Chapitre II

Traitement numérique des signaux

Sommaire

I Echantillonnage - Conversion analogique numérique (CAN)

I.1 Principe général

Le stockage des informations par un ordinateur se fait sous forme numérique (binaire) ; en outre l'enregistrement des valeurs prises par une fonction sera fatalement "**discontinu**", l'ordinateur ne pouvant enregistrer que des valeurs discrètes (cf cours IPT `Représentation des nombres`) ;

QUESTION : comment alors stocker un $\underbrace{\text{signal analogique}}_{\text{i.e. continu}}$ $f(t)$ en $\underbrace{\text{mémoire informatique}}_{\text{i.e. par valeurs discrètes}}$?

RÉPONSE : on a recours un "échantillonneur", suivi d'un CAN ou **C**onvertisseur **A**nalogique **N**umérique qui réalise le stockage du signal échantillonné sous forme numérique.

PRINCIPE : on prélève et on enregistre les valeurs de $f(t)$ tous les $T_e = \dfrac{1}{f_e}$, avec T_e appelée période d'échantillonnage (f_e "fréquence d'échantillonnage"). La machine enregistre donc en mémoire la suite des nombres suivants : $[f(0), f(T_e), f(2T_e), \ldots, f(nT_e), \ldots]$

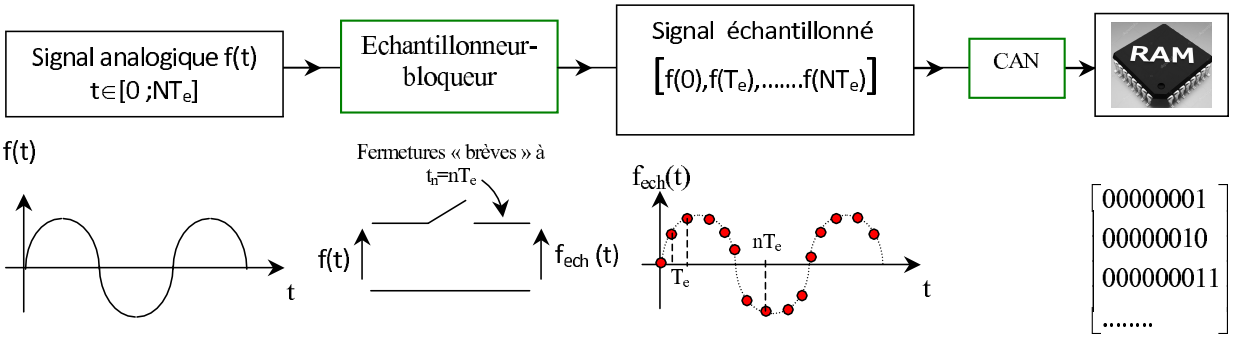

FIGURE II.1 – Principe de l'échantillonnage

Le signal échantillonné peut-être représenté par une fonction mathématique $f_{ech}(t)$ construite de la manière suivante :

- **pour l'échantillon $f_{ech}(nT_e)$ pris seul à la date $t_n = nT_e$ la valeur est donnée par le produit suivant :**

$$f(nT_e) \times \delta(t - nT_e) \qquad \text{avec } \delta(t - nT_e) \text{ pic de Dirac d'amplitude 1 centré sur la date } nT_e$$

- **pour un signal continue $f(t)$ ($t \in]-\infty;+\infty[$), la totalité du signal échantillonné $f_{ech}(t)$ est donnée par la sommation suivante :**

$$f_{ech}(t) = \sum_{n=-\infty}^{+\infty} f(nT_e) \times \delta(t - nT_e) \overset{f_{ech}(t \neq nT_e)=0}{=} f(t) \times \underbrace{\sum_{n=-\infty}^{+\infty} \delta(t - nT_e)}_{\text{peigne de Dirac période } T_e}$$

L'opération d'échantillonnage peut donc être schématisée ainsi :

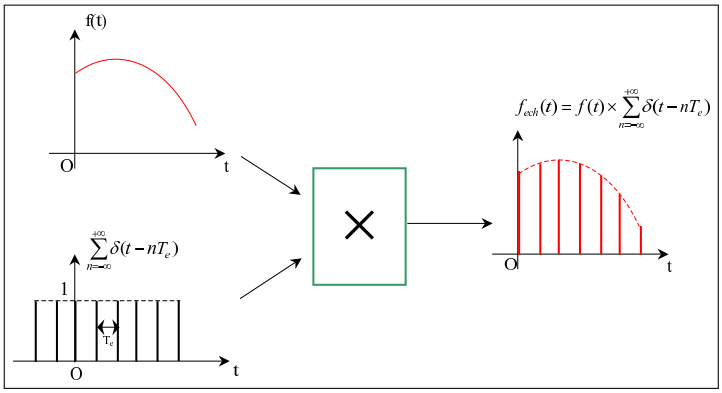

FIGURE II.2 – Schema de principe de l'échantillonnage

I.2 Quelques détails sur le codage par le CAN

Le CAN permet donc, une fois l'échantillonnage effectué, de numériser chaque valeur $f_{ech}(pT_e)$, c'est à dire de la convertir en un nombre binaire enregistrable dans une mémoire informatique.

a - Pas de quantification du CAN - profondeur de codage -précision

HYPOTHÈSE :

- le signal échantillonné $f_{ech}(pT_e)$ est compris entre $f_{ech_{min}}$ et $f_{ech_{max}}$
- le CAN peut coder N nombres différents dans l'intervalle $[f_{ech_{min}}, f_{ech_{max}}]$; ce nombre est naturellement une puissance de 2 puisque le stockage en puce mémoire est réalisé en binaire : $N = 2^p$

A RETENIR : (HORS PROGRAMME CEPENDANT !)

