TP_Filtrage_elem

J.-F. Bercher

November 24, 2013

Part I

TP Filtrage élémentaire

Le but de ce TP est de concevoir et appliquer différents filtres numériques sur un signal périodique de fréquence fondamentale f_0{=200 Hz}, échantillonné à la fréquence F_e{=8000 Hz}.

Ce signal est stocké dans le vecteur x sauvegardé dans le fichier sig1.npz. On peut le récupérer sous Python par l'instruction~: `f=numpy.load('sig1.npz')`

0.1 Mise en place des données

In [2]:
```
# -*- coding: utf-8 -*-
"""
Created on Tue Nov 19 15:46:37 2013
TP Filtrage
@author: bercherj@esiee.fr
"""

import scipy.io
from scipy.signal import lfilter
from numpy.fft import fft

# une fonction utilitaire
def freq(N,Fe=1):
    """ Retourne un vecteur de fréquences normalisées
    entre -0.5 et 0.5 sur N points """
    return (arange(N)/N-1/2)*Fe
```

Pour mémoire, les lignes suivantes permettent de lire un fichier Matlab, et de sauvegarder les résultats au format NumPy

In [3]:
```
#mat = scipy.io.loadmat('sig1.mat') # En fait mat est un dict
#mat.keys()   # dont les clés sont accessibles par .keys()
#x=mat['x'].flatten()
#m=mat['m'].flatten()
## On sauve les données au format numpy compressé
#numpy.savez('sig1',x=x, m=m)
```

In [4]:
```
# Pour recharger le truc :
f=numpy.load('sig1.npz')
#f.keys()
m=f['m']
```

```
x=f['x']

Fe=8000
Te=1/Fe
```

0.2 Analyse des des données :

On peut afficher le signal, en temps et en fréquence, par les instructions suivantes :

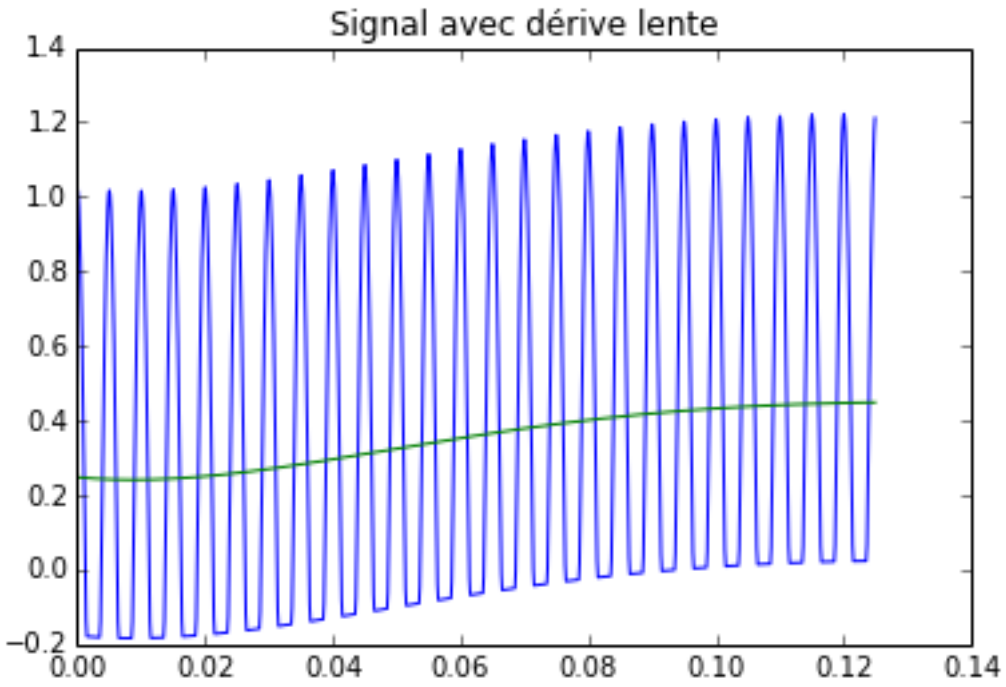
```
# Pour charger le truc :
f=numpy.load('sig1.npz')
m=f['m']
x=f['x']

Fe=8000
Te=1/Fe

# Affichage  temps
figure(1)
t=arange(len(x))*Te
plot(t,x,t,m)
title('Signal avec dérive lente')
show()
```

et

```
# Affichage fréquence
figure(2)
N=len(x)
f=freq(N)
plot(f,abs(fftshift(fft(x))))
xlim([-0.5, 0.5])
title('TF du signal (module)')
```

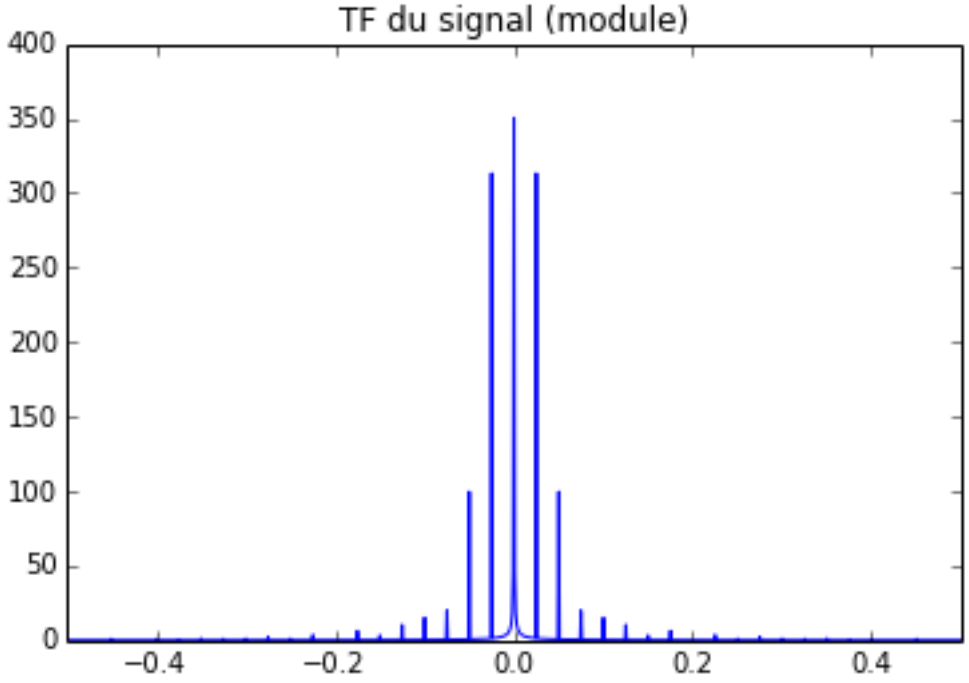
Commentez et interprétez.

