TP Filtrage numérique

Capacités exigibles du programme :

Analyse spectrale
- Mettre en évidence le phénomène de repliement du spectre provoqué par l'échantillonnage avec un oscilloscope numérique ou une carte d'acquisition.
- Choisir les paramètres d'une acquisition numérique destinée à une analyse spectrale afin de respecter la condition de Shannon, tout en optimisant la résolution spectrale.

Électronique numérique
- Utiliser un convertisseur analogique-numérique et un convertisseur numérique-analogique.

Liste du matériel :

- GBF
- Oscilloscope
- Module Arduino, câble USB
- Plaquette de de montage de composants
- Fils de connexion
- Boîte de résistance variable
- Boîte de capacité variable

1 Acquisition d'un signal et filtrage numérique

Repliement du spectre et critère de Shannon :
Déterminer et mettre en œuvre une analyse numérique permettant de visualiser le phénomène de repliement de spectre sur un signal sonore lorsque la condition de Shannon n'est pas vérifiée.

On étudiera avec soin la première annexe.

Filtrage numérique d'un signal :
Déterminer et mettre en œuvre une analyse numérique permettant de réaliser le filtrage numérique d'un son.

On étudiera avec soin la seconde annexe.

2 Filtrage numérique à l'aide d'un module Arduino

Réalisation d'un filtre passe-bas :
Déterminer et mettre en œuvre un protocole expérimental permettant de réaliser un filtre numérique passe-bas d'ordre 1.

Réalisation d'un filtre passe-haut :
Déterminer et mettre en œuvre un protocole expérimental permettant de réaliser un filtre numérique passe-haut d'ordre 1.

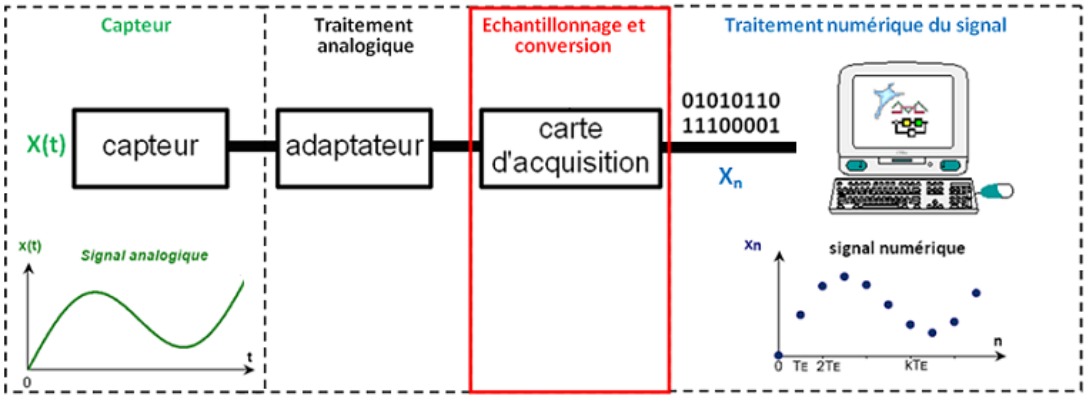
Réalisation d'un filtre passe-bande :
Déterminer et mettre en œuvre un protocole expérimental permettant de réaliser un filtre numérique passe-bande d'ordre 2.

A Annexe 1 - Électronique numérique

La mesure d'un signal, qualifié **d'analogique**, se réalise en général à l'aide d'un capteur auquel il faut souvent ajouter un adaptateur. Par exemple un microphone ne délivrera une tension que de l'ordre du mV, il faudra donc l'amplifier.

Si l'on désire réaliser un traitement numérique de ce signal, une **conversion analogique-numérique** (CAN) doit être réalisée. C'est ce que réalise en général un système d'acquisition de données (oscilloscope numérique, module Arduino, etc.).

