SAR Traitements audio : partie signal

Lianne Soo, Tinhinen Meniche

Introduction

La partie signal de la SAR traitements audio nous permet de mettre en oeuvre et d'analyser différents algorithmes pour comprendre leur impact sur les signaux audio. En manipulant des signaux réels et synthétiques, nous étudierons les caractéristiques des signaux sonores et les transformations qu'ils subissent lors du traitement numérique.

Synthèse additive

Analyse d'un son harmonique

Question 1.1

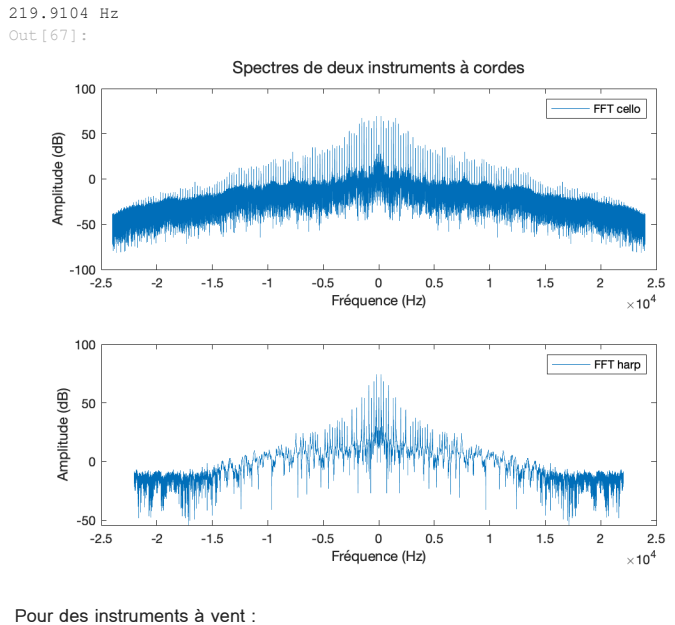
Visualisons le spectre d'amplitude obtenu avec une transformée de Fourier discrète de quelques instruments à cordes :

In [67]:

```
[f_cello,mod_cello]=module_fft("wav/single_tone_cello-a3.wav");
[f_harp,mod_harp]=module_fft("wav/single_tone_celtic-harp-a3.wav");

mod_cello_db = 20 * log10(mod_cello);
mod_harp_db = 20 * log10(mod_harp);

[f_cello_max,idx1_cello]=max(mod_cello);
f1_cello=f_cello(idx1_cello);
disp('fréquence fondamentale du violoncelle :');
disp([num2str(abs(f1_cello)), ' Hz']);


[f_harp_max,idx_harp]=max(mod_harp);
f1_harp=f_harp(idx_harp);
disp('fréquence fondamentale de lharpe :');
disp([num2str(abs(f1_harp)), ' Hz']);

figure;
t=tiledlayout(2,1);
title(t, 'Spectres de deux instruments à cordes');
nexttile
plot(f_cello,mod_cello_db);
legend('FFT cello');
xlabel('Fréquence (Hz)');
ylabel('Amplitude (dB)');
nexttile
plot(f_harp,mod_harp_db);
legend('FFT harp');
xlabel('Fréquence (Hz)');
ylabel('Amplitude (dB)');
```

```
fréquence fondamentale du violoncelle :
220.8047 Hz
fréquence fondamentale de lharpe :
219.9104 Hz
```

Out[67]:



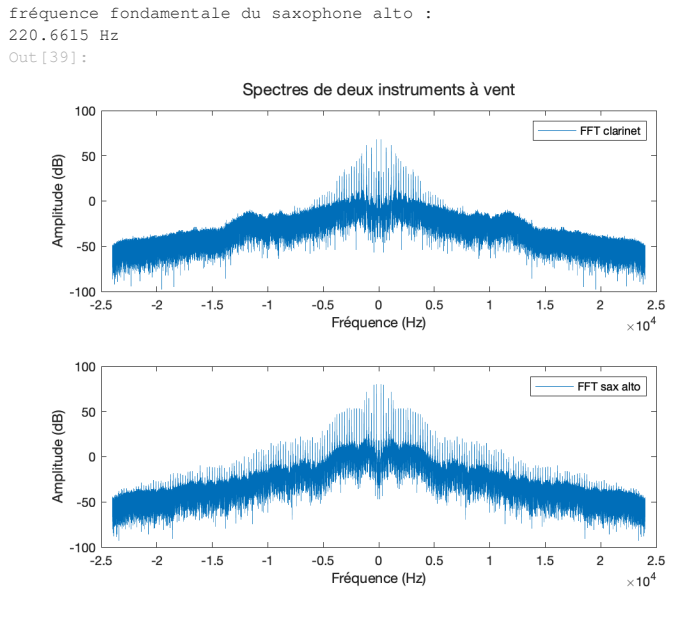
Pour des instruments à vent :

In [39]:

```
[f_clari,mod_clari]=module_fft("wav/single_tone_clarinet-a3.wav");
[f_sax_a,mod_sax_a]=module_fft("wav/single_tone_sax-alto-a3.wav");

mod_clari_db = 20 * log10(mod_clari);
mod_sax_a_db = 20 * log10(mod_sax_a);

[f_clari_max,idx1_clari]=max(mod_clari);
f1_clari=f_clari(idx1_clari);
disp('fréquence fondamentale de la clarinette :');
disp([num2str(abs(f1_clari)), ' Hz']);


[f_sax_a_max,idx_sax_a]=max(mod_sax_a);
f1_sax_a=f_sax_a(idx_sax_a);
disp('fréquence fondamentale du saxophone alto :');
disp([num2str(abs(f1_sax_a)), ' Hz']);

figure;
t=tiledlayout(2,1);
title(t, 'Spectres de deux instruments à vent');
nexttile
plot(f_clari,mod_clari_db);
legend('FFT clarinet');
xlabel('Fréquence (Hz)');
ylabel('Amplitude (dB)');
nexttile
plot(f_sax_a,mod_sax_a_db);
legend('FFT sax alto');
xlabel('Fréquence (Hz)');
ylabel('Amplitude (dB)');
```

```
fréquence fondamentale de la clarinette :
222.0918 Hz
fréquence fondamentale du saxophone alto :
220.6615 Hz
```

Out[39]:



En comparant les instruments de la même famille, nous remarquons que les instruments ont une fréquence fondamentale presque égale mais qu'ils ont un spectre d'amplitude différent. En effet, ils se différencient par leur répartition harmonique, ce qui explique pourquoi nous entendons des sons différents même lorsque 2 instruments de la même famille jouent la même note.

