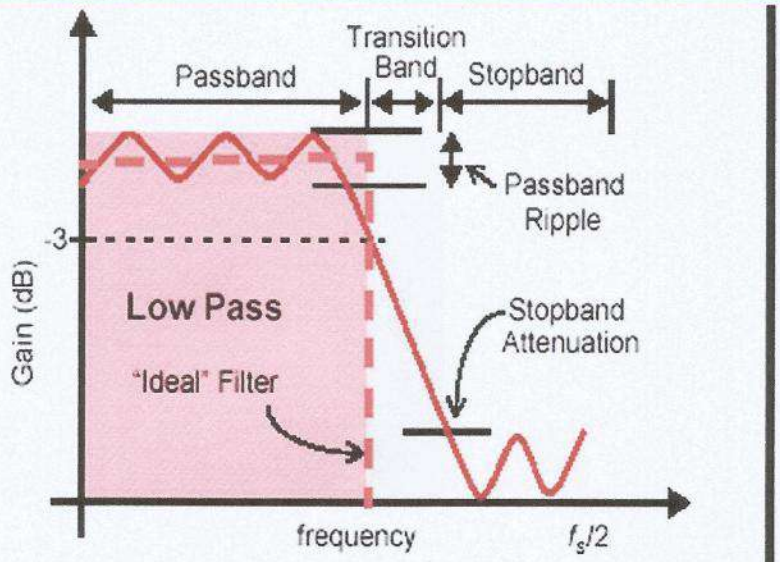
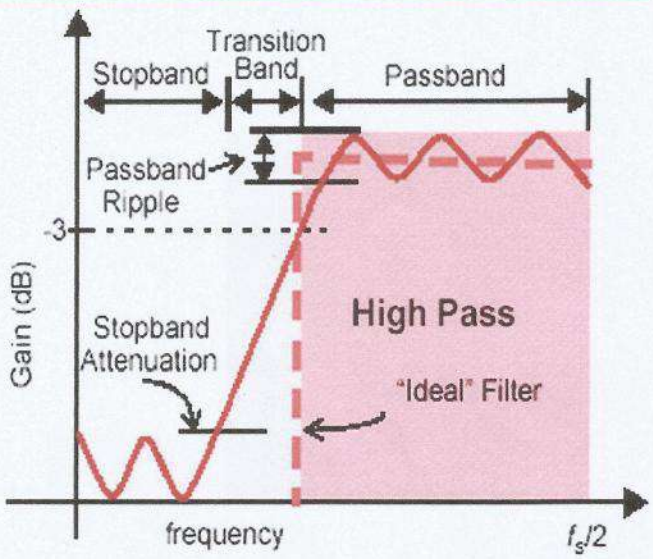
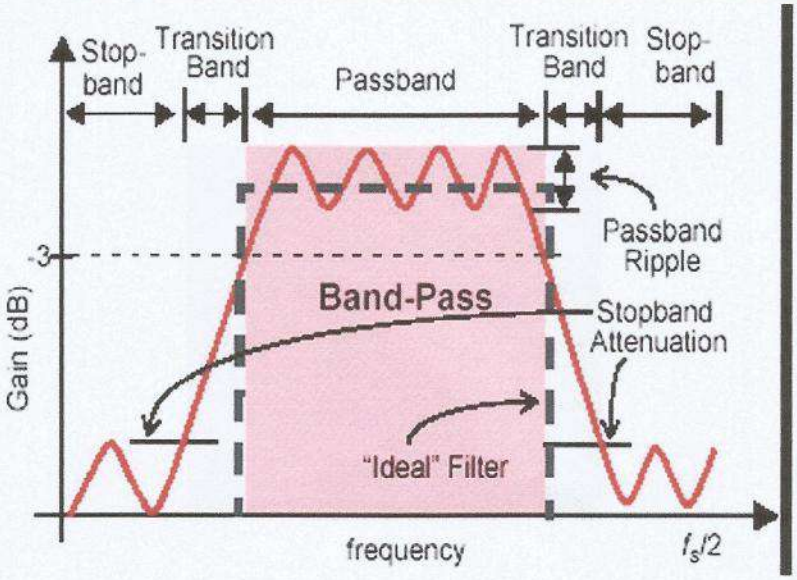
efrei
PARIS SUD
Ingénieurs en mouvement

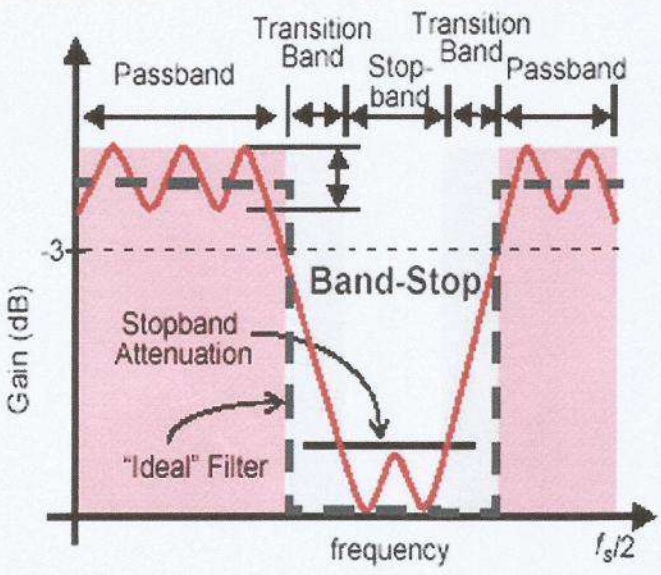
# Projet
# Du système à la fonction

## Synthèse de banc de filtres numériques





Julien Yoeurp
Paul Budowski
Baptiste Chazal


Vendredi 16 Mai 2014

# TABLE DES MATIERES

# INTRODUCTION

Un signal n'est pas toujours la représentation parfaite de notre volonté. Que ce soit un signal numérique, audio ou autre, nous voulons parfois filtrer ce signal afin qu'il puisse correspondre plus à une de nos attentes. On peut prendre comme exemple une chanson qui possède trop d'aigus qui tendent à être très désagréables à l'écoute.

Pour essayer de corriger ce problème, il existe ce que l'on appelle des filtres, ce sont des composants électroniques qui effectuent des opérations de traitement de signal. Concrètement, ils vont en atténuer certaines composantes et en laisser passer d'autres.

Pour étudier ce domaine du traitement de signal, nous aurons à disposition MATLAB, un langage de programmation et de modélisation. Dès lors, nous pouvons nous demander comment créer et simuler ces filtres au moyen d'une programmation qui s'apparente à de l'informatique pure ?

Pour répondre à cette question, nous verrons dans une première partie le fonctionnement général de MATLAB ainsi que les techniques de base pour réaliser des filtres. C'est dans une deuxième partie que nous ferons une approche pratique avec le filtrage d'un signal audio sous différentes formes.

# Initiation à la modélisation sous MATLAB

## 1.Introduction à MATLAB

MATLAB (MATrix LABoratory) est un langage haut niveau et aussi un environnement interactif utilisé dans le domaine du calcul numérique, de la modélisation et de la programmation.
Avec MATLAB, il est possible d'analyser des données, de développer des algorithmes et de créer des applications.

## Extrait de code MATLAB :

```
clc; clf;
N=100; t0=20;
k=1; k1=0.5; k2=20;            %Déclaratiion des constantes
e=zeros(1,N); temp=zeros(1,N);
d=zeros(1,N); s=zeros(1,N);      %Initialisation de tableaux
y=zeros(1,N); z=zeros(1,N);
ss=zeros(1,N); zz=zeros(1,N);
for t=t0:N-1   e(t+1)=1; end    %Programmation d'un échelon
for t=1:N      temp(t)=t-1; end  %Tableau des temps

%----------Programmation d'une dérivée----------

for t=2:N   d(t)=k*(e(t)-e(t-1)); end

%----------Tracé des Chronogrammes----------
figure(1);
subplot(2,2,1); plot(temp,e,'r');        % 'r'=red
grid; title ('Entrée');
subplot(2,2,2); plot(temp,d,'g');        % 'g'=green
grid; title ('Dérivée');
subplot(2,2,3); plot(temp,e,'x',temp,d,'g');
grid; title ('Lentree et la sortie');
```

Avec cet extrait de code, on peut facilement comprendre comment MATLAB est assimilable à un langage de programmation. Nous avons par exemple dans ce bout de code :

- Déclaration de constantes (lignes 2 et 3)
- Initialisation de tableaux (lignes 4 à 7)
- Implémentation de boucles à instructions (lignes 8 à 10)
- Appel de fonctions (ligne 1)
- Affichage de graphiques (lignes 11 à 17)
- Ecriture de commentaires (introduites par le signe '%')

C'est cet outil que nous allons utiliser pour programmer des filtres numériques.