On peut représenter l'ensemble de la chaîne d'acquisition ainsi :



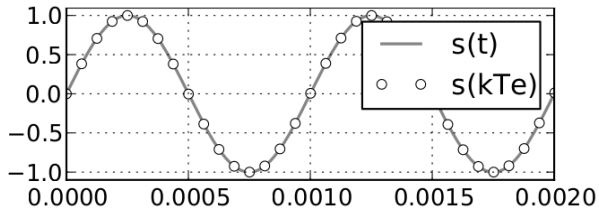
A.1 Échantillonnage

A.1.1 Définition

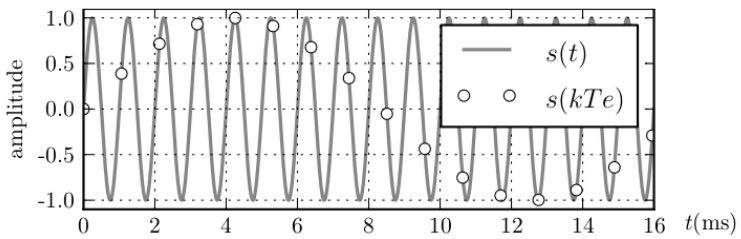
L'échantillonnage est l'opération qui consiste à mesurer un signal en capturant des valeurs à intervalles réguliers.

La durée entre deux mesures s'appelle la **période d'échantillonnage**, notée T_e. Se pose alors la question de savoir si les échantillons sont représentatifs du signal initial.

Si la période d'échantillonnage T_e est bien choisie, le signal obtenu est représentatif du signal initial, par exemple pour un signal sinusoïdal :



Si T_e est plus faible, cela ne changera rien. Si par contre elle est plus élevée, il peut se passer le phénomène suivant :



On constate que le signal obtenu n'est pas représentatif du signal initial, et de plus qu'il a une fréquence plus faible : on parle de **repliement de spectre**.

À l'aide d'une simulation (cf. notebook *Repliement de spectre et critère de Shannon*), on peut voir, pour un signal sinusoïdal, que pour que le signal obtenu ait la même fréquence que le signal initial, il faut que $Te < T/2$.

Si le signal est constitué de plusieurs composantes dans son spectre, cela donne :

$$Te < \frac{T_{\min}}{2}$$

On aboutit ainsi au théorème de Shannon.

A.1.2 Théorème de Shannon

À la suite des constatations précédentes, on admet le théorème de Shannon [1] :

> Un signal est correctement représenté à partir de ses échantillons, si la fréquence d'échantillonnage est supérieure à **deux fois** la fréquence maximale de son spectre.

Ce théorème s'énonce parfois à l'aide de la **condition de Shannon**, qui est la condition nécessaire à une bonne représentation :

$$\boxed{f_e > 2f_{\max}}$$

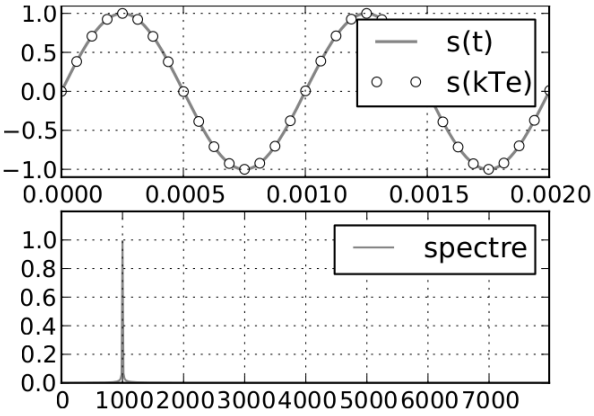
Un exemple d'application de ce théorème est fourni par le format des morceaux musicaux. La fréquence d'échantillonnage en HD est de 44 kHz, supérieure au double de la fréquence maximale audible par une oreille humaine, qui est de l'ordre de 20 kHz.

A.2 Spectre d'un signal échantillonné

A.2.1 Tracé du spectre

On a déjà vu que tout signal temporel possède une **représentation spectrale**. À l'aide des échantillons de ce signal, il existe un algorithme appelé FFT, acronyme de l'anglais *Fast Fourier Transform*, qui permet d'avoir une représentation approchée de son spectre. Cet algorithme, de complexité quasi-linéaire, est implanté par exemple dans les oscilloscopes numériques, les cartes graphiques des ordinateurs, les tableurs, et dans toutes les bibliothèques de calcul numérique.

Sur l'exemple suivant, le critère de Shannon est largement vérifié avec $f_e = 16f$:



