

Compte-rendu des travaux pratiques d'ES101

AS

Sommaire :

I. Projet 1

- I.1. Présentation du projet
- I.2. Réalisation du projet
- I.3. Code

II. Projet 2

- II.1. Sujet 1
 - II.1.1. Présentation du projet
 - II.1.2. Réalisation du projet
 - II.1.3. Code
- II.2. Bonus
 - II.2.1. Présentation du projet
 - II.2.2. Réalisation du projet
 - II.2.3. Code

III. Projet 3

- III.1. Sujet 1
 - III.1.1. Présentation du projet
 - III.1.2. Réalisation du projet
 - III.1.3. Code
- III.2. Sujet 2
 - III.2.1. Présentation du projet
 - III.2.2. Réalisation du projet
 - III.2.3. Code

I. Projet 1

I. 1. Présentation du projet

Le but de ce projet était de transformer un fichier audio brouillé en un signal non brouillé.

Un fichier audio brouillé en format wav nous avait été fourni : à l'oreille, nous entendions un signal aigu. A la fin du projet, nous entendions une musique d'opéra.

I. 2. Réalisation du projet

Pour réaliser notre projet, il a été nécessaire de procéder par étapes:

a) Récupération du signal

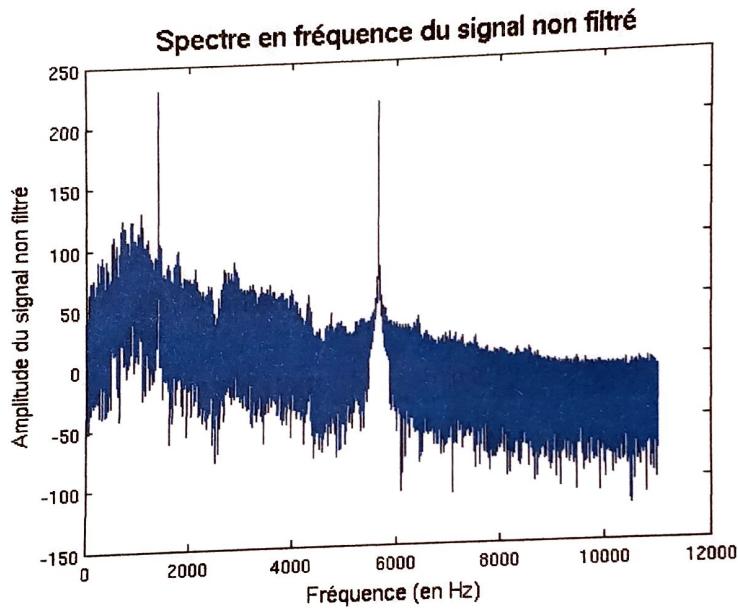
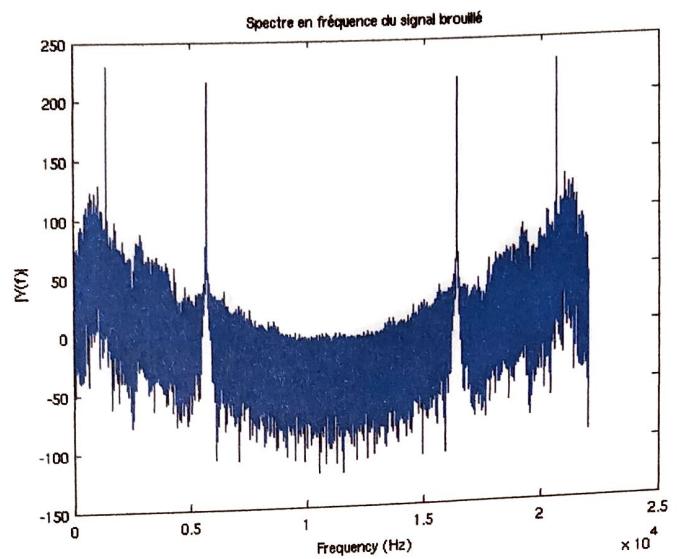
On considère un signal audio '*Mo11.wav*' brouillé par des fréquences. Dans un premier temps, à l'aide de la fonction wavread on extrait le signal dans le domaine temporel.

