Filtres de reconstruction analogiques et numériques

1. Introduction

La reconstruction d'un signal est l'opération qui consiste à convertir un signal numérique, par nature discret, en un signal analogique continu. La reconstruction se fait avec un convertisseur numérique-analogique (CNA), mais nécessite aussi un filtrage passe-bas pour éliminer les composantes spectrales propres au signal discret, comme expliqué dans le document Échantillonnage et reconstruction d'un signal périodique.

Le filtre de reconstruction est appelé filtre de lissage, ou filtre anti-image, car il doit enlever les composantes du spectre image, c'est-à-dire les composantes dont la fréquence est supérieure à la moitié de la fréquence d'échantillonnage. Le filtre de lissage est un filtre passe-bas dont la fréquence de coupure est égale à la moité de la fréquence d'échantillonnage.

La fréquence d'échantillonnage est fixée $f_e = 2000\ Hz$. La fréquence de Nyquist (moitié de celle d'échantillonnage) est $f_n = 1000\ Hz$. Le signal numérique considéré est une sinusoïde échantillonnée à f_e, dont la fréquence doit être inférieure à celle de Nyquist, afin de respecter la condition de Nyquist-Shannon. Cette fréquence sera variée de 100 Hz à 900 Hz. La conversion numérique-analogique est effectuée avec la centrale SysamSP5, en utilisant le module Python d'interface pour Sysam SP5 présenté dans CAN Eurosmart : interface pour Python. Le résultat de la reconstruction est observé à l'oscilloscope.

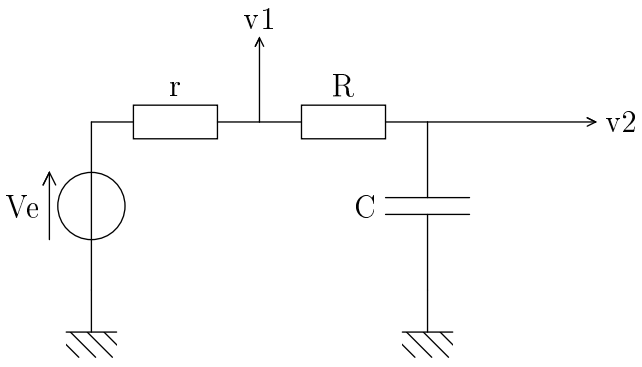
Nous allons voir deux exemples de filtres de lissage analogiques. Leur fréquence de coupure est fixée à 1000 Hz, soit la moitié de la fréquence d'échantillonnage.

Dans un deuxième temps, nous verrons la méthode numérique d'interpolation, qui consiste à augmenter la fréquence d'échantillonnage pour la conversion numérique-analogique. Il faut pour cela appliquer un filtre numérique passe-bas, appelé filtre d'interpolation. C'est la méthode utilisée pour la conversion numérique-analogique du son (par exemple dans les lecteurs de CD audio).

2. Filtres de lissage analogiques

2.a. Filtre du premier ordre

Le filtre passe-bas le plus simple est un filtre RC. On le place en sortie du CNA :



La source de tension représente la sortie SA1 de la centrale SysamSP5, avec sa

résistance de sortie $r = 50\ \Omega$. Les voies $v1$ et $v2$ de l'oscilloscope permettent d'observer le signal en sortie du CNA et en sortie du filtre. Les valeurs pour le filtre sont $R = 6.8 + 0.39 = 7.2\ k\Omega$ et $C = 22\ nF$, ce qui donne une fréquence de coupure de $1.0\ kHz$.