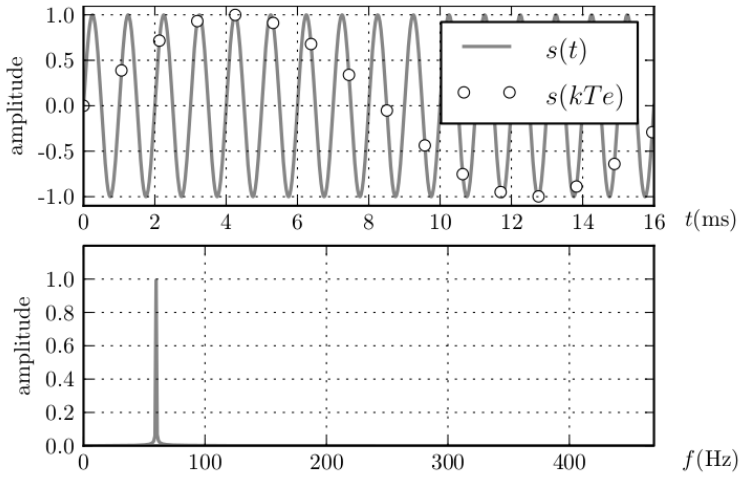
Le spectre obtenu est cohérent, mais s'étend inutilement jusqu'à 8 kHz $= \frac{f_e}{2}$, le spectre est tassé sur la gauche.

Ce résultat est important lorsque l'on cherche à calculer un spectre numériquement :

> L'étendue du spectre calculé par l'algorithme FFT s'étale de 0 à $\frac{f_e}{2}$.

Dans l'exemple suivant, le critère de Shannon n'est plus respecté avec $f_e = \frac{16}{17}f$:

1. Claude Shannon, 1916 - 2001, ingénieur et mathématicien américain



Les échantillons dessinent une sinusoïde de fréquence plus faible que celle du signal. C'est celle que l'œil reconstitue à partir des échantillons, c'est aussi celle que détermine l'algorithme FFT. On parle de **repliement de spectre**.

A.2.2 Repliement du spectre

On constate sur la figure précédente que le signal sinusoïdal $s(t)$ de fréquence $f > \dfrac{f_e}{2}$ est perçu comme un autre signal sinusoïdal $s'(t)$ de fréquence f' telle que $f' < \dfrac{f_e}{2}$, et de même amplitude que $s(t)$.

Ainsi toutes les fréquences ne vérifiant pas la condition de Shannon se retrouvent dans l'intervalle $[0, f_e/2[$. Ce phénomène s'appelle **le repliement du spectre**. Il est responsable de l'apparition de raies dans le spectre du signal échantillonné qui n'existent pas dans le spectre du signal réel.

A.2.3 Capacité numérique : application d'un algorithme FFT à un signal

Cf. le notebook « Appliquer un algorithme FFT à un signal ».

A.3 Acquisition d'un signal

A.3.1 Paramètres d'acquisition

Expérimentalement, on pourra régler trois paramètres :
- le nombre de points N ;
- la période d'échatillonnage Te ;
- la durée de l'acquisition D.

On utilisera donc la relation :

$$\boxed{T_e = \frac{D}{N}}$$

A.3.2 Choix de la période d'échantillonnage et de la durée d'acquisition

On prendra soin d'avoir :

$$\boxed{T_e << T}$$

inférieur au facteur 2 théorique minimum (condition de Shannon), expérimentalement souvent d'un facteur 20.

Le choix de la durée d'acquisition se fait souvent avec le nombre n de périodes que l'on souhaite voir à l'écran :

$$\boxed{D \sim nT}$$

B Annexe 2 - Filtrage numérique

B.1 Élaboration d'un filtre numérique

B.1.1 Exemple d'un filtre passe-bas du premier ordre

La forme générale d'un filtre passe-bas d'ordre 1 est donnée par :

$$\frac{\underline{s}}{\underline{e}} = \frac{1}{1+jx}$$

avec $x = \dfrac{\omega}{\omega_0} = \dfrac{\omega}{\omega_c} = \omega\tau$

L'équation différentielle associée est donc :

$$s + \tau \frac{\mathrm{d}s}{\mathrm{d}t} = e$$

On peut « discrétiser » cette formule de façon à utiliser la méthode d'Euler :

$$\mathrm{d}s = \frac{\mathrm{d}t}{\tau}(e - s)$$

Soit :

$$\boxed{s[n+1] = s[n] + \frac{T_e}{\tau} \times (e[n] - s[n])}$$

avec T_e la période d'échantillonnage.

B.1.2 Exemple d'algorithme, rédigé en Python

Il est alors très simple de programmer un tel filtre en Python :

```
def filtre_passe_bas_1o(e,te,tau):
    s=[0]
    for n in range(len(e)-1):
        s.append(s[n]+te/tau*(e[n]-s[n]))
    return(s)
```

filtre_passe_bas_exemple.py

La dernière page présente, sous forme de carte heuristique, le principe de construction de plusieurs filtres numériques en Python.