In [5]:
```
# Affichage  temps
figure(1)
t=arange(len(x))*Te
plot(t,x,t,m)
title('Signal avec dérive lente')
show()
```



On note un signal temporel approximativement périodique, période ~ 40 et une dérive lente du niveau moyen

In [6]:
```
# Affichage fréquence
figure(2)
N=len(x)
f=freq(N)
plot(f,abs(fftshift(fft(x))))
xlim([-0.5, 0.5])
title('TF du signal (module)')
show()
```



On voit que la TF est constituée de raies, signe que le signal sous-jacent comporte une partie périodique. Ces raies sont espacées de 0.025, en fréquences réduites, ce qui correspond bien à une période de 40 points, que l'on retrouve sur le graphique temporel

0.3 Filtrage

On souhaite à présent modifier le contenu spectral du signal x par différents filtres numériques de fonction de transfert $H(z) = B(z)/A(z)$. Une fonction standard Python vous sera très utile :

- *lfilter* qui implémente l'équation aux différences~: cette fonction calcule le vecteur y des sorties d'un filtre numérique spécifié par le vecteur B des coefficients du numérateur $B(z)$, le vecteur A des coefficients du dénominateur $A(z)$, pour un vecteur d'entrées x, suivant l'instruction~: `y=filter(B,A,x)`
- *freqz* qui calcule la réponse fréquentielle $H(e^{j2\pi f/Fe})$ en module et phase pour un filtre spécifié par les deux vecteurs B et A des coefficients de la fonction de transfert~: `freqz(B,A)`

Calcul du filtre moyenneur

Le signal est affecté par une dérive lente de son niveau moyen. On cherche alors à extraire cette valeur moyenne à variations lentes, d'une part, et à calculer le signal débarrassé de cette dérive d'autre part. On notera $M(n)$ la dérive, et $x_c(n)$ le signal centré.

Partie théorique :

Quelle expression permet de calculer la moyenne du signal x sur une période ?

En déduire un filtre, de réponse impulsionnelle $g(n)$, qui calcule cette moyenne $M(n)$.

En déduire un autre filtre, de réponse impulsionnelle $h(n)$, qui élimine cette moyenne : $x_c(n) = x(n) - M(n) = x(n) * h(n)$. Donner l'expression de $h(n)$.

Donner les expressions de $G(z)$ et de $H(z)$.

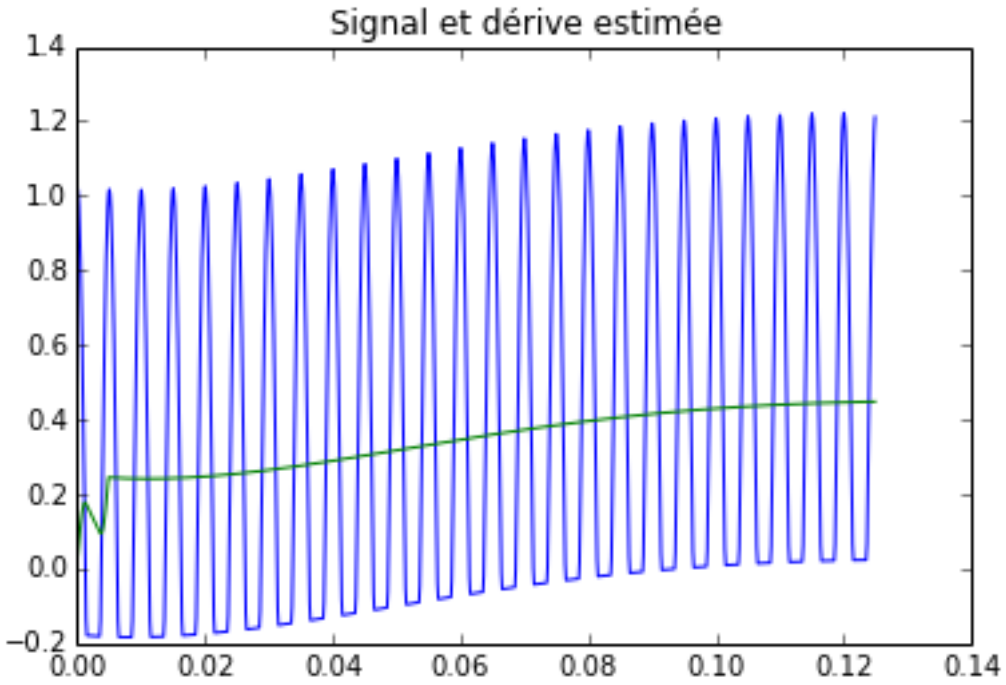
**Partie pratique **

Pour le filtre moyenneur, puis pour le filtre soustracteur :

- Créer et tracer les deux réponses impulsionnelles (on pourra se servir de l'instruction `ones(L)` qui crée un vecteur ligne de L uns.
- Tracer les réponses en fréquence de ces filtres. Vous pourrez utiliser la fonction `fft` qui calcule la TF, et tracer le module (`abs`) du résultat.
- Filtrer le signal x par ces filtres. Tracer les signaux de sortie ainsi que leurs spectres en fréquence. Conclure.

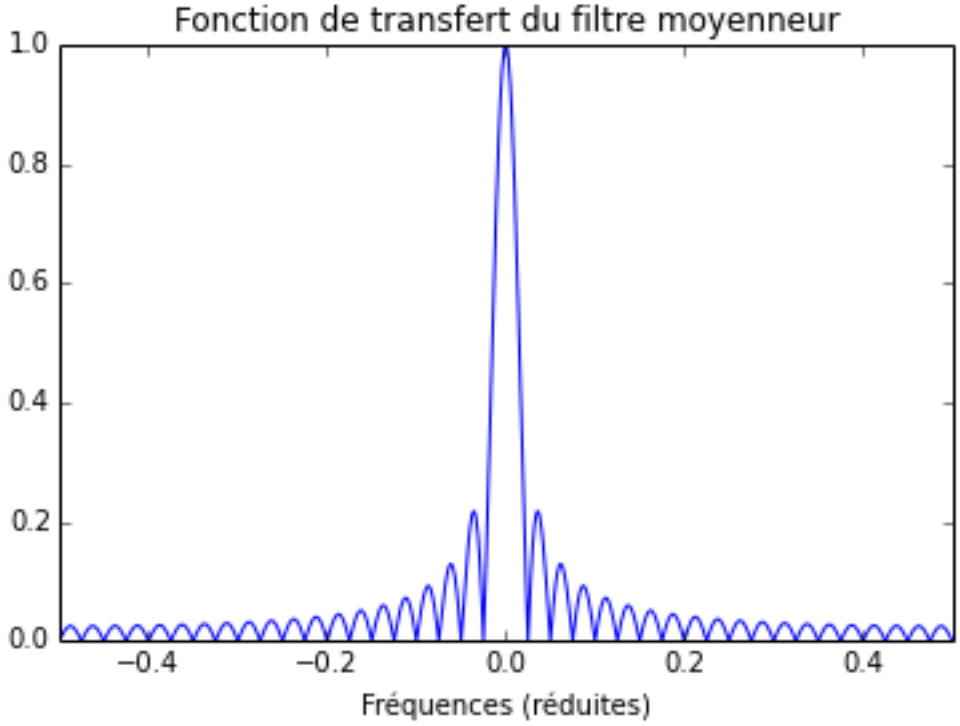
In [7]:
```
# Filtre moyenneur
#-------------------------------------------------------------
# Filtre g qui calcule la moyenne sur une période de 40 points
L=40
g=ones(L)/L
m_estimee=lfilter(g,1,x)
figure(3)
plot(t,x,t,m_estimee)
title('Signal et dérive estimée')
show()
#
# On vérifie quelle est l'allure de G(f)
```

```
G=fftshift(fft(g,1000))
figure(4)
plot(freq(1000),abs(G))
xlim([-0.5, 0.5])
xlabel('Fréquences (réduites)')
title('Fonction de transfert du filtre moyenneur')
```




```
<matplotlib.text.Text at 0x7fea6793ef50>
```

