Filtres à réponse impulsionnelle finie

1. Signal discret et transformée en Z

Soit $x(t)$ un signal continu échantillonné avec une période T_e. Le signal discret correspondant est la suite

$$x_n = x(nT_e) \tag{1}$$

Considérons la transformée de Fourier du signal continu :

$$\tilde{X}(f) = \int_{-\infty}^{+\infty} x(t) \exp(-i2\pi\, f\, t)\, dt \tag{2}$$

Une approximation de la transformée de Fourier est obtenue à partir du signal discret par la méthode des rectangles :

$$\tilde{X}(f) \simeq \sum_{n=-\infty}^{n=+\infty} x_n \exp(-i2\pi\, f\, n\, T_e) \tag{3}$$

Lorsque la somme est stoppée à un rang fini, on retrouve la transformée de Fourier discrète.

On pose :

$$Z = \exp(i2\pi\, f\, T_e) \tag{4}$$

La transformée en Z ([1],[2]) du signal discret est définie par :

$$X(Z) = \sum_{n=-\infty}^{n=+\infty} x_n Z^{-n} \tag{5}$$

La transformée en Z est ainsi l'équivalent pour les signaux discrets de la transformée de Fourier pour les signaux continus. En reportant l'expression (4) de Z dans la transformée en Z, on retrouve en effet l'approximation (3) de la transformée de Fourier.

2. Système linéaire causal à réponse impulsionnelle finie

2.a. Définition

Un système linéaire causal à réponse impulsionnelle finie calcule une suite y_n à partir de la suite x_n par la relation :

$$y_n = \sum_{k=0}^{N-1} h_k x_{n-k} \tag{6}$$

où les N coefficients h_k sont des constantes réelles. La valeur du signal discret y_n à l'instant n est donc obtenue par une combinaison linéaire des N valeurs précédentes du signal x_n. Ce système est dit causal car l'état de la sortie ne dépend que des états antérieurs de l'entrée.

On appelle impulsion un signal discret défini par :

$$x_0 = 1 \tag{7}$$
$$x_n = 0 \; si \; n \neq 0 \tag{8}$$

Pour une impulsion en entrée, le système linéaire fournit en sortie le signal :

$$y_n = h_n \tag{9}$$

C'est pourquoi la suite h_n est appelée la réponse impulsionnelle (discrète) du système. La réponse impulsionnelle est ici finie puisqu'elle comporte N valeurs.

On remarque que le signal de sortie est le produit de convolution du signal d'entrée par la réponse impulsionnelle.

2.b. Fonction de transfert en Z

Ce système constitue un filtre à réponse impulsionnelle finie (filtre RIF). Considérons alors la transformée en Z du signal de sortie :

$$Y(Z) = \sum_{n=-\infty}^{n=+\infty} \sum_{k=0}^{N-1} h_k x_{n-k} Z^{-n} \tag{10}$$
$$= \sum_{n'=-\infty}^{n'=+\infty} \sum_{k=0}^{N-1} h_k x_{n'} Z^{-n'} Z^{-k} \tag{11}$$
$$= \left(\sum_{k=0}^{N-1} h_k Z^{-k} \right) \sum_{n'=-\infty}^{n'=+\infty} x_{n'} Z^{-n'} \tag{12}$$
$$= H(Z)X(Z) \tag{13}$$

où $H(Z)$ est la fonction de transfert en Z définie par :

$$H(Z) = \sum_{k=0}^{N-1} h_k Z^{-k} \tag{14}$$

La transformée en Z du signal de sortie est donc le produit des transformées en Z du signal d'entrée et du filtre (les transformées de Fourier des signaux continus vérifient un théorème analogue).

2.c. Réponse fréquentielle

Considérons un signal d'entrée sinusoïdal, de fréquence f et d'amplitude unité :

$$x_n = \exp(i2\pi \, f \, nT_e) \tag{15}$$

Le signal en sortie s'écrit :

$$y_n = \sum_{k=0}^{N-1} h_k x_{n-k} \tag{16}$$

$$= \exp(i2\pi\, n\, f\, T_e) \sum_{k=0}^{N-1} h_k \exp(-i2\pi\, f\, k\, T_e) \tag{17}$$

$$= \exp(i2\pi\, n\, f\, T_e) H(Z) \tag{18}$$

$$= x_n H(Z) \tag{19}$$

avec

$$Z = \exp(i2\pi\, f\, T_e) \tag{20}$$

La réponse fréquentielle du filtre se déduit donc de la fonction de transfert en Z :

$$H(f) = \sum_{k=0}^{N-1} h_k \exp(-i2\pi\, k\, f\, T_e) \tag{21}$$

On remarque que cette réponse fréquentielle est une fonction de fT_e, qui est le rapport de la fréquence du signal sinusoïdal sur la fréquence d'échantillonnage. D'après le théorème de Shannon, ce rapport doit être inférieur à 1/2.