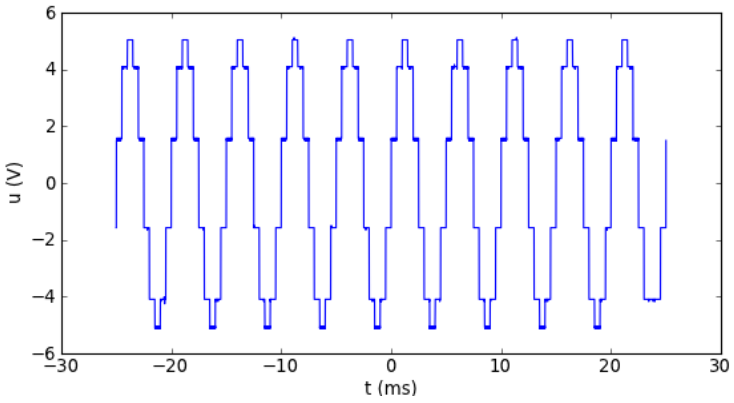
Le signal numérique est une sinusoïde de fréquence f ajustable échantillonnée à $2\ kHz$. Voici le programme python qui effectue l'échantillonnage d'un nombre Np de périodes et effectue la conversion numérique-analogique sur la sortie SA1 de la centrale SysamSP5 (avec répétition périodique). Le nombre de points total N ne doit pas dépasser 262142. La fréquence f de la sinusoïde est ici choisie à $100\ Hz$. Le spetre du signal discret obtenu est aussi calculé est tracé. L'exécution de la fonction `declencher_sorties` n'est pas bloquante, c'est-à-dire qu'elle retourne immédiatement. La fonction `show` , qui affiche la fenêtre graphique, permet de bloquer l'exécution du programme et donc de maintenir le fonctionnement de la sortie du CNA. Lorsqu'on ferme la fenêtre, l'interface avec la centrale SP5 est fermée ce qui arrête son fonctionnement.

sortie.py

```
import pycan.main as pycan
import numpy
import math
import time
from matplotlib.pyplot import *
import numpy.fft

sys = pycan.Sysam("SP5")
sys.ouvrir()
fe = 2000.0 # frequence d'echantillonnage
f = 100.0  # frequence sinusoide
Np=1000.0 # nombre de periodes
N=int(fe/f*Np) # nombre de points

v1 = numpy.zeros(N,numpy.double)
a = 2*math.pi*Np/N
for k in range(N):
    phi = a*k
    v1[k] = 5.0*math.cos(phi)

te = 1.0/fe # periode d'echantillonnage
sys.config_sortie(1,te*10**6,v1,-1)

sys.declencher_sorties(1,0)

a = numpy.absolute(numpy.fft.fft(v1))/N
freq = numpy.arange(N)*1.0/(te*N)
figure()
plot(freq,a)
xlabel("f (Hz)")
ylabel("A")
grid()
show()
```

```
sys.fermer()
```

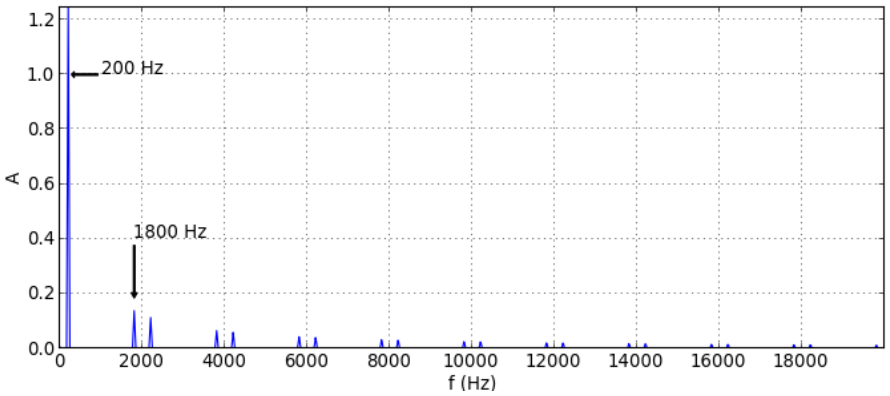
Voici la sortie du CNA (voie 1 de l'oscilloscope) pour $f = 200\ Hz$:



On voit l'effet de l'échantillonneur-bloqueur, qui maintient la tension constante entre deux échantillons consécutifs.