DÉFINITION - (I.2) - 1:

On appelle pas de quantification q l'incertitude sur chaque valeur numérisée du signal échantillonné f_{ech} : $q = \dfrac{f_{ech_{max}} - f_{ech_{min}}}{2^p - 1}$

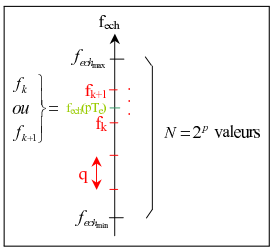

FIGURE II.3 – Pas de quantification

CONSÉQUENCE : la quantification engendre une perte d'information d'autant plus grande que le pas q est important ⇒ on veut q petit, donc on prend p élevé.

Exemple : dans le cas du codage CD on prends $p = 16$, soit $N = 2^{16} = 65536$ valeurs possibles.

b - Exemple de CAN : le CAN simple rampe

La finalité du CAN est de stocker sous forme binaire le signal échantillonné. On propose ci-dessous d'aborder dans les grandes lignes le CAN simple rampe qui est l'un des plus simples.

L'opération de conversion est réalisée en deux étapes :

- ECHANTILLONNAGE-BLOCAGE PAR CAPACITÉ :

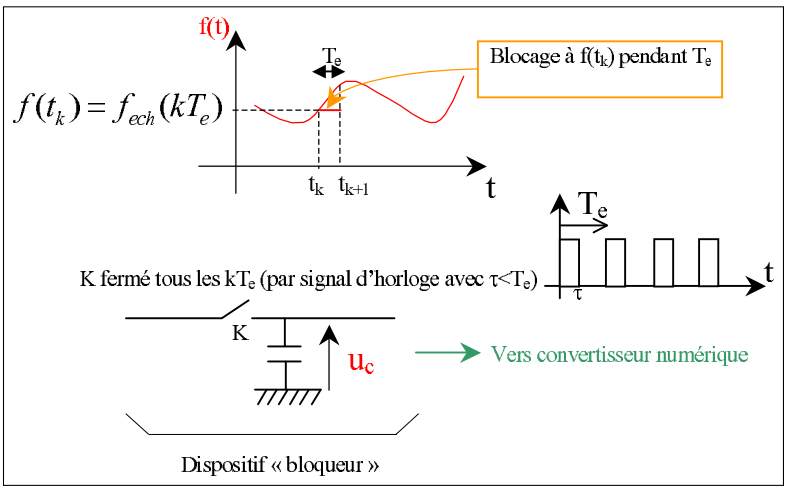

FIGURE II.4 – Principe échantillonneur-Bloqueur

Le signal d'horloge "pilote" le comportement de l'interrupteur K :
– à t_k K fermé $\rightarrow U_c(t_k) = f(t_k)$ échantillonnage
– à $t_k + \tau$ K ouvert $\rightarrow U_c(t \in [t_k, t_{k+1}]) = f(t_k)$ blocage

- CONVERSION NUMÉRIQUE PAR SIMPLE RAMPE :

HYPOTHÈSE : pour simplifier, on prendra un signal $f(t)$ évoluant entre 0 et $E > 0$. La conversion exploite ici une simple rampe évoluant entre 0 et E et de période T_e **synchrone de l'horloge de l'échantillonneur-bloqueur** :

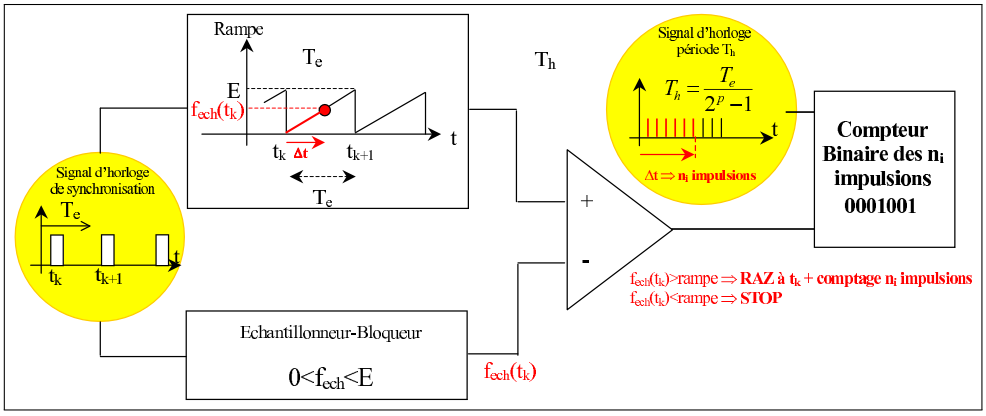

FIGURE II.5 – Conversion numérique simple rampe

PRINCIPE :

- A t_k : [l'échantillonneur-bloqueur fournit $f_{ech}(t_k)$; la rampe débute ; RAZ du compteur binaire]
- pour $t \in [t_k; t_k+\Delta t]$ on a $rampe < f_{ech}(t_k)$: [le comparateur est en état haut +1 ; le compteur compte les impulsions de l'horloge de période T_h ; RAZ du compteur binaire]
- A $t_k+\Delta t^+$ on a $rampe > f_{ech}(t_k)$: [le comparateur passe en état bas $\Rightarrow$ STOP du compteur ; le compteur affiche un codage binaire de $f_{ech}(t_k)$: $f_{ech}(t_k) = E\dfrac{n_i}{2^p-1}$]

I.3 Analyse spectrale d'un signal échantillonné

a - Rappel : spectre d'un peigne de Dirac

Le peigne de Dirac correspond ici à un train d'impulsion de largeur tendant vers 0 ($\Delta t \to 0$) de période T_e (période d'échantillonnage) et d'amplitude E (on rappelle la contrainte $E \cdot \Delta t = 1$:

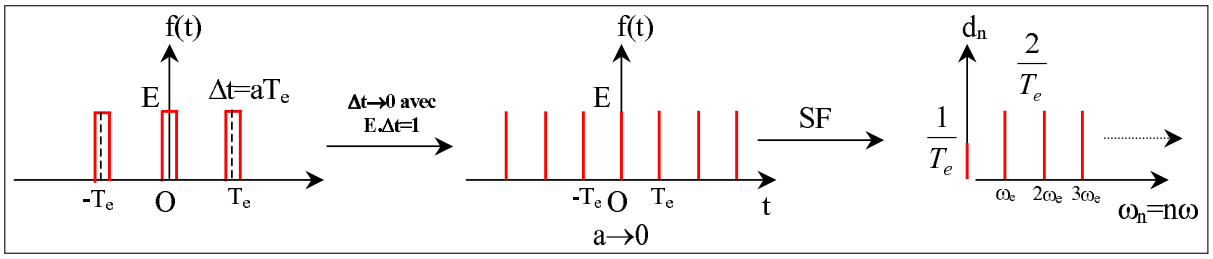

FIGURE II.6 – Série de Fourier d'un Peigne de Dirac ($a \to 0$)

On rappelle les coefficients de Fourier d'un tel signal : $\begin{cases} d_0 = |c_0| \overset{|sinc(na\pi)|\to 1\ \forall n}{\longrightarrow} Ea = E\dfrac{\Delta t}{T_e} = \dfrac{1}{T_e} \\ d_{n\geq 1} = |2\underline{c_n}| \overset{|sinc(na\pi)|\to 1\ \forall n}{\longrightarrow} 2E\dfrac{\Delta t}{T_e} = \dfrac{2}{T_e} \end{cases}$

Ainsi, la SF du peigne de Dirac de période T_e est : $\boxed{\sum_{n=-\infty}^{+\infty} \delta(t-nT_e) = \frac{1}{T_e} + \sum_{n=1}^{n=+\infty} \frac{2}{T_e} \cdot \cos(2\pi n \frac{t}{T_e})}$

b - Spectre d'un signal échantillonné

- CAS D'UN SIGNAL SINUSOÏDAL PUR
  Considérons dans un premier temps un signal sinusoïdal pur de fréquence f_0 : $f(t) = A\cos(2\pi f_0 t)$ que l'on échantillonne à T_e. Le signal échantillonné s'écrit :

$$f_{ech}(t) = A\cos(2\pi f_0 t) \times [\text{Peigne Dirac}] = A\cos(2\pi f_0 t) \times \left[\frac{1}{T_e} + \sum_{n=1}^{n=+\infty} \frac{2}{T_e} \cdot \cos(2\pi n F_e t)\right]$$

soit :

$$f_{ech}(t) = \frac{A}{T_e}\left[\cos(2\pi f_0 t) + \sum_{n=1}^{n=+\infty} \cos\left(2\pi(nF_e - f_0)t\right) + \cos\left(2\pi(nF_e + f_0)t\right)\right]$$

Le spectre de ce signal est donc :

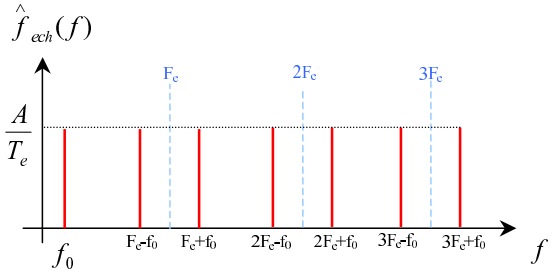

FIGURE II.7 – Spectre du signal sinusoïdal échantillonné

- CAS D'UN SIGNAL QUELCONQUE
  Considérons maintenant un signal quelconque $s(t)$; si ce dernier est périodique son spectre est obtenu par décomposition en SF, et s'il est apériodique, par TF. Il est en tout cas une somme de signaux sinusoïdaux. Le spectre $\hat{s}(f)$ de $s(t)$ peut par exemple avoir l'allure suivante :

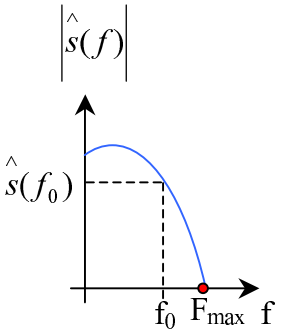

FIGURE II.8 – Spectre du signal $s(t)$ non périodique (seules les fréquences positive de la TF sont représentées)

NB : F_{max} représente la fréquence maximale présente dans le spectre. Lorsqu'il s'agit d'un signal périodique, l'étendue spectrale étant infinie, on retiendra pour F_{max} la fréquence pour laquelle l'amplitude de l'harmonique concernée est si faible que son "poids" dans le signal n'est plus représentatif[1].