En remarquant que la réponse fréquentielle est de périodicité $f_e = 1/T_e$ (la fréquence d'échantillonnage), considérons les N fréquences suivantes :

$$f_n = \frac{n}{NT_e} \tag{22}$$

On a alors

$$H(f_n) = \sum_{k=0}^{N-1} h_k \exp(-i2\pi\, k\, \frac{n}{N}) \tag{23}$$

qui est (à un facteur N près) la transformée de Fourier discrète de la réponse impulsionnelle.

2.d. Exemple

Considérons comme exemple simple un filtre qui réalise la moyenne arithmétique de deux valeurs consécutives de l'entrée :

$$y_n = \frac{1}{2}(x_n + x_{n-1}) \tag{24}$$

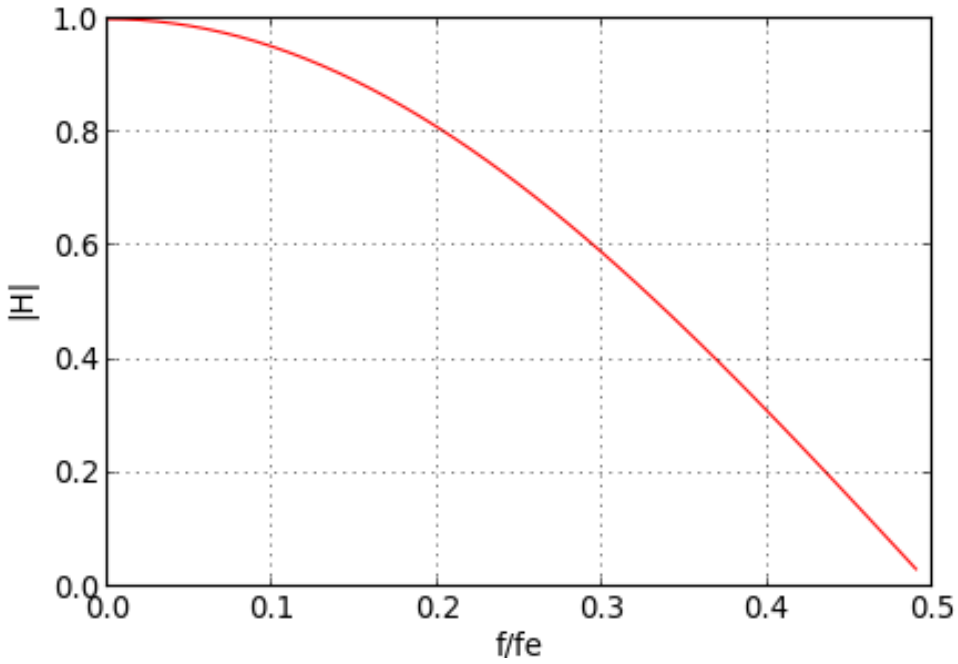
Sa fonction de transfert en Z est :