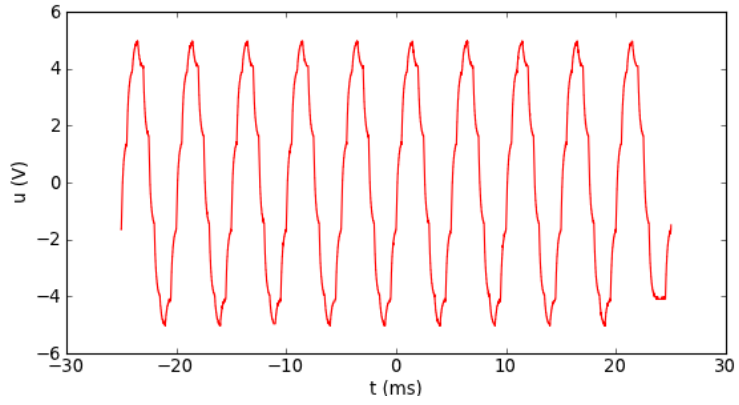
Nous allons récupérer les échantillons fournis par l'oscilloscope pour obtenir le spectre de ce signal. En pratique, on peut utiliser l'analyseur spectrale de l'oscilloscope (fonction FFT) pour obtenir ce spectre.

```
import numpy
import math
from matplotlib.pyplot import *
import numpy.fft
import scipy.signal

tableau = numpy.genfromtxt('sinus-200Hz-RC.txt',skip_header=3)
donnees = tableau.transpose()
t=donnees[0]
u=donnees[1]
N = t.size
T = (t[N-1]-t[0])*1e-3
a = numpy.absolute(numpy.fft.fft(u*scipy.signal.get_window("hann",N)))/N
f = numpy.arange(N)*1.0/T
figure(figsize=(10,4))
plot(f,a)
xlabel("f (Hz)")
ylabel("A")
axis([0,20000,0,a.max()])
xticks(numpy.arange(10)*2000)
arrow = dict(facecolor='black', width = 1, headwidth = 4, frac = 0.1, shrink=0.1)
annotate("200 Hz",[200,1.0],xytext=[1000,1.0],arrowprops=arrow)
annotate("1800 Hz",[1800,0.16],xytext=[1800,0.4],arrowprops=arrow)
grid()
```

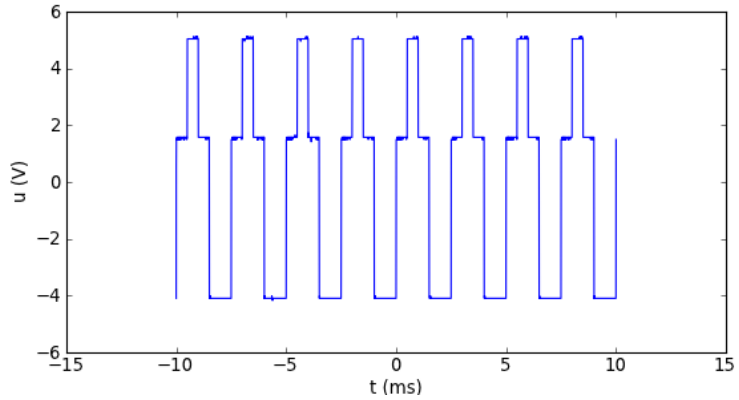


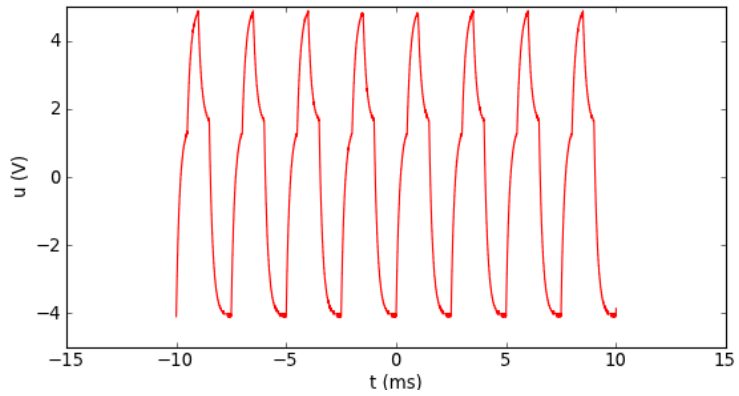
La fréquence d'échantillonnage de l'oscilloscope est ici de 200 kHz (seulement le début du spectre est représenté). La raie correspondant à la sinusoïde que l'on cherche à reconstruire se trouve tout à gauche, à 100 Hz. On voit que le spectre du signal à la sortie du CNA est tout à fait différent du spectre du signal numérique, qui ne comporte qu'une raie à 100 Hz et sa raie image. Il s'agit du résultat du blocage de la tension entre deux échantillons. La raie de la sinusoïde étant à 200 Hz, sa raie image est à $2000 - 200 = 1800$ Hz. Cette raie image est bien présente dans le spectre mais elle est atténuée. On voit par ailleurs que ces deux raies sont répliquées avec une période égale à la fréquence d'échantillonnage. Les troisième et quatrième raies sont à $2000 + 200$ Hz et $2000 + 1800$ Hz. L'amplitude de toutes ces raies suit une fonction en sinus cardinal (dont les maxima sont sur les fréquences multiples de f_e). Idéalement, le filtre de lissage ne doit laisser passer que la première raie, de manière à donner une sinusoïde. Il doit donc couper à partir de la moitié de la fréquence d'échantillonnage, soit ici 1000 Hz.