On peut par exemple isoler la composante de fréquence f_0, d'amplitude $|\hat{s}(f_0)|$; d'après la démarche ci-dessus, le spectre de cette composante une fois échantillonnée contient les fréquences $nF_e \pm f_0$. En reproduisant ce raisonnement pour l'ensemble des composantes spectrales du signal $s(t)$, on obtient le spectre $\hat{s}_{ech}(f)$ représenté ci-dessous, qui reproduit donc le motif spectral tous les nF_e :

1. pour un signal créneau ($\sim \frac{1}{n}$), abandonner le rang 20 est suffisant ; pour un signal triangulaire ($\sim \frac{1}{n^2}$), ce sera le rang 5

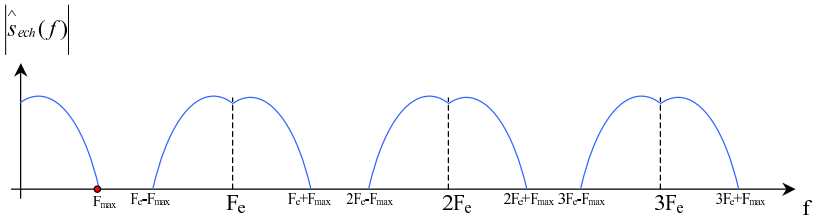

FIGURE II.9 – Spectre $\hat{s}_{ech}(f)$ du signal échantillonné $s_{ech}(t)$

c - Reconstitution d'un signal à partir de son échantillonnage (technique passe-bas) - théorème de Nyquist-Shannon

QUESTION : comment obtenir le spectre $\hat{s}(\omega)$ du signal $s(t)$ à partir du signal échantillonné ?

RÉPONSE : On peut retrouver le spectre $\hat{s}(\omega)$ de $s(t)$ **en réalisant un filtrage passe-bas** du signal échantillonné :

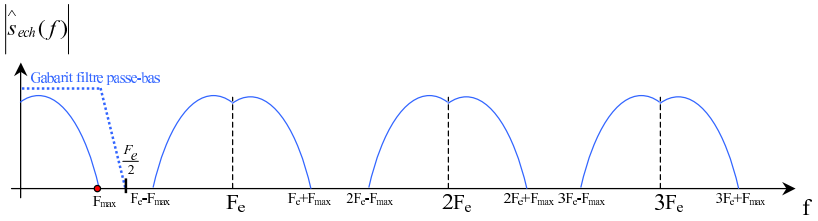

FIGURE II.10 – Filtrage passe-bas du signal échantillonné $s_{ech}(t)$

THÉORÈME DE NYQUIST-SHANNON : pour obtenir le spectre correct après filtrage, il faut s'assurer qu'aucune portion du "massif spectral" centré sur F_e ne pénètre dans le "massif spectral" centré sur 0 qui doit être le seul restitué par filtrage ; la condition de non chevauchement s'écrit naturellement :

$$F_{Max} < F_e - F_{Max} \qquad soit \qquad \boxed{2F_{Max} < F_e}$$

On en déduit le **théorème de Nyquist-Shannon imposant des conditions pour réaliser un échantillonnage correct** :

PROPRIÉTÉ - (I.3) - 1:

La reconstitution parfaite, *i.e. sans perte d'information (par synthèse de Fourier), d'un signal à partir de son échantillonnage à la fréquence F_e n'est possible qu'à condition qu' F_e soit au minimum deux fois supérieure à la plus haute fréquence présente dans le spectre du signal, ou bien la plus haute fréquence que l'on retiendra comme significative dans le spectre :*

$$\text{Echantillonnage correct} \Leftrightarrow Fe > 2F_{Max} \quad soit \quad F_{Max} < \frac{F_e}{2} = F_{Ny}$$

en posant $F_{Ny} = \dfrac{F_e}{2}$ **fréquence de Nyquist**.

NB : parfois, on fixe la fréquence de coupure du filtre f_c à $F_{Ny} = \dfrac{F_e}{2}$

EXEMPLE : cas du CD-AUDIO

Pour un CD-AUDIO, on a $\nu_e = 44,1\ kHz \simeq 44\ kHz$ et le spectre des audio-fréquences $\nu \in [20\ Hz\ 20\ kHz]$; l'échantillonnage est donc correct pour une restitution ultérieure du signal :

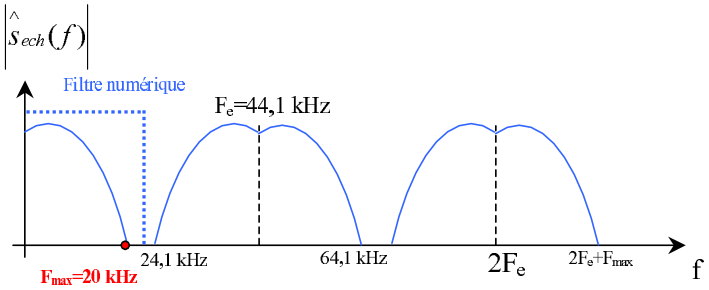

FIGURE II.11 – Echantillonnage à la norme CD-audio

d - Non respect du théorème de Nyquist-Shannon : le repliement de spectre et les fréquences "fantômes" (Expérience de cours/Simulation Python)

EXPÉRIENCE DE COURS : illustration du repliement sur un signal sinusoïdal $f_0 = 1\ kHz$.

Reprenons le cas simple de l'échantillonnage d'un signal sinusoïdal et fixons sa fréquence à $f_0 = 1kHz$.

Deux situations peuvent présenter :

- La fréquence d'échantillonnage est $F_e > 2F_{Max} = 2kHz$ par exemple $F_e = 3kHz \implies$ **l'échantillonnage est correct (condition de Nyquist-Shannon respectée)**
- La fréquence d'échantillonnage est $F_e < 2F_{Max} = 2kHz$ par exemple $F_e = 1,5kHz \implies$ **l'échantillonnage est incorrect ! (condition de Nyquist-Shannon violée)**

  Dans ce second cas, le spectre du signal échantillonné est le suivant :
  Il présente une raie dans la zone filtrée $\left[0, F_{Ny} = \frac{F_e}{2} = 0,75kHz\right]$ à la fréquence $F_e - f_0 = 1,5kHz - 1kHz = 0,5kHz$. Cette raie n'existe évidemment pas dans le signal originel, et est symétrique de la raie du signal à f_0 par rapport à la fréquence de Nyquist ; en effet, la fréquence médiane entre cette raie fantôme et la fréquence f_0 est :

$$\frac{(F_e - f_0) + f_0}{2} = \frac{F_e}{2} = F_{Ny}$$

A retenir : Si $F_e < 2f_0$ alors il apparait $f_{\text{fantôme}}$ symétrique de f_0 par rapport à $F_{Ny} = \dfrac{F_e}{2}$:

On parle de **repliement de spectre** par rapport à la fréquence de Nyquist.