$$H(Z) = \frac{1}{2}(1 + Z^{-1}) \tag{25}$$

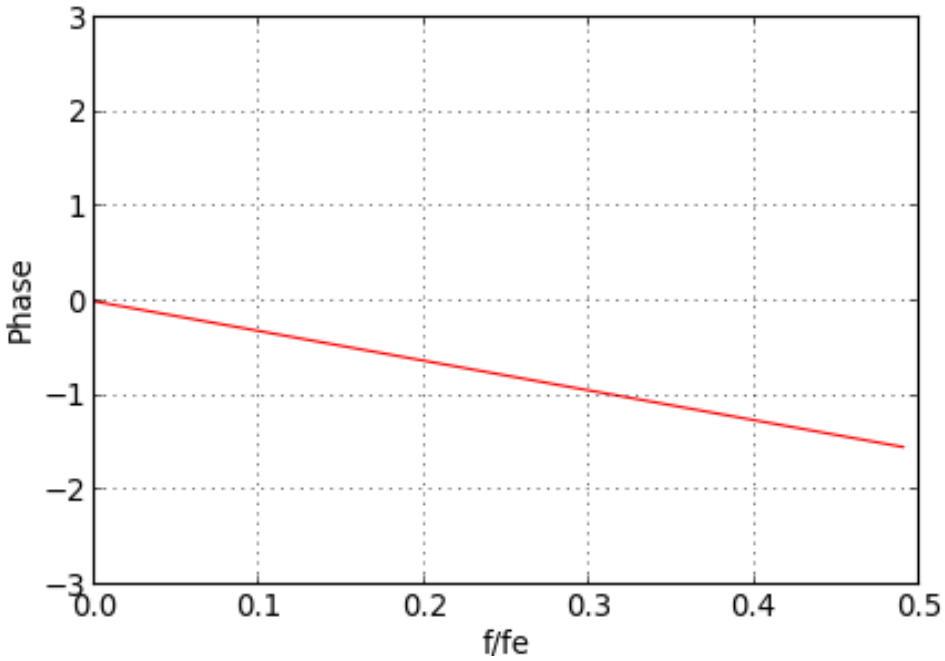
Voyons le tracé sa réponse fréquentielle avec Python :

```
import numpy as np
import math
import cmath
```

```
from matplotlib.pyplot import *
def H(Z):
    return 0.5*(1.0+Z**(-1))
def Hf(f):
    return H(np.exp(1j*2*math.pi*f))
f = np.arange(start=0.0,stop=0.5,step=0.01)
figure(figsize=(6,4))
plot(f,np.absolute(Hf(f)),'r')
xlabel('f/fe')
ylabel('|H|')
axis([0,0.5,0,1])
grid()
```




```
phase = np.unwrap(np.angle(Hf(f)))
figure(figsize=(6,4))
plot(f,phase,'r')
xlabel('f/fe')
ylabel('Phase')
axis([0,0.5,-3,3])
grid()
```

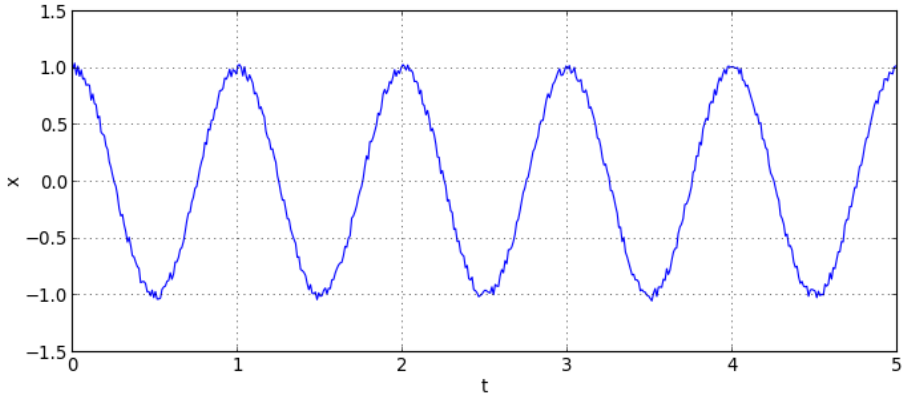


Le filtre moyenneur est un filtre passe-bas (assez peu sélectif). Le déphasage varie linéairement avec la fréquence. Cela se vérifie sur l'expression suivante de la réponse fréquentielle :

$$H(f) = \cos(\pi\, f\, T_e) \exp(-i\pi\, f\, T_e) \tag{26}$$

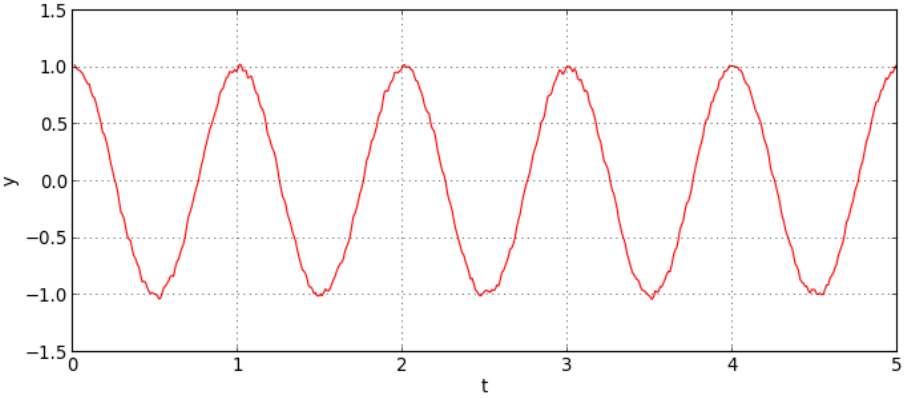
Pour simuler l'action de ce filtre sur un signal, considérons le signal continu suivant et son échantillonnage :

```
import random
def signal(t):
    return math.cos(2*math.pi*t)+0.05*random.uniform(-1.0,1.0)
fe = 100
te=1.0/fe
t = np.arange(start=0.0,stop=5.0,step=te)
n = t.size
x = np.zeros(n)
for k in range(n):
    x[k] = signal(te*k)
figure(figsize=(10,4))
plot(t,x)
xlabel('t')
ylabel('x')
axis([0,5,-1.5,1.5])
grid()
```