Voici la sortie du filtre pour la fréquence $f = 200$ Hz



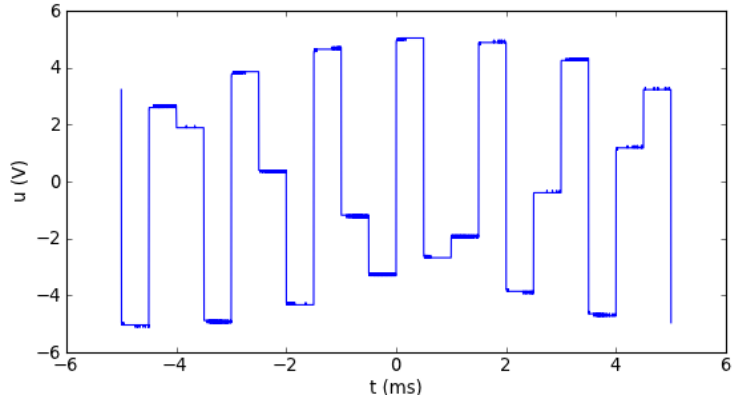
Le lissage est convenable à cette fréquence, égale au cinquième de la fréquence de Nyquist. Voyons l'entrée et la sortie du filtre à $f = 400$ Hz :

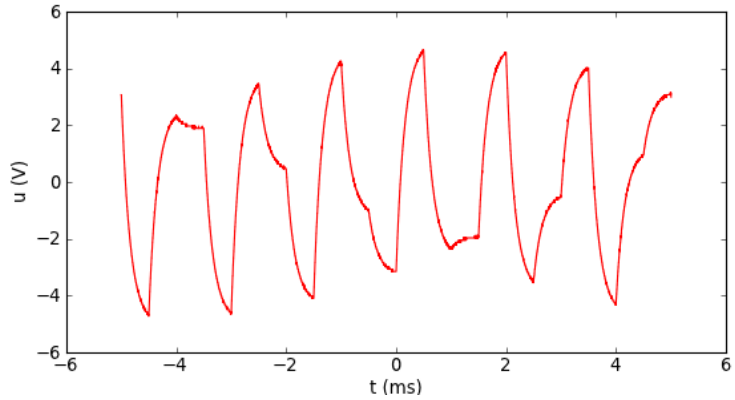




La réduction du nombre d'échantillons par période a pour conséquence une distorsion importante du signal reconstruit. On remarque que la valeur moyenne en sortie n'est pas nulle. Ce résultat médiocre est dû au fait que le filtre de lissage n'est pas assez sélectif. D'après le théorème de Shannon, la reconstruction de la sinusoïde à partir des échantillons est théoriquement possible.