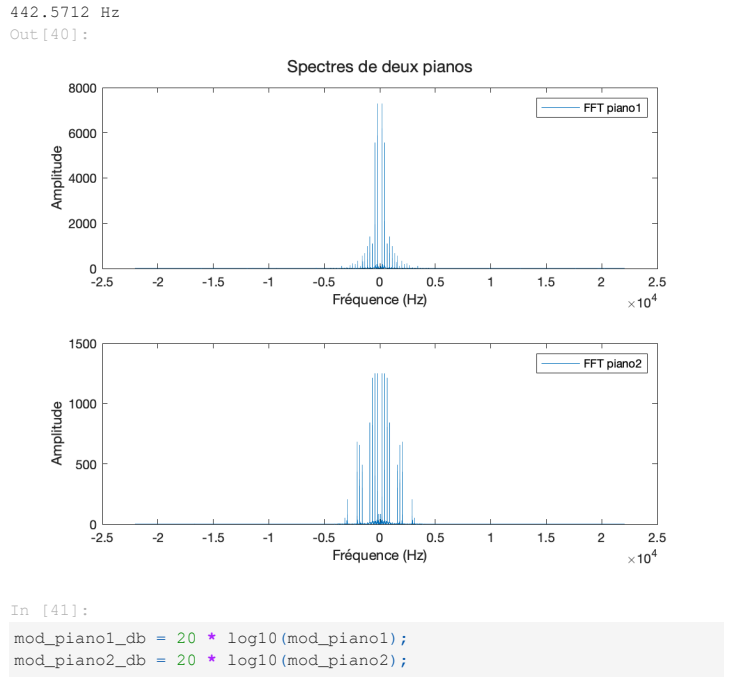
Question 1.2

In [40]:

```
[x,fe]=audioread("wav/single_tone_piano1.wav");
[f_piano1,mod_piano1]=module_fft("wav/single_tone_piano1.wav");
[f_piano2,mod_piano2]=module_fft("wav/single_tone_piano2.wav");

mod_piano1_db = 20 * log10(mod_piano1);
mod_piano2_db = 20 * log10(mod_piano2);

[f_piano1_max,idx_piano1]=max(mod_piano1);
f1_piano1=f_piano1(idx_piano1);
disp('fréquence fondamentale du piano1 :');
disp([num2str(abs(f1_piano1)), ' Hz']);

[f_piano2_max,idx_piano2]=max(mod_piano2);
f1_piano2=f_piano2(idx_piano2);
disp('fréquence fondamentale du piano2 :');
disp([num2str(abs(f1_piano2)), ' Hz']);

t = tiledlayout(2,1);
title(t, 'Spectres de deux pianos');
nexttile
plot(f_piano1,mod_piano1);
legend('FFT piano1');
xlabel('Fréquence (Hz)');
ylabel('Amplitude');

nexttile
plot(f_piano2,mod_piano2);
legend('FFT piano2');
xlabel('Fréquence (Hz)');
ylabel('Amplitude');
```

```
fréquence fondamentale du piano1 :
220.2939 Hz
fréquence fondamentale du piano2 :
442.5712 Hz
```

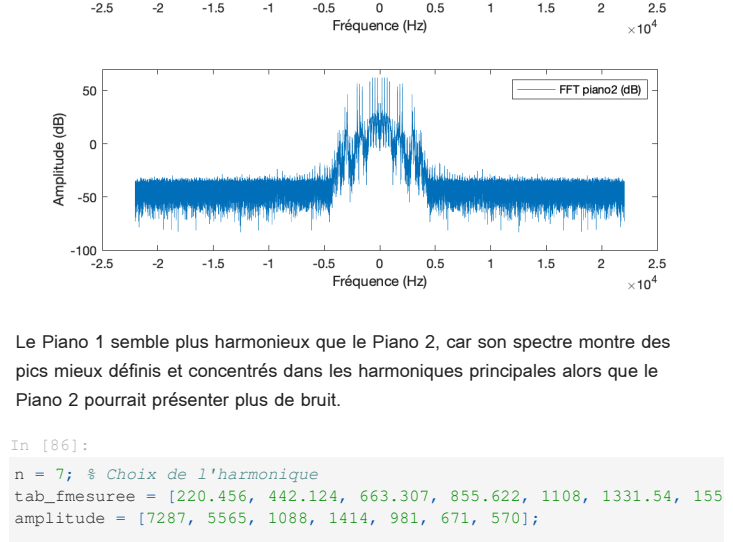
Out[40]:



In [41]:

```
mod_piano1_db = 20 * log10(mod_piano1);
mod_piano2_db = 20 * log10(mod_piano2);

t = tiledlayout(2,1);
title(t, 'Spectres de deux pianos (dB)');
nexttile
plot(f_piano1,mod_piano1_db);
legend('FFT piano1 (dB)');
xlabel('Fréquence (Hz)');
ylabel('Amplitude (dB)');
nexttile
plot(f_piano2,mod_piano2_db);
legend('FFT piano2 (dB)');
xlabel('Fréquence (Hz)');
ylabel('Amplitude (dB)');
ylim([-100 70]);
```

Out[41]:



Le Piano 1 semble plus harmonieux que le Piano 2, car son spectre montre des pics mieux définis et concentrés dans les harmoniques principales alors que le Piano 2 pourrait présenter plus de bruit.