La fonction wavread crée un vecteur y de taille $F_s \times \text{durée fichier}$ (où F_s désigne la fréquence d'échantillonnage du signal). Ainsi, on obtient le signal échantillonné y du fichier '*Mo11.wav*'.

b) Analyse spectrale

Pour effectuer l'analyse spectrale de notre signal, on utilise la transformée de Fourier (fonction *fft* sur matlab) avec laquelle est obtenu le spectre en fréquence du signal échantillonné Y .

Après avoir remarqué que le spectre en fréquence était symétrique par rapport à $f = F_s/2$, il convient de ne tracer le spectre en fréquence que pour des fréquences comprises entre 0 et $F_s/2$.



c) Repérage du brouillage

Le spectre en fréquence du signal échantillonné révèle la présence de deux pics d'amplitude extrêmement élevée: ces pics correspondent aux fréquences polluantes. Il est donc nécessaire de repérer précisément ces fréquences pour pouvoir ensuite filtrer notre signal et enlever le brouillage. Pour ce faire, il convient de chercher les deux maxima de Y puis de retrouver les fréquences f_1 et f_2 de brouillage correspondantes.

Nous avons trouvé:

$$f_1 = 1\ 397 \text{ Hz}$$

$$f_2 = 5\ 658 \text{ Hz}$$

9.

d) Filtrage

Afin d'obtenir le signal filtré, il a d'abord fallu trouver la fonction de transfert en Z de notre filtre. Pour cela, nous nous sommes servis des fréquences f_1 et f_2 de brouillage trouvées précédemment afin de déterminer les zéros de notre fonction de transfert en Z. Les zéros s'écrivent donc $Z_1 = e^{\frac{j2\pi f_1}{F_s}}$ et $Z_2 = e^{\frac{j2\pi f_2}{F_s}}$ et les pôles s'écrivent $P_1 = pe^{\frac{j2\pi f_1}{F_s}}$ et $P_2 = pe^{\frac{j2\pi f_2}{F_s}}$.

D'où notre fonction de transfert en Z du filtre à appliquer au signal initial :

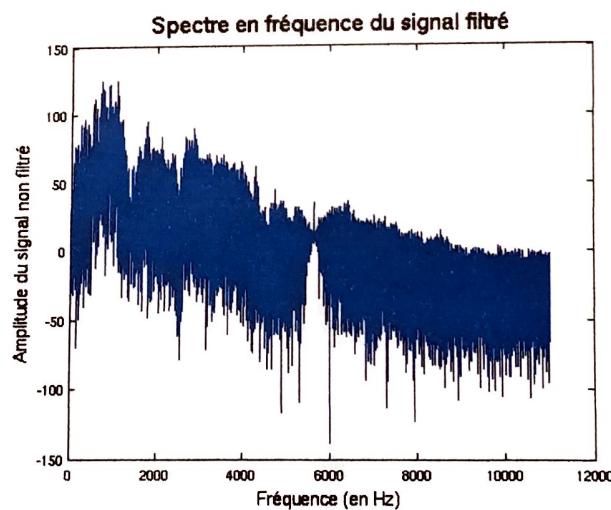
$$H(Z) = \frac{(1-Z_1Z^{-1})(1-Z_1^*Z^{-1})(1-Z_2Z^{-1})(1-Z_2^*Z^{-1})}{(1-P_1Z^{-1})(1-P_1^*Z^{-1})(1-P_2Z^{-1})(1-P_2^*Z^{-1})}$$

Ou encore :

$$H(Z) = \frac{1 - 2(Re(Z_1) + Re(Z_2))Z^{-1} + (|Z_1|^2 + |Z_2|^2 + 4Re(Z_1)Re(Z_2))Z^{-2} - 2(Re(Z_1)|Z_2|^2 + Re(Z_2)|Z_1|^2)Z^{-3} + |Z_1|^2|Z_2|^2Z^{-4}}{1 - 2(Re(P_1) + Re(P_2))Z^{-1} + (|P_1|^2 + |P_2|^2 + 4Re(P_1)Re(P_2))Z^{-2} - 2(Re(P_1)|P_2|^2 + Re(P_2)|P_1|^2)Z^{-3} + |P_1|^2|P_2|^2Z^{-4}}$$

e) Génération du signal corrigé

Après avoir appliqué le filtre de la fonction de transfert $H(Z)$ trouvée précédemment, on obtient le spectre en fréquence ci-dessous :



A partir de ce graphe, on remarque que les deux fréquences f_1 et f_2 de brouillage ont été supprimées. En écoutant le signal créé à l'aide de la fonction `wavwrite` de Matlab, le son est désormais clair et non brouillé.

