METHODES A HAUTE RESOLUTION

Augustin **HOFF** et Gustavo **CIOTTO PINTON**

Projet de synthèse, Supélec, 2014

Table des matières

1 Généralités

1.1 Introduction

Le but de ce projet est d'étudier les méthodes à haute résolution à travers la compréhension et l'approfondissement d'algorithmes connus sur ce sujet comme MUSIC. L'implémentation de ce dernier sur Matlab va permettre de mettre en pratique les études théoriques abstraites basées sur des statistiques ou des espaces hermitiens par exemple.

Une méthode à haute résolution est une méthode qui permet de mesurer des directions (ou des positions ou des fréquences en analyse spectrale) avec une erreur qui n'est limitée, dans le cas idéal, que par la durée d'observation du phénomène. Elle permet donc de résoudre le problème suivant : comment peut-on décrire la distribution géographique et même fréquentielle de sources finies à partir seulement de données fournies par des capteurs (le bruit constituera ici la principale difficulté).

Des applications sont toutes trouvées comme le sonar ou encore le radar, où le principe consiste à déterminer la direction d'arrivée des ondes électromagnétiques (radar) ou acoustiques (sonar) afin de localiser les sources.

Les signaux étudiés seront aléatoires à composante exponentielle (complexe) et feront intervenir des notions de probabilité.

1.2 Outils mathématiques

1.2.1 Quelques définitions

- L'autocorrelation d'un processus aléatoire pour deux indices de temps différents n_1 et n_2 est définie comme

$$r_{xx} = \mathcal{E}\{x[n_1]x^*[n_2]\} \tag{1}$$

- L'autocovariance est définie comme

$$c_{xx}[n_1, n_2] = \mathcal{E}\{(x[n_1] - \overline{x}[n_1])(x^*[n_2] - \overline{x}^*[n_2])\} \tag{2}$$

où

$$\overline{x}[n] = \mathcal{E}\{x[n]\} \tag{3}$$

D'après les équations 1 et 2, on peut alors écrire

$$c_{xx}[n_1, n_2] = r_{xx} - \overline{x}[n_1]\overline{x}^*[n_2] \tag{4}$$

On note ainsi que si la moyenne est nulle, l'autocovariance est égale à l'autocorrelation.

- Un processus aléatoire est dit stationnaire au sens large (*wide-sense stationary* en anglais) si $\overline{x}[n] = \mathcal{E}\{x[n]\} = constante\ \forall n$, et si son autocorrelation dépend seulement de la différence $m = n_2 - n_1$. En résumé

$$\overline{x}[n] = \overline{x}$$

et

$$r_{xx}[m] = \mathcal{E}\{x[n+m]x^*[n]\} \tag{5}$$

- Quelques propriétés remarquables peuvent être listées comme, par exemple

$$r_{xx}[0] \geq |r_{xx}[m]|$$

et

$$r_{xx}[-m] = r_{xx}^*[m]$$

- La matrice formée par les valeurs d'autocorrelation est notée R_{N-1} et est écrite comme

$$R_{N-1} = \begin{bmatrix} r_{xx}[0] & r_{xx}[-1] & r_{xx}[-2] & \cdots & r_{xx}[-(N-1)] \\ r_{xx}[1] & r_{xx}[0] & r_{xx}[-1] & \cdots & r_{xx}[-(N-2)] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{xx}[N-1] & r_{xx}[N-2] & r_{xx}[N-3] & \cdots & r_{xx}[0] \end{bmatrix} \tag{6}$$

- Le rapport entre les puissances du signal et du bruit est appellé SNR (*Signal-to-Noise Ratio*) dont expression est donnée par

$$SNR = \frac{P_{signal}}{P_{bruit}} \tag{7}$$

Il est souvent écrit en *décibels* [dB] :

$$SNR_{dB} = 10 * \log\left[\frac{P_{signal}}{P_{bruit}}\right] \tag{8}$$

- La puissance d'un processus aléatoire est calculée par l'équation suivante

$$P_x = \frac{1}{2\pi}\int_{-\pi}^{\pi} S_x(\omega)d\omega = \int_{-\frac{1}{2}}^{\frac{1}{2}} S_x(\nu)d\nu \tag{9}$$

où $S_x(f) = \mathcal{F}\{r_{xx}[k]\}$, la transformée de Fourier de la fonction d'autocorrelation, est appellé la *densité spectrale de puissance*. Pour les processus stationnaires, on note que

$$P_x = r_{xx}[0] \tag{10}$$

Démonstration :

$$P_x = \frac{1}{2\pi}\int_{-\pi}^{\pi} S_x(\omega)d\omega = \frac{1}{2\pi}\int_{-\pi}^{\pi}\left(\sum_{k=-\infty}^{\infty} r_{xx}[k]\exp(-j\omega k)\right)d\omega =$$

$$= \sum_{k=-\infty}^{\infty} r_{xx}[k]\left(\int_{-\pi}^{\pi}\exp(-j\omega k)d\omega\right) = \sum_{k=-\infty}^{\infty} r_{xx}[k]\delta[k] = r_{xx}[0]$$