## 2.Premiers programmes MATLAB

Avant de passer à la réalisation des filtres à proprement parler, nous devons d'abord implémenter une entrée « échelon » et une sortie « dérivée ».

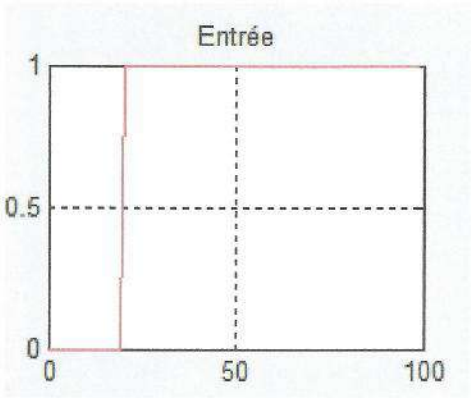
Une fonction échelon ou encore fonction de Heaviside est une fonction discontinue définie par :

$$\forall x \in \mathbb{R},\ H(x) = \begin{cases} 0 & \text{si} \quad x < 0 \\ 1 & \text{si} \quad x \geq 0. \end{cases}$$

Dans notre cas, ce sera un signal entrant qui vaudra 0 pour tout t<0 et qui vaudra 1 pour tout t égal ou supérieur à 0.

Voici l'instruction permettant de créer un tel signal :

```
for t=t0:N-1    e(t+1)=1; end
```




On peut remarquer que cette fonction ressemble bien à une fonction de Heaviside mais avec la particularité d'atteindre son maximum en un point autre que 0 comme cité précédemment.
La raison à ceci est la présence dans le code d'une constante fixant l'origine des temps :

```
t0=20;
```

Ainsi, c'est bien un signal « échelon » à condition de considérer que l'origine des temps se trouve à l'instant t=20.
Désormais, cette fonction sera notée e(t).

Dérivée : Pour la suite à venir, il y aura également nécessité d'avoir une fonction de dérivation.
Dans notre cas, il n'aura pas de calcul direct de dérivée mais nous allons implémenter une estimation de dérivée à partir d'un taux de variation.

Ainsi pour dériver notre signal d'entrée « échelon », nous avons :

$$\frac{de(t)}{dt}$$

Que l'on peut estimer grâce à son taux de variation.

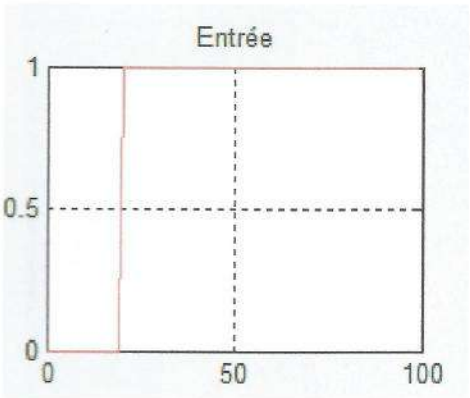
On a alors :

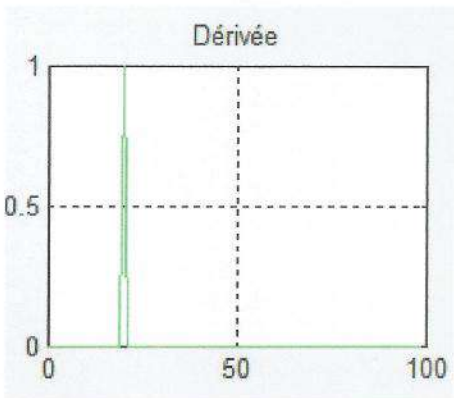
$$\frac{de(t)}{dt} = e(t)-e(t-1)$$

Il ne faut pas oublier que dans ce circuit, le dérivateur possède un gain que l'on notera k. C'est pourquoi le code MATLAB correspondant sera :

```
for t=2:N    d(t)=k*(e(t)-e(t-1)); end
```

Il s'agit d'une boucle commençant à l'instant t=2 et se finit à l'instant N (N=100 dans le code), la fonction dérivée est notée d(t) et on multiplie le taux de variation précédemment trouvé par le gain du dérivateur.




La dérivée obtenue correspond bien à la dérivée du signal d'entrée, la dérivée correspondant à la pente de la tangente en un point de la courbe.

Il existe aussi une autre manière de programmer cette même fonction échelon, avec le code :

```
e=[zeros (1,t0) ones(1,N-t0)]
```

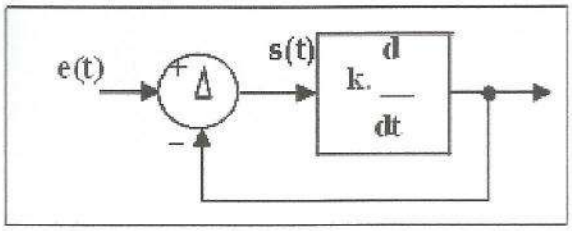
zeros (1,t0) : Cette partie va implémenter des zéros (signal nul) jusqu'en t0
ones(1,N-t0) : Cette partie va remplir de 1 le tableau correspondant.

Enfin, petite particularité, l'ajout d'un point virgule à la fin d'une instruction permet de cacher son effet dans la fenêtre de commande. Ceci peut être utile pour éviter d'encombrer inutilement cette fenêtre.

## 3.Programmation de filtres

### 3.1 Du premier ordre (I) :

Pour programmer un filtre passe bas sous MATLAB, nous devons implémenter dans le code une fonction mathématique permettant de recréer le comportement d'un filtre passe bas.




Montage d'un filtre passe-bas

En se basant sur ce montage, on peut en déduire :

$e(t)=s(t)+k\frac{de(t)}{dt}$ (avec s(t) sortie du filtre passe bas)

En remplaçant la dérivée par un accroissement fini, on obtient :

$e(t)=s(t)+k[e(t)-e(t-1)]$

D'où l'expression de la sortie s(t) :

$s(t)=\frac{e(t)}{1+k}+\frac{k}{1+k}s(t-1)$

La sortie d'un filtre passe bas du premier ordre peut être donc programmée au moyen d'une récurrence de la forme :

$s(t)=\alpha e(t)+\beta s(t-1)$

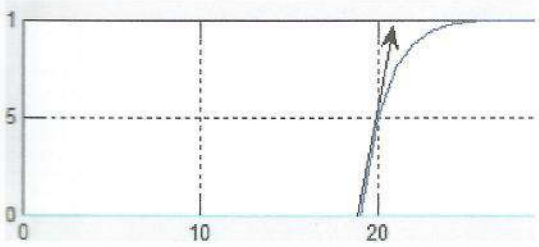
Avec : $\alpha=\frac{1}{1+k}$ et $\beta=\frac{k}{1+k}$

Pour implémenter ce filtre passe bas dans notre système, il faut ajouter la ligne suivante :

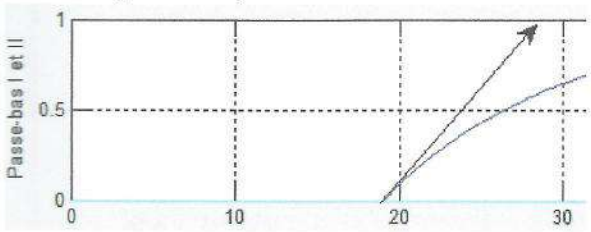
for t=2:N $\quad s(t)=\frac{1}{1+k}e(t)+\frac{k}{1+k}s(t-1)$ ; end

Nous allons maintenant relever la réponse indicielle de ce filtre passe bas, qui est la réponse d'un système linéaire soumis à une fonction échelon en entrée.