Voyons le résultat à 700 Hz :

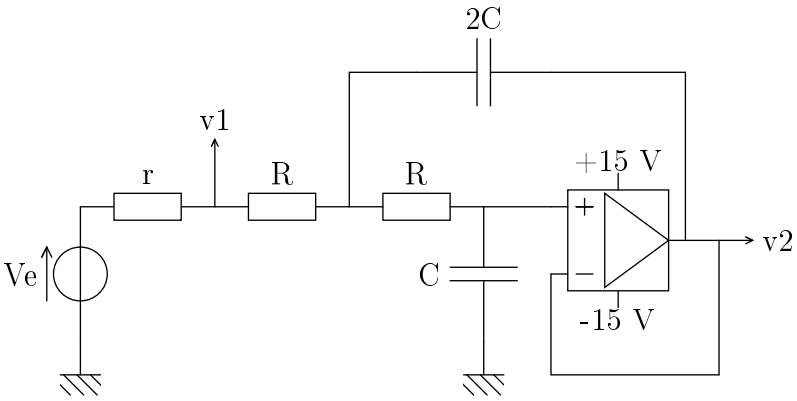




La reconstruction est mauvaise mais la périodicité est bien restituée, car la fréquence est toujours inférieure à celle de Nyquist.

2.b. Filtre du second ordre

Pour améliorer le lissage, on doit utiliser un filtre plus sélectif. Dans le montage suivant, un filtre passe-bas de Sallen et Key d'ordre 2 est utilisé :



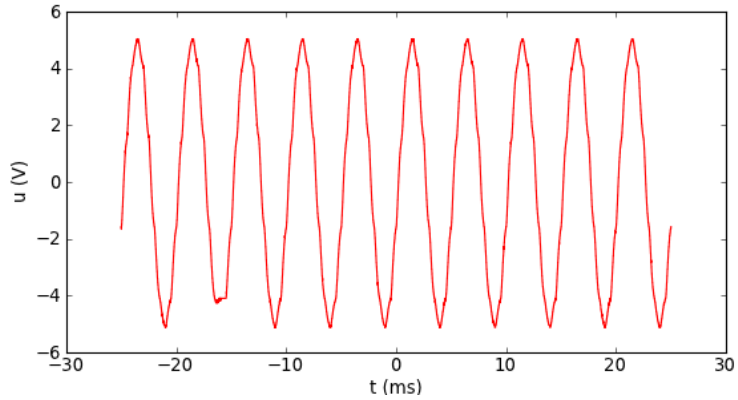
Avec $R = 10 + 1.2 = 11.2\ k\Omega$ et $C = 10\ nF$, la fréquence de coupure est :

$$f_c = \frac{1}{2\pi\sqrt{2}RC} = 1005\ Hz \tag{1}$$

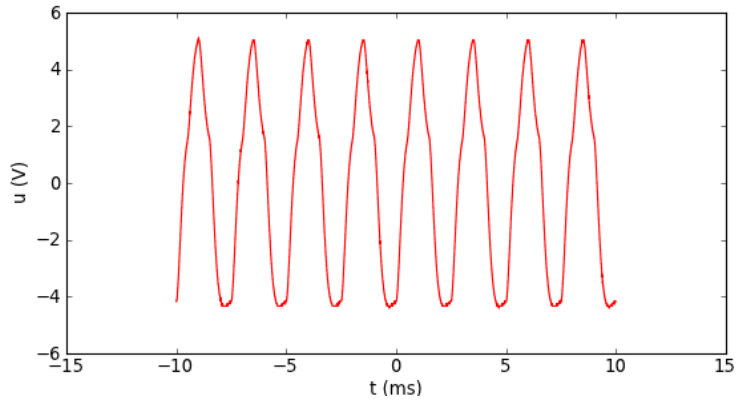
Le gain a en principe la forme suivante :