NB : dans le cas d'un signal non périodique (spectre continu), la situation de repliement correspond à l'allure spectrale suivante :

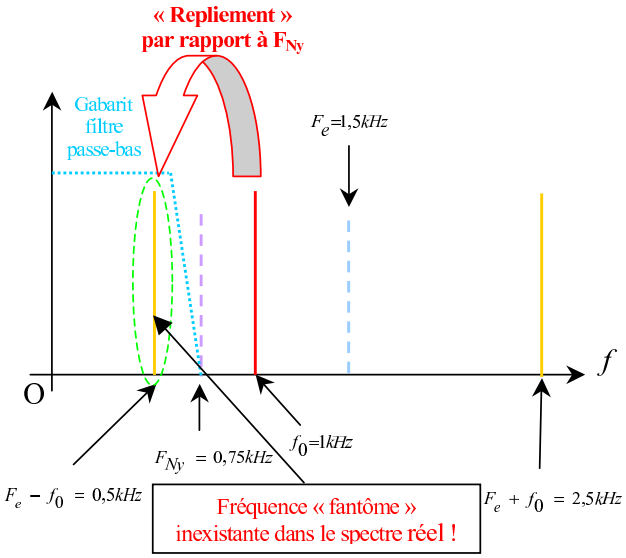

FIGURE II.12 – Phénomène de repliement de spectre pour un signal sinusoïdal

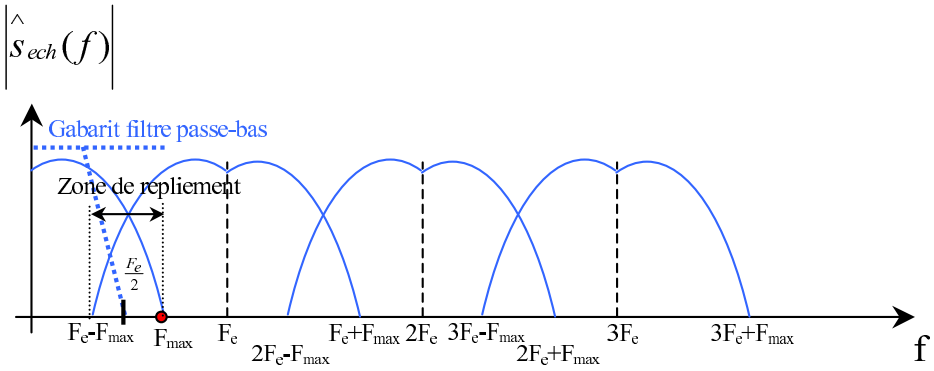

FIGURE II.13 – Phénomène de repliement de spectre

REMARQUE - (I.3) - 1:

Pour les signaux périodiques dont le spectre est non borné supérieurement, on procède généralement à un filtrage passe-bas avant échantillonnage appelé FILTRAGE ANTI-REPLIEMENT. Ceci permet de poser *arbitrairement* une borne supérieure au spectre, i.e. de définir F_{Max} afin d'assurer le respect rigoureux de la condition de Nyquist-Shannon lors du choix de la fréquence d'échantillonnage.

II Notions de base sur le filtrage numérique des signaux

II.1 Principe

La majorité des dispositifs modernes de traitement et analyse de signal, notamment en Hifi (élimination du bruit, enrichissement sélectif du spectre graves-médiums-aigües, effets de réverbération etc...) sont des dispositifs numériques travaillant donc sur des signaux préalablement échantillonnés. Ceci permet une latitude totale dans la

nature des traitements que l'on peut faire subir aux signaux puisqu'ils sont réalisées par des algorithmes programmés sur ordinateur.

Deux opérations de base du traitement numérique du signal en Hifi sont les filtrages passe-haut (élimination du bruit très basse fréquence type "Rumble"), et passe-bas (élimination de sifflements haute fréquence).

Question : quel est le principe de filtrage d'un signal échantillonné ?

2 méthodes : [temporel / spectral/fréquentiel

La suite expose les méthodes de filtrage numériques temporel et fréquentiel dont les principes sont résumés dans les synopsis ci-dessous :