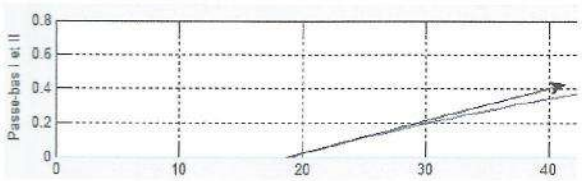
**Pour un gain k=1, nous avons :**




**Pour un gain k=10, nous avons :**




**Et pour un gain k=50, nous avons :**



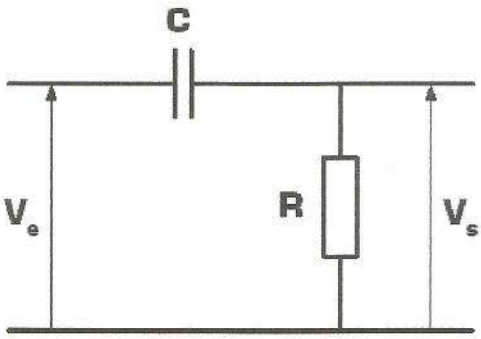

Nous pouvons voir qu'en augmentant le gain k, il se produit une modification de la réponse indicielle du filtre.
En effet, plus on augmente le gain, plus la pente est faible, c'est-à-dire que la rapidité de réponse du filtre décroit avec un gain montant.

C'est pourquoi le gain k a un rôle dans la formule de la fréquence de coupure du filtre :

$$f_c = \frac{1}{2\,k\,\pi}$$

Passons maintenant à la réalisation d'un filtre passe haut.

Pour ceci, nous pouvons reprendre le même raisonnement que pour le filtre passe bas.




Montage d'un filtre passe haut passif

On part de :

$$k\frac{de(t)}{dt} = s(t) + k\frac{ds(t)}{dt}$$

On remplace les dérivées par leurs accroissements finis :

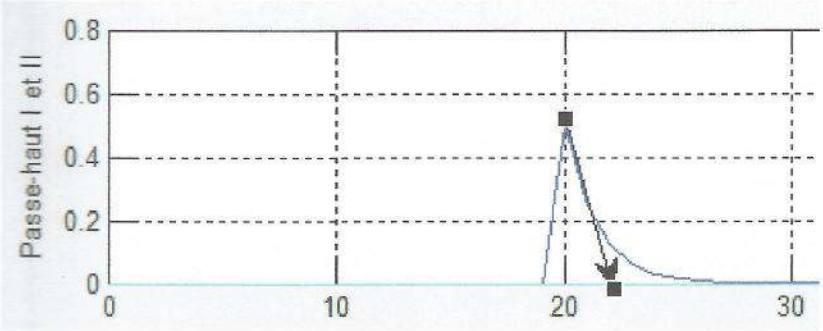
$$k[e(t)-e(t-1)] = s(t)+k[s(t)-s(t-1)]$$

Ainsi, nous trouvons comme fonction de sortie d'un filtre passe haut :

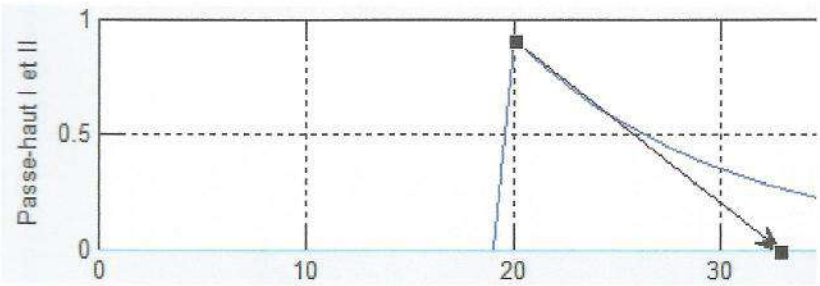
$$s(t)= \frac{k}{1+k}[e(t)-e(t-1)+s(t-1)]$$

Voyons maintenant sa réponse indicielle avec comme précédemment un même choix de gain montant :

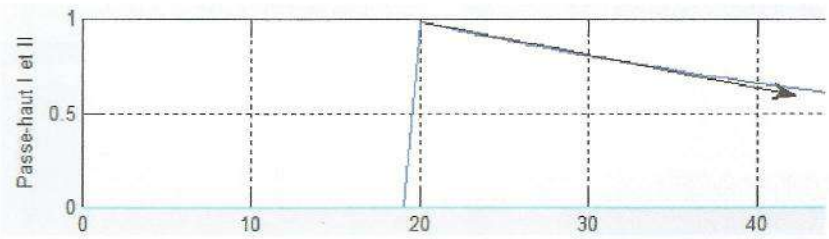
**Pour un gain k=1 :**




**Pour un gain k=10 :**




**Pour un gain k=50 :**




On retrouve le même comportement sur un filtre passe haut.

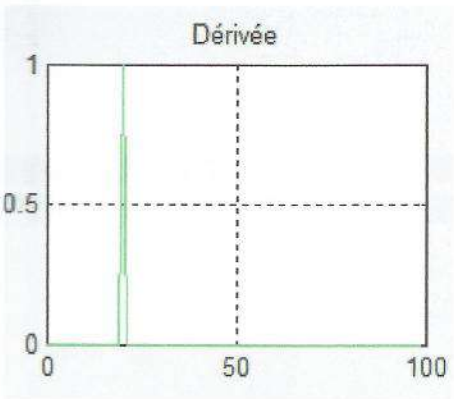
Ceci explicite clairement une des caractéristiques d'un filtre du premier ordre : Que ce soit un passe haut ou un passe bas, leur comportement est « équivalent ».

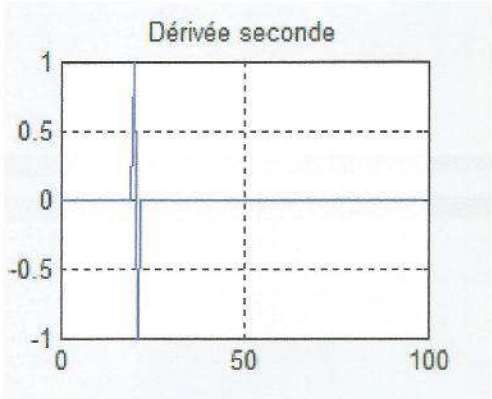
## 3.2 Du second ordre (II) :

Pour introduire les filtres du second ordre, on peut raisonner ainsi :
Si pour créer des filtres du premier ordre, on a eu recours à une dérivée simple, il nous faudrait approcher une dérivée seconde pour créer des filtres du second ordre.

Pour approcher une dérivée seconde, nous pouvons dériver la première dérivée, en analogie avec les mathématiques, il faut mettre sous MATLAB :

```
for t=3:N    d(t)=k*(e(t)-e(t-1)); end      % Dérivée première
for t=3:N    d2(t)=k*(d(t)-d(t-1)); end     % Dérivée seconde
```




Nous avons bien la dérivée seconde, qui est la dérivée de la première dérivée.

Mais ce n'est pas cette méthode que nous allons utiliser.

Programmation d'un filtre passe bas (second ordre) :

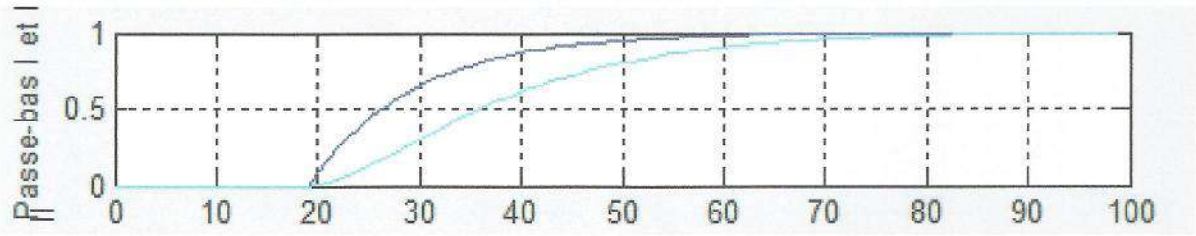
La réalisation du filtre passe bas (II) ne diffère pas tellement de celui du premier ordre, il s'agit juste de mettre en cascade deux filtres passe bas (I).
C'est pourquoi il faut mettre en entrée du second filtre passe bas du (I) la sortie du premier passe bas (I) pour obtenir le filtre passe bas (II).