In [86]:

```
n = 7; % Choix de l'harmonique
tab_fmesuree = [220.456, 442.124, 663.307, 855.622, 1108, 1331.54, 155
amplitude = [7287, 5565, 1088, 1414, 981, 671, 570];

% Calcul des fréquences théoriques
ftheo_piano1 = n * abs(tab_fmesuree(1)); % Fréquence théorique (exempl
disp('ftheo_piano1 :');
disp(ftheo_piano1);
disp('fmes_piano1 :');
disp(tab_fmesuree(n));
epsfh_piano1 = 1200 * (log2(tab_fmesuree(n)) - log2(ftheo_piano1));
disp('inharmonicité_piano1 :');
disp(abs(epsfh_piano1));

% Création du spectre (positif et négatif)
freqs_positive = tab_fmesuree;
freqs_negative = -tab_fmesuree;
amplitudes = amplitude;

% Combinaison des fréquences et amplitudes
freqs = [freqs_negative, freqs_positive];
amps = [amplitudes, amplitudes];
```

```
ftheo_piano1 :
   1.5432e+03

fmes_piano1 :
   1.5563e+03

inharmonicité_piano1 :
   14.6320

   1.5432e+03

fmes_piano1 :
   1.5563e+03

inharmonicité_piano1 :
   14.6320
```

Fréquence (Hz)	f2 (Hz)	f3 (Hz)	f4 (Hz)	f5 (Hz)	f6 (Hz)	f7 (Hz)
Théorique	440.6	660.9	881.2	1101.5	1321.8	1542.1
Mesurées	442.1	663.3	855.6	1107.94	1331.5	1556.3
Inharmonicité	6.7524	5.0682	52.22	8.9606	11.484	14.632

Question 1.3

In [43]:

```
size=length(tab_fmesuree);
duree =2; %length(f_piano1)/fe;
tmp = 0:1/fe:duree-1/fe;
signal=0;
for k = 1:size
    signal=signal + amplitude(k)*sin(2*pi*tab_fmesuree(k)*tmp);
end
signal_norma = signal / max(abs(signal)); %signal normalisé
sound(signal_norma, fe);
```

En comparant le son synthétisé au son original, nous entendons la même note mais le son synthétisé sonne beaucoup plus robotique et monotone car il n'a pas de variation de volume (sans l'enveloppe), ni de bruit qui crée toute la richesse du son du piano.

Question 1.4

In [68]:

```
n_sig = length(signal_norma);
len_attack = round(n_sig * 0.001);
len_decay = round(n_sig * 0.008);
len_sustain = round(n_sig * 0.80);
len_release = n_sig - (len_attack + len_decay + len_sustain);

A = linspace(0, 1, len_attack);  % Montée rapide
D = linspace(1, 0.8, len_decay); % Déclin rapide
S = linspace(0.8, 0.8, len_sustain); % Faible sustain
R = linspace(0.8, 0, len_release);  % Relâchement progressif

envelope = [A, D, S, R];

if length(envelope) > n_sig
    envelope = envelope(1:n_sig);
elseif length(envelope) < n_sig
    envelope = [envelope, zeros(1, n_sig - length(envelope))];
end

tone = signal_norma .* envelope;

sound(tone, fe);
```

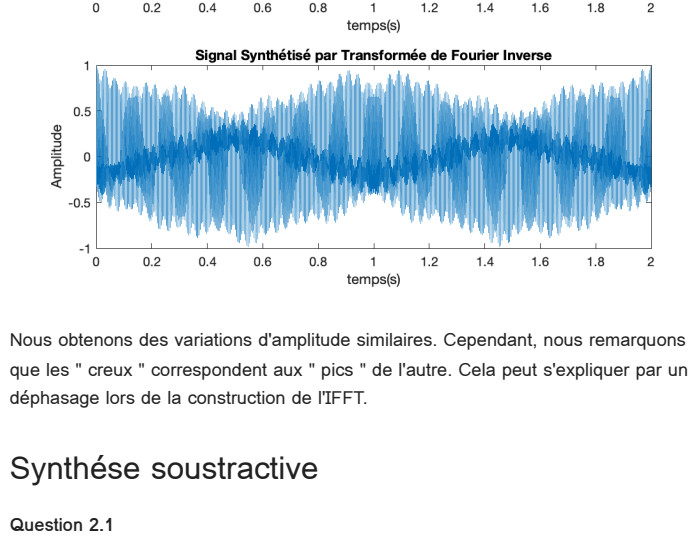
Comme dit précédemment, le son synthétisé sonne toujours robotique sans les bruits mais l'enveloppe permet d'approcher ce son au son original du piano en créant des variations de volume.

Question 1.5

In [45]:

```
N = length(tmp);
spectrum = zeros(1, N);

% Index correspondant aux fréquences mesurées
freq_indices = round(tab_fmesuree / fe * N);

% Construction du spectre
for k = 1:length(tab_fmesuree)
    spectrum(freq_indices(k) + 1) = amplitude(k);
    spectrum(N - freq_indices(k) + 1) = amplitude(k);
end

% Transformée de Fourier inverse
tf_inverse_modpiano1 = ifft(spectrum, 'symmetric');

tf_inverse_modpiano1 = tf_inverse_modpiano1 / max(abs(tf_inverse_modpi

subplot(2,1,1);
plot(tmp, signal_norma);
xlabel('temps(s)');
ylabel('Amplitude');
title('Signal somme sinusoïdale');

subplot(2,1,2);
plot(tmp, tf_inverse_modpiano1);
xlabel('temps(s)');
ylabel('Amplitude');
title('Signal Synthétisé par Transformée de Fourier Inverse');

%sound(tf_inverse_modpiano1, fe); % Pour écouter le son synthétisé
```

Out[45]:



Nous obtenons des variations d'amplitude similaires. Cependant, nous remarquons que les " creux " correspondent aux " pics " de l'autre. Cela peut s'expliquer par un déphasage lors de la construction de l'IFFT.

Synthése soustractive

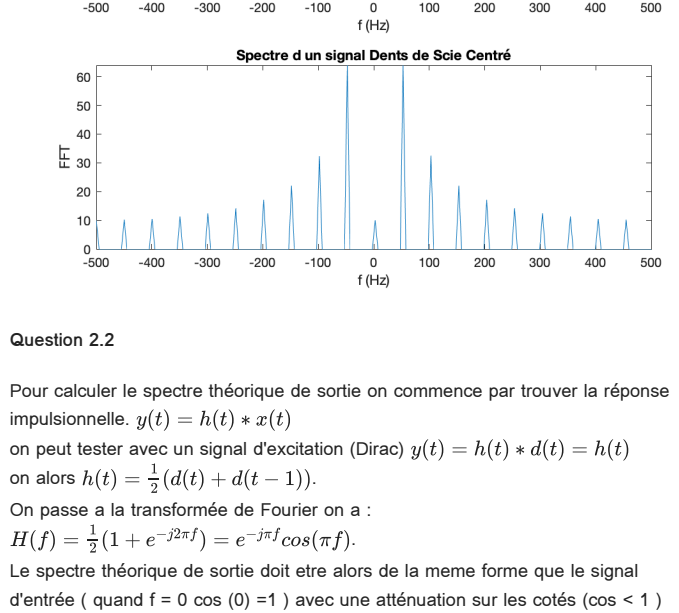
Question 2.1

Pour vérifier numériquement les calculs, on peut utiliser la décomposition de Fourier. En effet, un signal temporel périodique y(t) de fréquence f se décompose en une somme infinie de sinus et cosinus de fréquence f , 2f , 3f, etc.
La transformée de Fourier est plus simple à calculer (somme)
Cette méthode est assez complexe, surtout pour des signaux complexes. Les calculs deviennent assez difficiles et l'interprétation des résultats encore plus