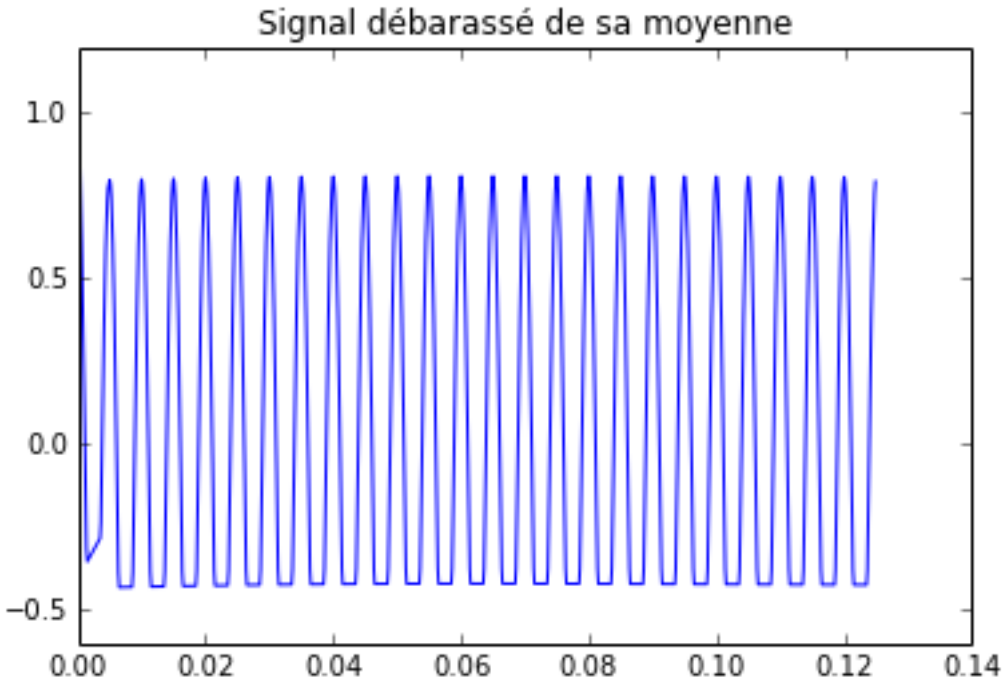
Out [7]:

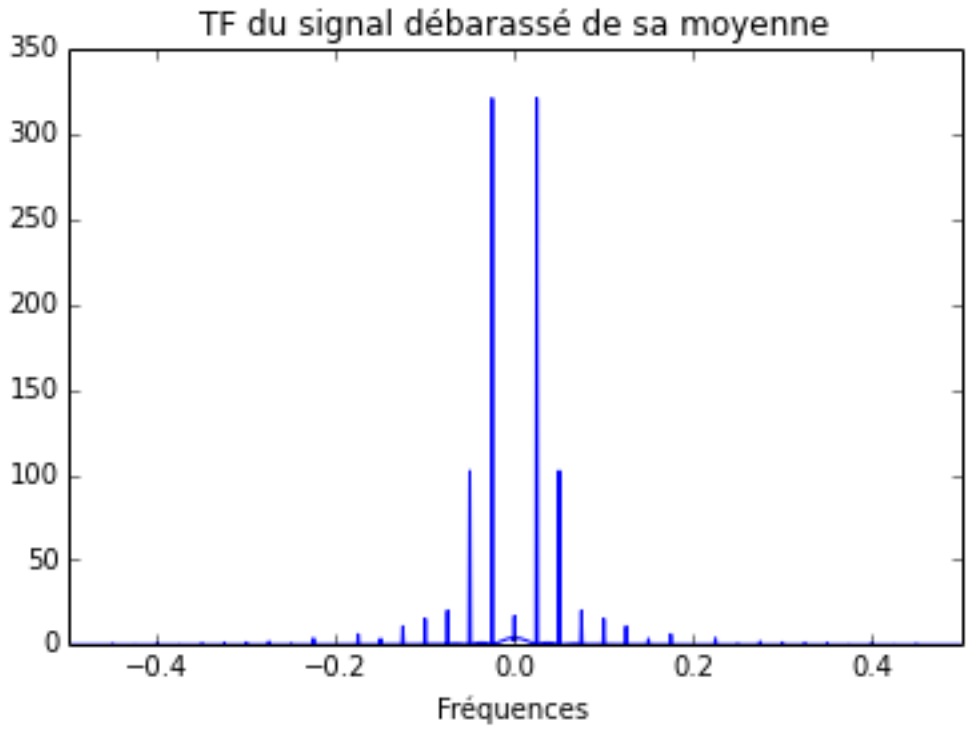


Et on notera, oh bonheur, que cette FT a le bon goût de s'annuler tous les 1/L : ainsi les raies du signal périodique de départ sont parfaitement éliminées; seuls persiste la partie fréquentielle autour de f=0