Le code MATLAB correspondant est :

```
for t=2:N   ss(t)=(1/(k+1))*s(t)+(k/(k+1))*ss(t-1);  end
```

On remarque que l'expression de la sortie du passe bas (II) (notée ici ss(t)) dépend bien de l'expression du filtre passe bas (I) (avec la présence de s(t)).

Voici la réponse indicielle simultanée d'un filtre passe bas (I) en bleu foncé et celle d'un filtre passe bas (II) en bleu cyan avec un gain k=10 :



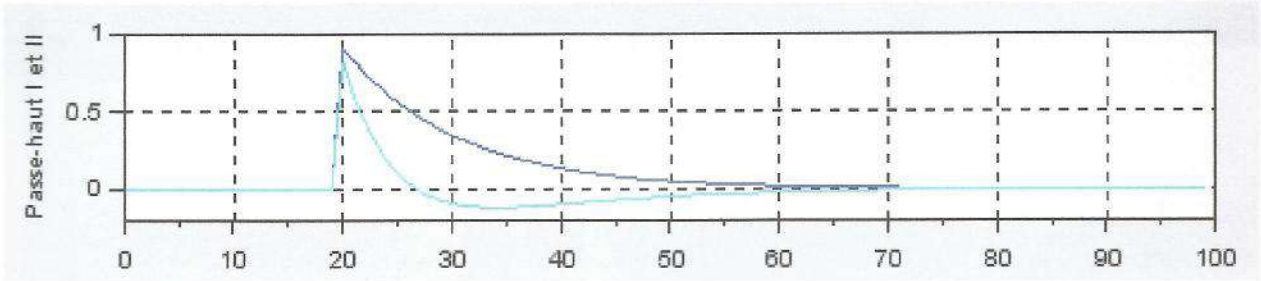

La réponse (pente à l'origine des temps) est plus forte pour un filtre passe bas (I) que pour un filtre passe bas (II).

Qu'en est-il pour le filtre passe haut ?

Même raisonnement que pour le filtre passe bas, il faut ajouter le code MATLAB suivant :

```
for t=2:N   zz(t)=(k/(k+1))*(z(t) - z (t-1))+(k/(k+1))*zz(t-1);  end
```

Et maintenant, le chronogramme correspondant :



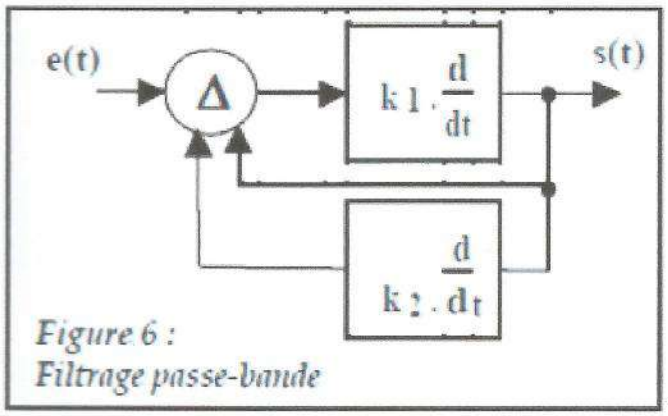

Le comportement des filtres passe haut est l'inverse de celui des filtres passe bas.

Le filtre passe bas (I) a une rapidité de réponse plus forte que celui d'ordre 2,
mais le filtre passe haut (I) a une rapidité de réponse plus faible que celui d'ordre 2.

De plus, on peut remarquer que le régime transitoire du filtre passe bas (II) est plus long que celui d'ordre 1, mais que dans le cas des filtres passe haut, le régime transitoire est de même durée quelque soit l'ordre mais le passe haut(II) passe par un état de dépassement (valeur<0).

Programmation d'un filtre passe bande :

Pour programmer un filtre passe bande, il faut mettre en cascade un filtre passe haut et un filtre passe bas.




Montage d'un filtre passe bande

Son expression se résume ainsi :

y(t)=e(t)-(1+k2)*y(t)+k2*y(t-1)-y(t)-k2*(y(t)-y(t-1))

Il est caractérisé par deux constantes de temps nommés k1 et k2 qui définissent :

-La pulsation propre $\omega 0$ : On la nomme propre car c'est la pulsation « de base » du système, sans intervention d'une quelconque force extérieure.
-Le facteur de qualité q : C'est la « qualité du filtre », en d'autres termes plus q est grand, plus la largeur de la bande passante sera étroite et donc plus le filtre sera sélectif.

Nous avons :

$k1=\frac{1}{q\omega 0}$

$k2=\frac{q}{\omega 0}$

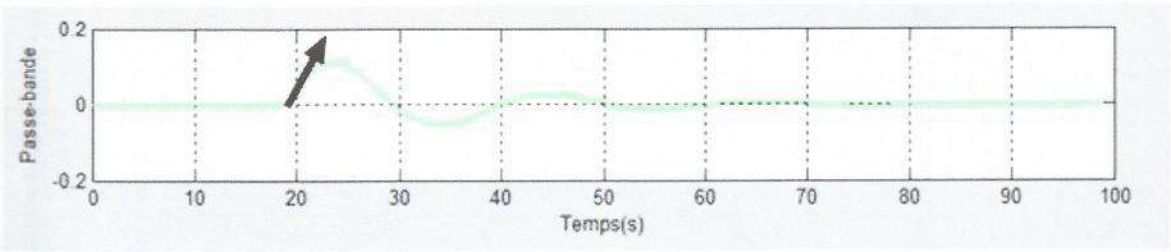
Pour notre filtre passe bande, nous allons choisir les constantes suivantes :
k1=0,5 et k2=20, nous avons donc :

$k1=\frac{1}{q\omega 0}=0,5$ $\quad q=20\,\omega 0$ $\quad \boxed{\omega 0=\sqrt{0,1}}$ $\quad \boxed{q=20\sqrt{0,1}}$

$\Leftrightarrow$ d'où $k1=\frac{1}{20\omega 0^2}=0,5$ $\Leftrightarrow$ $k2=\frac{q}{\sqrt{0,1}}=20$ $\Leftrightarrow$

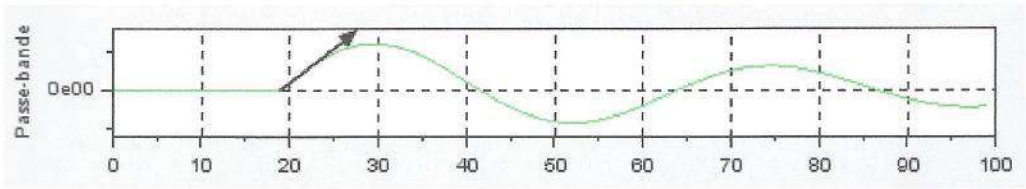
$k2=\frac{q}{\omega 0}=20$

Son chronogramme est le suivant :




Sa réponse indicielle est ici semblable à un filtre du premier ordre soumis au même gain k=10.
On retrouve ici une pseudo période d'une valeur de 20s.

Modifions maintenant le facteur de qualité en prenant q=100 :




Avec un facteur de qualité plus important, la pseudo période augmente pour avoir ici une valeur de 43 secondes.
De plus, ce facteur q joue un rôle dans la réponse indicielle du filtre : Plus le facteur qualité est important, moins la réponse indicielle sera forte.

# APPLICATION AUDIO

## 1. Observation de signaux du spectre sonore

Nous étudierons en premier temps des signaux audio :

Pour ceci, nous devons nous assurer du respect de certains paramètres :
-Le nombre d'échantillons nécessaires
-Normalisation de la fréquence

Dans le code, la ligne « fréquence » correspondra à la fréquence que l'on voudra générer.