I. 3. Code

```
clear all
close all

%% Récupération du signal :
[y,Fs]=wavread('M011.wav');

N=length(y);

%% Analyse spectrale :
Y=(fft(y,N)).^2;
f=linspace(0, Fs/2, N/2);

figure
plot(f,10*log(abs(Y(1:N/2))));
title('Spectre en fréquence du signal non filtré', 'FontSize', 15)
xlabel('Fréquence (en Hz)', 'FontSize', 13)
ylabel('Amplitude du signal non filtré', 'FontSize', 13)

%% Repérage du brouillage :
[amp1,ind1]=max(Y(1:N/2));
'1ère fréquence qui brouille :'
freq1=f(ind1);
freq1

Y(ind1)=0;
[amp2,ind2]=max(Y(1:N/2));
'2ème fréquence qui brouille :'
freq2=f(ind2); freq2

%% Filtrage :
rho=0.9

%% Zéros et pôles de notre fonction de transfert :
z1=exp(j*2*pi*freq1/Fs);
z2=exp(j*2*pi*freq2/Fs);
p1=rho*exp(j*2*pi*freq1/Fs);
p2=rho*exp(j*2*pi*freq2/Fs);

% Coefficients du numérateur de notre fonction de transfert en z-1 :%
a4 = 1;
a3 = -2*(real(z1)+real(z2));
a2 = 4*real(z1)*real(z2)+norm(z1)^2+norm(z2)^2;
a1 = -2*(real(z1)*norm(z2)^2+real(z2)*norm(z1)^2);
```

```

a0 = norm(z1)^2*norm(z2)^2;
num = [a4 a3 a2 a1 a0];

% Coefficients du dénominateur de notre fonction de transfert en z-1 :%
b4 = 1;
b3 = -2*(real(p1)+real(p2));
b2 = 4*real(p1)*real(p2)+norm(p1)^2+norm(p2)^2;
b1 = -2*(real(p1)*norm(p2)^2+real(p2)*norm(p1)^2);
b0 = norm(p1)^2*norm(p2)^2;
den = [b4 b3 b2 b1 b0];

y_filtre=filter(num,den,y);

%% Spectre en fréquence du signal filtré : %%

N=length(y_filtre);

Y_filtre=(fft(y_filtre,N)).^2;

figure
plot(f,10*log(abs(Y_filtre(1:N/2))));
title('Spectre en fréquence du signal filtré', 'FontSize', 15)
xlabel('Fréquence (en Hz)', 'FontSize', 13)
ylabel('Amplitude du signal non filtré','FontSize', 13)

%% Génération du fichier corrigé : %%

y_filtre=y_filtre/max(y_filtre); %Normalisation de y_filtre
wavwrite(y_filtre(4:), Fs, 'son_corrigé.wav'); %On enlève les 4
premiers échantillons correspondant au délais du filtre

```

II. Projet 2

II. 1. Sujet 1

II.1.1. Présentation du projet

Le but de ce projet était de transformer un fichier audio contenant un écho en un signal sans écho.

Un fichier audio (contenant un écho) en format wav nous avait été fourni. A la fin du projet, nous entendions une musique nette.

