sin(\omega t) + \frac{1}{3}\sin(3\omega t) + \frac{1}{5}\sin(5\omega t) + \frac{1}{7}\sin(7\omega t)\ldots\right)$$

instruction éventuellement utile : np.zeros(n) créee un tableau vide de n éléments ;

2. Créér deux couples de tableaux A_10,phi_10 et A_20,phi_2 contenant les listes des coefficients des harmoniques n=10 ou 20. Superposer les deux courbes correspondantes, comparer leurs allures. On prendra n=20 pour la suite du TP.
3. Superposer les tracés des spectres en amplitude en entrée et en sortie du filtre passe bas ;
4. Superposer les tracés des courbes en entrée et en sortie du filtre passe bas ;
5. Adapter le programme de façon à vérifier ce qui est avancé dans l'exercice 5 de la feuille ALI : appliquer un filtre passe bande de facteur de qualité 20 à un signal créneau de manière à restituer uniquement l'harmonique 3 :

comparer les spectres et signaux en entrée et en sortie.

```
def coeffs (n):
    tab_A = np.zeros(n)
    tab_Phi = np.zeros(n)
    for i in range(n):
        if (i+1)%2==0:
            tab_A[i]=4/(np.pi*i)
            tab_Phi[i]=np.pi/2

    return tab_A,tab_Phi

A_10,phi_10=coeffs(10)
A_20,phi_20=coeffs(20)


P = len(A_20)+1
f=1.0 # fréquence
(a,b)=(0,2/f) # intervalle du tracé (4 périodes)
N = 30*P #  (tracé précis, plus de points si plus d'harmoniques)
t = np.linspace(a,b,N)
plt.figure()
plt.plot(t,signal(t,f,A_10,phi_10),'b',label='10 harmoniques')
plt.plot(t,signal(t,f,A_20,phi_20),'r',label='20 harmoniques')
plt.grid()
plt.xlabel('t',fontsize=16)
plt.ylabel('u',fontsize=16)
plt.legend(loc='upper right')
plt.show()



def filtrage(H,A,phi):
    '''
        Paramètres :
            H : fonction de transfert (premier paramètre = fréquence)
            A : liste des amplitudes des harmoniques
            phi : liste des phases à l'origine des harmoniques

        Objets renvoyés :
            A_f : liste des amplitudes du signal filtré
            phi_f : liste des phases à l'origine du signal filtré
    '''
    A_f = A.copy() # on doit faire une copie des listes pour ne pas les modifier
    phi_f = phi.copy()
    for k in range(len(A)):
        h = H(k*f) # harmonique de fréquence k*f
        A_f[k] *= np.absolute(h)
        phi_f[k] += np.angle(h)
    return (A_f,phi_f)

def H(f):
    fc=3
    if f==0:
        return 0
    return 1/(1+1j*20.0*(f/fc-fc/f))


(A_s,phi_s) = filtrage(H,A_20,phi_20)

plt.figure()
for i in range(20):
    plt.vlines(i,0,A_20[i],'b',label='e(t)')
    plt.vlines(i+0.1,0,A_s[i],'r',label='s(t)')
plt.xlabel('f')
plt.title ('spectres')
plt.ylabel('amplitude(V)')
plt.show()

plt.figure()
t = np.linspace(a,b,N)
plt.plot(t,signal(t,f,A_20,phi_20),'b',label='e(t)')
plt.plot(t,signal(t,f,A_s,phi_s),'r',label='s(t)')
plt.legend(loc='upper right')
plt.show()
```