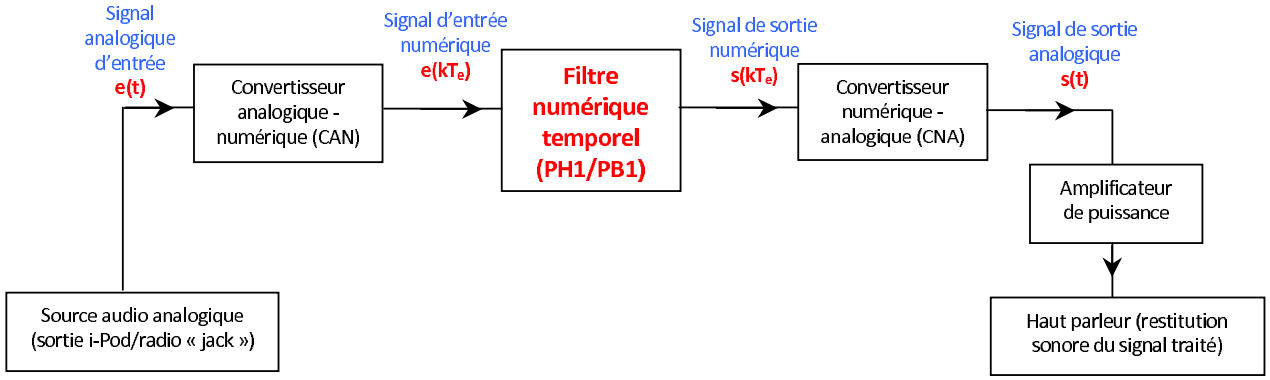

FIGURE II.14 – Chaine complète de filtrage numérique temporel d'un signal audio

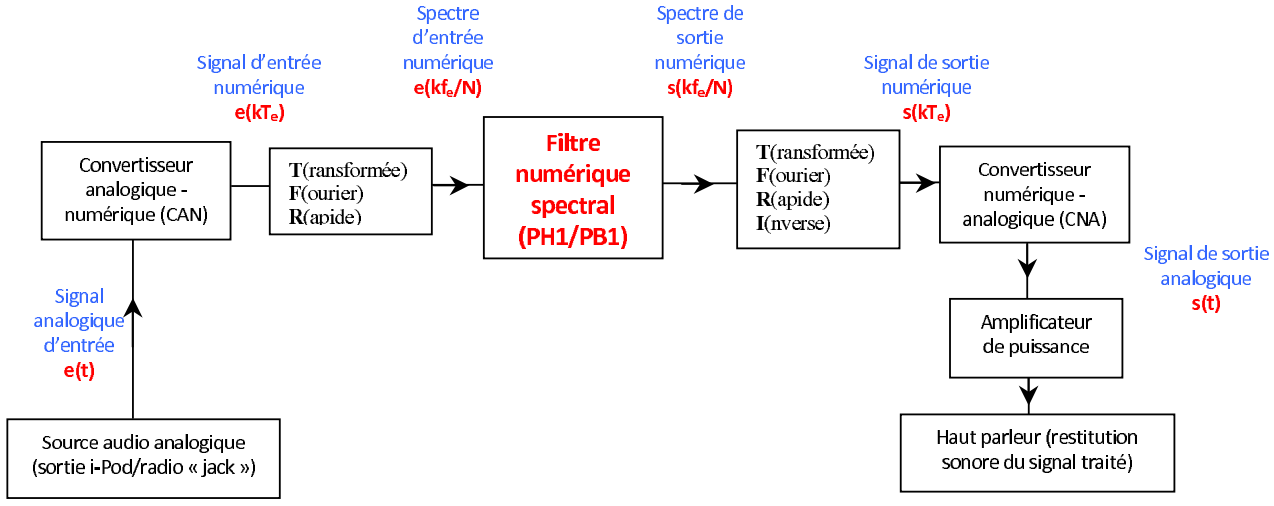

FIGURE II.15 – Chaine complète de filtrage numérique frequentiel d'un signal audio

II.2 Mise en oeuvre

a - Filtrage numérique temporel (Expérience de cours/Simulation Python)

Le signal à traiter présente deux composantes sinusoidales l'une BF à $f_b = 100\ Hz$ et l'autre HF à $f_h = 4\ kHz$:

$$e(t) = U_{0_b}\cos(2\pi f_b t) + U_{0_h}\cos(2\pi f_h t)$$

On se propose par exemple de réaliser l'élimination de la composante HF par filtrage numérique temporel passe-bas. Les différentes étapes que vous exécuterez en TP sont les suivantes :

- Acquisition du signal par conversion analogique-numérique (CAN) avec la fréquence d'échantillonnage $f_e = \dfrac{1}{T_e} =\geq 100\ kHz > 2f_h$ (cf th. de Shannon) avec LatisPro $\longrightarrow e(kT_e) \quad (k \in \mathbb{N})$.
- Enregistrement du signal échantillonné au format .csv (séparation des données par une virgule), sous forme de couples $(kT_e, e(kT_e))$
- Lecture du fichier .csv et **Filtrage numérique temporel** par un programme Python du signal échantillonné $e(kT_e)$, par exemple **passe-bas 1er ordre** $e(kT_e) \overset{\text{passe bas}}{\Longrightarrow} s(kT_e)$
- Enregistrement par python du signal filtré $s(kT_e)$ au format .csv
- Ouverture du fichier signal filtré par LatisPro
- Conversion numérique-analogique (CNA) du signal $s(kT_e) \longrightarrow s(t)$ et restitution par HP.

■ CALCUL DE LA "FORMULE" DE FILTRAGE DU PASSE-BAS $e(kT_e) \overset{\text{PASSE BAS}}{\Longrightarrow} s(kT_e)$

La fonction de transfert d'un passe bas 1er ordre passif de pulsation de coupure ω_c est de forme :

$$H(j\omega) = \frac{1}{1 + j\dfrac{\omega}{\omega_c}}$$

dont l'équation différentielle correspondante est :

$$\frac{ds(t)}{dt} + \omega_c \cdot s(t) = \omega_c \cdot e(t)$$

On intègre cette équation différentielle entre les instants "discrets" kT_e et $(k+1)T_e$ et l'on réordonne les deux membres pour obtenir :