B.1.3 Capacité numérique : réalisation du filtrage numérique d'un signal

Cf. le notebook « Réaliser le filtrage numérique d'un signal ».

B.2 Réalisation d'un filtrage numérique à l'aide d'un module Arduino

B.2.1 Introduction

Le traitement numérique d'un signal (calculs) est en général confié à des composants dédiés à cette tâche, qui sont souvent des microcontrôleurs.

Ainsi, on réalise l'acquisition du signal analogique en entrée, son traitement, puis on envoie en sortie le signal traité. Ainsi il faut deux composants : un **convertisseur analogique-numérique** en entrée et un **convertisseur numérique-analogique** en sortie, souvent abrégés en **CAN** et **CNA**.

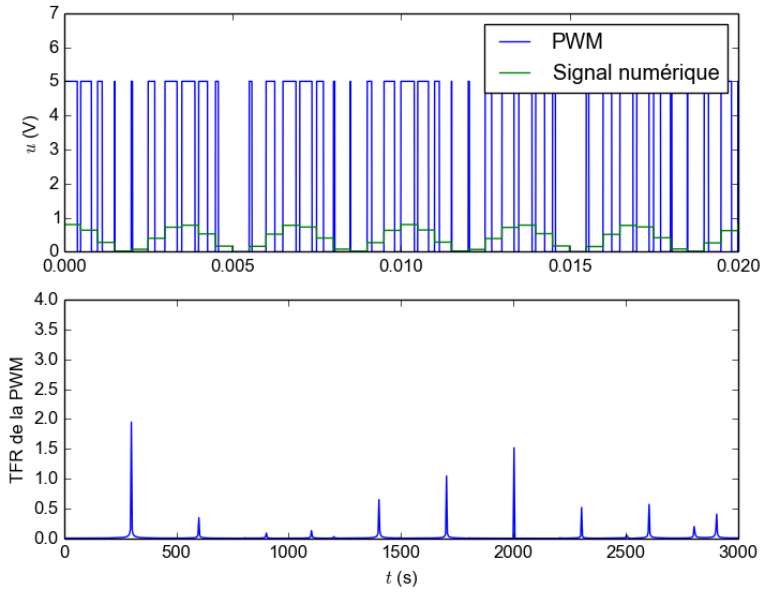
B.2.2 Utilisation d'une carte Arduino

Cette carte possède des entrées analogiques qui permettent la conversion analogique numérique. **Attention** les entrée analogiques fonctionne entre 0 et 5 V uniquement :

Le signal d'entrée devra toujours être positif.

La carte possède également des sorties qui permettent la conversion numérique analogique, mais sous une forme un peu particulière. Le signal de sortie se présente sous la forme d'un signal créneau, dont la **largeur de chaque**

créneau est modulée par le signal. Ainsi, un signal sinusoïdal sera par exemple représenté par le signal ci-dessous.



On parle de **Modulation de Largeur d'Impulsion**, notée MLI en français ou PWM en anglais (Pulse Width Modulation). On peut montrer que le spectre de ce signal comporte une composante basse fréquence correspondant au signal à transmettre (voir ci-dessus). Il convient alors simplement **d'ajouter un filtre passe-bas en sortie de la carte**, de fréquence de coupure supérieure à la fréquence du signal et très inférieure à celle du signal créneau de la PWM :

$$f_s < f_c << f_{\text{PWM}}$$

Un simple circuit RC pourra ici faire l'affaire.

Au final, un entier N tel que $0 \leqslant N \leqslant 1023$ sera converti en une tension $s(t)$ telle que :