Pour obtenir le signal discret filtré, il faut effectuer la convolution avec la réponse impulsionnelle. La fonction `scipy.signal.convolve` permet de le faire. On utilise l'option `mode='valid'` , qui restreint le calcul aux points valides. Ainsi le premier point calculé est la moyenne des deux premiers et le dernier point est la moyenne des deux dernier. Le premier point calculé correspond à l'instant T_e, c'est pourquoi l'échelle de temps doit être modifiée.

```
import scipy.signal
h = np.array([0.5,0.5])
y = scipy.signal.convolve(x,h,mode='valid')
ny = y.size
ty = np.zeros(ny)
for k in range(ny):
    ty[k] = te+te*k
figure(figsize=(10,4))
plot(ty,y,'r')
xlabel('t')
ylabel('y')
axis([0,5,-1.5,1.5])
grid()
```



3. Filtres RIF à phase linéaire

3.a. Définition et propriétés

Pour un filtre à phase linéaire, le déphasage est une fonction linéaire de la fréquence. La réponse fréquentielle a donc la forme suivante :

$$H(f) = G(f)\exp(-i\phi(f)) \tag{27}$$
$$\phi(f) = 2\pi f\tau \tag{28}$$

La phase d'une composante de fréquence f devient en sortie du filtre :

$$2\pi\, f\, t - \phi(f) = 2\pi\, f(t-\tau) \tag{29}$$

Toutes les fréquences du signal subissent le même décalage τ en traversant le filtre. τ est le temps de propagation. Si toutes les composantes spectrales sont dans la bande passante, pour laquelle $G(f) = 1$, on obtient :

$$y(t) = x(t-\tau) \tag{30}$$

La forme du signal n'est pas modifiée par le filtrage en bande passante. On définit le retard en nombre d'échantillons par $P = f_e\tau$ (P entier). La fonction de transfert en Z d'un filtre à phase linéaire dans la bande passante est donc :

$$H_{bp}(Z) = Z^{-P} \tag{31}$$

En isolant le terme contenant la phase, la réponse fréquentielle s'écrit d'après l'expression(15) :

$$H(f) = \exp(-i2\pi\, f\, P\, T_e)\sum_{k=0}^{N-1} h_k \exp(-i2\pi(k-P)fT_e) \tag{32}$$

Après un changement de variable dans la somme, on en déduit l'expression du gain :

$$G(f) = \sum_{k=-P}^{N-1-P} h_{k+P}\exp(-i2\pi\, k\, f\, T_e) \tag{33}$$

On veut que $G(f)$ soit réel. Il faut donc que la réponse impulsionnelle soit symétrique par rapport à l'indice P, c'est-à-dire :

$$h_{k+P} = h_{-k+P} \tag{34}$$

On en déduit que $N = 2P+1$. N est donc impair. Il est toutefois possible d'obtenir un filtre RIF à phase linéaire avec N pair en posant $P - \frac{1}{2} = f_e\tau$.

3.b. Synthèse par série de Fourier

Soit la suite définie par :

$$g_k = h_{k+P} \quad (-P \leq k \leq P) \tag{35}$$

Elle vérifie la propriété $g_{-k} = g_k$. La réponse fréquentielle s'écrit

$$H(f) = \exp(-i2\pi\, f\, \tau) \sum_{k=-P}^{P} g_k \exp(-i2\pi\, k\, f\, T_e) = \exp(-i2\pi\, f\, \tau) G(f) \tag{36}$$

En considérant la limite $P \to \infty$, on obtient alors :

$$G(f) = \sum_{k=-\infty}^{+\infty} g_k \exp(-i2\pi\, k\, f\, T_e) \tag{37}$$

On obtient un filtre à phase linéaire mais à réponse impulsionnelle infini (impossible à réaliser). Cette expression a toutefois l'avantage de faire apparaître g_k comme les coefficients de Fourier de la fonction $G(f)$, fonction de période $f_e = 1/T_e$. On a ainsi :

$$g_k = \frac{1}{f_e} \int_0^{f_e} G(f) \exp\left(i2\pi\, k \frac{f}{f_e}\right) df \tag{38}$$

Puisque $g_{-k} = g_k$, ces coefficients sont réels et la fonction $G(f)$ est paire.