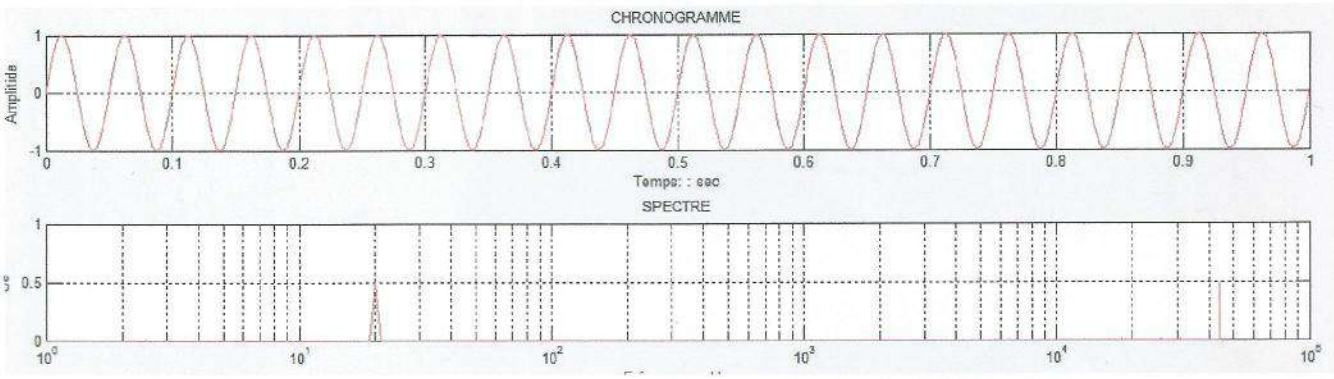
En sachant que notre fréquence d'échantillonnage est ici de $f_E$=44100 Hz (Fréquence standard en audio)
Cela signifie que toutes les secondes, il y aura traitement de 44100 valeurs.
Ainsi, pour représenter un signal sur une fenêtre d'une seconde, on aura besoin de 44100 échantillons (avec un taux d'échantillonnage de 44,1 Khz)
Ce qui justifie la ligne N=44100

Enfin, il faut s'assurer de la normalisation de la fréquence selon la condition de Shannon.
La condition de Shannon explique que l'on ne perd pas d'information en reconstruisant un signal à partir de ses échantillons si la fréquence d'échantillonnage est au moins égale à deux fois la plus élevée des fréquences contenues dans le spectre du signal qu'on échantillonne.

C'est pourquoi la fréquence maximale des signaux que nous pouvons échantillonner sera égale à au plus la moitié du taux d'échantillonnage.

$f_{max}=\frac{fe}{2}$=22050Hz

Par exemple, pour une fréquence de 20Hz, nous avons ce chronogramme et ce spectre :




On observe bien un signal sinusoïdal en entrée et le spectre révèle la présence d'une fréquence basse($f_b$) et d'une fréquence haute($f_h$).

On peut ainsi dresser le tableau suivant, qui correspond au relevé des fréquences basses et hautes d'un panel de fréquences de base allant de 20Hz à 50000hz.

| f(Hz) | 20 | 40 | 80 | 100 | 1000 | 2000 | 4000 | 15000 | 20000 | 40000 | 50000 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $f_B$(Hz) | 20 | 40 | 80 | 100 | 1000 | 2000 | 4000 | 15000 | 20000 | 40000 | 38300 |
| $f_H$(kHz) | 44,08 | 44,04 | 44,02 | 44 | 43,1 | 42,1 | 40,1 | 29,1 | 24,1 | 4 | 5,8 |
| Audible | Oui | Oui | Oui | Oui | Oui | Oui | Oui | Oui | Non | Non | Non |

Avec $f_E$=44100 Hz, (Taux d'échantillonnage)

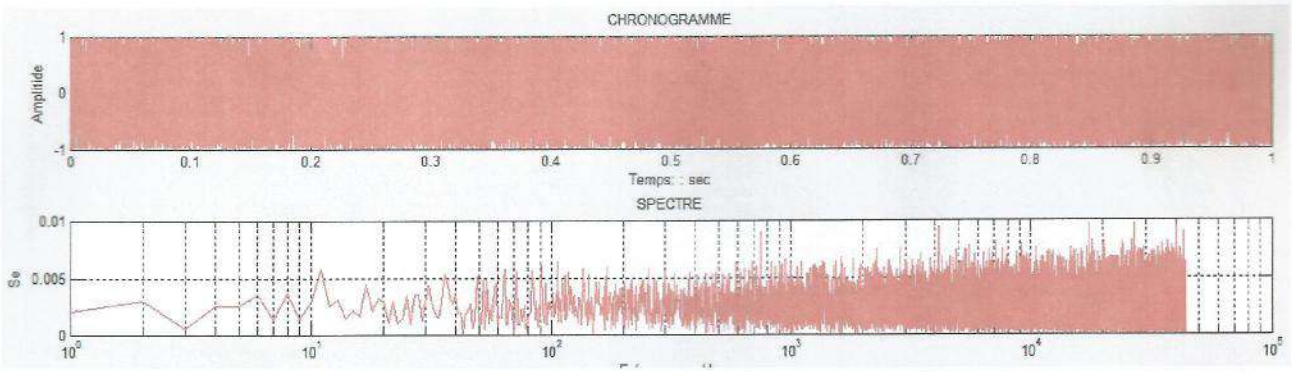
On peut remarquer que $f_E=f+f_H$ ou encore $f_E=f_B+f_H$ mais ce n'est pas vérifié sur les hautes fréquences, on peut donc penser qu'il existe une partie linéaire et une autre partie non-linéaire (comme sur les AOP).

Sur les signaux à 40kHz et à 50kHz, la fréquence du signal réellement entendu correspond en réalité aux fréquences basses correspondantes.

Maintenant, programmons un bruit blanc. C'est un signal contenant toutes les fréquences mais avec des amplitudes aléatoires. Il est nommé ainsi par analogie avec la lumière blanche, qui contient toutes les couleurs.
La perception de ce bruit blanc est celle d'un souffle, typique de celui entendu sur les téléviseurs CRT lorsque l'on bascule sur une chaîne mal programmée.

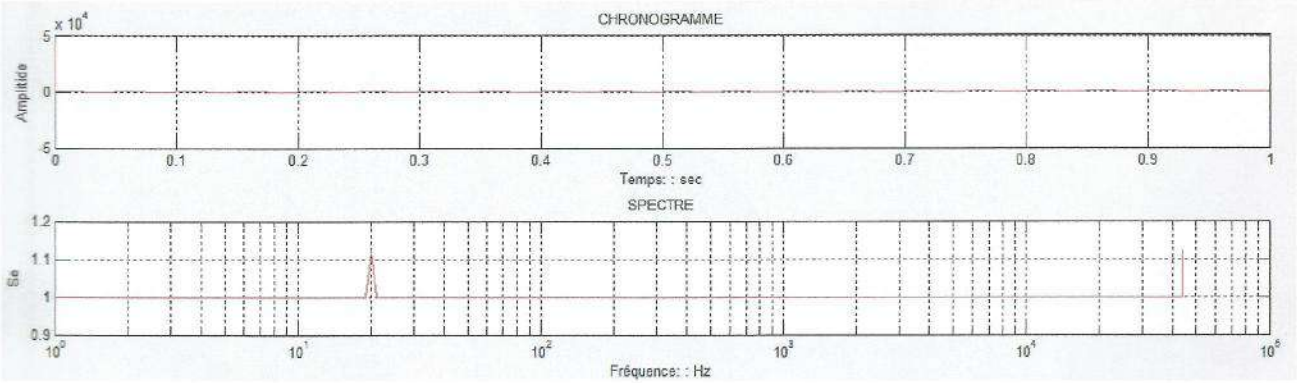
Le code MATLAB correspondant est :

```
e=(rand(1,N)-0.5)*2;
```




Le spectre du bruit blanc s'apparente ici à une pseudo sinusoïde à amplitude croissante et dont la pseudo période diminue en avançant sur l'axe des abscisses.

Comparons maintenant cela à un signal impulsionnel :




Le code correspondant au signal d'entrée est :

```
t0=3; e(t0)=N;
```

La perception auditive est ici une sorte de « cliquetis ».
Le spectre de ce signal impulsionnel ressemble au spectre d'un sinus à une fréquence réelle, avec la présence de deux « pics ».