$$s(t) = \frac{N}{1023} \times 5\ V$$

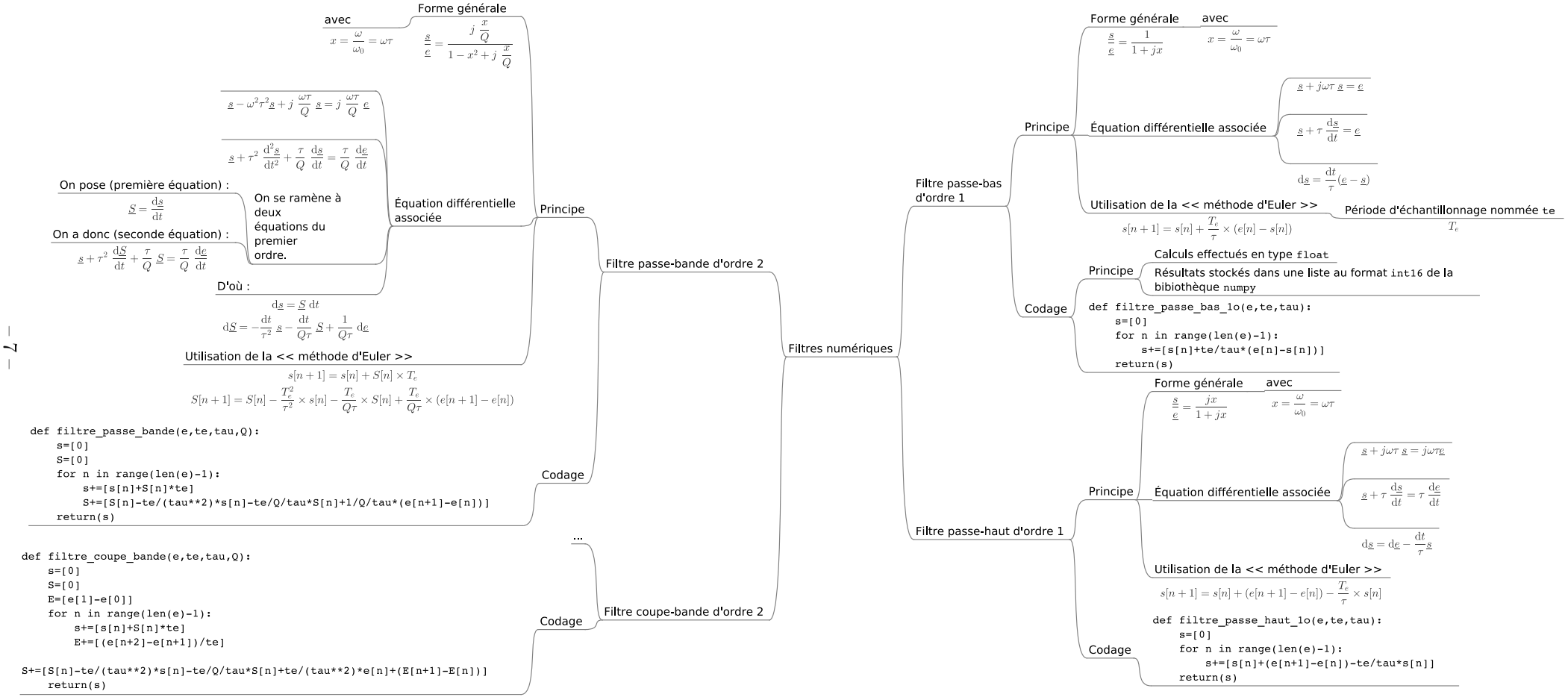
B.2.3 Mise en œuvre

Un filtre passe-bas de période d'échantillonnage $Te = 0,1$ ms et de fréquence de coupure $f_c = 100$ Hz pourra ainsi être codé de la façon suivante :

```
// broches
int entree = A0; // la broche d'entree
int sortie = 9; // la broche de sortie

// grandeurs
float e; // le signal d'entree
float s=0; // le signal de sortie
int s_num; // le signal numerique de sortie

void setup()
{
  TCCR1B = 1; // initialisation de la PWM a 31  kHz sur les broches 9 et 10
}

void loop()
{
  e=analogRead(entree)*5.0/1023;
  s=s+0.0628*(e-s); // le fitre passe bas, fc=100 Hz, Te/tau=Te*wc=0,0628
  s_num=s/5.0*1023;
  analogWrite(sortie,s_num);
  delay(0.1); // Te=0,1 ms (fe=10 kHz)
}
```

filtre_passe_bas_arduino/filtre_passe_bas_arduino.ino

Remarque : Il faut bien indiquer `5.0` pour la valeur de la tension de 5 V, pour indiquer qu'il s'agit d'un flottant et donc forcer la division flottante au lieu de la division entière.