1 Calcul du filtre soustracteur

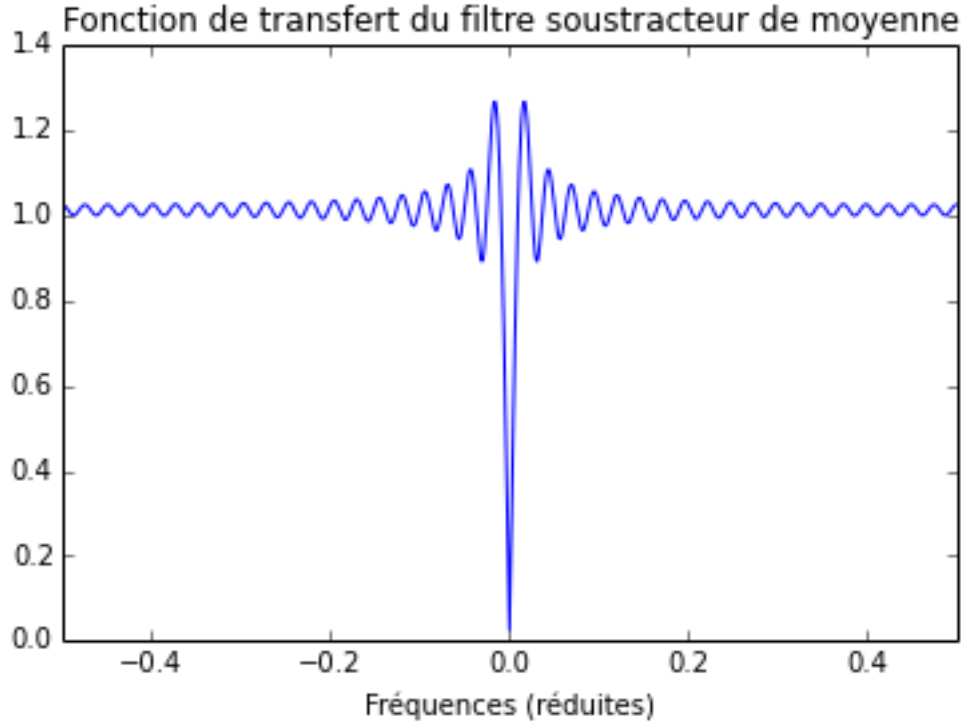
In [8]:
```
# Filtre soustracteur de moyenne
#-------------------------------------------------------------
# Filtre h qui soustrait la moyenne sur une période de 40 points
# soit on définit h comme
h=-ones(L)/L
h[0]=1
# soit on définit l'opération comme
# $xc(n)=x(n)-(g*x)(n)$
xc=lfilter(h,1,x)
figure(5)
plot(t,xc)
title('Signal débarassé de sa moyenne')
show()

figure(6)
plot(freq(len(xc)),abs(fftshift(fft(xc))))
xlabel('Fréquences')
xlim([-0.5, 0.5])
title('TF du signal débarassé de sa moyenne')
show()

# On vérifie quelle est l'allure de H(f)
H=fftshift(fft(h,1000))
figure(7)
plot(freq(1000),abs(H))
xlabel('Fréquences (réduites)')
xlim([-0.5, 0.5])
title('Fonction de transfert du filtre soustracteur de moyenne')
```






```
<matplotlib.text.Text at 0x7fea6799a190>
```

Out [8]:



2 Deuxième partie : accentuation d'une bande de fréquence

On souhaite maintenant accentuer (très nettement) la zone de fréquence autour de 1000 Hz sur le signal initial.

2.1 Partie théorique

Après un rappel éventuel de l'enseignant sur les filtres rationnels, déterminez les pôles p_1 et p_2 d'un filtre permettant de réaliser cette accentuation. Calculer la fonction de transfert, $H(z)$, correspondante ainsi que la réponse impulsionnelle $h(n)$.**Corrigé** : Calcul de la RI en plaçant deux pôles complexes conjugués pour la fréquence qui nous intéresse. La RI théorique se calcule comme suit : soit une fonction de transfert possédant deux pôles p_0 et p_1:

$H(z) = \frac{1}{(1-p_0 z^{-1})(1-p_1 z^{-1})}$

Par décomposition en éléments simples, on a

$$H(z) = \frac{A}{1-p_0 z^{-1}} + \frac{B}{1-p_1 z^{-1}}$$

avec $A = \frac{p_0}{p_0-p_1}$ et $B = -\frac{p_1}{p_0-p_1}$. On a donc

$$H(z) = \frac{1}{p_0-p_1}\left(\frac{p_0}{1-p_0 z^{-1}} - \frac{p_1}{1-p_1 z^{-1}}\right)$$

L'original, c'est-à-dire la fonction dont la transformée en z est $1/(1-az^{-1})$ est $a^n u(n)$, où $u(n)$ est l'échelon de Heaviside. Par suite, on obtient

$$h(n) = \frac{1}{p_0-p_1}\left(p_0^{n+1} - p_1^{n+1}\right)u(n).$$

Finalement, si on prend des pôles complexes conjugués de la forme $p_0 = p_1^* = \rho e^{j2\pi f_0}$, il reste :

$$h(n) = \rho^n \frac{\sin(2\pi(n+1)f_0)}{\sin(2\pi f_0)}\,u(n).$$

2.2 Partie pratique

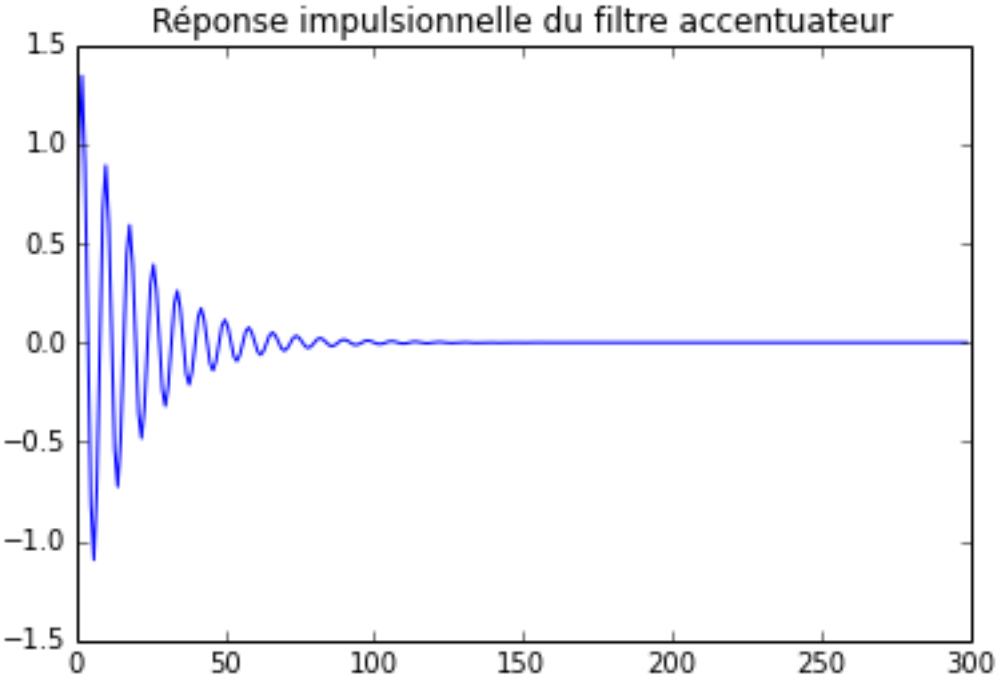
- Le vecteur de coefficients du dénominateur $A(z)$ sera calculé par : ***A=poly([p1,p2])***
  et vous vérifierez que vous obtenez les coefficients déterminés "à la main".
- Tracer la réponse fréquentielle du filtre
- Calculer sa réponse impulsionnelle par : # calcul de la RI d=zeros(300) d[1]=1 h_accentue=lfilter([1],a,d) (réponse à une impulsion de dirac calculée sur 300 points). La tracer.
- Calculer et tracer la réponse impulsionnelle obtenue avec la formule théorique. La comparer à la précédente.
- Calculer et tracer la sortie du filtre soumis à l'entrée x_c ainsi que son spectre. Conclure.