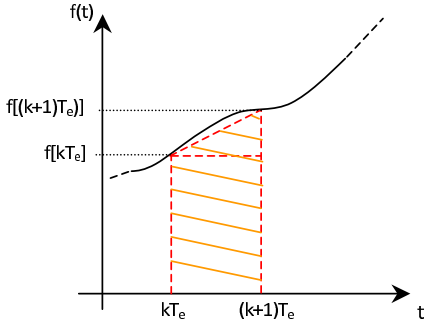

FIGURE II.16 – Intégration par méthode des trapèzes

$$s(k+1) = +s(k) + \omega_c \underbrace{\int_{kT_e}^{(k+1)T_e} [e(t) - s(t)] \cdot dt}_{\text{aire sous la courbe} \Rightarrow \text{méthode des trapèzes}} \tag{II.1}$$

or l'intégrale entre deux instants kT_e et $(k+1)T_e$ d'une fonction $f(t)$ peut être approchée par méthode des trapèzes avec :

$$\int_{kT_e}^{(k+1)T_e} f(t)\cdot dt = \overbrace{T_e\cdot f(kT_e)}^{\text{aire du rectangle inférieur}} + \overbrace{\frac{T_e}{2}\left[f[(k+1)T_e]-f(kT_e)\right]}^{\text{aire du triangle supérieur}} = \frac{T_e}{2}\left[f[(k+1)T_e]+f(kT_e)\right]$$

Compte tenu de ceci, l'équation II.1 devient :

$$s(k+1) = s(k) + \frac{\omega_c T_e}{2}\left[e(k)+e(k+1)-s(k)-s(k+1)\right]$$

soit finalement : $\boxed{s(k+1) = \frac{2-\omega_c T_e}{2+\omega_c T_e}\cdot s(k) + \frac{\omega_c T_e}{2+\omega_c T_e}\left[e(k)+e(k+1)\right]}$

Cette dernière relation de récurrence permet d'obtenir le signal numérique (échantillonné) de sortie du filtre.

Le code Python complet sera commenté "en live" :

Script Python Filtrage numérique passe-bas et passe haut 1er ordre

```
tiny
# -*- coding: utf-8 -*-
from math import *
import numpy as np
import matplotlib.pyplot as plt
#ouverture du fichier de donnees source
mesdonnees=open('Donnees_formatees_CSV.csv','r')
#Donnees numeriques
Te=0.00001 #PÃ©riode d'Ã©chantillonnage
wc=2*np.pi*200.0 #Pulsation de coupure des passe-bas et passe-haut
#Tableaux vierges pour signal entree pour enregistrement de 4096 donnees
t=np.zeros(4096)
e=np.zeros(4096)
#Lecture en-tete du fichier de donnees
entete=mesdonnees.readline().rstrip('\n\r').split(',')
#Initialisation d'un compteur et du max d'Ã©chelle graphique
k,max=(0,0)
#Boucle de construction de e par lecture du fichier source
for Ligne in mesdonnees:
    tL,eL=Ligne.rstrip('\n\r').split(",") #Lit la ligne, supprime espace
#et retour chariot, coupe a la virgule et stocke le temps et la valeur signal
    t[k]=float(tL) # affectation de la kiÃ¨me valeur du temps
    e[k]=float(eL) # affectation de la kiÃ¨me valeur du signal
    if e[k]>max:
        max=1.05*e[k] #Max est une variable d'ajustement automatique de l'echelle
    k+=1
#Fermeture du fichier source
mesdonnees.close()
#Tableaux vierges pour signal sortie pour enregistrement 4096 donnees
spb1=np.zeros(4096)
sph1=np.zeros(4096)
#ouverture des fichiers de donnees sortie
masortie_PB1=open('Donnee_sortie_PB1_CSV.csv','w')
masortie_PH1=open('Donnee_sortie_PH1_CSV.csv','w')
#Ecriture des signaux de sortie des filtres par rÃ©currence
for k in range(4095):
    spb1[k+1]=((2-wc*Te)/(2+wc*Te))*spb1[k]+((wc*Te)/(2+wc*Te))*(e[k]+e[k+1])#PB
    sph1[k+1]=((2-wc*Te)/(2+wc*Te))*sph1[k]+(2/(2+wc*Te))*(e[k+1]-e[k])#PH
    masortie_PB1.write(str(t[k])+','+str(spb1[k])+'\n') #Ecriture de ligne dans le fichier de sortie Passe
     bas1 aprÃ¨s conversion en chaine de caractere
    masortie_PH1.write(str(t[k])+','+str(sph1[k])+'\n') #Ecriture de ligne dans le fichier de sortie passe
     haut1 aprÃ¨s conversion en chaine de caractere
#Fermeture des fichiers de sortie passe-bas1 et passe-haut1
masortie_PB1.close()
masortie_PH1.close()
#Trace evolution des tensions entree et sortie
plt.grid()
plt.xlabel(r'$t(s)$', fontsize=10)
plt.ylabel(r'$s(t)\ (en\ V)$', fontsize=10, rotation=0)
plt.title(r"Tension de sortie du filtre",size=20)
plt.axis([t[2048],t[4095],-max,max])
#entree=plt.plot(t,e)
sortie_PB1=plt.plot(t,spb1)
#sortie_PH1=plt.plot(t,sph1)
plt.show()
```

Listing II.1 – Sources_Python/Filtrage_numerique_version_2_boucles_pour_compil_latex.py

NB : le fichier Python est disponible sur le site MP3, et libre de modification/amélioration si le coeur vous en dit.

■ Calcul de la "formule" de filtrage du passe-haut

A faire en live à titre d'exercice.

b - Filtrage numérique spectral

Même exemple mais cette fois **par la technique fréquentielle** $\Longrightarrow$ réalisation en TP à l'aide d'un programme python fourni.

PRINCIPE :
On calcule l'amplitude et la phase de chaque composante du signal de sortie à partir de celles du signal d'entrée et de la fonction de transfert du filtre ;

$$\underline{s}\left(k\frac{F_e}{N}\right) = H\left(j2\pi k\frac{F_e}{N}\right)\cdot\underline{e}\left(k\frac{F_e}{N}\right) \quad\Longrightarrow\quad \left[\begin{array}{l} \left|s\left(k\frac{F_e}{N}\right)\right| = G\left(k\frac{F_e}{N}\right)\times\left|e\left(k\frac{F_e}{N}\right)\right| \\ \\ \varphi_s\left(k\frac{F_e}{N}\right) = \varphi_e\left(k\frac{F_e}{N}\right) + arg\left[H\left(j2\pi k\frac{F_e}{N}\right)\right] \end{array}\right.$$

avec N nombre total d'échantillons prélevés et $k \in \left[0, \lfloor\frac{N}{2}\rfloor\right]$ (afin d'assurer $f \in \left[0, F_{Ny} = \frac{F_e}{2}\right]$).

III Complément TP : notions de fenêtrage

III.1 Principe

L'analyse de Fourier numérique d'un signal échantillonné est généralement réalisée par implémentation d'un algorithme de transformée de fourier rapide ou FFT (Fast Fourier Transform) dans les logiciels de traitement (LatisPro), ou les oscilloscopes numériques (Tektronix). Cette analyse est fatalement réalisée sur une portion limitée du signal ; cette troncature du signal entraîne l'introduction de composantes spectrales inattendues ; on propose d'exposer schématiquement le principe de fenêtrage qui limite ces effets.

EXEMPLE : acquisition numérique d'un signal sinusoïdal, donc tronqué : $\begin{cases} f(t) = A\cdot cos(\omega t) & |t| < \dfrac{\tau}{2} \\ 0 \quad ailleurs \end{cases}$

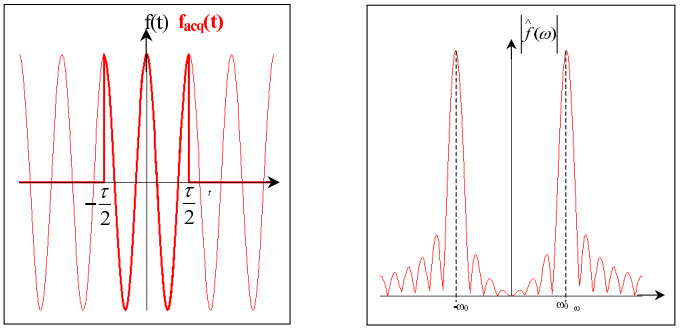

FIGURE II.17 – Sinusoïde tronquée et sa TF

La TF donne : $\hat{f}(\omega) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} f_{acq}(t)\cdot e^{-j\omega t}\cdot dt = \frac{A}{\sqrt{2\pi}}\int_{-\tau/2}^{\tau/2}\cos(\omega t)\cdot e^{-j\omega t}\cdot dt$

soit après calcul :

$$\hat{f}(\omega) = \frac{A\tau}{2\sqrt{2\pi}}\left[sinc(\frac{\omega_0-\omega}{2}\tau) + sinc(\frac{\omega_0+\omega}{2}\tau)\right]$$

On constate sur le tracé du spectre de ce signal tronqué, que des lobes latéraux "entachent" le spectre. Deux solutions se présentent alors : soit on augmente τ ce qui ralentit le traitement en temps réel (pas souhaitable),

soit **on exploite la technique du fenêtrage.**

Le fenêtrage consiste à multiplier le signal par une fonction du temps dite "fenêtre", puis réaliser la TF sur la nouvelle fonction ainsi crée. En fonction de la "forme" choisie pour la fenêtre, cette opération a pour effet de modifier l'allure spectrale du signal ainsi traité en améliorant certains défauts, notamment ceux engendrés par la troncature du signal.

On propose par exemple ici un fenêtrage de la sinusoïde tronquée par la fonction $g(t) = e^{-\left(\frac{4t}{\tau}\right)^2}$

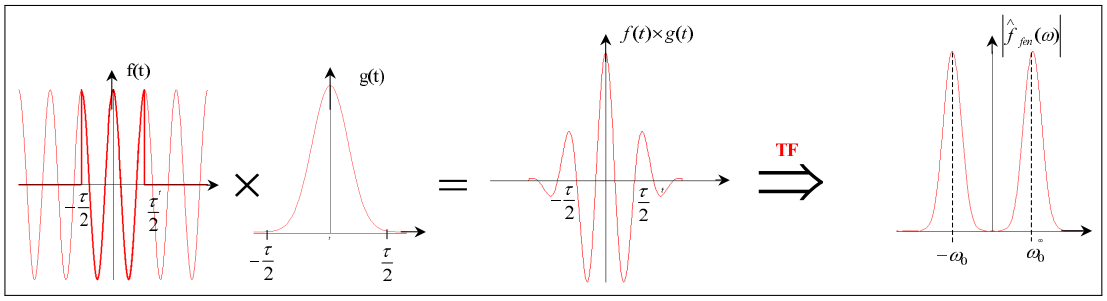

FIGURE II.18 – Fenetrage et TF de la sinusoïde tronquée

Bilan du fenêtrage : on constate un effet d'affinement des raies spectrales -> meilleure lecture de la fréquence ω_0 et plus proche du spectre d'une sinusoïde infinie malgré la troncature.

REMARQUE - (III.1) - 2:

En pratique (cf TP), le type de fenêtrage choisi doit être adapté au signal à analyser, ainsi qu'à la mesure que l'on souhaite réaliser :

- Si l'on souhaite mesurer la fréquence d'une composante, la fenêtre la plus adapté est la fenêtre de Hann (dite "Hanning").
- Si l'on souhaite mesurer l'amplitude d'une composante, on choisira la fenêtre type "Flattop" (sommet plat).

III.2 Exemples d'exploitation

En «live» sur oscilloscope Tektronix en TP !