$$G(f) = \frac{1}{\sqrt{1 + \left(\frac{f}{f_c}\right)^4}} \tag{2}$$

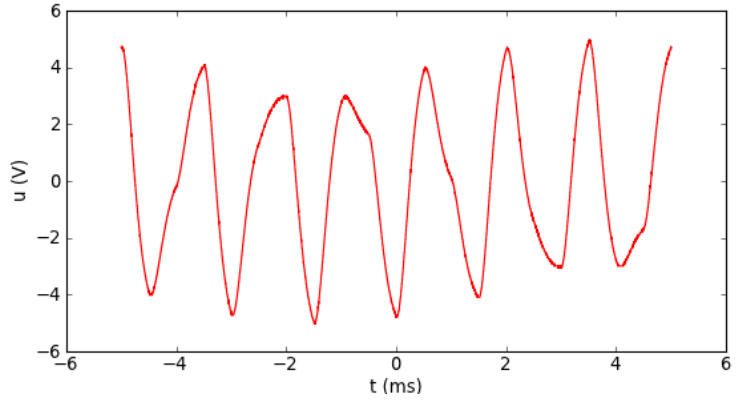
Pour comparer avec le filtre RC, on montre les résultats aux mêmes fréquences. Tout d'abord à $f = 200\ Hz$:



L'amélioration par rapport au filtre RC est sensible, mais pas spectaculaire. Voyons $f = 400\ Hz$:



Voici $f = 700\ Hz$:



Pour conclure, le filtre d'ordre 2 apporte une amélioration sensible du lissage, mais la reconstruction reste médiocre lorsqu'on s'approche de la fréquence de Nyquist. Le plus gênant est sans doute l'apparition de valeurs moyennes non nulles dans le signal de sortie. Il faut toutefois relativiser ces résultats négatifs car un signal numérique comporte en pratique assez peu de fréquences proches de celle de Nyquist. Par exemple dans un son numérisé à 44 kHz, les composantes fréquentielles entre 10 kHz et 22 kHz, qui sont des sons extrêmement aigus, sont relativement peu importantes. Néanmoins, ces composantes

interviennent dans les harmoniques de certains sons, et leur reconstruction correcte est importante pour la restitution haute-fidélité. Nous allons voir une méthode numérique beaucoup plus efficace pour reconstruire le signal.

3. Filtres de lissage numérique

La méthode consiste à augmenter la fréquence d'échantillonnage d'un facteur entier n. Il faut bien sûr que le convertisseur numérique-analogique puisse effectuer la conversion à cette nouvelle fréquence. Avec les convertisseurs modernes, cela se fait sans difficulté.

La première étape consiste à créer les nouveaux échantillons, en recopiant n fois chaque échantillon du signal d'origine. Dans un deuxième temps, il faut filtrer ces nouveaux échantillons pour effectuer un lissage numérique. Le filtre utilisé souvent pour cela procède par interpolation pour obtenir les nouveaux échantillons. Nous allons plutôt faire ce filtrage au moyen d'un filtre RIF, analogue à celui étudié dans le document Filtres passe-bas analogiques et numériques. C'est une méthode qui exige plus de calculs qu'une simple interpolation, mais qui donne une meilleure reconstruction du signal.

Notons $f'_e = nf_e$ la nouvelle fréquence d'échantillonnage. La fréquence de coupure du filtre est la moitié de $f_e/2$. Le rapport de la fréquence de coupure sur la nouvelle fréquence d'échantillonnage est donc :

$$a = \frac{f_c}{f'_e} = \frac{0.5}{n} \tag{3}$$

Pour calculer la réponse impulsionnelle du filtre, nous utilisons la fonction `scipy.signal.firwin`, qui permet aussi d'appliquer un fenêtrage progressif pour réduire les ondulations de la réponse fréquentielle.

Voici le programme python qui effectue l'échantillonnage de la sinusoïde à $f_e = 2000\ Hz$ (comme plus haut) puis augmente la fréquence d'échantillonnage d'un facteur $n = 5$ avant d'appliquer le filtre d'interpolation. L'indice de troncature P n'a pas besoin d'être très grand pour ce filtrage. On prend $P = 10$. Enfin les nouveaux échantillons sont envoyés au CNA à la fréquence $f'_e = 10\ kHz$.