## 2.Synthèse de filtres numériques à usage audio

Pour la synthèse des différents filtres qui vont servir, nous devons prendre en compte des paramètres supplémentaires, ce sont les constantes de temps k, k1 et k2 des dérivateurs de chaque filtre :

On a alors :

$$k=\frac{1}{\omega c}$$

$$k_1=\frac{1}{q\omega 0}=\frac{2\pi\Delta f}{\omega 0^2}$$

$$k_2=\frac{q}{\omega 0}=\frac{1}{2\pi\Delta f}$$

(avec $\Delta\omega=2\pi\Delta f$ (largeur de bande) et $q=\frac{\omega 0}{\Delta\omega}$ (facteur de qualité))

$\omega c$ étant la pulsation de coupure des filtres (I)
$\omega 0$ étant la pulsation centrale du filtre passe bande.

Ceci dit, il est également possible de programmer ce banc de filtres sans en exprimer la sortie intermédiaire en s'inspirant d'une mise en cascade de type s(t), s(t-1), s(t-2), etc...
L'avantage de cette méthode est d'éliminer les sorties intermédiaires, ainsi cela nous permet de réduire les erreurs dûs au codage à la main.
De plus, il permet de repérer « l'ordre ».

## 3. Test des filtres audio (signal sinusoïdal)

Il est maintenant temps de tester ces filtres :
Nous allons commencer par un signal sinusoïdal composite, c'est-à-dire comprenant une basse fréquence et une haute fréquence.

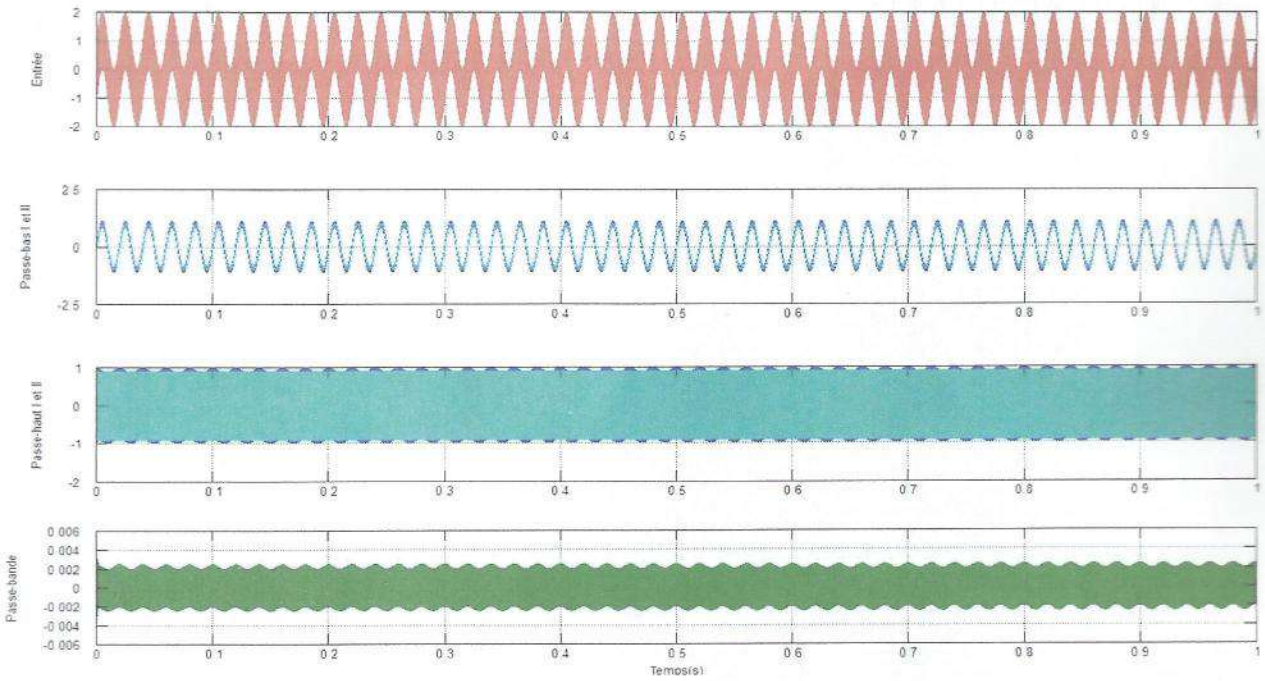
Nous devons mettre sous MATLAB l'entrée suivante :

e(t)=sin(2*pi*50*t/N)+sin(2*pi*10000*t/N)

Nous avons bien deux composantes :
-La première est une basse fréquence (50 Hz)
-La deuxième est une haute fréquence (10 kHz)

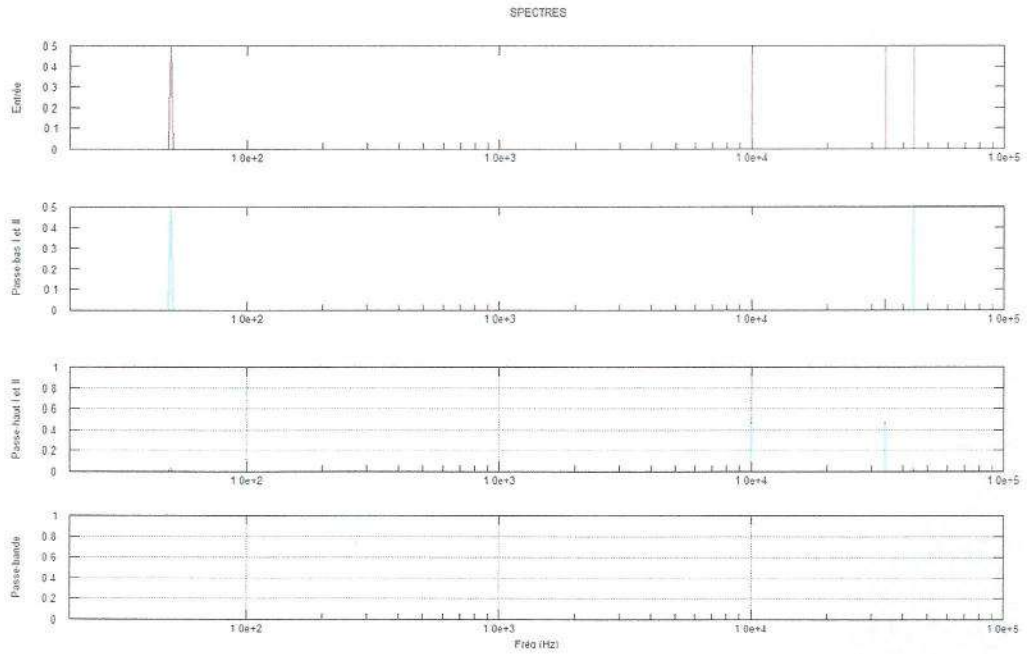
La présence du quotient par le nombre d'échantillons N se justifie ici par la nécessité de normaliser ces fréquences.

Voici les chronogrammes correspondant respectivement à l'entrée, aux filtres passe bas (I et II), aux filtres passe haut (I et II) et enfin au filtre passe bande.




On voit bien comment à partir d'un signal composite, on peut en observer la décomposition avec l'utilisation de ces trois filtres.

Observons maintenant les spectres correspondants :




Limitons nous à l'intervalle [0 ; 10000] :
Dans le signal d'entrée, on observe deux « pics » qui représentent la basse et la haute fréquence de notre signal composite (50 Hz et 10000 Hz).

Ici, le résultat du filtrage est évident :
-Dans le spectre correspondant au filtrage passe bas, on ne retrouve que le « pic » correspondant à la basse fréquence.
-Dans le spectre correspondant au filtrage passe haut, on ne retrouve plus que le « pic » correspondant à la haute fréquence.
-Le filtre passe bande ne révèle rien, ce qui est normal puisqu'il existe pas de fréquence intermédiaire en entrée.

Ceci est validé par notre écoute de ces différents signaux filtrés, ainsi après un filtrage passe bas, la partie aigue du son est fortement atténuée, ce qui est également le cas après un filtrage passe haut où c'est la partie grave du son qui tend à être négligeable.