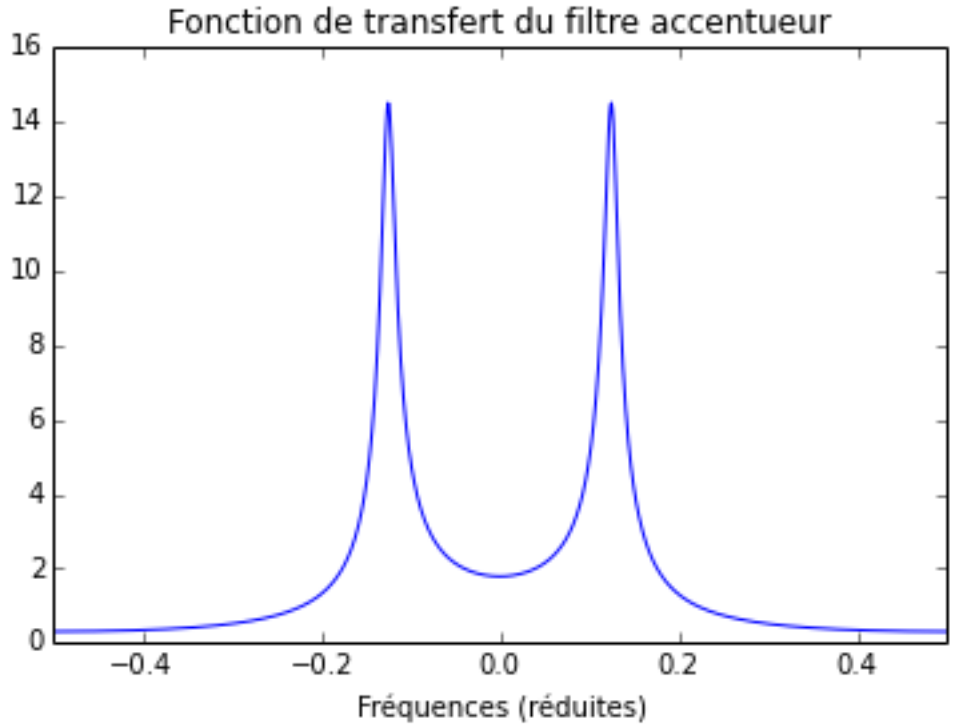
In [9]:
```
rho=0.95
fo, Fe=1000, 8000
a=poly([rho*exp(1j*2*pi*fo/Fe), rho*exp(-1.j*2*pi*fo/Fe)])


# calcul de la RI
d=zeros(300)
d[1]=1
h_accentue=lfilter([1],a,d) # calcul de la RI
figure(8)
plot(h_accentue)
title('Réponse impulsionnelle du filtre accentuateur')
```

```
# en fréquence
H_accentue=fftshift(fft(h_accentue,1000))
figure(9)
plot(freq(1000),abs(H_accentue))
xlabel('Fréquences (réduites)')
xlim([-0.5, 0.5])
title('Fonction de transfert du filtre accentueur')
```

```
<matplotlib.text.Text at 0x7fea678789d0>
```

Out [9]:

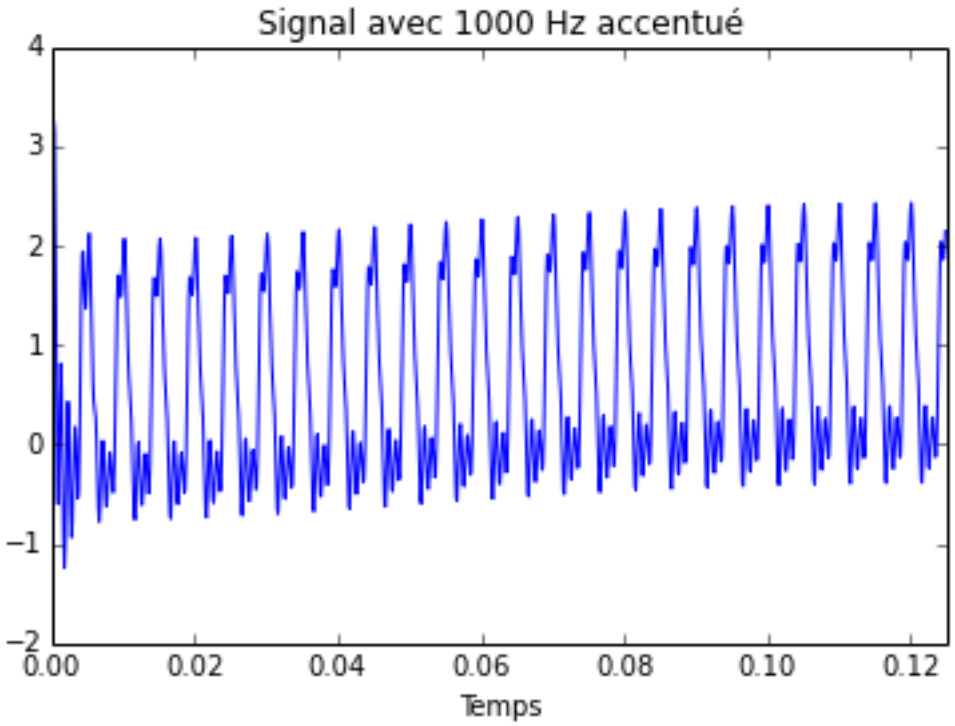


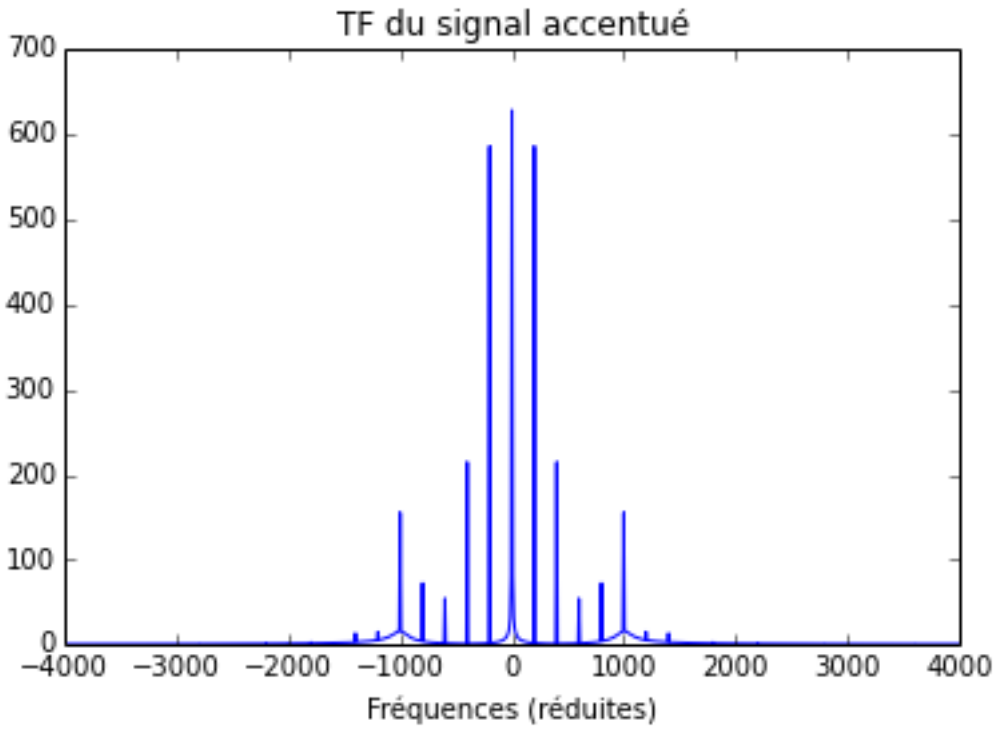


In [10]:
```
# calcul de la filtrée
sig_accentue=lfilter([1],a,x) #
figure(10)
plot(t,sig_accentue)
xlabel('Temps')
xlim([0, len(x)*Te])
title('Signal avec 1000 Hz accentué')

# en fréquence
S_accentue=fftshift(fft(sig_accentue,1000))
figure(11)
plot(freq(1000,Fe),abs(S_accentue))
xlabel('Fréquences (réduites)')
xlim([-Fe/2, Fe/2])
title('TF du signal accentué')
```

```
<matplotlib.text.Text at 0x7fea677e07d0>
```

Out [10]:

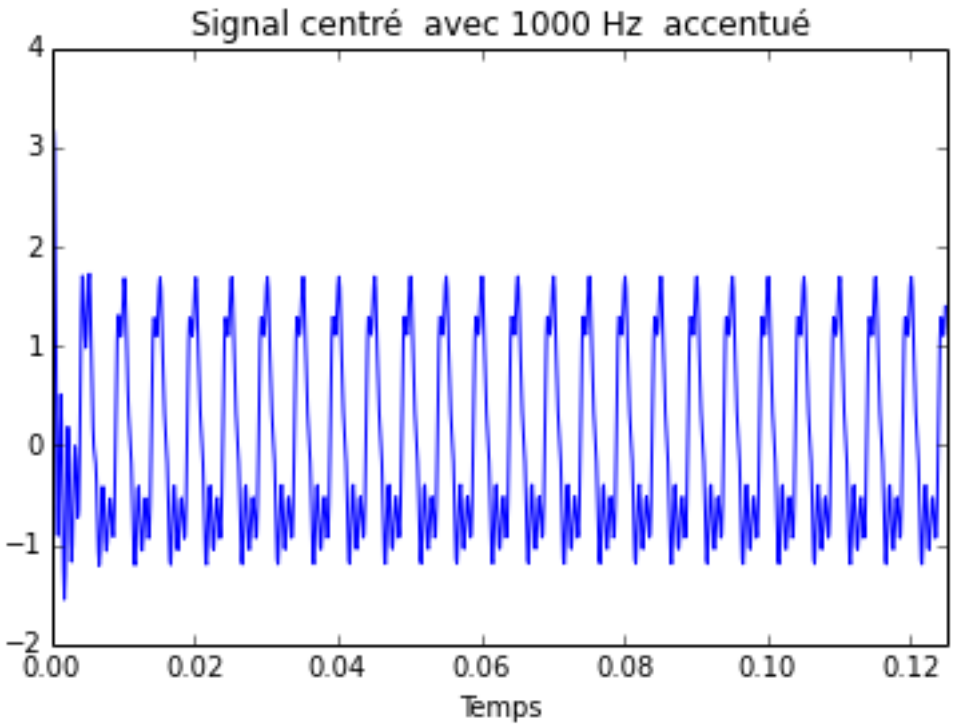




- Comment peut-on simultanément amplifier autour de 1000 Hz et supprimer la dérive ? Proposer un filtre qui réalise les deux opérations.

In [11]:
```
# Les deux à la fois :
# calcul de la filtrée
sig_accentue_centre=lfilter(h,a,x) #
figure(12)
plot(t,sig_accentue_centre)
xlabel('Temps')
xlim([0, len(x)*Te])
title('Signal centré  avec 1000 Hz  accentué')
```

Out [11]:
```
<matplotlib.text.Text at 0x7fea67911110>
```



#Filtrage passe-bas [0- 250 Hz] par la méthode de la fenêtre

On cherche maintenant à ne conserver que les basses fréquences (0 à 250 Hz) du signal x_c en filtrant celui-ci par un filtre passe-bas FIR à N=101 coefficients.

2.3 Partie théorique

On considère un filtre passe-bas idéal dont le module de la fonction de transfert, $H(f)$, est une fonction rectangulaire. Calculer la réponse impulsionnelle (infinie) du filtre numérique qui réaliserait ce passe-bas de façon idéale.

2.4 Partie pratique

a. On veut limiter le nombre de coefficients à L (RIF). Calculer le vecteur h à L coefficients représentant cette réponse pondérée (tronquée) par une fenêtre rectangulaire rect$_T$(t) où $T = L * Te$.

b. Tracer la réponse fréquentielle de ce filtre.

c. Calculer et tracer la sortie de ce filtre soumis à l'entrée x_c ainsi que son spectre.

d. Observer le temps de propagation de groupe de la réponse fréquentielle~: `plot(f,grpdelay(B,A,N))`. Commenter.

Corrigé : Pour une impulsion rectangulaire centrée, de largeur $2B$, la transformée de Fourier inverse vaut

$$\begin{aligned} h(n) &= \int_{[1]} \text{rect}_{2B}(f) e^{j2\pi fn} \mathrm{d}f \\ &= \int_{-B}^{B} e^{j2\pi fn} \mathrm{d}f \\ &= B \frac{\sin(\pi Bn)}{\pi Bn} \end{aligned}$$

Le problème qui se pose alors est que :