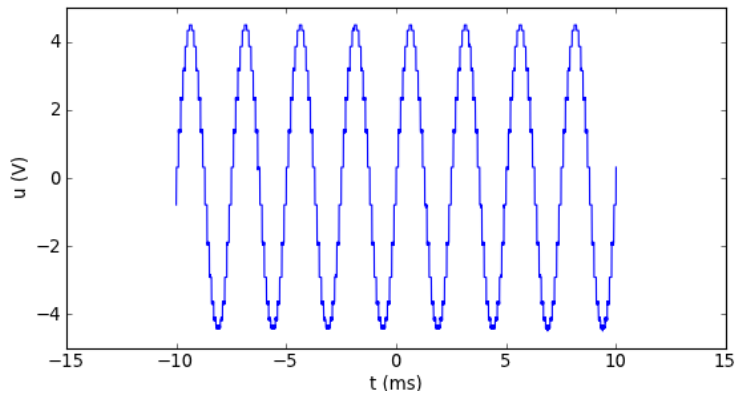
sortieInterpolation.py

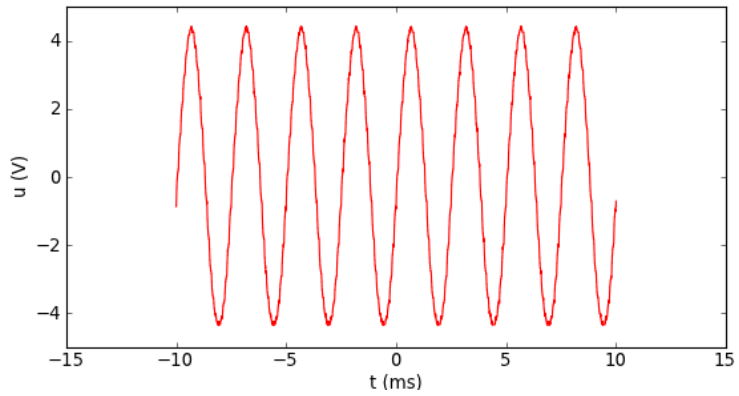
```
import pycan.main as pycan
import numpy
import math
import time
import scipy.signal
from matplotlib.pyplot import *
import numpy.fft


sys = pycan.Sysam("SP5")
sys.ouvrir()

fe = 2000.0 # frequence d'echantillonnage initiale
f = 100.0  # frequence sinusoide
Np=1000.0 # nombre de periodes
N=int(fe/f*Np) # nombre de points
```

```
print(N)
v1 = numpy.zeros(N,numpy.double)
a = 2*math.pi*Np/N
for k in range(N):
    phi = a*k
    v1[k] = 5.0*math.cos(phi)

n=5 # on augmente fe d'un facteur 5
v2 = numpy.zeros(0)
i=0
while i<v1.size:
    for j in range(n):
        v2 = numpy.append(v2,v1[i])
    i += 1

P = 10
#filtre d'interpolation
h = scipy.signal.firwin(numtaps=2*P+1,cutoff=[0.5/n],nyq=0.5,window='hann')
v3 = scipy.signal.convolve(v2,h)

fe = fe*n # nouvelle frequence d'echantillonnage
te = 1.0/fe
sys.config_sortie(1,te*10**6,v3,-1)

sys.declencher_sorties(1,0)

a = numpy.absolute(numpy.fft.fft(v3))/N
freq = numpy.arange(N)*1.0/(te*N)
figure()
plot(freq,a)
xlabel("f (Hz)")
ylabel("A")
grid()
show()
sys.fermer()
```

Un filtre analogique de lissage est placé en sortie du CNA. Un simple filtre RC est ici suffisant. On prend $R = 15\ k\Omega$ et $C = 2.2\ nF$, ce qui donne une fréquence de coupure de $4.8\ kHz$. Comme cette fréquence est grande devant les fréquences du signal à reconstruire (lesquelles sont inférieures à $1000\ Hz$), le filtre amènera très peu de distorsion.