In [46]:

```
%signal carre
Tc = 1; %période
fca=1000;
tc = linspace(-2,2,1000);
xc = square(2*pi*tc/Tc);
yc=fftshift(fft(xc));
modulec=abs(yc);
fc=linspace(-fca/2,fca/2,length(yc));
figure ;
subplot(2,1,1);
plot(fc, modulec);
xlabel('f (Hz)');
ylabel('FFT');
title('Spectre d un signal Carré Centré');

%signal dent de scie
Td = 10*(1/50);
fs = 1000;
td = 0:1/fs:Td-1/fs;
xd = sawtooth(2*pi*50*td);
yd=fftshift(fft(xd));
moduled=abs(yd);
fd=linspace(-fs/2,fs/2,length(yd));


subplot(2,1,2);
plot(fd, moduled);
xlabel('f (Hz)');
ylabel('FFT');
title('Spectre d un signal Dents de Scie Centré');
```

Out[46]:



Question 2.2

Pour calculer le spectre théorique de sortie on commence par trouver la réponse impulsionnelle. $y(t) = h(t) * x(t)$
on peut tester avec un signal d'excitation (Dirac) $y(t) = h(t) * d(t) = h(t)$
on alors $h(t) = \frac{1}{2}(d(t) + d(t-1))$.
On passe a la transformée de Fourier on a :
$H(f) = \frac{1}{2}(1 + e^{-j2\pi f}) = e^{-j\pi f}cos(\pi f)$.
Le spectre théorique de sortie doit etre alors de la meme forme que le signal d'entrée (quand f = 0 cos (0) =1) avec une atténuation sur les cotés (cos < 1)

In [47]:

```
%signal dent de scie
xd_filtered=filter([1/2],[1/2],xd);
yd_filtered = fftshift(fft(xd_filtered));
moduled_filtered=abs(yd_filtered);
sound(xd_filtered,fs);
```

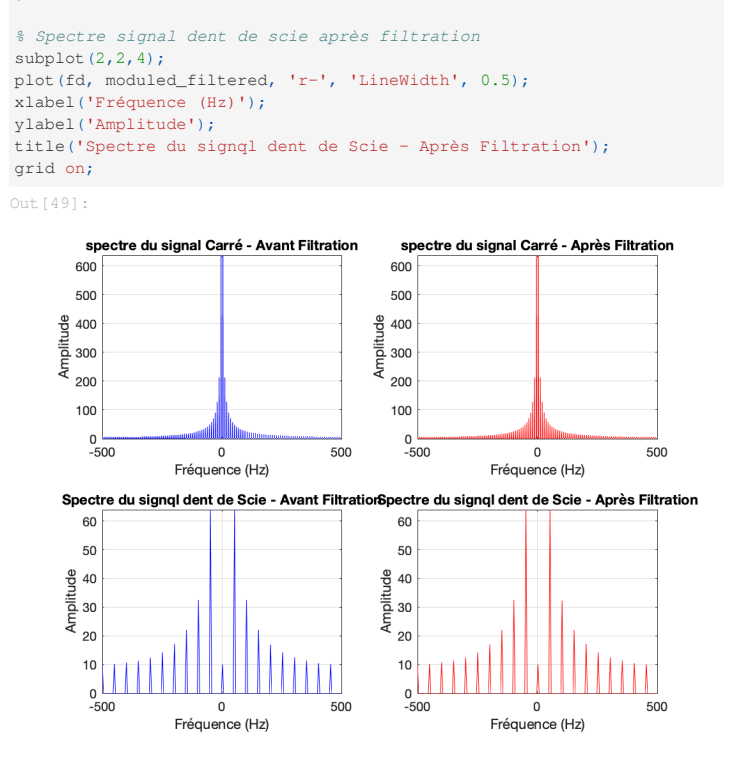
In [48]:

```
%signal carre
xc_filtered=filter([1/2],[1/2],xc);
yc_filtered = fftshift(fft(xc_filtered));
modulec_filtered=abs(yc_filtered);
sound(xc_filtered,fca);
```

In [49]:

```
figure;
% Spectre signal carré avant filtration
subplot(2,2,1);
plot(fc, modulec, 'b-', 'LineWidth', 0.5);
xlabel('Fréquence (Hz)');
ylabel('Amplitude');
title('spectre du signal Carré - Avant Filtration');
grid on;

% Spectre signal carré après filtration
subplot(2,2,2);
plot(fc, modulec_filtered, 'r-', 'LineWidth', 0.5);
xlabel('Fréquence (Hz)');
ylabel('Amplitude');
title('spectre du signal Carré - Après Filtration');
grid on;

% Spectre signal dent de scie avant filtration
subplot(2,2,3);
plot(fd, moduled, 'b-', 'LineWidth', 0.5);
xlabel('Fréquence (Hz)');
ylabel('Amplitude');
title('Spectre du signql dent de Scie - Avant Filtration');
grid on;

% Spectre signal dent de scie après filtration
subplot(2,2,4);
plot(fd, moduled_filtered, 'r-', 'LineWidth', 0.5);
xlabel('Fréquence (Hz)');
ylabel('Amplitude');
title('Spectre du signql dent de Scie - Après Filtration');
grid on;
```

Out[49]:



En comparant les figures du spectre du signal carré et dent de scie avant et après filtration, nous confirmons bien la théorie qui dit que le spectre théorique de sortie doit etre de la meme forme que le signal d'entrée (quand f = 0 cos (0) =1). Cependant, nous n'observons pas d'atténuation car les signaux ne s'étendent pas sur une plage de fréquence assez élevée.