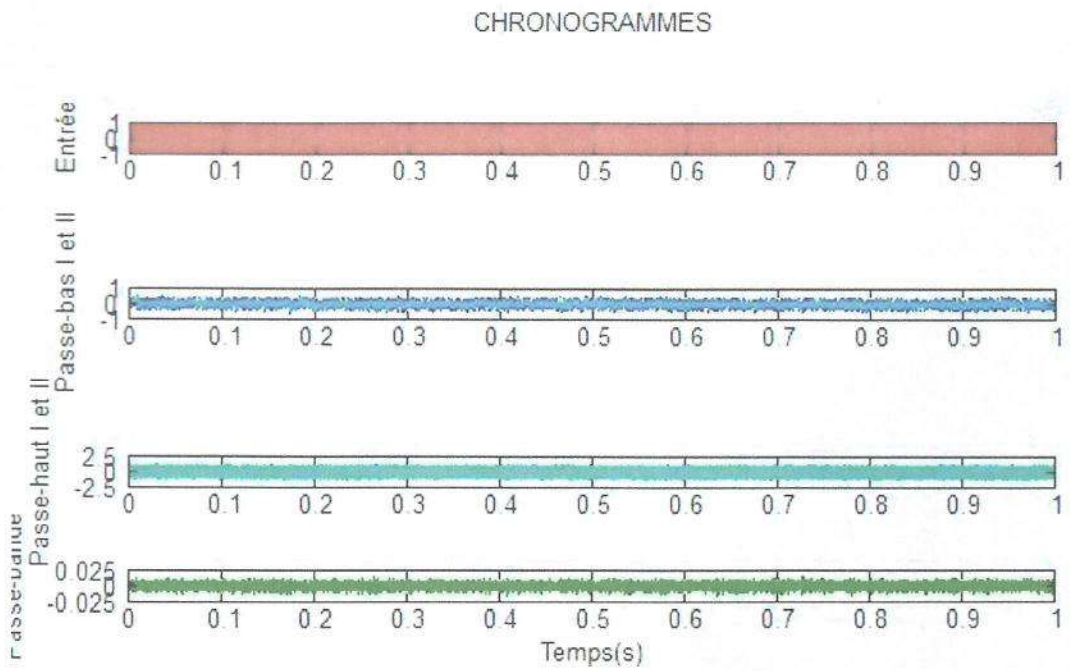
## 4. Test des filtres audio (bruit blanc)

Testons maintenant ces filtres avec un bruit blanc en entrée, ce qui nous permettra d'observer l'effet de ces filtres sur l'ensemble des fréquences audibles.

Sous MATLAB, il faut mettre en entrée :

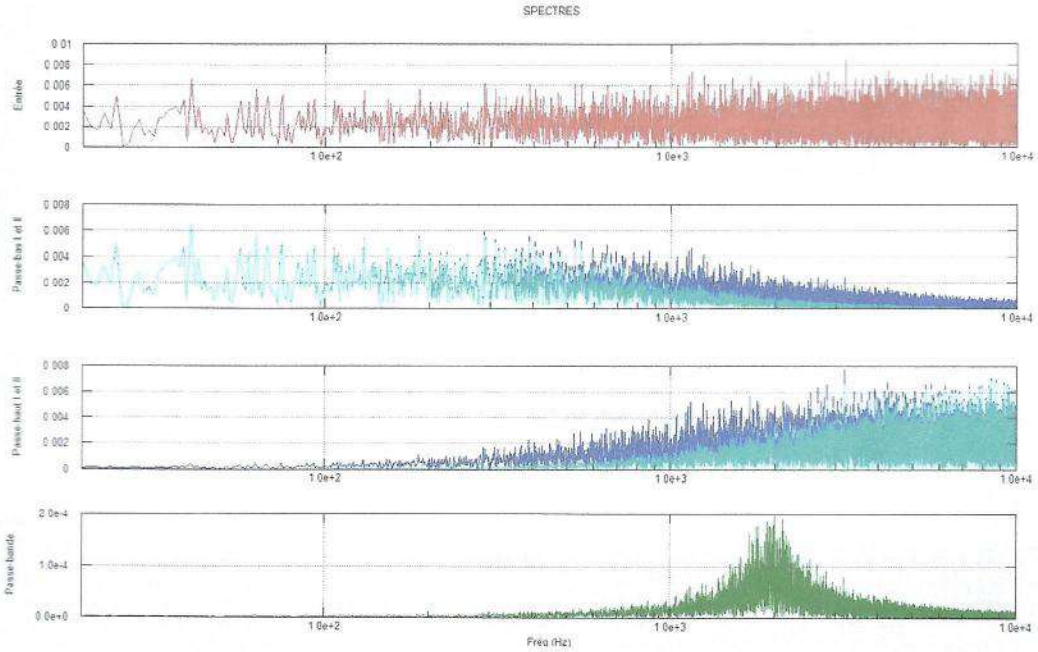
```
e=(rand(1,N)-0.5)*2 ;
```

Observons les chronogrammes correspondants :




Nous retrouvons le même constat que pour le signal composite, à savoir une décomposition du signal d'entrée par les différents filtres.

Du côté des spectres :




On retrouve toujours le même constat, nous avons d'abord le bruit blanc en entrée, que nous avons introduit précédemment.

Ici, le passe bas va conserver la partie « gauche » du signal de base, à savoir les basses fréquences et atténuer le reste, alors que le passe haut va conserver la partie « droite » du signal et atténuer ainsi les basses fréquences.

Le bruit blanc contenant également des fréquences « intermédiaires », on a ici la présence d'un spectre pour la partie du passe bande, qui conserve justement les fréquences à « l'interface » d'atténuation des filtres passe bas et passe haut.

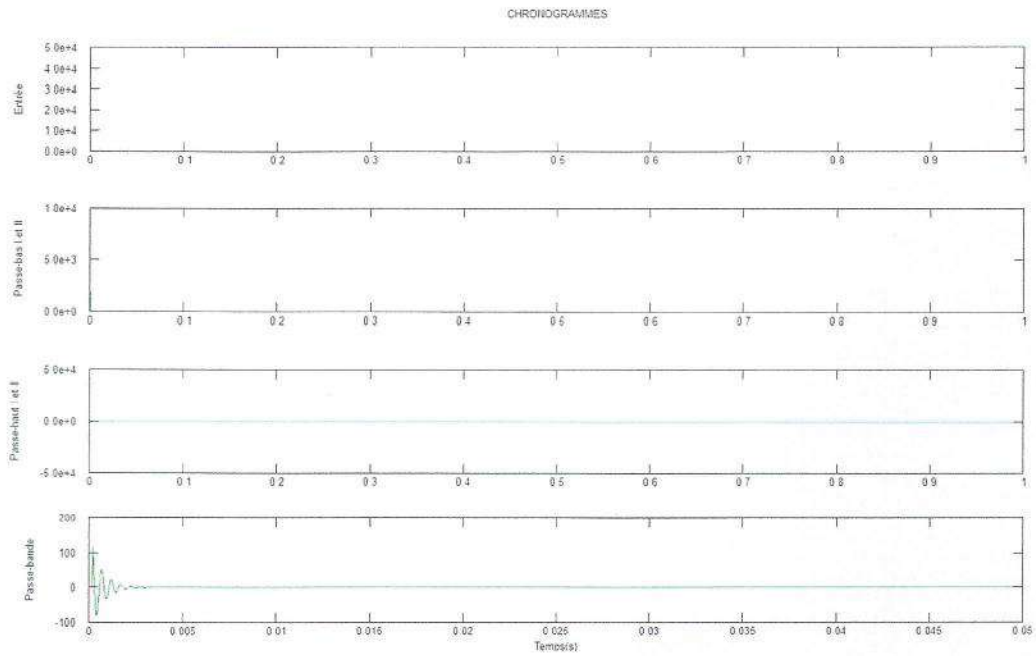
## 5. Test des filtres audio (impulsion)

Nous allons enfin tester ces filtres audio cette fois-ci avec en entrée une impulsion (qui a également été présenté plus tôt).