II.1.2. Réalisation du projet

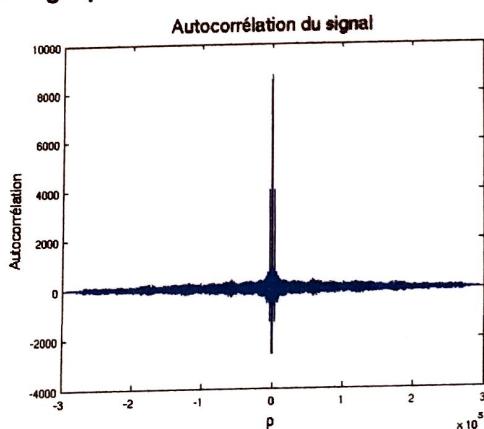
a) Récupération du signal

Comme dans le premier projet, il a d'abord fallu récupérer le signal à l'aide de la fonction `wavread` de Matlab. Cette fonction nous donne le signal y et la fréquence d'échantillonnage F_s .

b) Mise en évidence de l'écho

Afin de constater qu'effectivement, un écho est bien présent dans le fichier audio initial, on trace le graphe d'autocorrélation que l'on obtient en calculant la fonction d'autocorrélation du signal à l'aide de la fonction `xcorr` de Matlab.

On obtient ainsi le graphe d'autocorrélation ci-dessous :



Grâce à ce graphe, on remarque la présence d'un pic important pour p proche de 0 provenant de l'écho.

c) Caractérisation de l'écho

Sachant qu'un écho peut être représenté pour sa fonction de transfert $C(Z) = \frac{1}{1 + \alpha Z^{-n_0}}$ (de sorte que $y(n) = s(n) + \alpha s(n - n_0)$) , il faut trouver le retard n_0 et l'amplitude α du signal.

1. Recherche du retard n_0 de l'écho

Le retard n_0 de l'écho se trouve facilement à l'aide du graphe d'autocorrélation. Il s'agit en effet de la différence entre le deuxième pic le plus important (celui pour p proche de 0) et le pic le plus important (celui pour $p = 0$).

En recherchant sur Matlab l'indice pour lequel le graphe d'autocorrélation présente ce deuxième pic le plus important à l'aide de la fonction *max*, on obtient directement le retard $n_0 = 3418$.

2. Recherche de l'amplitude α de l'écho

Afin de trouver l'amplitude α du signal, nous allons considérer que notre signal est stationnaire (c'est-à-dire $E[s^2(n)] = E[s^2(p)]$) et aléatoire (c'est-à-dire $E[s(n)s(n - p)] = 0$). Ces hypothèses nous permettent d'écrire :

$$\begin{aligned} \bullet \quad r_{yy}(n_0) &= E[y(n)y(n - n_0)] \\ &= E[(s(n) + \alpha s(n - n_0))(s(n - n_0) + \alpha s(n - 2n_0))] \\ &= E[s^2(n) + \alpha(s(n)s(n - 2n_0) + s^2(n - n_0)) + \alpha^2 s(n - n_0)s(n - 2n_0)] \\ &= \alpha E[s^2(n - n_0)] \end{aligned}$$

et :

$$\begin{aligned} \bullet \quad r_{yy}(0) &= E[y^2(n)] \\ &= E[(s(n) + \alpha s(n - n_0))^2] \\ &= E[s^2(n) + 2\alpha s(n)s(n - n_0) + \alpha^2 s^2(n - n_0)] \\ &= (1 + \alpha^2)E[s^2(n - n_0)] \end{aligned}$$

On obtient donc, par la dernière ligne : $E[s^2(n - n_0)] = \frac{r_{yy}(0)}{1 + \alpha^2}$.

Puis, en injectant ce résultat dans la première équation, on a :

$$r_{yy}(n_0) = \frac{\alpha}{1 + \alpha^2} r_{yy}(0)$$

$$\Leftrightarrow \alpha^2 r_{yy}(n_0) - \alpha r_{yy}(0) + r_{yy}(0) = 0$$

En prenant la seule solution inférieure à 1 (il s'agit d'un écho, donc d'amplitude plus petite que celle du signal original), on obtient :

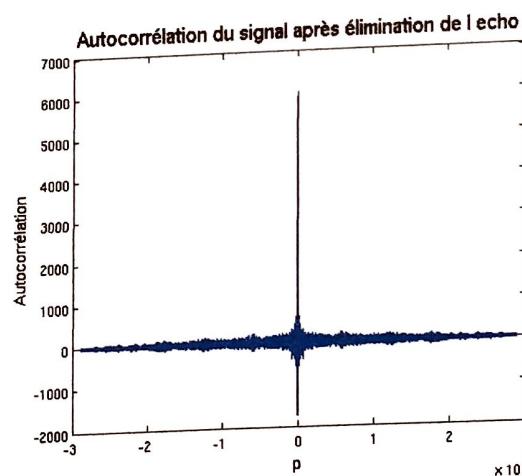
$$\alpha = \frac{r_{yy}(0) + \sqrt{r_{yy}^2(0) - 4r_{yy}^2(n_0)}}{2r_{yy}(n_0)}$$

En cherchant sur Matlab les valeurs de $r_{yy}(0)$ et de $r_{yy}(n_0)$ à l'aide de la fonction *max*, on obtient $\alpha = 0,6525$.

d) Filtrage de l'écho et récupération du signal corrigé

Maintenant que nous connaissons la fonction de transfert $C(Z) = \frac{1}{1 + \alpha Z^{-n_0}}$ de notre écho, il s'agit de filtrer cet écho afin de l'éliminer de notre fichier audio initial. Pour cela, nous procédons comme pour le premier TP en utilisant la fonction *filt er* de Matlab.

Le graphe d'autocorrélation du signal après filtrage se trouve ci-dessous.



On remarque que ce graphe ne présente qu'un pic dominant pour $p = 0$: l'écho a bien été supprimé.

On récupère ensuite ce signal corrigé à l'aide de la fonction *wavwrite* de Matlab. Le son est désormais net.

II.1.3. Code

```
clear all
close all

%% Récupération du signal %%
[y,Fs]=wavread('Pall.wav');

%% Corrélation du signal %%
c=xcorr(y);

%% Affichage du graphe d'autocorrélation %%
N=length(c);

p=linspace(-N/2, N/2, N);

figure
plot(p,c);
title('Autocorrélation du signal', 'Fontsize', 15)
xlabel('p', 'Fontsize', 13)
ylabel('Autocorrélation', 'Fontsize', 13)

%% Recherche du retard de l'écho %%
[Rxx_n0, Retard]=max(c(N/2+10:N));
Retard=Retard+10-1 ✓

%% Recherche de l'amplitude de l'écho %%
Rxx_0=max(c);
Alpha1=(Rxx_0+sqrt(Rxx_0^2-4*Rxx_n0^2))/(2*Rxx_n0);
Alpha2=(Rxx_0-sqrt(Rxx_0^2-4*Rxx_n0^2))/(2*Rxx_n0);

if Alpha1<1
    Alpha=Alpha1
end

if Alpha2<1
    Alpha=Alpha2
end

%% Filtrage de l'écho %%
Num=1;

Den=zeros(1, Retard+1);
Den(1)=1;
Den(Retard+1)=Alpha;
```

```

y_filtre=filter(Num,Den,y);

%% Vérification de l'élimination de l'écho %%
c_filtre=xcorr(y_filtre);
N=length(c_filtre);

p=linspace(-N/2, N/2, N);

figure
plot(p,c_filtre);
title('Autocorrélation du signal après élimination de l echo', 'FontSize',
15)
xlabel('p','FontSize', 13)
ylabel('Autocorrélation','FontSize', 13)

%% Récupération du signal %%

wavwrite(y_filtre(:), Fs, 'Pall_corrige.wav'); %On enlève les 4 premiers
échantillons correspondant au délais du filtre

```

II. 2. Bonus

II.2.1. Présentation du projet

On se propose de corriger un signal sonore qui a été modifié de la façon suivante :
 Les hautes fréquences et les basses fréquences ont été permutées, à l'exception de la fréquence nulle et de la fréquence $F_e/2$.

II.2.2. Réalisation du projet

Notre idée a été de ré-inverser les fréquences pour revenir au signal de départ.

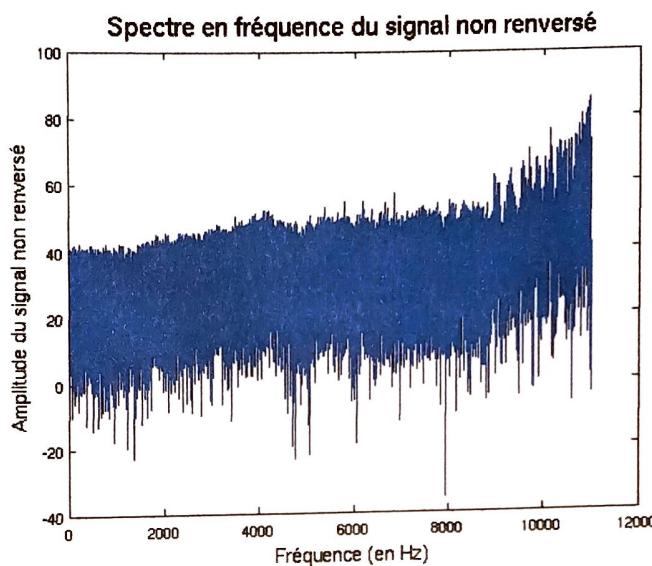
Pour réaliser notre projet nous avons toujours procédé par étapes :

a) Récupération du signal

Comme dans le premier projet, il a d'abord fallu récupérer le signal à l'aide de la fonction `wavread` de Matlab. Cette fonction nous donne le signal y et la fréquence d'échantillonnage F_s .

b) Analyse spectrale

On utilise la transformée de Fourier (fonction *fft* sur matlab) avec laquelle est obtenu le spectre en fréquence du signal échantillonné *Y*.



c) Modification du vecteur Y

On inverse les composantes du vecteur *Y*, grâce à la fonction *flipdub*.

d) Génération du signal corrigé

On applique la transformée de Fourier inverse à *Y* puis on utilise la fonction *wavwrite* pour écrire le fichier au format *wav*.

II.2.3. Code

```
%% Spectre en fréquence du signal brouillé : %%
[y,Fs]=wavread('canal.wav');

N=length(y);

Y=(fft(y,N));
f=linspace(0, Fs/2, N/2);
```

%% Permutation des fréquences :

```
Y(2:round(N/2))=flipud(Y(2:round(N/2)));
Y(round(N/2)+2:N)=flipud(Y(round(N/2)+2:N));
```

%% FFT inverse du signal avec les fréquences renversées :

```
y=real(ifft(Y));
```

```
wavwrite(y, Fs, 'canal_corrigé.wav');
```

III. Projet 3

III. 1. Sujet 1

III.1.1. Présentation du projet

On considère non plus un son mais une image brouillée. Chaque ligne de cette image a été décalée de quelques pixels par rapport à la précédente.

III.1.2. Réalisation du projet

a) Récupération et affichage de l'image

On considère une image 'fichier2.bmp' brouillée. Dans un premier temps, à l'aide de la fonction *imread*, on lit le fichier et on le stocke dans une variable *B* qui va être une matrice de *n* lignes et *m* colonnes (dimensions de l'image). Ainsi, un pixel correspondra à la valeur d'une case de cette matrice. Les valeurs prises sont 0 (blanc) et 255 (noir).

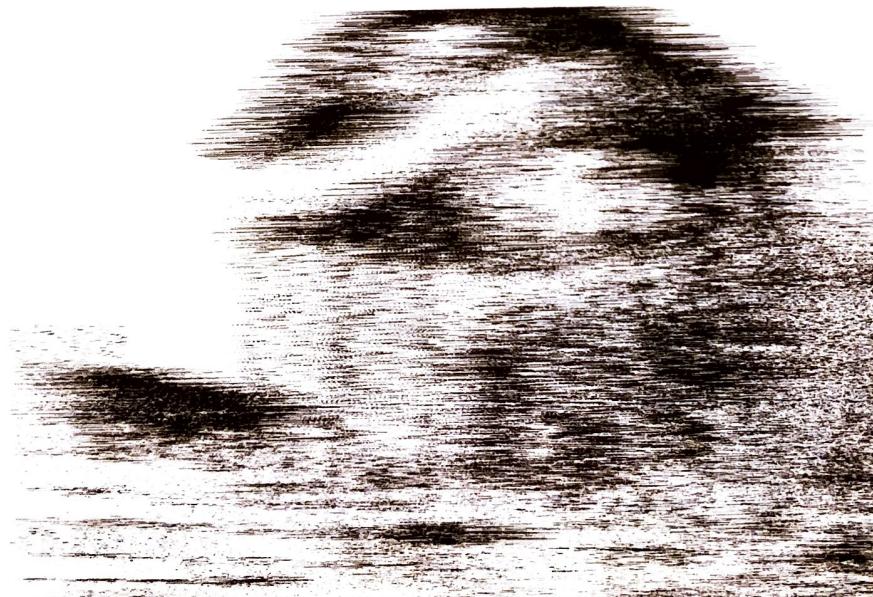


Image brouillée

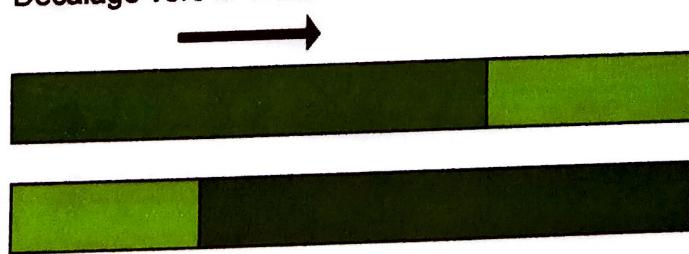
b) Repérage et correction des décalages

Chaque ligne $i+1$ ($i > 0$) ont été décalées de quelques pixels par rapport à la précédente (ligne i), soit vers la gauche, soit vers la droite

Décalage vers la gauche :



Décalage vers la droite :



Afin de savoir quel est le décalage entre la ligne i et la ligne $i+1$ il convient de tracer le graphe d'autocorrélation entre les deux lignes à l'aide de la fonction `xcorr`. En effet, l'indice du maximum du graphe obtenu nous renseignera le décalage qui existe entre les deux lignes. Ainsi, le décalage correspondra à $m - \text{indice}$ où m est le nombre de colonnes de l'image et indice l'indice du maximum obtenu (puisque le graphe d'autocorrélation possède $2m+1$ valeurs).

Pour rétablir l'image d'origine, nous avons corrigé notre image « au fur et à mesure » à l'aide d'une boucle `for` : nous corrigeons la ligne $i+1$ par rapport à la ligne i pour des valeurs de i croissantes (ainsi les lignes étaient toujours corrigées par rapport à la ligne 1). Nous avons décidé d'utiliser la fonction `circshift` prenant en paramètre un vecteur et un décalage et décalant ce vecteur selon le décalage donné. Ainsi, le décalage à appliquer correspondait à l'opposé du décalage trouvé avec le graphe d'autocorrélation ($\text{indice} - m$).

Remarque : la fonction `circshift` prenant en paramètre un vecteur sous forme de colonne, nous avons dû convertir la ligne en colonne (il suffisait de prendre la transposée du vecteur $B(i+1, :)$).

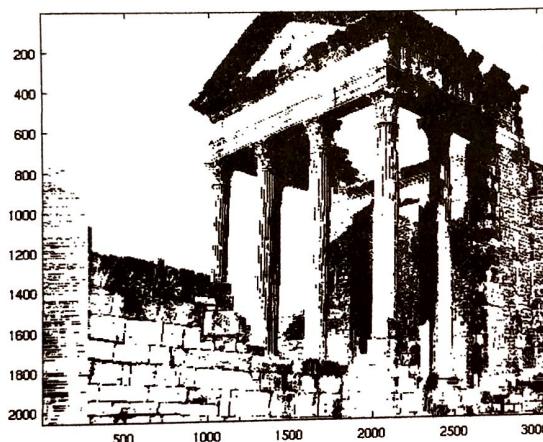


Image corrigée

III.1.3. Code

```
%% Récupération et affichage de l'image %  
  
B=imread('fichier2.bmp','bmp') ;  
B=255*B ;  
Image(B) ;  
  
colormap('Gray');  
n=length(B(:,1));
```

```

m=length(B(1,:));

%% Repérage et correction des décalages %%
for i=1:(n-1)
    ligne1=B(i,:); %On récupère la ligne i
    ligne2=B(i+1,:); %On récupère la ligne i+1
    c=xcorr(ligne1,ligne2);
    [maximum, indice]=max(c);
    decalage=indice-m;

    %On transforme notre ligne i+1 en colonne pour pouvoir appliquer
    %circshift :
    colonne=B(i+1,:)';
    colonne_decalee=circshift(colonne, decalage);

    %On remet en place notre ligne :
    B(i+1,:)=colonne_decalee';
end

%% On lit l'image %%
figure
image(B);
colormap('Gray');

```

III. 2. Sujet 2

III.2.1. Présentation du projet

Ce projet illustre la récupération puis l'écoute d'un signal via deux signaux initialement brouillés. Pour cela, on dispose de deux signaux x_1 et x_2 tels que :

$$x_1(n) = a_1 x(n) + b_1 w(n)$$

$$x_2(n) = a_2 x(n) + b_2 w(n)$$

où $x(n)$ est le signal utile recherché, $w(n)$ un signal brouilleur additif décorrélé de $x(n)$ et les coefficients a_1 , b_1 , a_2 et b_2 sont tels que $b_1 \gg a_1$ et $b_2 \gg a_2$.

Afin d'arriver à l'objectif, on utilisera pour cela la technique d'inversion de puissance.

III.1.2. Réalisation du projet

La technique d'inversion de puissance consiste à trouver la valeur de ρ qui minimise la puissance du signal $\varepsilon(n) = x_1(n) - \rho x_2(n)$ afin d'inverser le rapport signal sur brouilleur de $\varepsilon(n)$ vis-à-vis de celui du signal x_2 .

1. Calcul de la puissance de $\varepsilon(n)$

La puissance de $\varepsilon(n)$ s'écrit :

$$\begin{aligned} E[\varepsilon^2(n)] &= E[x_1^2(n) - 2\rho x_1(n)x_2(n) + \rho^2 x_2^2(n)] \\ &= E[x_1^2(n)] - 2\rho E[x_1(n)x_2(n)] + \rho^2 E[x_2^2(n)] \end{aligned}$$

2. Minimisation de la puissance de $\varepsilon(n)$

La valeur ρ_0 minimisant la puissance de $\varepsilon(n)$ s'écrit :

$$\begin{aligned} \frac{dE[\varepsilon^2(n)]}{d\rho}(\rho_0) = 0 &\Leftrightarrow -2E[x_1(n)x_2(n)] + 2\rho_0 E[x_2^2(n)] = 0 \\ &\Leftrightarrow \rho_0 = \frac{E[x_1(n)x_2(n)]}{E[x_2^2(n)]} \end{aligned} \tag{1}$$

3. Inversion du rapport signal sur brouilleur sur $\varepsilon(n)$

Notons $\varepsilon(n) = a_3 x(n) + b_3 w(n)$. Nous supposerons dans la suite que $x(n)$ et $w(n)$ sont deux signaux décorrélés, centrés et normalisés.

Afin de montrer que le rapport signal sur brouilleur sur $\varepsilon(n)$ est inverser vis-à-vis celui de $x_2(n)$ (c'est-à-dire que $b_3 \ll a_3$), on va d'abord montrer que $x_2(n)$ et $\varepsilon(n)$ sont décorrélés.

- Montrons que $x_2(n)$ et $\varepsilon(n)$ sont décorrélés.

$$\begin{aligned} E[x_2(n)\varepsilon(n)] &= E[x_2(n)(x_1(n) - \rho_0 x_2(n))] \\ &= E[x_1(n)x_2(n)] - \rho_0 E[x_2^2(n)] \\ &= 0 \text{ d'après (1)} \end{aligned}$$

- Montrons que $b_3 \ll a_3$.

On a également :

$$\begin{aligned} E[x_2(n) \varepsilon(n)] &= E[(a_2x(n) + b_2w(n))(a_3x(n) + b_3w(n))] \\ &= a_2a_3 + b_2b_3 \\ &= 0 \text{ d'après le résultat précédent} \end{aligned}$$

Donc $\left| \frac{a_3}{b_3} \right| = \left| \frac{a_2}{b_2} \right| \ll 1$, ce qui montre que $b_3 \ll a_3$.

III.1.3. Code

```
clear all
close all

%% Récupération des signaux
[x1,Fs1]=wavread('x1.wav');
[x2,Fs2]=wavread('x2.wav');
n=length(x1);

%% Détermination de rho par calcul littéral.
rho=mean(x1.*x2)/mean(x2.^2);

%% Ecriture du signal wav :
epsi=x1-rho*x2;
wavwrite(epsi, Fs1, 'x_corrige.wav');
```