Pour une fonction de gain $G(f)$ souhaitée, on calcule les coefficients de Fourier g_k puis on tronque la série de Fourier à un rang P de manière à obtenir la réponse impulsionnelle finie définie par :

$$h_k = g_{k-P} \quad (0 \leq k \leq 2P) \tag{39}$$

Cette méthode revient à appliquer un fenêtrage rectangulaire aux coefficients de Fourier. La fonction $G(f)$ effectivement obtenue après troncature est donc le produit de convolution de la fonction souhaitée par la transformée de Fourier de la fenêtre. En pratique, il est préférable d'appliquer d'autres types de fenêtres, comme les fenêtres de Hann ou de Hamming.

3.c. Exemple : filtre RIF passe-bas

La fonction de gain souhaitée est définie sur l'intervalle $[-f_e/2, f_e/2]$ par :

$$G(f) = 1 \; pour \; -f_c \leq f \leq f_c \tag{40}$$

$$= 0 \; pour \; f_c < |f| \leq \frac{f_e}{2} \tag{41}$$

Les coefficients de Fourier de cette fonction sont :

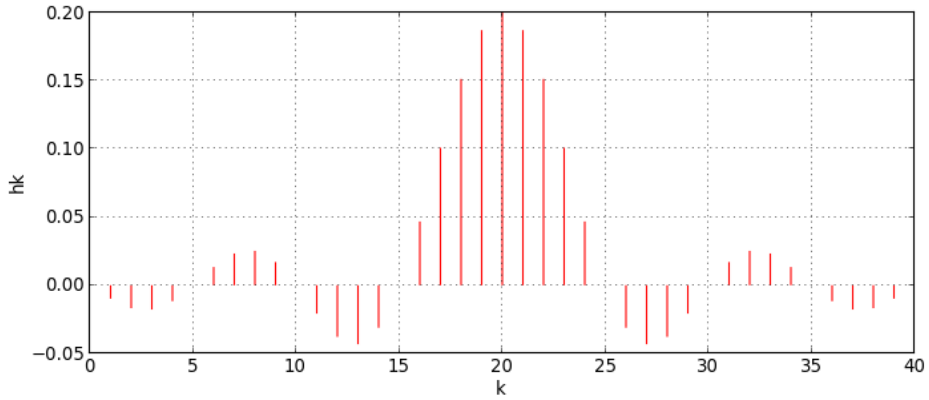
$$g_k = \frac{1}{k\pi} \sin\left(2\pi\, k \frac{f_c}{f_e}\right) \tag{42}$$

Le résultat peut s'exprimer avec la fonction sinus cardinal et ne dépend que du rapport de la fréquence de coupure sur la fréquence d'échantillonnage $a = \frac{f_c}{f_e}$:

$$g_k = 2a\, sinc(2\pi\, k\, a) \tag{43}$$

Considérons le cas d'une troncature par une fenêtre rectangulaire au rang P. On fixe la fréquence de coupure, le rang P de troncature et on calcule la réponse impulsionnelle :

```
a=0.1
P=20
def g(k):
    return 2*a*np.sinc(2*(k-P)*a)
N=2*P+1
liste_k = np.arange(start=0,stop=N)
h = g(liste_k)
figure(figsize=(10,4))
vlines(liste_k,[0],h,'r')
xlabel('k')
ylabel('hk')
grid()
```