Question 2.3

In [50]:

```
%signal dent de scie
n_sig = length(yd);
len_attack = round(n_sig * 0.001);
len_decay = round(n_sig * 0.008);
len_sustain = round(n_sig * 0.90);
len_release = n_sig - (len_attack + len_decay + len_sustain);

A = linspace(0, 1, len_attack);
D = linspace(1, 0.8, len_decay);
S = linspace(0.8, 0.8, len_sustain);
R = linspace(0.8, 0, len_release);

envelope = [A, D, S, R];

if length(envelope) > n_sig
    envelope = envelope(1:n_sig);
elseif length(envelope) < n_sig
    envelope = [envelope, zeros(1, n_sig - length(envelope))];
end

toned = xd .* envelope;

% Jouer le son avec l'enveloppe
sound(toned, fs);
```

In [51]:

```
%signal carre
n_sig = length(yc);
len_attack = round(n_sig * 0.001);
len_decay = round(n_sig * 0.008);
len_sustain = round(n_sig * 0.90);
len_release = n_sig - (len_attack + len_decay + len_sustain);

A = linspace(0, 1, len_attack);
D = linspace(1, 0.8, len_decay);
S = linspace(0.8, 0.8, len_sustain);
R = linspace(0.8, 0, len_release);

envelope = [A, D, S, R];

if length(envelope) > n_sig
    envelope = envelope(1:n_sig);
elseif length(envelope) < n_sig
    envelope = [envelope, zeros(1, n_sig - length(envelope))];
end

tonec = xc .* envelope;

sound(tonec, fca);
```

En écoutant les sons avec la même enveloppe, nous constatons que les sons ne sont pas du tout similaires par rapport à la synthèse additive,

Question 2.4

In [52]:

```
d = designfilt('lowpassfir','FilterOrder',2,'HalfPowerFrequency',0.25)
```

In [53]:

```
%signal carre
xc_filtered_2 = filter(d, xc);
tonec_2 = xc_filtered_2 .* envelope;
sound(tonec_2, fca)
```

In [54]:

```
%signal dent de scie
xd_filtered_2 = filter(d, xd);
n=min(length(xd_filtered_2),length(envelope))
toned_2 = xd_filtered_2(1:n) .* envelope(1,n);
sound(toned_2, fs)
```

Out[54]:

```
n = 200
```

Avec d'autres paramètres :

In [55]:

```
d2 = designfilt("bandpassfir", ...
    FilterOrder=20,CutoffFrequency1=400, ...
    CutoffFrequency2=450,SampleRate=1500);
```

In [56]:

```
 %signal carre
xc_filtered_2 = filter(d2, xc);
tonec_2 = xc_filtered_2 .* envelope;
sound(tonec_2, fca)
```

In [57]:

```
%signal dent de scie
xd_filtered_2 = filter(d2, xd);
n=min(length(xd_filtered_2),length(envelope))
toned_2 = xd_filtered_2(1:n) .* envelope(1,n);
sound(toned_2, fs)
```

Out[57]:

```
n = 200
```

Nous avons testé différents paramètres mais nous sommes toujours très loin du son d'un piano.

Effets audio-numériques

Mesure de réponse impulsionnelle

Question 3.1

On veut exprimer R_{yx} en fonction de R_{xx} et de h
On a $R_{yx}(u) = y(u) * x^*(-u)$
, $R_{xx}(u) = x(u) * x^*(-u)$
et $y(u) = x(u) * h(u)$.
On a alors $R_{yx} = x(u) * h(u) * x^*(-u) = x(u) * x^*(-u) * h(u) = R_{xx} * h(u)$
d'ou $R_{yx} = R_{xx} * h(u)$

Question 3.2

D'après la question 3.1 on a $R_{yx} = R_{xx} * h(u)$ en supposant que
$R_{xx}(u) \approx d(u)$
$R_{yx} = d(u) * h(u)$
d'où $y(u) * x^*(-u) = d(u) * h(u)$
on peut passer à la transformer de fourier et on obtient
$Y(v). X^*(-v) = D(v). H(v)$
Sachant que $D(v) = 1$ (transformé de Fourié d'un Dirac)
on obtient alors $H(v) = Y(v). X(-v)^*$
$h(u) = IFFT(H)$

Ainsi, en repassant dans le domaine temporel, $h(u)$ est également l'intercorrélation entre $y(u)$ et $x(u)$.

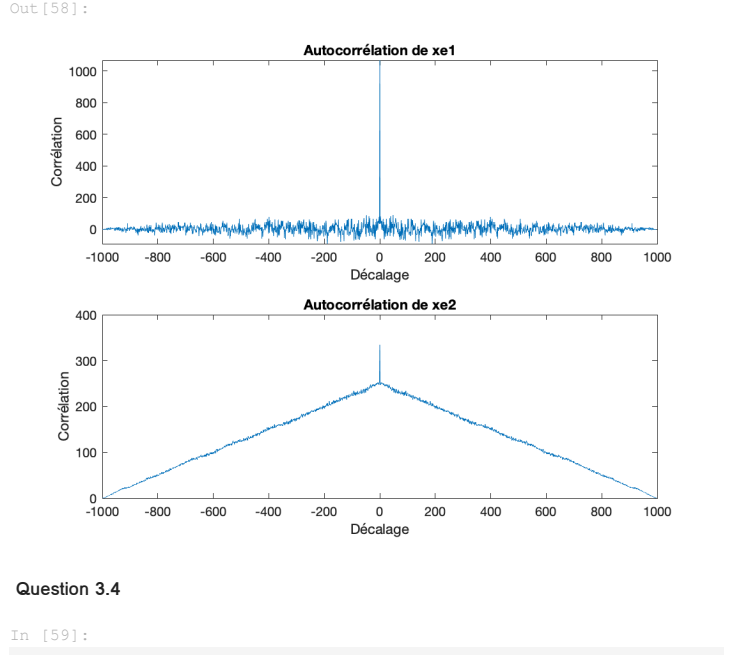
Question 3.3

Le signal le plus adapté comme signal d'excitation pour mesurer la réponse impulsionnelle de la pièce est le signal xe1 : En effet pour supposer que $R_{xx}(u) = d(u)$ soit nul ou casi nul partout sauf en 0 donc on prend le signal à qui l'autocorrélation ressemble le plus à un Dirac