Filtres numériques
Filtre passe-bas d'ordre 1
Principe
Forme générale
$\frac{\underline{s}}{\underline{e}} = \frac{1}{1+jx}$
avec
$x = \frac{\omega}{\omega_0} = \omega\tau$
Équation différentielle associée
$\underline{s} + j\omega\tau\,\underline{s} = \underline{e}$
$\underline{s} + \tau\frac{\mathrm{d}\underline{s}}{\mathrm{d}t} = \underline{e}$
$\mathrm{d}\underline{s} = \frac{\mathrm{d}t}{\tau}(\underline{e} - \underline{s})$
Utilisation de la << méthode d'Euler >>
$s[n+1] = s[n] + \frac{T_e}{\tau} \times (e[n] - s[n])$
Période d'échantillonnage nommée te
T_e
Codage
Calculs effectués en type float
Résultats stockés dans une liste au format int16 de la bibiothèque numpy
def filtre_passe_bas_1o(e,te,tau):
s=[0]
for n in range(len(e)-1):
s+=[s[n]+te/tau*(e[n]-s[n])]
return(s)
Filtre passe-haut d'ordre 1
Principe
Forme générale
$\frac{\underline{s}}{\underline{e}} = \frac{jx}{1+jx}$
avec
$x = \frac{\omega}{\omega_0} = \omega\tau$
Équation différentielle associée
$\underline{s} + j\omega\tau\,\underline{s} = j\omega\tau\underline{e}$
$\underline{s} + \tau\frac{\mathrm{d}\underline{s}}{\mathrm{d}t} = \tau\frac{\mathrm{d}\underline{e}}{\mathrm{d}t}$
$\mathrm{d}\underline{s} = \mathrm{d}\underline{e} - \frac{\mathrm{d}t}{\tau}\underline{s}$
Utilisation de la << méthode d'Euler >>
$s[n+1] = s[n] + (e[n+1] - e[n]) - \frac{T_e}{\tau} \times s[n]$
Codage
def filtre_passe_haut_1o(e,te,tau):
s=[0]
for n in range(len(e)-1):
s+=[s[n]+(e[n+1]-e[n])-te/tau*s[n]]
return(s)
Filtre passe-bande d'ordre 2
Principe
Forme générale
$\frac{\underline{s}}{\underline{e}} = \frac{j\,\frac{x}{Q}}{1 - x^2 + j\,\frac{x}{Q}}$
avec
$x = \frac{\omega}{\omega_0} = \omega\tau$
Équation différentielle associée
$\underline{s} - \omega^2\tau^2\underline{s} + j\,\frac{\omega\tau}{Q}\,\underline{s} = j\,\frac{\omega\tau}{Q}\,\underline{e}$
$\underline{s} + \tau^2\,\frac{\mathrm{d}^2\underline{s}}{\mathrm{d}t^2} + \frac{\tau}{Q}\,\frac{\mathrm{d}\underline{s}}{\mathrm{d}t} = \frac{\tau}{Q}\,\frac{\mathrm{d}\underline{e}}{\mathrm{d}t}$
On pose (première équation) :
$\underline{S} = \frac{\mathrm{d}\underline{s}}{\mathrm{d}t}$
On a donc (seconde équation) :
$\underline{s} + \tau^2\,\frac{\mathrm{d}\underline{S}}{\mathrm{d}t} + \frac{\tau}{Q}\,\underline{S} = \frac{\tau}{Q}\,\frac{\mathrm{d}\underline{e}}{\mathrm{d}t}$
On se ramène à deux équations du premier ordre.
D'où :
$\mathrm{d}\underline{s} = \underline{S}\,\mathrm{d}t$
$\mathrm{d}\underline{S} = -\frac{\mathrm{d}t}{\tau^2}\,\underline{s} - \frac{\mathrm{d}t}{Q\tau}\,\underline{S} + \frac{1}{Q\tau}\,\mathrm{d}\underline{e}$
Utilisation de la << méthode d'Euler >>
$s[n+1] = s[n] + S[n] \times T_e$
$S[n+1] = S[n] - \frac{T_e^2}{\tau^2} \times s[n] - \frac{T_e}{Q\tau} \times S[n] + \frac{T_e}{Q\tau} \times (e[n+1] - e[n])$
Codage
def filtre_passe_bande(e,te,tau,Q):
s=[0]
S=[0]
for n in range(len(e)-1):
s+=[s[n]+S[n]*te]
S+=[S[n]-te/(tau**2)*s[n]-te/Q/tau*S[n]+1/Q/tau*(e[n+1]-e[n])]
return(s)
...
Filtre coupe-bande d'ordre 2
Codage
def filtre_coupe_bande(e,te,tau,Q):
s=[0]
S=[0]
E=[e[1]-e[0]]
for n in range(len(e)-1):
s+=[s[n]+S[n]*te]
E+=[(e[n+2]-e[n+1])/te]
S+=[S[n]-te/(tau**2)*s[n]-te/Q/tau*S[n]+te/(tau**2)*e[n]+(E[n+1]-E[n])]
return(s)