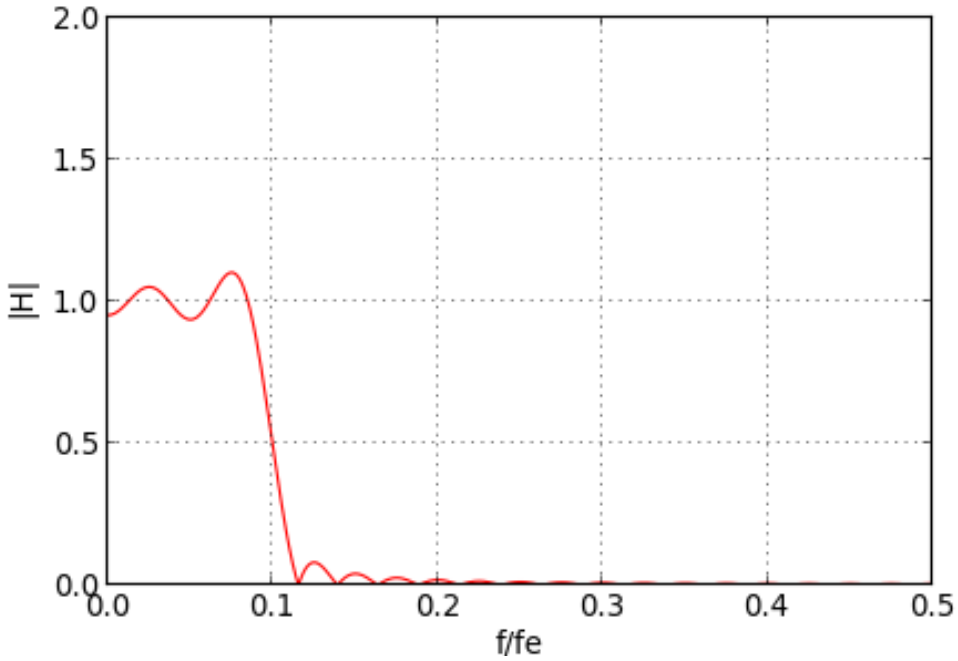
La réponse fréquentielle est donnée par la fonction suivante :

```
def Hf(f):
    s = 0.0
    for k in range(N):
       s += h[k]*np.exp(-1j*2*math.pi*k*f)
    return s
```

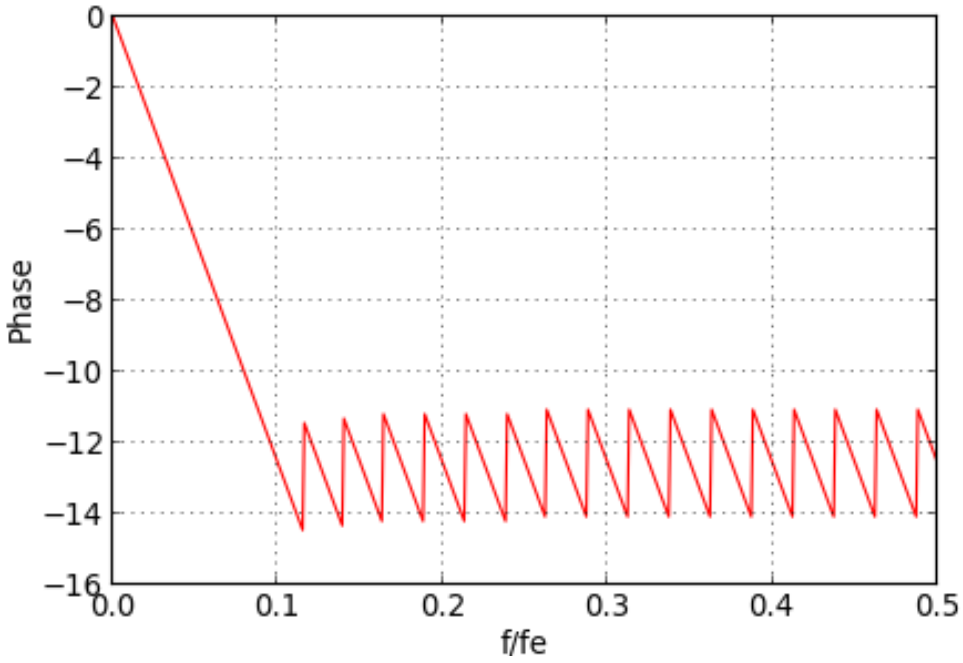
On trace le gain et le déphasage :

```
f = np.arange(start=0.0,stop=0.5,step=0.001)

figure(figsize=(6,4))
plot(f,np.absolute(Hf(f)),'r')
xlabel('f/fe')
ylabel('|H|')
axis([0,0.5,0,2])
grid()
```




```
phase = np.unwrap(np.angle(Hf(f)))
figure(figsize=(6,4))
plot(f,phase,'r')
xlabel('f/fe')
ylabel('Phase')
grid()
```

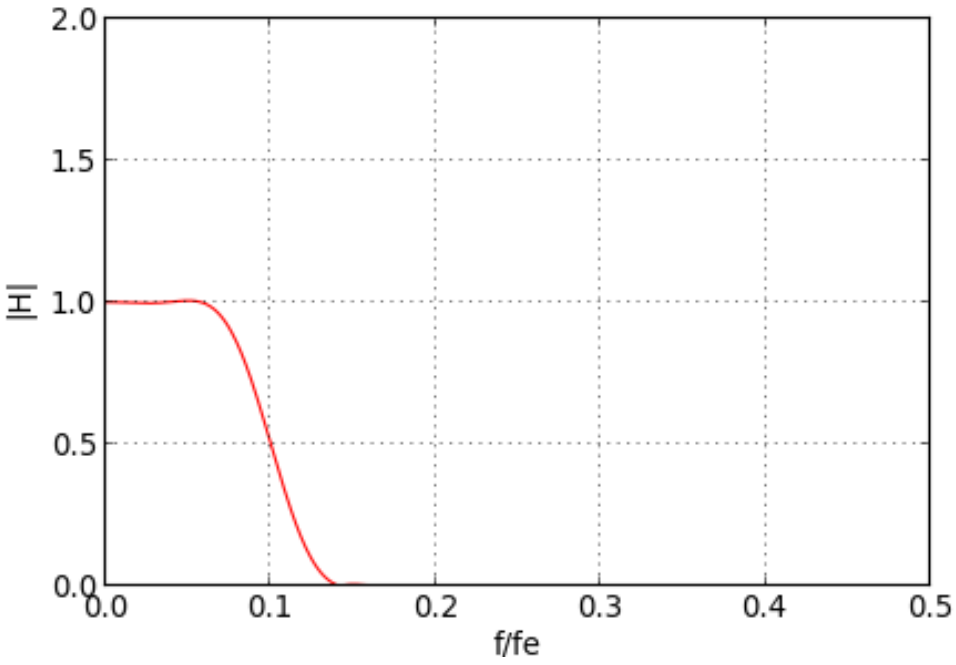


On constate que la phase est bien linéaire dans la bande passante, mais le gain présente des ondulations très fortes. Dans la bande atténuée, il y a des discontinuités de π de la

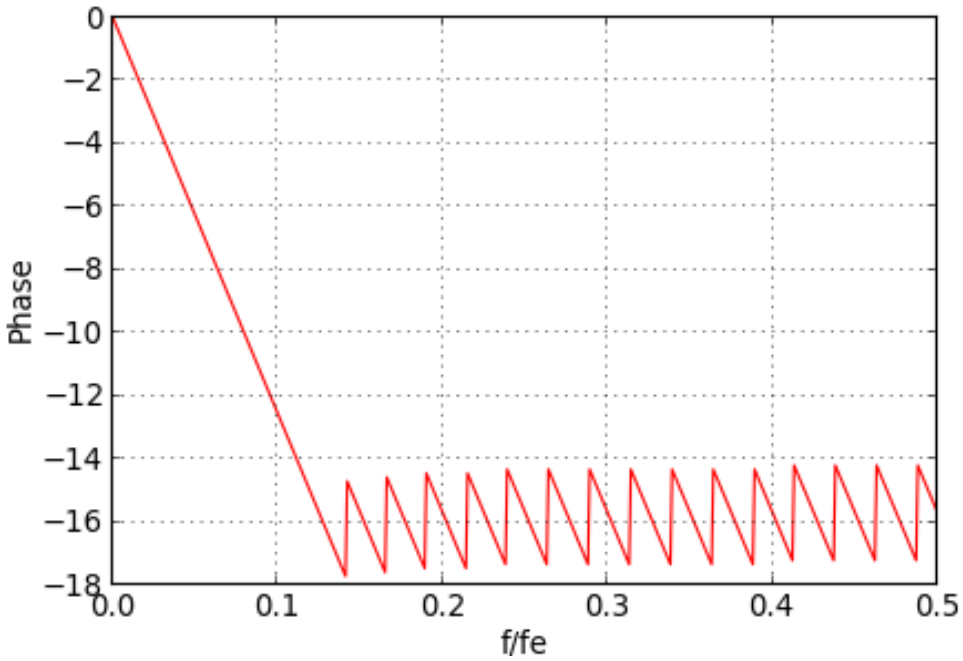
phase. Bien sûr, les différences par rapport à la fonction de transfert souhaitée sont dûes à la troncature de la réponse impulsionnelle.

Essayons une troncature par une fenêtre de Hann :

```
h = h*scipy.signal.hann(N)
figure(figsize=(6,4))
plot(f,np.absolute(Hf(f)),'r')
xlabel('f/fe')
ylabel('|H|')
axis([0,0.5,0,2])
grid()
```




```
phase = np.unwrap(np.angle(Hf(f)))
figure(figsize=(6,4))
plot(f,phase,'r')
xlabel('f/fe')
ylabel('Phase')
grid()
```



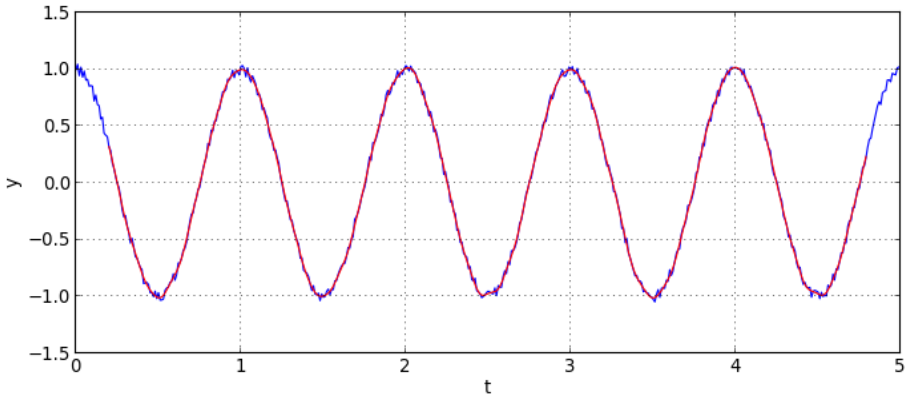
Les ondulations dans la bande passante et dans la bande atténuée sont considérablement réduites. La linéarité de la phase dans la bande passante est toujours assurée.

Lorsque ce filtre est utilisé dans un système de traitement du signal temps-réel, pour un signal situé dans la bande passante, la sortie présente un retard de $\tau = PT_e$ par rapport à l'entrée. Cela se comprend aisément en remarquant que le point prépondérant pour le calcul de y_n est x_{n-P} (terme g_0). Pour obtenir un filtre très sélectif, il faut augmenter P. Si le retard τ doit rester fixe, il faut parallèlement augmenter la fréquence d'échantillonnage.

Pour l'analyse de données expérimentales (échantillon fini), le retard τ n'a aucune signification physique et peut être annulé de la manière suivante. Soit Q le nombre de points de l'échantillon. Le premier point de la sortie calculé est y_{2P} puisqu'il faut au moins $2P+1$ valeurs antérieures de x_n pour appliquer la convolution. Ce premier point est en retard de $\tau = PT_e$ par rapport à l'entrée. Annuler le retard revient donc à attribuer l'instant $(2P-P)T_e = PT_e$ à ce premier point. Ainsi, le dernier point calculé y_{Q-2P} correspond à l'instant $(Q-P)T_e$. L'application du filtrage revient donc à enlever les P premiers instants et les P derniers instants de l'échantillon. Voyons cela sur le signal déjà défini plus haut.

```
y = scipy.signal.convolve(x,h,mode='valid')
ny = y.size
ty = np.zeros(ny)
for k in range(ny):
    ty[k] = P*te+te*k
figure(figsize=(10,4))
plot(t,x,'b')
plot(ty,y,'r')
xlabel('t')
ylabel('y')
axis([0,5,-1.5,1.5])
```

```
grid()
```



Le signal traité est bien synchrone avec le signal d'origine ; les P premiers et les P derniers points sont perdus. Dans un filtre numérique temps-réel, la sortie serait retardée, c'est-à-dire décalée de $\tau = PT_e$ vers la droite.

Remarquer l'utilisation de l'option `mode='valid'` qui restreint le calcul de la convolution aux points valides. Ainsi le premier point de la convolution est calculé à partir des $2P + 1$ premiers points de `x` et le dernier point est calculé à partir des $2P1 + 1$ derniers points de `x`.

Références

[1] M. Bellanger, *Traitement numérique du signal*, (Dunod, 1998)
[2] E. Tisserand, J.F. Pautex, P. Schweitzer, *Analyse et traitement des signaux*, (Dunod, 2008)