1.2.2 Application à un signal sinusoïdal complexe

Dans notre cas, on considère $s[n]$ comme une somme de P sinusoïdes complexes telles que

$$s[n] = \sum_{k=1}^{P} A_k \exp(j(2\pi f_k nT + \theta_k)) \tag{11}$$

T correspond à la période d'échantillonnage. On suppose que les variables aléatoires θ_k sont idépendantes et uniformement distribuées sur l'intervalle $[0, 2\pi[$. Ainsi le calcul de la moyenne et de l'autorrelation pour s[n] est simplifié et donne grâce à 1 et 3 :

$$\overline{s}[n] = 0 \tag{12}$$

et

$$r_{ss}[m] = \sum_{k=1}^{P} A_k^2 \exp(j2\pi f_k mT) \tag{13}$$

Démonstration :

$$\overline{s}[n] = \int_{-\infty}^{\infty} A_k \exp(j(2\pi f_k nT + \theta)) \frac{\mathbf{1}_{[0,2\pi[}}{2\pi} \, \mathrm{d}\theta = 0$$

et

$$r_{xx}[n_1, n_2] = \mathcal{E}\{s[n_1]s^*[n_2]\} = \mathcal{E}\{\sum_{i=1}^{P} A_i \exp(j(2\pi f_i(n_1)T + \theta_i))(\sum_{j=1}^{P} A_j^* \exp(-j(2\pi f_j n_2 T + \theta_j))\} =$$

$$= \mathcal{E}\{\sum_{i=1}^{P} A_i^2 \exp(j2\pi f_i(n_1 - n_2)T)\} + \mathcal{E}\{\sum_{\substack{(i,j)\in[\![1,P]\!] \\ i\neq j}} A_i A_j^* \exp(j(2\pi(f_i n_1 - f_j n_2)T + \theta_i - \theta_j))\}$$

En effet, les θ_k étant indépendantes, alors les variables aléatoires résultantes d'une composition avec une exponentielle le sont aussi (car c'est une fonction continue donc mesurable). On en déduit que

$$\mathcal{E}\{\sum_{\substack{(i,j)\in[\![1,P]\!] \\ i\neq j}} A_i A_j^* \exp(j(2\pi(f_i n_1 - f_j n_2)T + \theta_i - \theta_j))\} =$$

$$= \sum_{\substack{(i,j)\in[\![1,P]\!] \\ i\neq j}} \mathcal{E}\{A_i \exp(j(2\pi f_i n_1 T + \theta_i))\}\mathcal{E}\{A_j^* \exp(-j(2\pi f_j n_2 T + \theta_j))\}$$

Ce terme est donc nul de la même façon que pour le calcul de $\overline{s}[n]$. Le terme restant donne le résultat. On note que le processus $s[n]$ est stationnaire (WSS).

Par analogie avec la lumière blanche qui contient toutes les fréquences lumineuses avec la même intensité, un bruit blanc est un processus stochastique qui possède la même densité spectrale de puissance à toutes les fréquences. Ceci correspond à une autocorrélation nulle en tout point sauf à l'origine : le processus est décorrélé.

Si on ajoute alors un bruit blanc $w[n]$ de variance p_w à $s[n]$, le processus $x[n] = s[n] + w[n]$ aura comme autocorrelation l'expression

$$r_{xx}[m] = r_{ss}[m] + r_w[m] = \sum_{k=1}^{P} A_k^2 \exp(2\pi f_k m T) + p_w \delta[m] \tag{14}$$

et matrice d'autocorrelation

$$R_{xx} = R_{ss} + R_w = \sum_{k=1}^{P} A_k^2 \boldsymbol{e}_{N-1}(f_k) \boldsymbol{e}_{N-1}^H(f_k) + p_w \boldsymbol{I} \tag{15}$$

où

$$\boldsymbol{e}_{N-1}(f_k) = \begin{bmatrix} 1 \\ \exp(2\pi j f_k T) \\ \vdots \\ \exp(2\pi j f_k (N-1) T) \end{bmatrix} \tag{16}$$

Les deux matrices R_{ss} et R_w ont pour dimension $N \times N$.

Enfin, on calcule la puissance associée au signal d'entrée à partir de l'équation 10. Cela donne

$$P_{ss} = r_{ss}[0] = \sum_{k=1}^{P} A_k^2 \tag{17}$$

et d'après l'équation 8, on obtient le rapport signal-bruit (SNR) correpondant

$$SNR_{dB} = 10 * \log \left[\frac{P_{signal}}{P_{bruit}} \right] = 10 * \log \left[\frac{\sum_{k=1}^{P} A_k^2}{p_w} \right] \tag{18}$$

1.2.3 Estimation de fréquences par l'analyse des éléments propres

Étant donnée l'équation 15, on peut en déduire que si la matrice d'autocorrelation a un ordre N plus grand ou égal au nombre de sinusoïdes complexes P, i.e $N-1 > P$, alors la matrice d'autocorrelation des signaux, R_{ss} a pour rang P. En effet, les vecteurs $\boldsymbol{e}_{N-1}(f_k)$ engendrent l'image de R_{ss}. Cela peut se voir soit en revenant à la définition 6, soit en remarquant tout simplement que $\boldsymbol{e}_{N-1}(f_k)\boldsymbol{e}_{N-1}^H(f_k)$ est un des P projecteurs de rang 1.

De façon plus simple , on peut vérifier que la matrice d'autocovariance du bruit R_w a rang N puisqu'elle est multiple d'une matrice identité $N \times N$.

En outre, la matrice R_{ss} qui est hermitienne par définition peut être écrite grâce à ses valeurs et vecteurs propres

$$R_{ss} = \sum_{i=1}^{N} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^H \tag{19}$$

où $\lambda_1 > \lambda_2 > \lambda_3 > \ldots > \lambda_N \geq 0$ et $\boldsymbol{v}_i \boldsymbol{v}_j^H = 0$ si $i \neq j$ et 1 sinon (vecteurs orthogonaux et de norme 1).

On peut démontrer [1] qu'une matrice de dimension N avec $P < N$ a $N-P$ valeurs propres égales à 0. L'équation 19 devient donc

$$R_{ss} = \sum_{i=1}^{P} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^H \tag{20}$$

Une representation alternative de la matrice identité $N \times N$ en fonction des vecteurs propres est

$$\boldsymbol{I} = \sum_{i=1}^{N} \boldsymbol{v}_i \boldsymbol{v}_i^H \tag{21}$$

En remplaçant les équations 20 et 21 dans 15 on obtient

$$R_{xx} = R_{ss} + R_w = \sum_{i=1}^{P} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^H + p_w \sum_{i=1}^{N} \boldsymbol{v}_i \boldsymbol{v}_i^H = \sum_{i=1}^{P} (\lambda_i + p_w) \boldsymbol{v}_i \boldsymbol{v}_i^H + p_w \sum_{i=P+1}^{N} \boldsymbol{v}_i \boldsymbol{v}_i^H \tag{22}$$

Ainsi, les vecteurs propres $\boldsymbol{v}_{P+1}, \boldsymbol{v}_{P+2}, \ldots, \boldsymbol{v}_N$ construisent l'espace bruit de R_w, tous avec la même valeur propre p_w et les vecteurs $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_P$, dits principaux, construisent l'espace des signaux de R_{xx} et R_{ss} avec comme valeurs propres $\lambda_1 + p_w, \lambda_2 + p_w, \ldots, \lambda_P + p_w$.

C'est important de remarquer que les valeurs propres des vecteurs principaux dépendent du signal mais aussi du bruit, ce qui peut générer des complications dans le cas où la variance du bruit blanc est relativement grande.

Enfin, des algorithmes (comme *MUSIC* et *ESPRIT*), abordés plus précisement dans la suite, sont basés sur le fait que les vecteurs qui répresentent les signaux sont tous orthogonaux aux vecteurs qui décrivent le bruit. Ainsi, lorsqu'une entrée $\boldsymbol{s}$ est captée, il est possible déterminer si une telle entrée est essentiellement composée de bruit ou non grâce au produit scalaire

$$\boldsymbol{s}^H \left(\sum_{k=P+1}^{N} \alpha_k \boldsymbol{v}_k \right) \tag{23}$$

qui théoriquement vaut 0 si l'entrée est un signal sinusoïdal complexe.

1. Le théorème du rang donne la dimension du noyau : N-P

2 Algorithme MUSIC

2.1 Aspect théorique

L'algorithme MUSIC [2] se base exactement sur le même principe que l'équation 23 pour détecter des pics mais, par contre, utilise une autre expression qui dépend explicitement de la fréquence. Celle-ci s'écrit :

$$\sum_{k=P+1}^{N} \alpha_k \left| \boldsymbol{e}_{N-1}^H(f)\boldsymbol{v}_k \right|^2 = \boldsymbol{e}_{N-1}^H(f) \left(\sum_{k=P+1}^{N} \alpha_k \boldsymbol{v}_k \boldsymbol{v}_k^H \right) \boldsymbol{e}_{N-1}(f) \tag{24}$$

où, seulement pour cet algorithme, $\alpha_k = 1 \ \forall k$.

On a que 23, 24 valent 0 si $\boldsymbol{e}_{N-1}(f)$ est un des vecteurs de l'espace signal i.e si $f = f_k$ pour k $\in [\![1, P]\!]$, puisqu'il est orthogonal à toute combinaison linéaire des vecteurs qui construisent l'espace du bruit.

L'estimateur est donc donné par l'expression

$$P_{MUSIC}(f) = \frac{1}{\boldsymbol{e}_{N-1}^H(f) \left(\sum_{k=P+1}^{N} \boldsymbol{v}_k \boldsymbol{v}_k^H \right) \boldsymbol{e}_{N-1}(f)} \tag{25}$$

et tend vers l'infini pour des fréquences qui composent le signal.

2.2 Mise en oeuvre

En s'appuyant sur la théorie, on peut alors écrire un algorithme afin de déterminer les fréquences d'un signal source dissimulé dans du bruit. Le resultat est obtenu grâce à la détermination des pics de la fonction Pmusic, explicitée ci-dessus. Voici le code détaillé :

```
%%  Fonction P_music.m
%   Gustavo CIOTTO PINTON, Augustin HOFF
%   Supelec 2014
%
%   Parametres: signal_x   : signal d'entree (vecteur 1xN)
%               frequences: vecteur des frequences recherchees
%                           (vecteur 1xM)
%               P          : nombre de signaux complexes
%
%   Sortie:     resultat   : vecteur avec les resultats pour chaque
%                            frequence (vecteur 1xM)

function [ resultat ] = P_music( signal_x,  frequences , P)

N = size(signal_x ,2);
T = 1/N;

%% Calcul de la matrice d'autocorrelation
z   = corrmtx( signal_x , 0 );
Rxx = toeplitz(z);

%% Calcul des valeurs et vecteurs propres de la matrice d'autocorrelation
[mVecteursPropres, mValeursPropres] = eig(Rxx);
```

2. MUltiple SIgnal Classification

```
24
25 % classement dans l'ordre croissant des valeurs propres
26 [valeursPropresClassees,ordre] = sort(abs(diag(mValeursPropres)));
27
28 % valeurs propres correspondantes au sous espace du bruit
29 valeursPropresBruit = valeursPropresClassees (1:N-P);
30
31 % indices correspondant aux N-P plus petites vap du bruit
32 ordreBruit = ordre (1:N-P);
33
34 %% Calcul de la somme des produits v*vH (ou v est un des vecteurs propres
35 %% qui construisent le sous espace du bruit et vH est son transpose
36 %% conjugue) selon equation 18
37 resultatPartiel = zeros( N, N );
38
39 % on sait qu'il en y a N - P. Pour chacun d'eux, on calcule le produit
40 % correspondant et on l'ajoute au resultat partiel initialise a la matrice
41 % nulle
42 for k = 1:N-P
43
44     indice = ordreBruit(k);
45     vecteurPropreBruit = mVecteursPropres(:, indice);
46
47     % variable auxiliaire pour le calcul v*vH
48     aux = vecteurPropreBruit * conj (vecteurPropreBruit');
49
50     resultatPartiel = resultatPartiel + aux;
51
52 end
53
54 %% Pour chacune des frequences, on construit le vecteur exponentiel (eq. 12)
55 %  et le multiplie par la somme des v*vH. Par fin, on applique l'eq. 19.
56 NFrequences = size(frequences, 2);
57 resultat = zeros (1, NFrequences );
58
59 for n = 1:NFrequences
60
61     % vecteur exponentiel (eq 2)
62     temps = (0 : 1 : (N-1))';
63     eFrequences = exp(2*pi*1i*T*frequences(n)*temps);
64
65     % equation 18
66     aux = conj(eFrequences') * resultatPartiel * eFrequences;
67
68     resultat (n) = abs (1/aux); % equation 19
69 end
```

Après avoir écrit l'algorithme principal, on réalise une fonction test avec les mêmes notations que pour la théorie.

```
%%   Programme test.m
%    Gustavo CIOTTO PINTON, Augustin HOFF
%    Supelec 2014
%    Programme testant la fonction P_music

%% Parametres initiaux
clear all; close all; clc;

N = 100;              % nombre de points mesures
n = 0: 1 : N - 1;

P = 3;                % nombre de frequences differentes
T = 1/N;
fDebut = 1e0;         % frequence la plus petite recherchee
fFin   = 7e1;         % frequence la plus haute recherchee
fPas   = 1e-1;

%% Informations sur les signaux complexes
A = [1 2.5 3];          % amplitudes
F = [4.9e1 5e1 6e1];  % frequences recherchees
Theta = 2*pi*[rand(1) rand(1) rand(1)]; % phases aleatoires
                                        % uniformement reparties
                                        % sur [0,2pi[

SNR = [0.001 40 100 500]; %(dB)

bAux = randn(1,N);

for l = 1 : size(SNR, 2)

    varianceNoise = sum (A.*A) / ( 10 ^ (SNR(l) / 10)); % variance p du
                                                        %l bruit blanc

    s = zeros (1, N);

    for k = 1:P
        s = s + A(k) * exp(2*pi*1i*n*T*F(k) + 1i*Theta(k));
    end

    %% Calcul du bruit blanc et du signal d'entree
    b = sqrt(varianceNoise) * bAux;

    x = s + b;  % le signal est donc la somme du bruit avec
                % les parties complexes


    %% Calcus des pics
    frequences = fDebut : fPas : fFin; % vecteur des frequences parcourues
    result = P_music (x, frequences, P);

```

```
    %% Affichage des resultats
    figure
    plot(n, abs(x));
    grid on
    title('Signal x[k]');
    xlabel ('k')
    legend (['SNR ' num2str(SNR(1),3) ' dB']);
    ylabel ('x[k]')

    figure
    subplot(2,1,1);
    plot(frequences, result);
    grid on
    title('Pics trouves par la fonction P_{music}');
    xlabel ('Frequence (Hz)')
    legend (['SNR ' num2str(SNR(1),3) ' dB']);
    ylabel ('P_{music} (f)')

    subplot(2,1,2)
    plot (0 : 1/N : 1 - 1/N ,abs(fft (x)));
    grid on
    title(['Transformee de Fourier de x[k] pour N = ' num2str(N,3)]);
    xlabel ('Frequence (v)')
    legend (['SNR ' num2str(SNR(1),3) ' dB']);
    ylabel ('| X(v) |')

end
```

2.3 Resultats

Les figures representées ci-dessous sont les résultats des programmes présentés dans la section 2.2 pour différentes valeurs du SNR (eq. 8) à un nombre de mesures constant $N = 100$. Afin de mieux mettre en évidence l'évolution des pics, nous avons adopté $P = 3$ dans nos tests, contrairement à $P = 2$ indiqué par le sujet. Les fréquences du signal d'entrée choisies, et qui alors doivent être identifiés par le programme, sont $f_1 = 49Hz$, $f_2 = 50Hz$ et $f_3 = 60Hz$. Les deux premières ont été choisis très proches, ce qui peut, dans les cas où la puissance du signal n'est pas assez forte comparée à celle du bruit, compliquer leur identification. Enfin, les amplitudes utilisées sont $A_1 = 1$, $A_2 = 2.5$ et $A_3 = 3$.

Dans chacune des figures, on trouve deux graphiques : le premier, au dessus, répresente le résultat de l'éxécution de la fonction $\mathrm{P}^{\mathrm{music}}$ pour fréquences comprises dans l'interval de $1Hz$ jusqu'à $100Hz$, et le deuxiéme montre la transformée de Fourier discrète du respectif signal x, $X(\nu) = \mathcal{F}\{x[k]\}$, réalisée à partir de la commande *fft*, implementée sous Matlab. L'objectif de cette approche consiste à comparer les méthodes conventionnelles d'analyse fréquentielle à celles dites de haute résolution, afin de déterminer les situations où la *fft* n'arrive pas à identifier des fréquences proches.

La figure 2 répresente la sortie du programme pour $SNR = 0.001dB$, situation où les puissances sont presque égales ($\frac{P_s}{P_b} = 10^{\frac{0.001}{10}} \approx 1$) et, par conséquent, dans ce cas là, le bruit interfère fortement dans le signal $x(t)$, comme l'illustre la figure 1. On vérifie déjà que la fréquence correpondante à $f_1 = 49Hz$ est indiscernable aux deux méthodes et que les pics obtenus par la fonction $\mathrm{P}^{\mathrm{music}}$ ne sont pas aussi accentués comme ceux des figures 3b et 4, dont les ordres de grandeur sont de 10^{10} et 10^{14} respectivement. On vérifie aussi que le bruit intense a provoqué l'apparition de certains sommets pour fréquences qui n'appartiennent pas au signal original dans les deux figures.

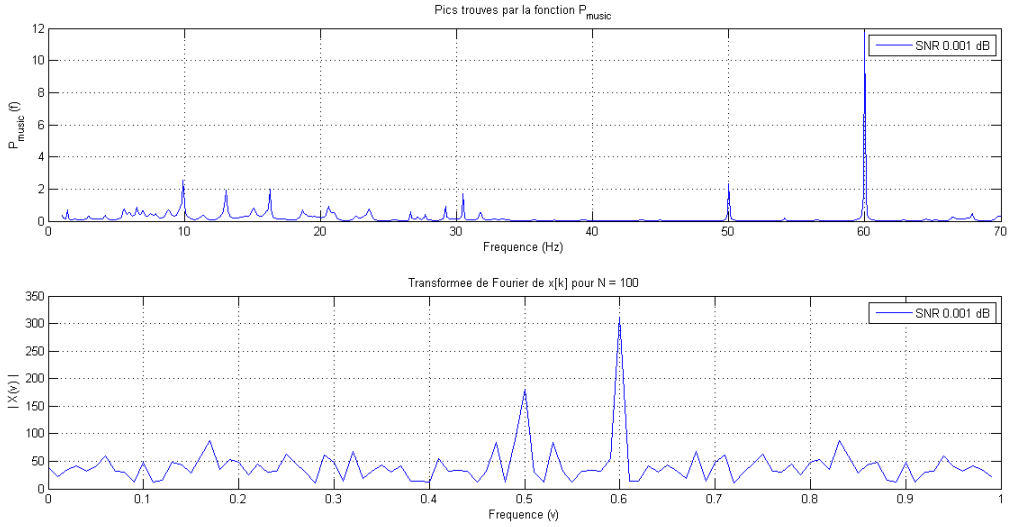


FIGURE 1 – Resultats obtenus pour $SNR_{dB} = 0.001dB$

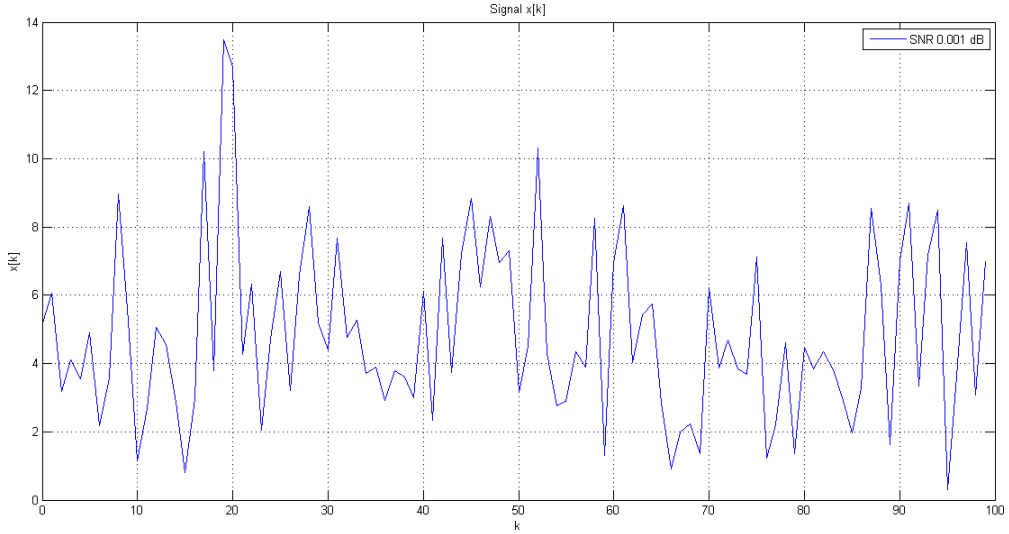
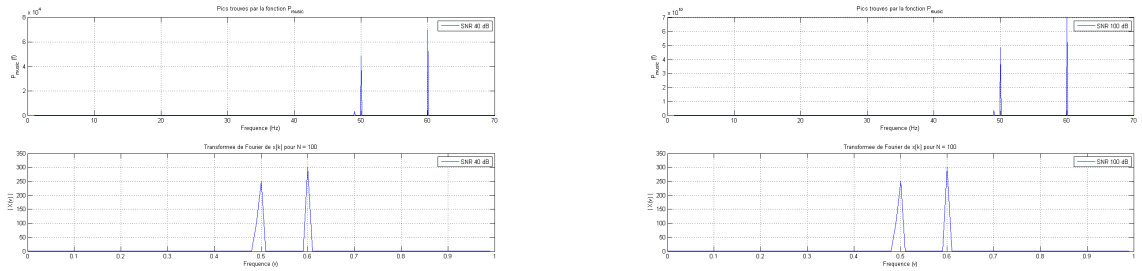


FIGURE 2 – Signal avec du bruit pour $SNR_{dB} = 0.001dB$

Les figures 3a et 3b montrent les résultats de l'éxécution pour $SNR = 40dB$ et $SNR = 100dB$, c'est-à-dire que la puissance du signal est assez grande comparée à celle du bruit. Plus spécifiquement, $P_s = 10^{\frac{40}{10}} P_b$ et $P_s = 10^{\frac{100}{10}} P_b$ respectivement. Dans ces deux cas, on vérifie que les pics correspondants aux fréquences ajoutées par le bruit disparaissent, restant seulement ceux du signal. Par contre, l'algorithme *MUSIC* est déjà capable d'identifier séparément $f_1 = 49Hz$ et $f_2 = 50Hz$ (le pic relatif à f_1 est faible comparé aux autres mais peut être bien distingué quand même). Néanmoins la transformée de Fourier n'a pas été capable quand à elle de distinguer ces deux fréquences.

On comprend donc l'intérêt ici des méthodes hautes résolution : les pics sont beaucoup plus accentués et localisés que ceux générés par la transformée.



(a) $SNR_{dB} = 40dB$ (b) $SNR_{dB} = 100dB$

FIGURE 3 – Resultats obtenus pour différentes valeurs du SNR

Finalement, un dernier exemple, répresenté par la figure 4, qui illustre le cas où le bruit interfère de façon très faible sur le signal (comparer avec le premier cas, en regardant les figures 5 et 1). Dans ce cas là, la puissance du signal est 10^{50} plus forte que celle du bruit et on obtient , comme espéré, les trois pics bien définis pour la fonction $\mathrm{P}^{\mathrm{music}}$. On note notamment que les deux fréquences très proches ont pu être bien distingué. Cependant, le résultat relatif à la *fft* n'a pas montré d'avancés significatives : les fréquences f_1 et f_2 n'ont pas pu être différencié quand même.

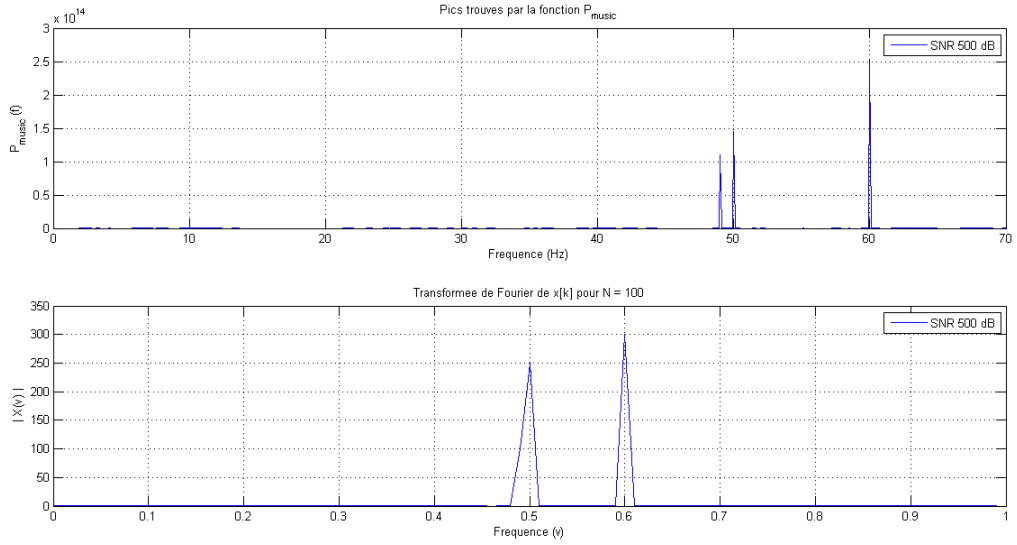


FIGURE 4 – Resultats obtenus pour $SNR_{dB} = 500dB$

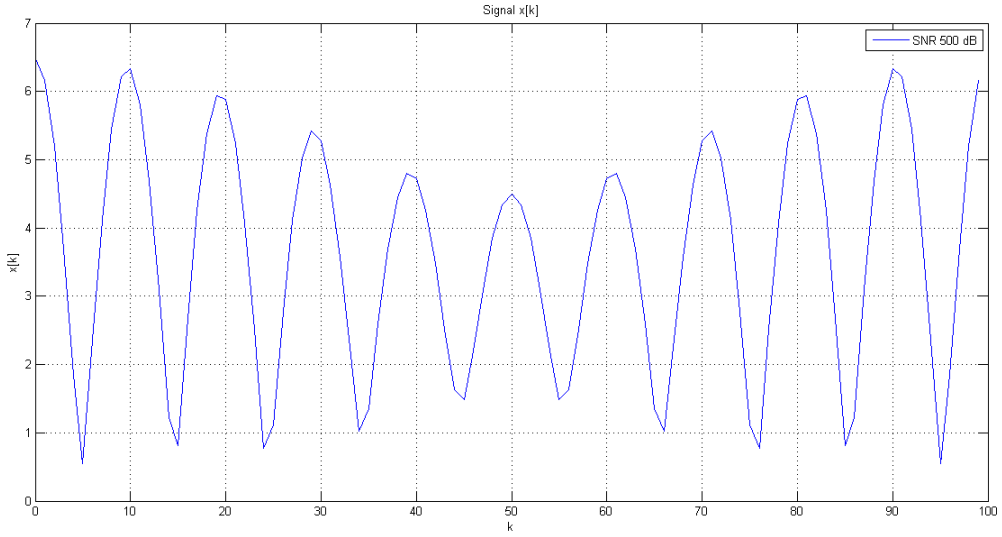


FIGURE 5 – Signal pour $SNR_{dB} = 500dB$

3 Algorithme ESPRIT

ESPRIT (Estimation of Signal Parameters via Rotational Invariance Techniques) est une méthode de localisation de sources mais qui ne s'applique que dans le cas particulier d'un réseau de capteurs constitués de deux sous-antennes identiques translatés l'une par rapport à l'autre. Elle a été implémenté afin de gagner en temps de calcul par rapport aux algorithmes plus conventionels comme MUSIC. Son succès repose également sur sa simplicité et ses bonnes performances.

On considère toujours le même le processus $x[n] = s[n] + w[n]$ où $w[n]$ est un bruit blanc $w[n]$ de variance p_w et s[n] est défini comme pour l'équation 11.

Supposons que les antennes soient disposées comme ci-dessous :

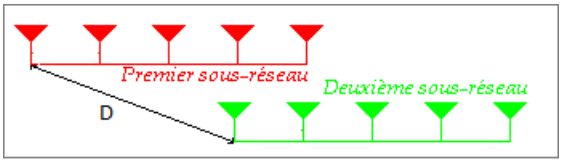


FIGURE 6 – Disposition des antennes.

On a un déphasage temporel de $\frac{Dsin(\phi)}{c}$ entre les 2 capteurs qui se correspondent où c est la célérité de la lumière et ϕ l'angle d'incidence du signal. Soit N le nombre de capteurs d'un réseau.

Ainsi on peut écrire que

$$S_2 = S_1\Omega \tag{26}$$

si

$$S_1 = \begin{bmatrix} \boldsymbol{e}_{N-1}(f_1) & \cdots & \boldsymbol{e}_{N-1}(f_P) \end{bmatrix}$$

avec $\boldsymbol{e}_{N-1}(f_k)$ définie comme en 16

et

$$\Omega = \begin{bmatrix} \exp(2\pi j f_1 \frac{Dsin(\phi_1)}{c}) & & 0 \\ & \ddots & \\ 0 & & \exp(2\pi j f_p \frac{Dsin(\phi_P)}{c}) \end{bmatrix}$$

La matrice d'autocorrélation s'écrit alors

$$R_{xx} = R_{ss} + R_w = \sum_{k=1}^{P} A_k^2 \begin{bmatrix} \boldsymbol{e}_{N-1}(f_k) \\ \boldsymbol{e}_{N-1}(f_k) \exp(2\pi j f_k \frac{Dsin(\phi_k)}{c}) \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_{N-1}(f_k) \\ \boldsymbol{e}_{N-1}(f_k) \exp(2\pi j f_k \frac{Dsin(\phi_k)}{c}) \end{bmatrix}^H + p_w \boldsymbol{I} \quad (27)$$

ou encore

$$R_{xx} = \begin{bmatrix} S_1 \\ S_1\Omega \end{bmatrix} \begin{bmatrix} A_1^2 & & 0 \\ & \ddots & \\ 0 & & A_P^2 \end{bmatrix} \begin{bmatrix} S_1 \\ S_1\Omega \end{bmatrix}^H + p_w \boldsymbol{I} \quad (28)$$

Celle-ci possède 2N valeurs propres et les P plus grandes correspondent à l'espace signal. Les vecteurs propres associés notés $u_1, ..., u_P$ engendrent l'espace signal tout comme les colonnes de la matrice des vecteurs sources $\begin{bmatrix} S_1 \\ S_1\Omega \end{bmatrix}$. Ainsi il existe une matrice T inversible telle que :

$$U = [u_1 \cdots u_P] = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} S_1 \\ S_1\Omega \end{bmatrix} T = \begin{bmatrix} S_1 T \\ S_1 \Omega T \end{bmatrix} \quad (29)$$

Ainsi $S_1 = U_1 T^{-1}$ et $U_2 = S_1 \Omega T$ ce qui donne :

$$U_2 = U_1 T^{-1} \Omega T = U_1 \Psi \quad (30)$$

Ψ possède les mêmes valeurs propres que Ω. Donc la connaissance de cette matrice permet d'obtenir les informations sur le signal. Pour obtenir une estimation de Ψ, on utilise communément le critère des moindres carrés qui donne :

$$\Psi = (U_1 U_1^H)^{-1} U_1^H U_2 \quad (31)$$

4 Références

— Institute for Dynamics Systems and Control. *White noise and power spectral density.* `http://www.idsc.ethz.ch/Courses/signals_and_systems/ArchiveFall10/lectureNotes8.pdf`
— S. Lawrence Marple Jr. *Digital Spectral Analysis with Applications.* Prentice-Hall Inc, 1988
— Bernard Picinbono. *Signaux aléatoires - Tome 2 Fonctions aléatoires et modèles avec problèmes résolus.* Dunod Université, 1993
— *Les méthodes à haute résolution.* HERMES, 1998
— Yves Grenier. *Méthodes haute-résolution en analyse spectrale et localisation.* `http://perso.telecom-paristech.fr/~rioul/liesse/2012liesse4/YG-slides.pdf`