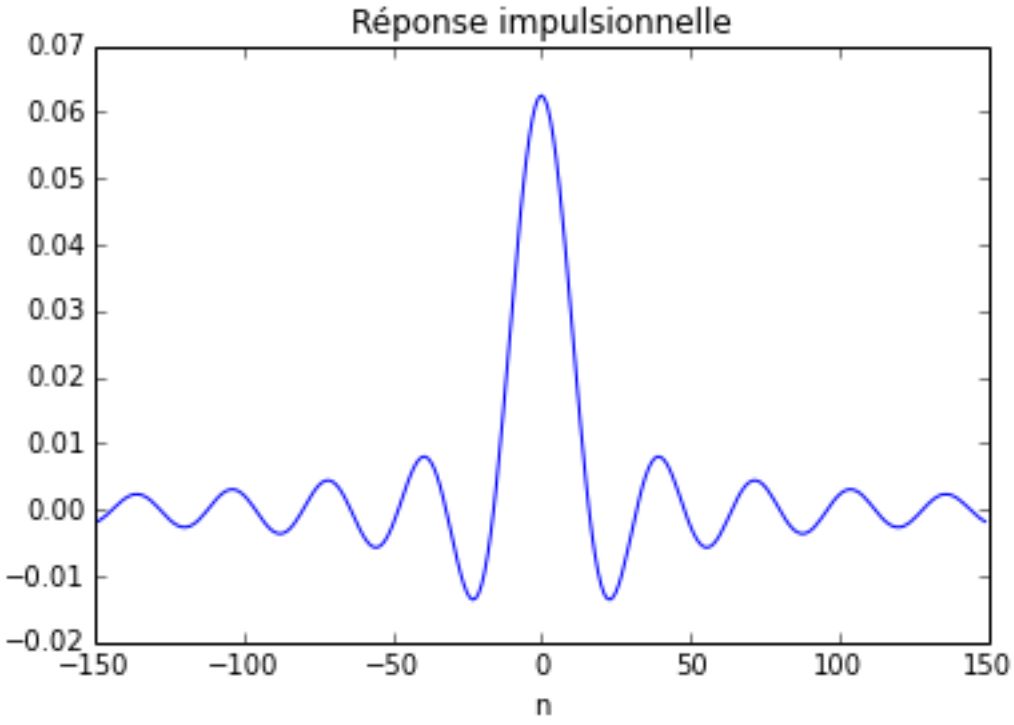
- la réponse impulsionnelle $h(n)$ est à support non compact,
- et non causale (définie pour les temps négatifs)

Pour obtenir une RI sur L points, il faut donc

- tronquer la RI, puis
- décaler celle-ci de manière à rendre la RI causale.

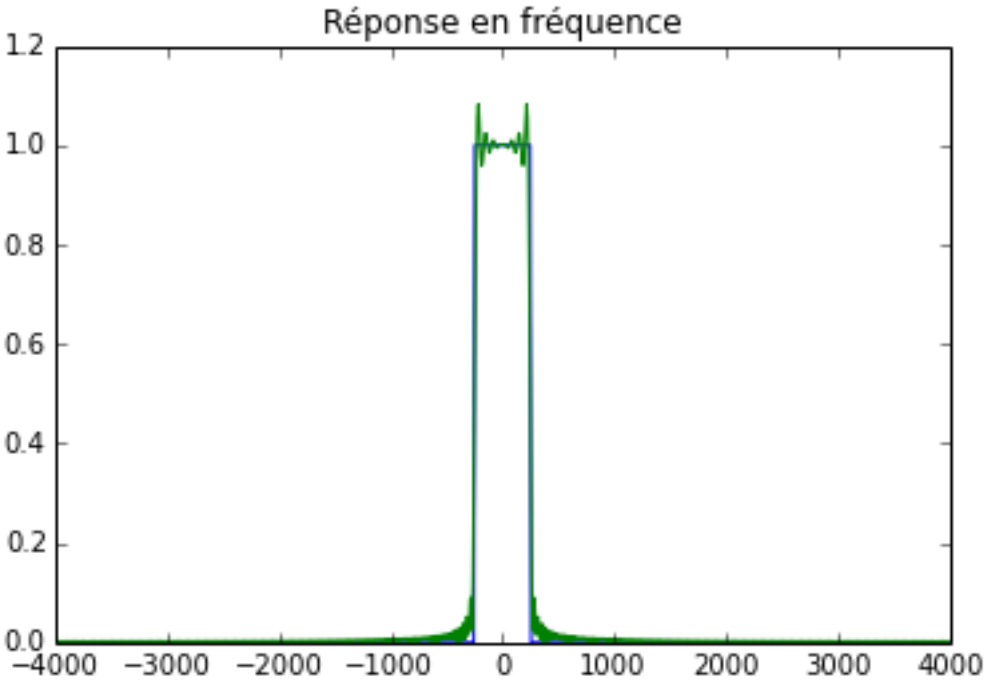
In [12]:
```
from numpy import *
from pylab import *
B=250
Fe=8000
B=B/Fe # Bande en fréquences réduites
n=arange(-150,150)

def sinc(x):
    x=array(x)
    z=[sin(n)/n if n!=0 else 1 for n in x]
    return array(z)

#h=B*sin(pi*B*n)/(pi*B*n) # Mais ici il y a une division par zéro
#h[0]=B
h=2*B*sinc(pi*2*B*n)
plot(n,h)
xlabel('n')
title('Réponse impulsionnelle')
```

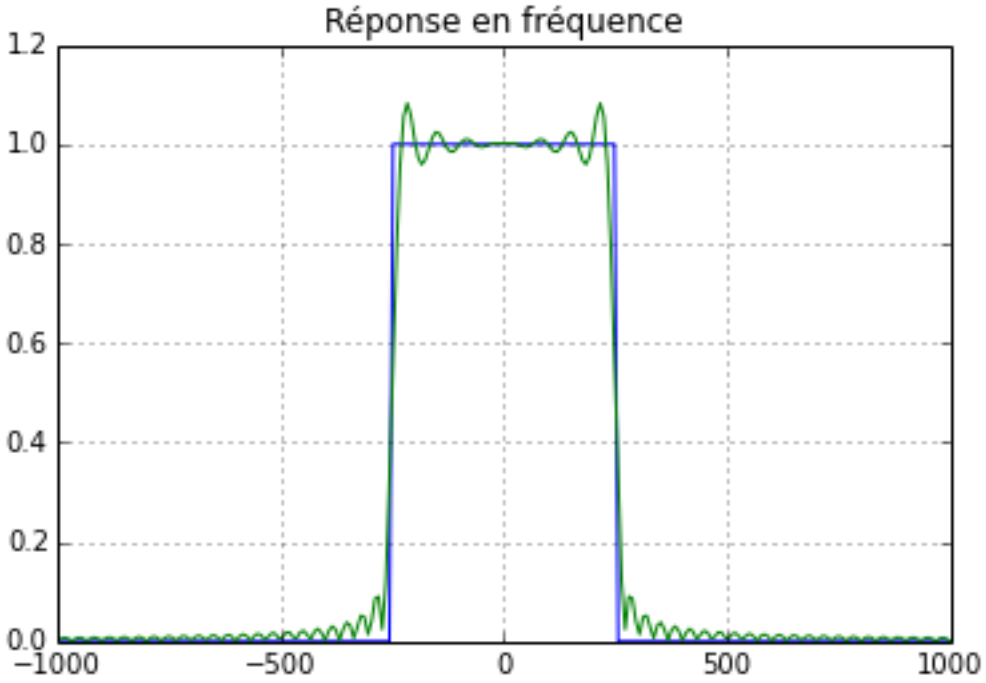
Out [12]:
```
<matplotlib.text.Text at 0x7fea677da750>
```



Pour tronquer, il suffit de ne garder que les points intéressants ; ici les points entre -50 et 50 (pour un total de 101 points)