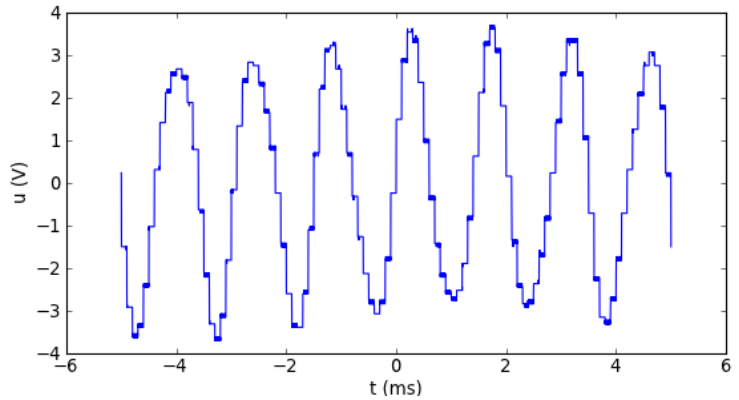
Voici le résultat pout $f = 400\ Hz$:

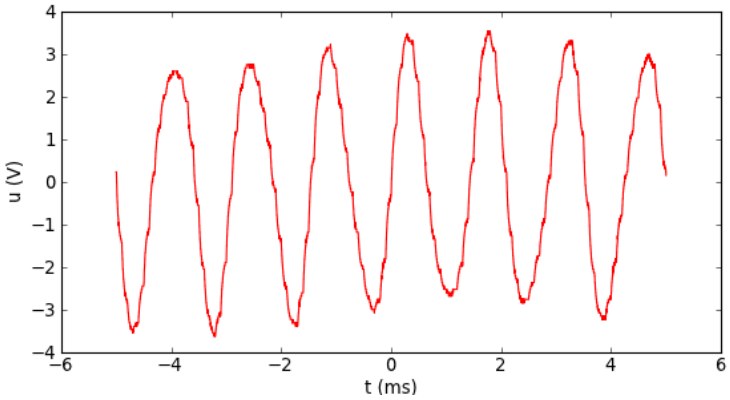




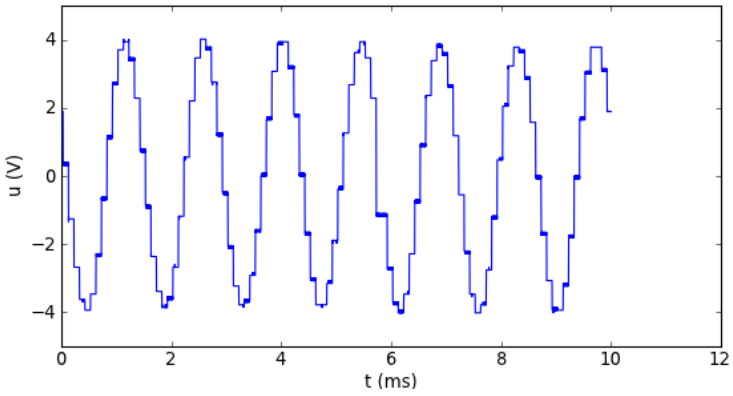
Comme le nombre d'échantillons a été multiplié par 5, le résultat est bien meilleur qu'avec le filtre analogique seul.

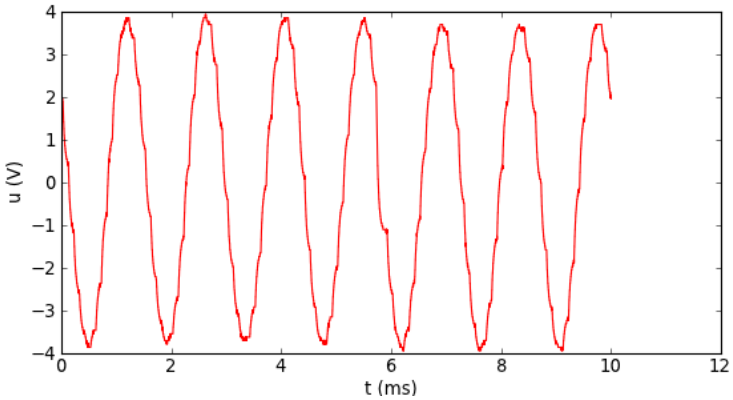
Voici le résultat pour $f = 700\ Hz$:





Il reste des ondulations, auquelles on remédie facilement en augmentant l'indice de troncature. Voici la même fréquence avec $P = 20$:





Conclusion : la méthode de l'augmentation de la fréquence d'échantillonnage suivie d'un filtrage d'interpolation est bien plus efficace que le lissage analogique seul. C'est d'ailleurs la méthode utilisée pour la conversion numérique-analogique du son.