Sous MATLAB, il faut mettre en entrée :

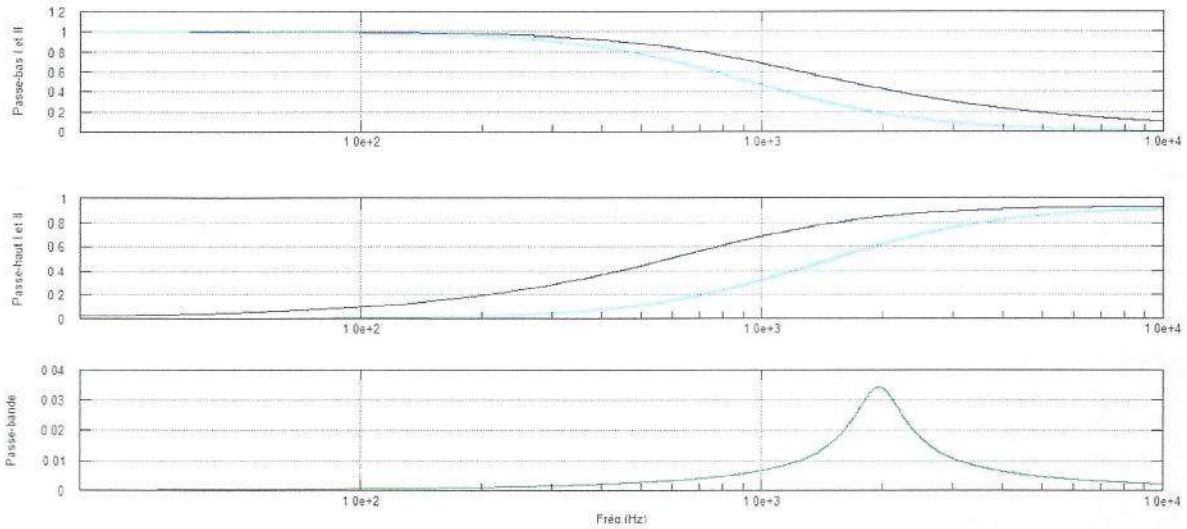
```
e=zeros(1,N) ; e(10)=N ;
```

Les chronogrammes suivants ne révèlent rien de concret, faute au signal impulsionnel qui est par nature fugace :

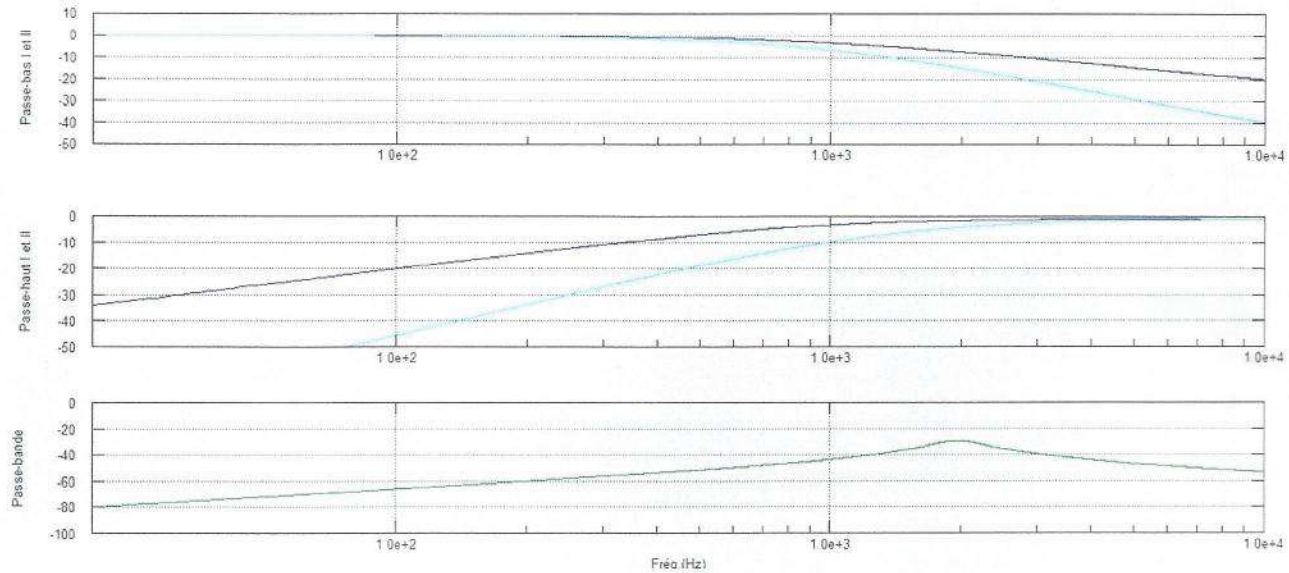



Cependant, on retrouve une pseudo oscillation sur le chronogramme du spectre passe bande dont la fréquence est à vue d'œil d'environ 200 Hz.

Observons maintenant les spectres associés (vue non logarithmique):




Echelle logarithmique :

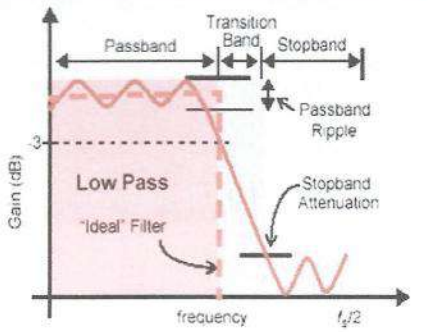



Nous retrouvons à peu près la même tendance que pour un bruit blanc, à la différence que les différentes courbes se révèlent être plus « lisses » grâce au signal impulsionnel.

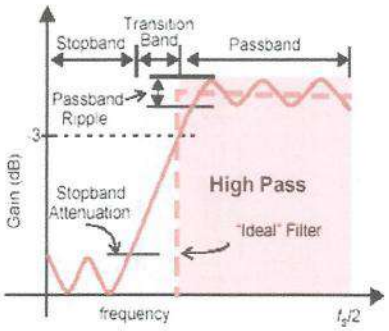
Notons ici que nous avons utilisé une échelle logarithmique pour observer le deuxième jeu de spectres.

En observant bien ces spectres, il est légitime de se dire qu'elles ont le même aspect que leur diagramme de Bode respectifs :

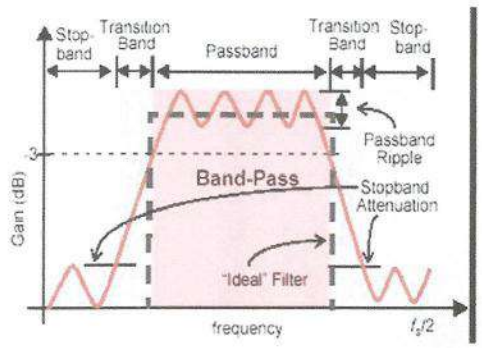
Filtre passe bas :




Filtre passe haut :




Filtre passe bande :




C'est pourquoi, on peut tenter de déterminer graphiquement les caractéristiques de chacun des ces filtres :

<u>Fréquence de coupure</u> :

-Filtre passe bas I : 1000 Hz

-Filtre passe bas II : 750 Hz

-Filtre passe haut I : 1000 Hz

-Filtre passe haut II : 1250 Hz

<u>Filtre passe bande</u> :

-Fréquence centrale : 2000 Hz

-Pente en basse fréquence : +20 dB/décade

-Pente en haute fréquence : -20 dB/décade

Voilà qui conclut notre projet, nous disposons à présent de nos propres filtres fonctionnels !

# CONCLUSION

Les filtres numériques sont des outils très intéressants dans la mesure où ils sont fabriqués non pas physiquement mais à la suite d'une programmation à l'aide d'outils mathématiques.

C'est comme ça qu'à l'aide d'un calculateur numérique tel qu'un ordinateur associé à un logiciel de programmation comme MATLAB que l'on peut modéliser toutes sortes de filtres numériques.

Ainsi, il est possible de tester toutes sortes de paramètres tels que la modification des constantes de temps, du facteur de qualité, de l'amortissement, afin de concevoir le filtre correspondant le plus à notre besoin.

Ces filtres sont de plus présents dans notre vie quotidienne même s'ils passent inaperçus : On peut les retrouver par exemple dans les amplificateurs pour instruments de musique : Leur action se retrouve dans les potentiomètres « Bass », « Middle », « Treble » !

Voilà qui conclut ce projet, nous espérons que vous aurez autant de plaisir à lire ce dossier que nous en avons eu à le rédiger.