In [58]:

```
load('signal_excitation.mat');
[acf1, lags1] = xcorr(xe1);
[acf2, lags2] = xcorr(xe2);

figure;
subplot(2,1,1);
plot(lags1, acf1);
title('Autocorrélation de xe1');
xlabel('Décalage');
ylabel('Corrélation');

subplot(2,1,2);
plot(lags2, acf2);
title('Autocorrélation de xe2');
xlabel('Décalage');
ylabel('Corrélation');
```

Out[58]:

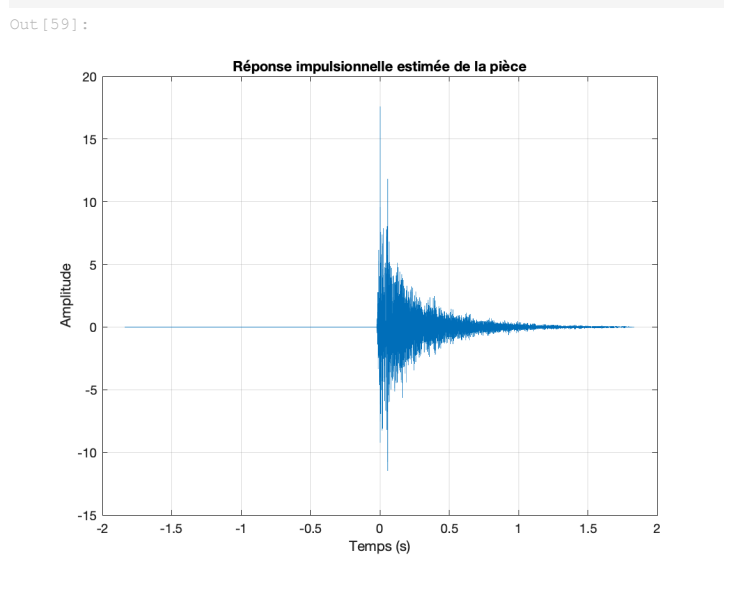


Question 3.4

In [59]:

```
load('signal_excitation.mat');
xe1 = xe1 / max(abs(xe1));
y = simule_piece(xe1, fe);
h_estimee = xcorr(y, xe1);

N = length(h_estimee);
temps = (-N/2:N/2 - 1)/fe;


figure;
plot(temps, h_estimee);
xlabel('Temps (s)');
ylabel('Amplitude');
title('Réponse impulsionnelle estimée de la pièce');
grid on;
```

Out[59]:



Nous observons bien la réponse impulsionnelle typique d'une pièce réverbérante En effet,nous avons bien un pic principal qui correspond au trajet direct du son puis une décroissance progressive et oscillante qui correspond aux réflexions sur les surfaces.

Question 3.5

La fonction effet_reverb se trouve dans le dosser src

Question 3.6

In [60]:

```
[x,fe]=audioread("wav/single_tone_guitar_nylon_string_a2.wav");
tic
y=effet_reverb(x,h);
toc
```

```
Elapsed time is 0.009078 seconds.
```

Convolution rapide

Question 3.7

In [61]:

```
[x1,fe1]=audioread("wav/single_tone_guitar_nylon_string_a2.wav");
tic
y=effet_reverb_FFT(x,h);
toc
```

```
Elapsed time is 0.009528 seconds.
```

Le temps de calcul à l'aide de la fonction effet_reverb_FFT est inférieur à celui par application de convolution directe d'un ordre de 10.

Question 3.8

Oui , la méthode utilisant la FFT est équivalente à la convolution classique de la section.
En effet on obtient la méthode utilisant la FFT en partant de la convolution classique
On a la convolution classique

$$y(k) = h(k) * x(k)$$

On passe a la transformée de Fourier on obtient :

$$FFT(y(k)) = FFT(h(k)). IFFT(x(k))$$

On fait la transformée de Fourier inverse :

$$y(k) = IFFT(FFT(h(k)). FFT(x(k)))$$

Effet de retard

L'équation de récurrence du système est donnée par :

$$y(k) = x(k) - gy(k - \tau)$$

Question 3.9

La réponse impulsionnelle $h(k)$ est la réponse du système lorsque l'entrée $x(k)$ est une impulsion de Dirac $\delta(k)$.

$$x(k) = \delta(k)$$

$$\text{avec } \delta(k) = \begin{cases} 1 & \text{si } k = 0, \\ 0 & \text{sinon.} \end{cases}$$

On a donc :

$$y(k) = \delta(k) - gy(k - \tau)$$

- Pour $k = 0$:

$$y(0) = \delta(0) - gy(0 - \tau)$$
$$y(0) = 1$$

car $\delta(0) = 1$ et $y(0 - \tau) = 0$

- Pour $0 < k < \tau$:

$$y(k) = 0$$

car $\delta(k) = 0$ pour $k \neq 0$ et $k - \tau < 0$, donc $y(k - \tau) = 0$

- Pour $k = \tau$:

$$y(\tau) = -g$$

car $\delta(\tau) = 0$ et $y(\tau - \tau) = y(0) = 1$

- Pour $\tau < k < 2\tau$:

$$y(k) = 0$$

car $\delta(k) = 0$ pour $k \neq 0$ et $y(k - \tau) = 0$ pour $k - \tau \neq 0$ et $k - \tau \neq \tau$

- Pour $k = 2\tau$:

$$y(2\tau) = \delta(2\tau) - gy(2\tau - \tau)$$
$$y(2\tau) = -g \cdot (-g) = g^2$$

car $\delta(2\tau) = 0$ et $y(2\tau - \tau) = y(\tau) = -g$

Par récurrence, on peut établir que :

$$h(k) = \begin{cases} (-g)^n & \text{si } k = n\tau, \\ 0 & \text{sinon.} \end{cases}$$

Question 3.10

Pour que le filtre Effet de delay soit stable, sa réponse impulsionnelle doit être sommable, c'est-à-dire :

$$\sum_{k=0}^{\infty} |h(k)| < \infty$$

Pour $k = n\tau$, on a :

$$\sum_{n=0}^{\infty} |(-g)^n| = \sum_{n=0}^{\infty} |g^n|$$

Cette somme est une série géométrique donnée par :

$$\sum_{n=0}^{\infty} |g|^n = \begin{cases} \frac{1}{1-g} & \text{si } |g| < 1, \\ +\infty & \text{si } |g| \geq 1. \end{cases}$$

Le filtre est stable pour $|g| < 1$.

Question 3.11

On a l'équation du filtre delay :

$$y(k) = x(k) - gy(k - \tau)$$

En isolant $x(k)$, on obtient :

$$x(k) = y(k) + gy(k - \tau)$$

- $a = [a_0, \ldots, a_N]$ est un vecteur comportant les coefficients de la partie récursive
  $a_0 = 1$
  $a_m = 0$ pour $1 \leq m \leq \tau - 1$
  $a_\tau = g$
  On a donc

$$a = [1, \underbrace{0, \ldots, 0}_{\tau-1}, g]$$

- $b = [b_0, \ldots, b_M]$ est un vecteur comportant les coefficients de la partie non-récursive

$$b = [1]$$

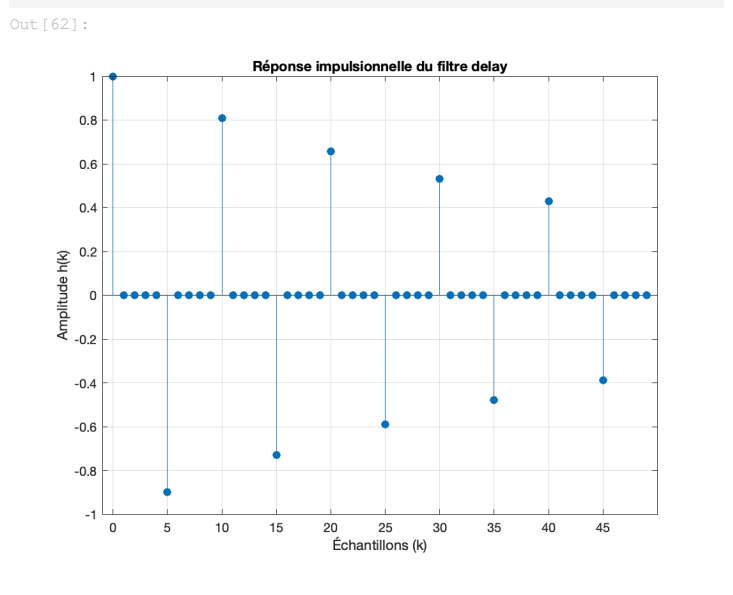
Question 3.12

Utilisons le script analyse_delay pour obtenir la réponse impulsionnelle et crée la représentation temporelle de $h(k)$ en choisissant nous-même le coefficient d'atténuation g, le retard τ et la longueur de la réponse N

In [62]:

```
N=50; %longueur de la réponse
h = analyse_delay(0.9,5,N);

figure;
stem(0:N-1, h, 'filled');
xlabel('Échantillons (k)');
ylabel('Amplitude h(k)');
title('Réponse impulsionnelle du filtre delay');
grid on;
```

Out[62]:



Cette représentation temporelle de $h(k)$ correspond bien à ce qu'on attendait. En effet, le signal de sortie est renvoyé avec 5 échantillons de retard et son amplitude est progressivement atténuée, traduisant la stabilité du filtre, ce qui est normal, comme nous avons choisi $g = 0, 9$. De plus, nous avons bien des oscillations qui proviennent de l'effet de rétroaction $-g$.

Question 3.13

$y(k) = x(k) * h(k)$

Ce qui nous donne dans le domaine fréquentielle :

$$\hat{h}(\nu) = \frac{Y(\nu)}{X(\nu)}$$

Appliquons la transformée de Fourier à l'équation du filtre de retard :

$$Y(\nu) = X(\nu) - gY(\nu)e^{-j2\pi\nu\tau}$$
$$Y(\nu) \cdot (1 + g \cdot e^{-j2\pi\nu\tau}) = X(\nu)$$
$$\hat{h}(\nu) = \frac{Y(\nu)}{X(\nu)} = \frac{1}{1 + g \cdot e^{-j2\pi\nu\tau}}$$

- Déterminons le module de $\hat{h}(\nu)$:

Posons $z = g \cdot e^{-j2\pi\nu\tau}$, alors :

$$\left|\hat{h}(\nu)\right| = \left|\frac{1}{1+z}\right| = \frac{1}{|1+z|}$$

$$z = g \cdot e^{-j2\pi\nu\tau} = g \cdot \cos(2\pi\nu\tau) - j \cdot g \cdot \sin(2\pi\nu\tau)$$

Donc :

$$1 + z = 1 + g \cdot \cos(2\pi\nu\tau) - j \cdot g \cdot \sin(2\pi\nu\tau)$$

Le module de $1 + z$ est :

$$|1 + z| = \sqrt{(1 + g \cdot \cos(2\pi\nu\tau))^2 + (-g \cdot \sin(2\pi\nu\tau))^2}$$
$$= \sqrt{1 + 2g \cdot \cos(2\pi\nu\tau) + g^2 \cdot \cos^2(2\pi\nu\tau) + g^2 \cdot \sin^2(2\pi\nu\tau)}$$
$$= \sqrt{1 + 2g \cdot \cos(2\pi\nu\tau) + g^2}$$

Ainsi, le module de la réponse en fréquence est :

$$\left|\hat{h}(\nu)\right| = \frac{1}{\sqrt{1 + 2g \cdot \cos(2\pi\nu\tau) + g^2}}$$

- Déterminons la phase de $\hat{h}(\nu)$:

$$arg(\hat{h}(\nu)) = -arg((1 + g \cdot e^{-j2\pi\nu\tau}))$$

Posons $z = g \cdot e^{-j2\pi\nu\tau}$, alors :

$$arg(1 + z) = \arctan\left(\frac{\Im(1 + z)}{\Re(1 + z)}\right)$$

où : $\Im(1 + z) = -g \cdot \sin(2\pi\nu\tau), \quad \Re(1 + z) = 1 + g \cdot \cos(2\pi\nu\tau)$

Ainsi :

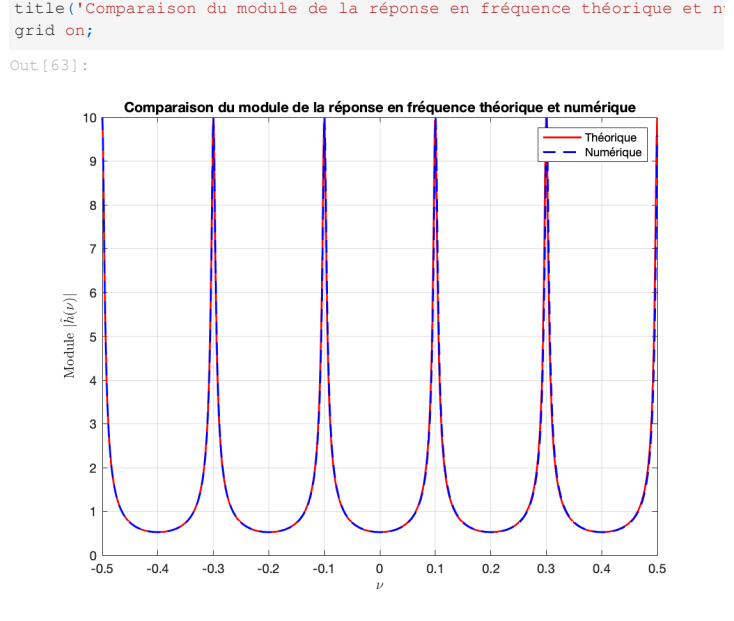
$$arg(1 + z) = \arctan\left(\frac{-g \cdot \sin(2\pi\nu\tau)}{1 + g \cdot \cos(2\pi\nu\tau)}\right)$$
$$arg(\hat{h}(\nu)) = -\arctan\left(\frac{-g \cdot \sin(2\pi\nu\tau)}{1 + g \cdot \cos(2\pi\nu\tau)}\right)$$
$$= \arctan\left(\frac{g \cdot \sin(2\pi\nu\tau)}{1 + g \cdot \cos(2\pi\nu\tau)}\right)$$

Question 3.14

In [63]:

```
g = 0.9;
tau = 5;
N = 1000;
nu = linspace(-0.5, 0.5, N);

% Théorique
H_theorique = 1 ./ sqrt(1 + 2 * g * cos(2 * pi * nu * tau) + g^2);

% Numérique
h = analyse_delay(g, tau, N);
H_num = abs(fft(h, N)); % DFT et module
H_numerique = fftshift(H_num);


figure;
plot(nu, H_theorique, 'r', 'LineWidth', 1.5); hold on;
plot(nu, H_numerique, 'b--', 'LineWidth', 1.5);
xlabel('$\nu$', 'Interpreter', 'latex');
ylabel('Module $|\hat{h}(\nu)|$', 'Interpreter', 'latex');
legend('Théorique', 'Numérique');
title('Comparaison du module de la réponse en fréquence théorique et n
grid on;
```

Out[63]:



Le module de la réponse en fréquence obtenu théoriquement et numériquement sont bien superposés, nous n'observons pas de différence.

Question 3.15

La fonction se trouve dans le dossier src

Question 3.16

In [64]:

```
[x_chords, fe_chords] = audioread("wav/piano_chord.wav");
test_effet_delay = effet_delay(x_chords, 0.25, 0.9,fe_chords);
sound(test_effet_delay, fe_chords);
```

Nous entendons bien l'effet de retard : comme g est presque égal à 1, nous entendons à peine une diminution de volume entre chaque répétition de l'accord, cette répétition est par ailleurs bien distinguable car τ est assez grand.

L'équation de récurrence du système est donnée par :

$$y(k) = x(k) - \frac{g}{K}\sum_{n=0}^{K-1} y(k - \tau - n)$$

Question 3.17

La fonction effet_delay_filtre se trouve dans le dossier src

Question 3.18

In [65]:

```
delay = 0.25;
g=0.9;
K=10;
[x,Fe]=audioread("wav/piano_chord.wav");
y=effet_delay_filtre(x,delay,g,K,Fe);
sound(y, Fe);
audiowrite("wav/piano_delay_filtre.wav", y, Fe);
```

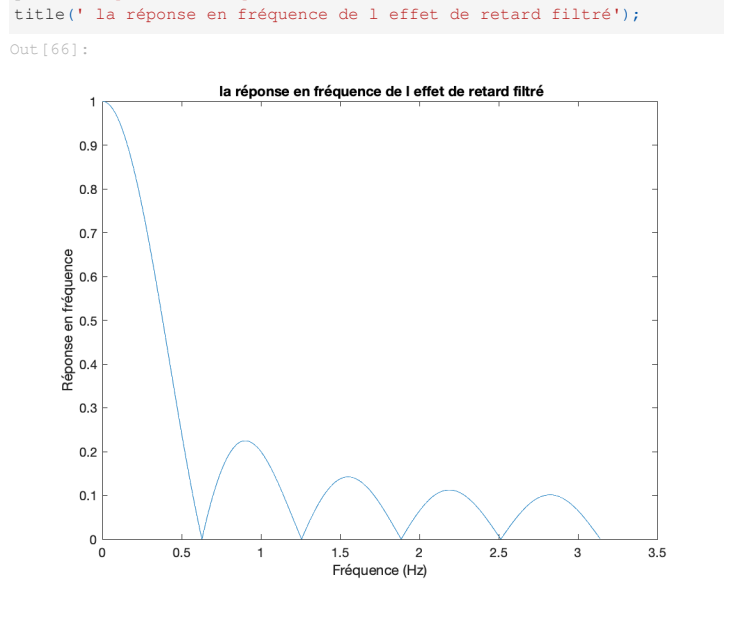
Avec cette nouvelle équation de récurrence, nous prenons en compte la longueur K. De plus, le coefficient d'amortissement n'est plus appliqué qu'à seulement le signal de sortie d'une période d'avant mais à la moyenne des 10 signaux de sortie. Nous entendons donc cette fois-ci une diminution du volume du signal qui s'estompe petit à petit.

Question 3.19

In [66]:

```
K=10 ;
h=ones(1,K)/K;
[H,f]=freqz(h,1,1000);

figure;
plot(f,abs(H));
xlabel('Fréquence (Hz)');
ylabel('Réponse en fréquence');
title(' la réponse en fréquence de l effet de retard filtré');
```

Out[66]:



D'après la figure observé, le filtre "coupe" bien progressivement les hautes-fréquences du signal.

Conclusion

L'utilisation de différentes techniques permettent d'approcher plus ou moins un son réel. Pour notre part, la synthèse additive a plutôt été concluante bien que le son synthétisé ne sonne pas naturel sans les bruits. Cependant, nous n'avons pas réussi à approcher un son réel à l'aide de la synthèse soustractive. Enfin, nous avons tester différents effets qui peuvent corriger des défauts ou rajouter un coté artistique.