In [42]:
```
L=50
h_tronq=h=2*B*sinc(pi*2*B*arange(-L,L))
H_tronq=fft(h_tronq,1000)
f=(arange(1000)/1000-1/2)*8000
R=[1 if abs(ff)<250 else 0 for ff in f]
#
figure()
plot(f,R, f,abs(fftshift(H_tronq)))
title("Réponse en fréquence")
figure()
plot(f,R,f,abs(fftshift(H_tronq)))
title("Réponse en fréquence")
xlim([-1000, 1000])
grid('on')
```



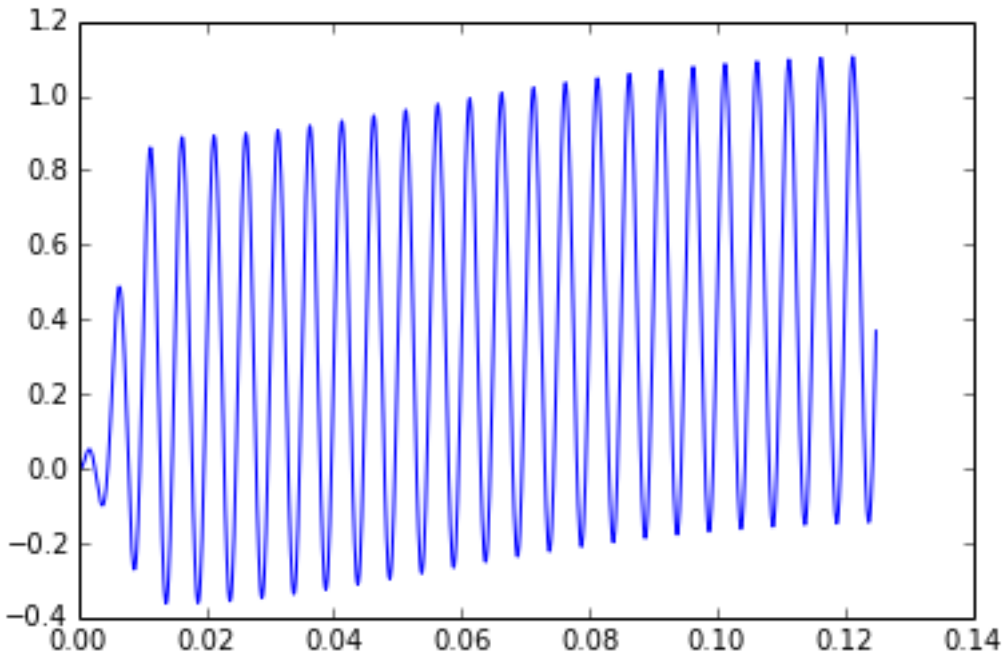


Calcul de la sortie de ce filtre passe-bas :

In [16]:
```
sig_passe_bas=lfilter(h,[1],x) #
plot(t,sig_passe_bas)
```

```
[<matplotlib.lines.Line2D at 0x7fea67599190>]
```

Out [16]:



Temps de propagation de groupe

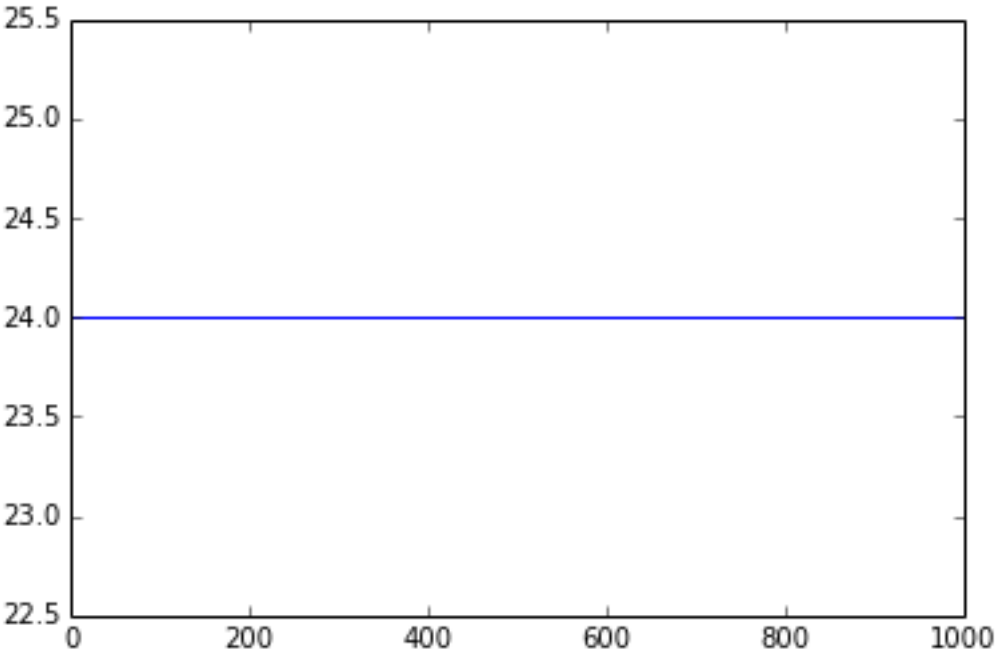
Le temps de propagation de groupe est calculé comme indiqué ici, cf https://ccrma.stanford.edu/~jos/fp/Numerical_Computation_Group_Delay.html

In [58]:

```
def grpdelay(h):
    N=len(h)
    hn=h*arange(N)
    Td=real(fft(hn.flatten(),1000)/(fft(h.flatten(),1000)))
    return Td
hh=zeros(200)
#hh[20:25]=array([1, -2, 70, -2, 1])
hh[24]=1
plot(grpdelay(hh))
```

```
[<matplotlib.lines.Line2D at 0x7fea6694dc50>]
```

Out [58]:



3 Echantillonnage, périodisation, repliement

Vous disposez d'un signal $x(n)$, échantillonné à $Fe = 32$.

1. Chargez ce signal par `load Signal`. Le signal est chargé dans l'environnement sous le nom x. Visualisez x dans le domaine temporel et fréquentiel. Quelle est sa durée temporelle ? Quelle est approximativement la bande occupée ?
2. On étudie d'abord l'effet d'une répétition du signal. Créez un nouveau signal, $x_r(n)$ et répétant 8 fois le motif $x(n)$ (fonction `repeat`). Visualisez le signal temporel, puis comparez les réponses en fréquence de $x(n)$ et $x_r(n)$. Conclusions.
3. On s'intéressera ensuite aux effets de l'échantillonnage : rééchantillonnez le signal aux fréquences $Fse = 16$, $Fse = 8$, $Fse = 4$ (créez les signaux $x_{e1}(n), x_{e2}(n)$ et $x_{e3}(n)$), en utilisant la fonction `echant`. Visualisez les signaux temporels, et comparez les réponses fréquentielles (toujours sur $[-Fe/2, Fe/2]$, avec $Fe = 32$, la fréquence d'échantillonnage initiale).
4. Créez enfin un signal périodique échantillonné $x_{re}(n)$, en périodisant le signal initial (en créant par exemple 8 périodes) puis en échantillonnant le signal résultant. Analysez le signal obtenu en temps et en fréquence. Conclusions.

In [14]: