Troisième partie .

Travaux pratiques avancés

6. Analyse de la parole

6.1. Introduction

L'information portée par le signal parole peut être considéré de bien des façons. On distingue généralement plusieurs niveaux de description non exclusifs : acoustique, phonétique et phonologique.

Au niveau *acoustique*, on s'intéresse essentiellement au signal que l'on tentera de caractériser par son intensité, sa fréquence, son timbre et ses propriétés statistiques. Au plan *phonétique*, on considère la génération des sons, les phonèmes qui composent un mot et les classes auxquels ils se rattachent. Enfin, la *phonologie* s'attache à décrire le rythme, la prosodie, la mélodie d'une phrase.

Le texte qui suit traitera du signal acoustique seulement. Il n'a d'autre but que de servir d'introduction à une deuxième partie où l'on abordera le codage et décodage LPC de la parole. Une présentation plus complète peut êtrer lue avec profit dans [1].

6.2. Analyse de la parole

Avant de vouloir traiter ou coder le signal de la parole, il est important de commencer par comprendre ce qu'est la parole, quel est son contenu spectral, quelles sont les parties qui la composent. De plus, il est primordial de réaliser que l'analyse des signaux est basée sur la stationnarité de ceux-ci alors que, par essence, un message ne peut pas être stationnaire. On sera donc constamment confronté au dilemme posé par l'analyse d'un signal transitoire considéré comme stationnaire.

Pour la suite, vous utiliserez les deux outils suivants :

- le programme CoolEdit 2000 pour l'enregistrement, l'analyse auditive et visuelle des sons et de leurs spectres ;
- le programme Matlab pour l'analyse et le traitement numériques des signaux.

6.2.1. Classification des phonèmes

Lorsqu'on recherche les composants élémentaires du langage articulé, on en trouve environ une trentaine pour la langue française. Ces éléments désignés sous le nom de *phonèmes* sont répartis en 7 classes ; ils suffisent pour représenter l'ensemble des sons. Il s'agit des :

- voyelles voisées : l**i**t, **é**té, mar**ai**s, Urs**u**le, p**eu**r, p**e**tit, j**eu**, p**a**tte, p**â**te, s**o**l, s**au**le, bij**ou** ;
- voyelles nasales : br**in**, br**un**, ch**an**t, b**on**jour ;
- semi-voyelles : pa**ill**e, l**ui**, L**oui**s ;
- consonnes fricatives : **s**au**c**is**s**on, **z**èbre, **ch**at, **j**anvier, **f**ameux, **v**ert ;
- consones nasales : N**an**tes, m**en**the, a**gn**eau ;
- consonnes liquides : sa**l**on, bu**r**eau ;
- consonnes plosives : **p**ari, **b**ar**b**are, ba**t**eau, ba**d**aud, é**c**art, lan**g**ue.

Ces classes de phonèmes font intervenir à des degrés divers les lèvres, la cavité nasale, la langue, le palais, la glotte et les cordes vocales. Des différences subtiles entre phonèmes déterminent le sens du mot et modifient sensiblement la forme de l'onde sonore et son spectre. Ces différences ne sont pas faciles à détecter et à mettre en oeuvre.

Dans certaines applications, en téléphonie par exemple, on peut se contenter d'une approche plus grossière et de répartir les phonèmes dans deux classes seulement, les sons voisés et non voisés. Les premiers sont modélisés par un signal périodique, alors que les seconds sont représentés par un bruit. Une tâche difficile du codage de la parole consiste à déterminer si un son est voisé ou non.

6.2.2. Période des sons voisés

Considérant que les sons voisés ont un contenu périodique bien marqué, le problème à résoudre consiste à trouver la période de la composante fondamentale et à décider si le son analysé est voisé ou non. Cette période (communément appelée le pitch), est un paramètre très important pour la synthèse de la parole car l'oreille est très sensible à ses variations.

On a observé que la fréquence de la fondamentale se situe entre 40 Hz et 250 Hz pour les voix masculines alors qu'elle est comprise entre 150 Hz et 700 Hz pour les voix féminines. De manière générale, on admettra donc qu'un son est voisé si sa période ou le pitch est compris entre 2 msec et 20 msec.

6.3. Acquisition et analyse avec CoolEdit

On utilisera le logiciel CoolEdit pour acquérir des sons, sélectionner et écouter des phonèmes ou des parties de phrases et visualiser des ondes sonores à l'aide de graphes, de spectres ou de spectrogrammes.

6.3.1. Paramètres pour l'enregistrement

Considérant que l'on s'intéresse ici à des sons de la bande téléphonique, on les enregistrera en monophonie à la fréquence de 8 kHz avec un convertisseur 16 bits après un filtrage antirepliement des fréquences supérieures à 4 kHz.

Pour que les fichiers soient directement utilisables par Matlab, on les sauvegardera dans un fichier *.txt de type ASCII et on prendra garde à supprimer les 4 premières lignes qui contiennent des informations sur l'enregistrement.

6.3.2. Visualisation des signaux et de leur spectre

Dans la figure 6.1, on présente le signal correspondant au mot "bonjour". On y voit le graphe du signal complet, son spectrogramme et deux zones du son "bonjour". La première correspond au son non voisé "j", alors que la deuxième illustre le phonème voisé "ou". Pour chacune de ces deux zones, on a tracé les signaux temporels et les spectres correspondants.

Pour le son "j", on relèvera le caractère aléatoire du signal, sa faible puissance et le fort taux de passages par zéro. Pour le son "ou", on notera d'abord son caractère périodique qui conduit aux raies spectrales du domaine fréquentiel. La périodicité basse fréquence des signaux voisés conduit à un faible taux de passages par zéro. De plus, la puissance des sons voisés est sensiblement plus grande que celle des sons non voisés.

6.4. Analyse du signal acoustique avec Matlab

Le logiciel Matlab servira pour traiter les signaux par tranches successives, extraire leurs caractéristiques et mettre en évidence les résultats obtenus.

Après avoir enregistré une phrase ou un son avec CoolEdit, celui-ci doit être sauvegardé dans un fichier *.txt de type ASCII afin de pouvoir être lu par Matlab. Il ne doit contenir aucune autre information que les valeurs échantillonnées du signal.

6.4.1. Lecture du fichier de données

Le fichier *.txt comportera une colonne de N valeurs échantillonnées. Il sera lu par Matlab sous la forme d'un vecteur dont l'amplitude sera normalisée à 1 :

```
[filename,path] = uigetfile('*.txt','Choix de la phrase');
phrase = load(filename);
phrase = phrase/max(abs(phrase));
```

On désignera une tranche à l'aide de la variable `st`. La tranche désirée peut être sélectionnée en prenant une partie des composantes du vecteur `phrase` avec la commande

```
st = phrase(Ndebut :Nfin);
```

Si une analyse spectrale doit être faite, on choisira de préférence une tranche de longueur égale à une puissance de 2. Par exemple, 128 ou 256.

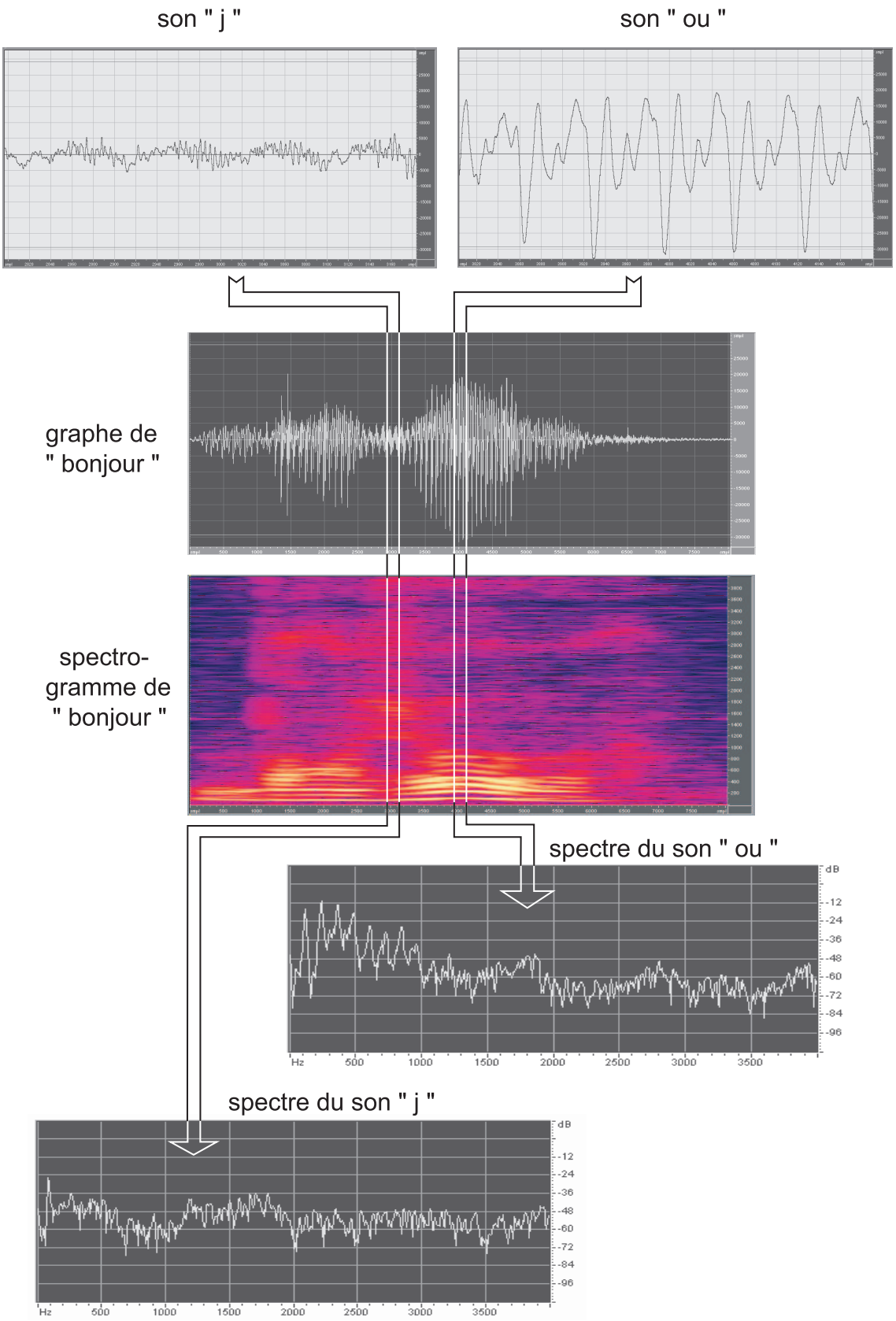

Fig. 6.1.: Graphes correspondant au mot “bonjour” ; mise en évidence des sons “j” et “ou”

6.4.2. Initialisation

La visualisation du signal temporel et de son spectre débute par l'initialisation de quelques variables et la suppression de la valeur moyenne qui n'a aucun intérêt en traitement des signaux :

```
fe = 8e3 ; Te = 1/fe ;
Npoints = length(st) ;
duree = Npoints*Te ;
temps = 0 :Te :duree-Te ;
df = 1/duree ;
Ndemi = fix(Npoints/2) ;
frequence = df*(0 :Ndemi-1) ;  % 0 <= frequence < fe/2
st = st - mean(st) ;           % suppression de la valeur moyenne
```

6.4.3. Valeur efficace

Vous remarquerez par la suite que les sons voisés sont généralement plus intenses que les sons non voisés. Pour évaluer l'amplitude des signaux, on calcule la valeur efficace de la tranche considérée. Celle-ci est égale à la déviation standard du signal

```
Seff = std(st) ;               % valeur efficace du signal
```

6.4.4. Taux de passages par zéro

Le taux de passages par zéro peut également aider à la décision voisé / non voisé. Il est défini comme le rapport entre le nombre de passages par zéro et le nombre d'échantillons considérés

$$n_{xz} = \frac{N_{xz}}{N_{ech}}$$

Le nombre passages par zéro peut se calculer comme suit :

```
function [Nxz] = zcross(xt)
  xz = xt - mean(xt) ;
  xz = (1+sign(xz))/2 ; % transformation en un signal binaire 0 / 1
  xz = diff(xz) ; % derivee du signal binaire = +/- 1
  Nxz = sum(abs(xz)) ; % nombre de passages par 0
```

6.4.5. Spectre

L'analyse spectrale est faite à l'aide de la FFT. Idéalement, le nombre de points de la tranche analysée devrait être une puissance de 2. Afin d'éviter les effets de bords de la tranche qui peuvent conduire à un étalement spectral, il est nécessaire d'effectuer préalablement un fenêtrage de la tranche. Ces opérations sont réalisées à l'aide des commandes suivantes :

```
stHm = st.*Hamming(Npoints) ;
spectre = fft(stHm) ;
spectre = spectre(1 :Ndemi) % limitation à fe/2
module = abs(spectre) ; phase = angle(spectre) ;
plot(frequence,20*log10(abs(spectre)) ;
```

Une illustration de sons voisés et non voisés est donnée dans les figures 7.6 et 7.7. On notera les raies spectrales bien visibles dans le spectre du signal voisé et, en particulier, la correspondance entre la fréquence de la fondamentale et la période du signal voisé.

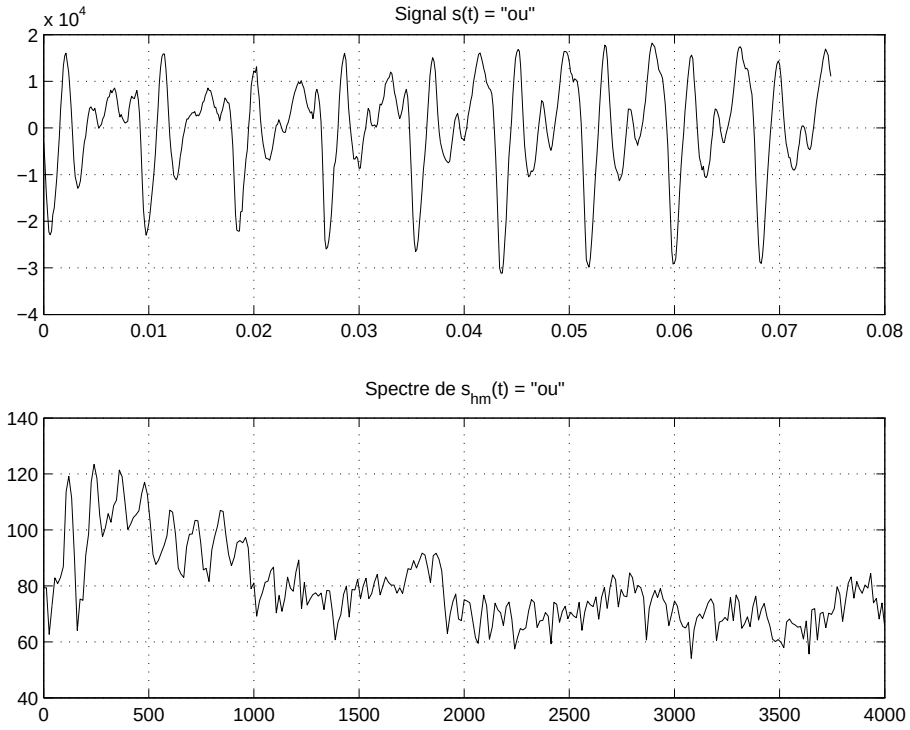

Fig. 6.2.: Signal voisé et son spectre

6.5. Recherche du pitch

6.5.1. Filtrage du signal

Comme on l'a dit plus haut, on admet généralement que la période du pitch de la voix humaine est comprise entre 2 et 20 msec. Le domaine spectral qui nous préoccupe ici est donc inférieur à 500 Hz. Il est ainsi préférable, puisqu'on s'intéresse à un signal dont le spectre est limité, de commencer par éliminer les fréquences supérieures à 500 Hz. Ceci peut être fait à l'aide d'un filtre numérique passe-bas ; celui-ci est généralement du type Butterworth et d'ordre 8 :

```
fc = 500 ; fn = fe/2 ; ordre = 8 ;
[nbtw dbtw] = butter(ordre, fc/fn) ;
stf = filter(nbtw, dbtw, st) ;
```

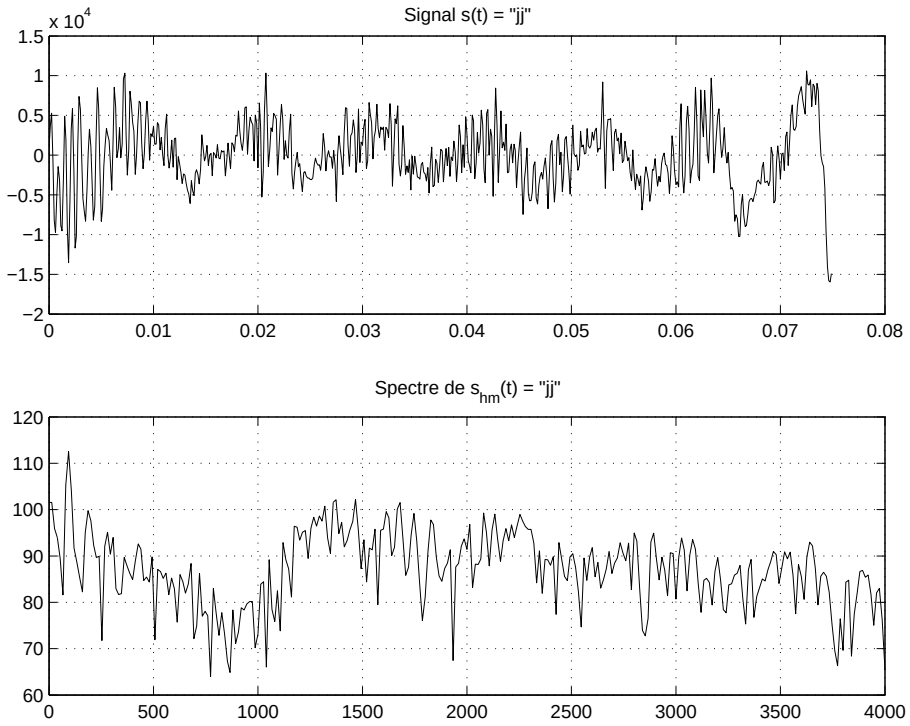

FIG. 6.3.: Signal non voisé et son spectre

6.5.2. Autocorrélation

On a vu que la tranche considérée est périodique si le son est voisé et aléatoire dans le cas contraire. Afin de faciliter la recherche de la période, on travaille de préférence avec la fonction d'autocorrélation car celle-ci est généralement moins bruitée que le signal lui-même (figure 6.4).

Le résultat de l'autocorrélation est un vecteur de longueur $2N$ avec un maximum en son milieu. Si le signal est périodique, d'autres pics distants de la valeur du pitch seront présents. Pour trouver ce dernier, il suffit donc de mesurer la distance entre pics successifs.

Les commandes sont alors les suivantes :

```
% autocorrélation d'une tranche st filtrée
  rss = real(xcorr(stf))/Npoints;
  [rssmax k0] = max(rss);          % k0 = position du max central
  rss = rss(k0 :length(rss));      % partie droite de rss

% le 1er pic latéral doit se trouver entre Tpmin et Tpmax
  fpmax = 500;                     % fréquences min et max du pitch
  fpmin = 50;
  Tpmax = 1/fpmin;                 % périodes min et max du pitch
  Tpmin = 1/fpmax;
  kpmin = round(Tpmin/Te);         % compteurs liés à Tpmin et Tpmax
  kpmax = round(Tpmax/Te);
```

```
% recherche du premier pic
  rss = rss(kpmin :kpmax) ;          % domaine limité par Tpmin et Tpmax
  [ymax k1] = max(rss) ;             % k1 = position du 1er max latéral

% entier correspondant à la période du pitch
  kp = kpmin + k1 ;
  Tp = kp * Te ;
```

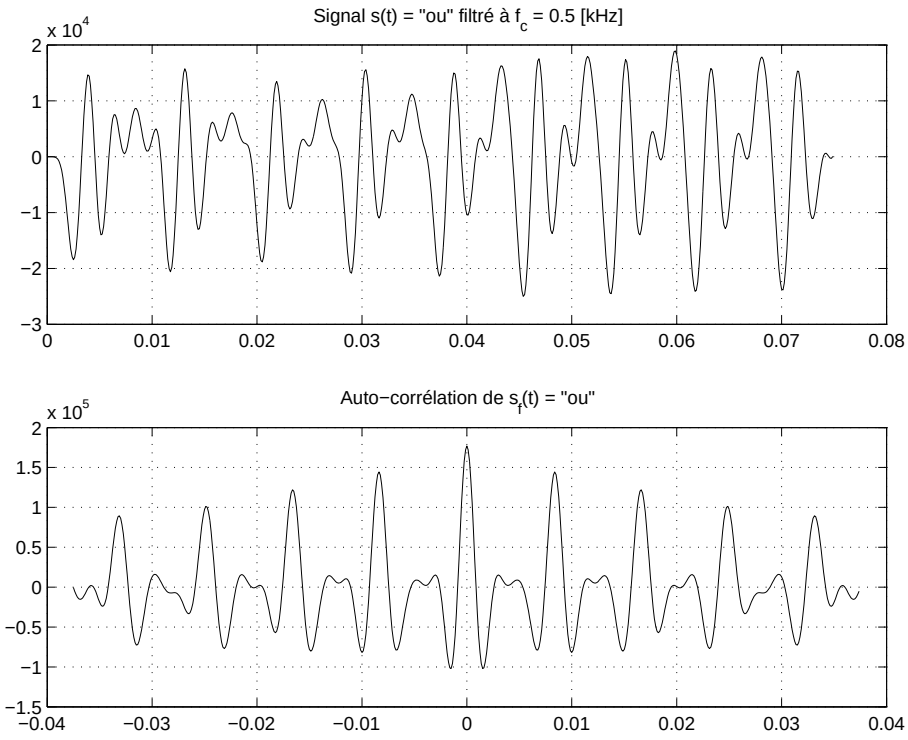

Fig. 6.4.: Autocorrélation d'un son voisé

6.6. Travail pratique

Pour aborder l'analyse de la parole, je vous propose de travailler sur la phrase "Le colibri a chanté" (fichier colibri.txt) ou une phrase de votre choix.

6.6.1. Avec CoolEdit :

1. Chargez le fichier `colibri.txt` en mode `mono / 16 bits / 8 kHz`.
2. Écoutez la phrase ; observez le graphe temporel et le spectre (`Analyze/Frequency Analysis`) de diverses parties de la phrase ; notez que le temps peut être gradué en secondes ou en échantillons avec le bouton droit de la souris placé sur l'axe temporel.

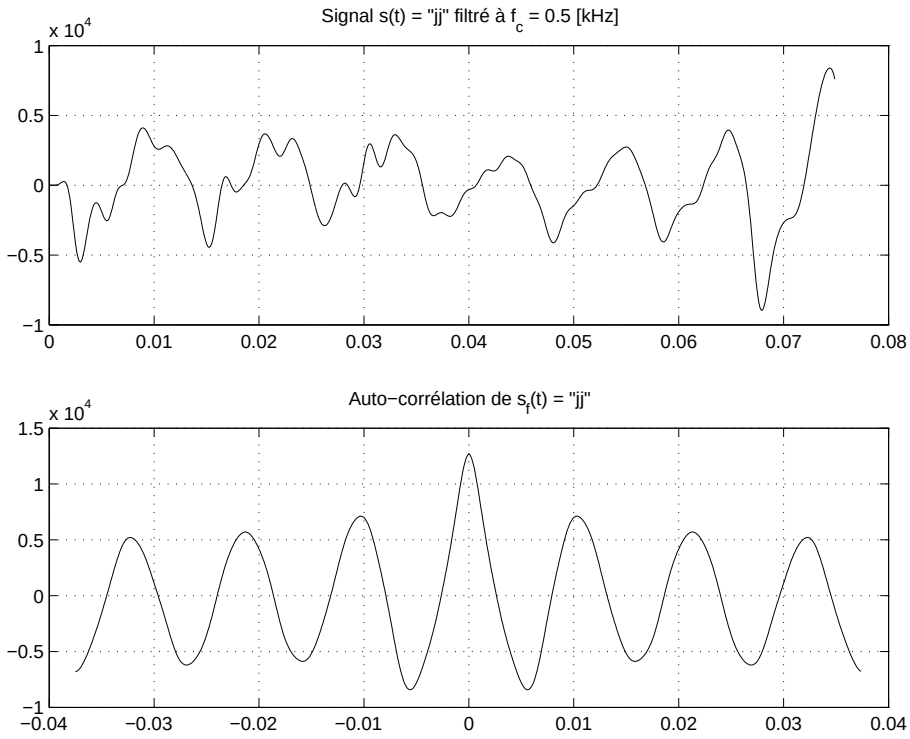

FIG. 6.5.: Autocorrélation d'un son non voisé

3. Sur le spectrogramme (`View/Spectral View`), observez l'effet de la durée d'analyse N sur les résolutions temporelle et fréquentielle en prenant 128, 256 et 512 échantillons (`Options/ Settings/ Spectral`).
4. Qu'en est-il de la relation liant les résolutions temporelle et spectrale ? quelle durée d'analyse choisissez-vous ?
5. Mentionnez tous les phonèmes que l'on trouve dans cette phrase ; à quelles familles appartiennent-ils ?
6. Relevez les numéros d'échantillons correspondant au début et à la fin des phonèmes.
7. Dans l'enregistrement, choisissez librement au moins deux tranches voisées et non voisées.
8. Observez leurs spectres et spectrogramme puis analysez plus en détail leurs caractéristiques temporelles et spectrales.

6.6.2. Avec Matlab

Procédez à l'analyse détaillée de plusieurs tranches voisées / non voisées et tentez de les séparer automatiquement. Pour cela :

1. Extrayez de la phrase les zones que vous souhaitez analyser.
2. Mesurez leur valeur efficace et taux de passages par zéro ($< 100\%$!).
3. Recherchez la période du signal avec la fonction de corrélation.

4. Voyez-vous un moyen de séparer automatiquement les sons voisés / non voisés ?

Bibliographie

[1] R. Boite et al., *Traitement de la parole*, PPUR, 2000

7. Codage et décodage LPC de la parole

7.1. Introduction

Le codage linéaire prédictif LPC (Linear Predictive Coding) de la parole est utilisé, en particulier, en téléphonie où il permet de transmettre les communications avec un débit d'environ 12 kbits/sec au lieu de 64 kbits/sec si on se contentait de numériser la parole.

Cette forte diminution du débit est basée sur une modélisation très simplifiée du conduit vocal dont on transmet les paramètres toutes les 10 ou 20 ms.

7.2. Prédiction linéaire

7.2.1. Mesure de l'erreur de prédiction

Le codage LPC consiste à estimer la valeur de l'échantillon à venir sur la base de quelques valeurs mesurées précédemment $s[n-k]$ (figure 7.1).

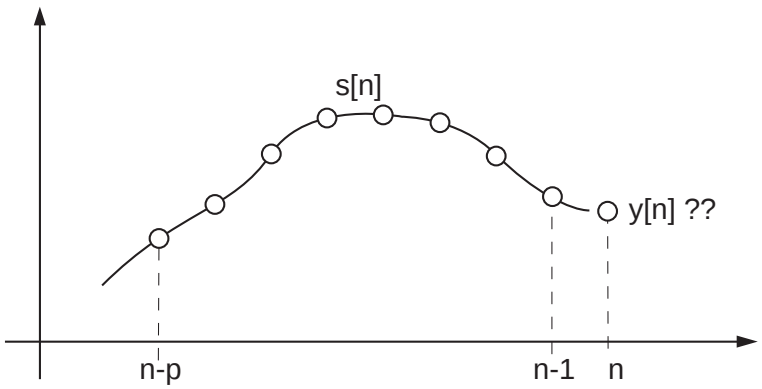

FIG. 7.1.: Les échantillons $s[n-p]$ à $s[n-1]$ sont utilisés pour estimer la valeur à venir

La valeur estimée $y[n]$ est calculée à partir des échantillons précédents pondérés par des coefficients a_k qui sont généralement au nombre de 8 à 12 :

$$y[n] = -\left(a_1 s[n-1] + a_2 s[n-2] + \cdots + a_p s[n-p]\right) = -\sum_{k=1}^{p} a_k s[n-k] \qquad (7.1)$$

La valeur des coefficients de prédiction a_k s'obtient par minimisation de la variance σ_e^2 de l'écart $e[n]$. Celui-ci est défini comme la différence entre la valeur réelle $s[n]$ et la valeur estimée $y[n]$:

$$e[n] = s[n] - y[n] = s[n] + \sum_{k=1}^{p} a_k s[n-k] \tag{7.2}$$

La puissance ou variance de l'écart de l'ensemble des N échantillons $e[n]$ à disposition dépend du choix des coefficients de prédiction a_k et elle vaut :

$$\sigma_e^2(a_k) = \frac{1}{N}\sum_{n=0}^{N-1} e^2[n] = \frac{1}{N}\sum_{n=0}^{N-1}\left(s[n] + \sum_{k=1}^{p} a_k s[n-k]\right)^2 \tag{7.3}$$

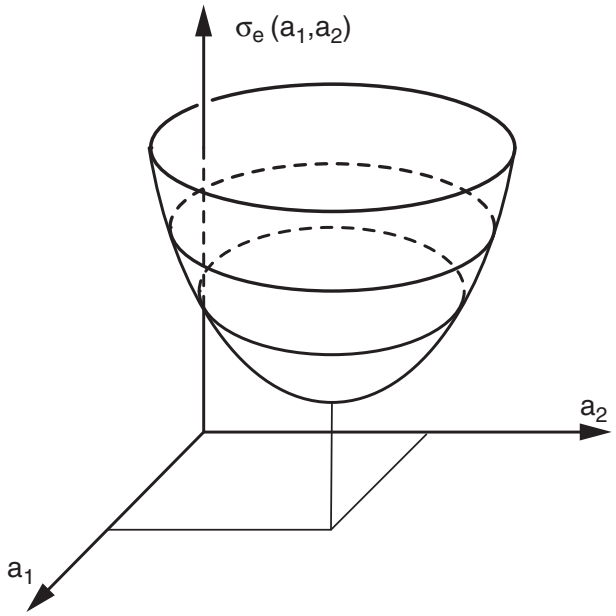

Fig. 7.2.: Variance de l'erreur de prédiction

7.2.2. Calcul des coefficients de prédiction linéaire

La procédure pour obtenir la valeur optimum des coefficients a_k consiste à rendre minimum la puissance de l'erreur commise lors de la prédiction. Un schéma fonctionnel traduisant cette démarche est présentée dans la figure 7.3.

Mathématiquement, la variance est une fonction des paramètres de prédiction a_k :

$$\sigma_e^2 = \sigma_e^2(a_1, a_2, \ldots, a_p) = \sigma_e^2(a_k) \tag{7.4}$$

Sa valeur minimum s'obtient donc lorsque l'ensemble des dérivées partielles de σ_e^2 par rapport aux paramètres a_k sont nulles :

$$\sigma_{e,min}^2 \;\Rightarrow\; \frac{\delta\sigma_e^2(a_k)}{\delta a_k} = 0,\; k = 1, \ldots, p \tag{7.5}$$

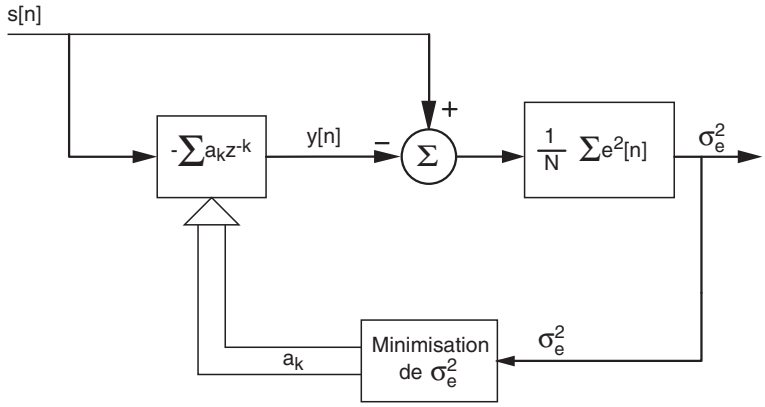

FIG. 7.3.: Schéma fonctionnel de la prédiction linéaire

Le calcul de ces p dérivées partielles conduisent à p équations pour les p paramètres inconnus a_k (voir annexe 7.9) :

$$\begin{aligned} a_1 r_{ss}[0] + a_2 r_{ss}[-1] + \cdots + a_p r_{ss}[1-p] &= -r_{ss}[1] \\ a_1 r_{ss}[1] + a_2 r_{ss}[0] + \cdots + a_p r_{ss}[2-p] &= -r_{ss}[2] \\ \vdots &= \vdots \\ a_1 r_{ss}[p-1] + a_2 r_{ss}[p-2] + \cdots + a_p r_{ss}[0] &= -r_{ss}[p] \end{aligned}$$

avec :

$$r_{ss}[m] = \sum_{n=0}^{N-1} s[n]s[n-m],\ m = 1, \ldots, p \tag{7.6}$$

Les coefficients des paramètres a_k sont les p premières valeurs de la fonction d'autocorrélation $r_{ss}[m]$ du signal $s[n]$ comportant N échantillons.

Cet ensemble d'équations linéaires peut s'écrire sous forme matricielle :

$$\mathbf{R}_{ss}\,\mathbf{a} = -\mathbf{r}_{ss} \tag{7.7}$$

où $\mathbf{R}_{ss}$ est la matrice $p \times p$ d'autocorrélation, $\mathbf{r}_{ss}$ le vecteur $p \times 1$ d'autocorrélation et $\mathbf{a}$ le vecteur $p \times 1$ des paramètres de prédiction. On notera que, la fonction d'autocorrélation étant paire, la matrice $\mathbf{R}_{ss}$ est symétrique. On voit donc que les coefficients de prédiction linéaire peuvent s'obtenir par inversion de la matrice $\mathbf{R}_{ss}$:

$$\mathbf{a} = -\mathbf{R}_{ss}^{-1}\,\mathbf{r}_{ss} \tag{7.8}$$

L'estimation de la valeur à venir sera d'autant meilleure que le nombre de points N utilisés pour calculer la fonction d'autocorrélation $r_{ss}[m]$ sera élevé. L'évaluation des paramètres a_k se fait donc après analyse d'une tranche t_k suffisamment longue ($N \sim 200$) alors que le calcul de la valeur à venir $y[n]$ n'utilise que les p dernières valeurs de $x[n]$.

Il est important de relever que si les points $s[n]$ ne sont pas corrélés entre eux, aucune prévision n'est possible (cas du bruit blanc). Pour plus d'informations, on consultera avantageusement la référence [1].

7.2.3. Interprétation de la prédiction linéaire

Dans ce qui précède, le signal $e[n]$ a été considéré comme une erreur de prédiction dont on a minimisé la variance pour calculer les coefficients de prédiction linéaire a_k. Sa définition était la suivante :

$$e[n] = s[n] - y[n] = s[n] + \sum_{k=1}^{p} a_k s[n-k] \tag{7.9}$$

Appliquant la transformation en z à cette équation, on en tire la relation suivante :

$$E(z) = S(z)\left(1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_p z^{-p}\right) = S(z)A(z) \tag{7.10}$$

Elle montre que l'on peut reconstruire le résidu $e[n]$ de l'estimation à partir du signal $s[n]$ à l'aide d'un filtre non récursif représenté par la fonction de transfert $A(z)$.

Une deuxième représentation, plus fructueuse,

$$S(z) = E(z)\frac{1}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_p z^{-p}} = E(z)\frac{1}{A(z)} \tag{7.11}$$

met en évidence le fait que le résidu $e[n]$ peut être considéré comme un signal d'excitation servant à créer le signal $s[n]$ avec l'aide d'un filtre récursif tous pôles $H(z) = 1/A(z)$. Dans le cas du codage LPC de la parole, le signal d'excitation $e[n]$ est choisi périodique pour les sons voisés et aléatoire pour les sons non voisés.

7.3. Modèle du conduit vocal

La production de sons met en oeuvre un certain nombre de muscles modifiant la forme du conduit vocal dans lequel circule un flux d'air. On y trouve les cordes vocales qui vibrent pour les sons voisés et restent au repos pour les sons non voisés. Viennent en suite le pharynx et la cavité buccale en parallèle avec la cavité nasale. La forme de ces parties est constamment modifiée pour créer le message sonore.

Le modèle généralement adopté pour créer artificiellement des sons est grossier par rapport à la complexité du système phonatoire mais il est tout à fait satisfaisant pour les besoins de la téléphonie. Ce modèle comprend (figure 7.4) :

- un générateur périodique d'impulsions unité ;
- un générateur de nombres aléatoires à valeur moyenne nulle et variance unité ;
- un commutateur servant à choisir les sons voisés ou non ;
- un gain proportionnel à la valeur efficace du signal $s[n]$;
- un filtre tous pôles $H(z) = 1/A(z)$.

Comme les sons évoluent constamment, le générateur et le filtre doivent être modifiés en permanence. L'extraction de ces paramètres du générateur et du filtre constituent le codage de la parole. A l'émission, on décompose le son en tranches pour en extraire les paramètres qui seuls seront transmis. A la réception, chaque tranche du signal sonore est reconstruite à partir des paramètres du générateur et du filtre.

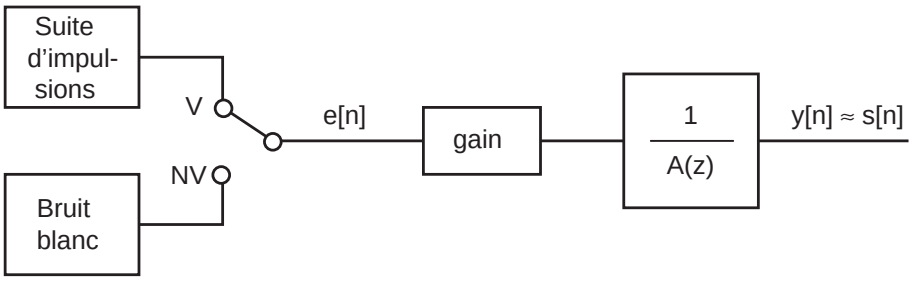

FIG. 7.4.: Modèle du conduit vocal

L'analyse de la parole se fait par tranches de 20 à 30 msec. Dans le cas où la fréquence d'échantillonnage est de 8 kHz, chaque tranche comporte donc 160 à 240 échantillons. C'est à partir de ces échantillons que l'on décide si le son est voisé ou non et que l'on calcule le gain et les paramètres du filtre.

Pour assurer une certaine continuité du signal sonore, les tranches successives peuvent être décalées d'une valeur inférieure à leur durée (figure 7.5). Généralement, ce décalage est de 10 msec (80 échantillons).

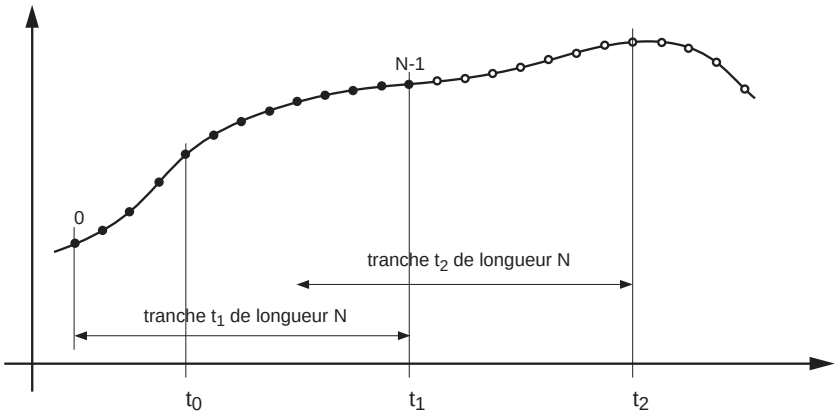

FIG. 7.5.: Découpage en tranches t_k du signal $s[n]$

Dans le cas où l'on doit effectuer une analyse spectrale à l'aide de la FFT, il est préférable de travailler avec des tranches dont le nombre de points est une puissance de 2, généralement 128 ou 256. La durée des tranches est alors de 16 ou 32 msec et le décalage de 8 msec (64 échantillons).

Considérant que les sons voisés ont un contenu périodique bien marqué, le problème à résoudre consiste à trouver la période de la composante fondamentale et à décider si le son analysé est voisé ou non. Cette période (communément appelée le pitch), est un paramètre important pour la synthèse de la parole car l'oreille est sensible à ses variations.

On a observé que la fréquence de la fondamentale se situe entre 40 Hz et 250 Hz pour les voix masculines alors qu'elle est comprise entre 150 Hz et 700 Hz pour les voix féminines. De manière générale, on admettra donc qu'un son est voisé si sa période ou le pitch est compris entre 2 msec et 20 msec.

7.4. Analyse du signal

Après avoir enregistré une phrase ou un son, celui-ci doit être sauvegardé dans un fichier *.txt de type ASCII afin de pouvoir être lu par MatLab. Il ne doit contenir aucune autre information que les valeurs échantillonnées du signal.

Le fichier *.txt comportera une colonne de N valeurs échantillonnées. Il sera lu par MatLab sous la forme d'un vecteur dont l'amplitude sera normalisée à 1 :

```
[filename,path] = uigetfile('*.txt','Choix de la phrase');
phrase = load(filename);
phrase = phrase/max(abs(phrase));
```

Par la suite, on désignera une tranche à l'aide de la variable `st`. La tranche désirée peut être sélectionnée en prenant une partie des composantes du vecteur `phrase` avec la commande

```
st = phrase(Ndebut :Nfin);
st = st - mean(st);           % suppression de la valeur moyenne
```

Si une analyse spectrale doit être faite, on choisira de préférence une tranche de longueur égale à une puissance de 2. Par exemple, 128 ou 256.

7.4.1. Initialisation

La visualisation du signal temporel et de son spectre débute par l'initialisation de quelques variables, la suppression de la valeur moyenne et le calcul de la valeur efficace :

```
fe = 8e3; Te = 1/fe;
Npoints = length(st);
temps = Te*(0 :Npoints-1);
duree = Npoints*Te;
df = 1/duree;
Ndemi = fix(Npoints/2);
frequence = df*(0 :Ndemi-1);  % 0 <= frequence < fe/2
```

7.4.2. Spectre

L'analyse spectrale se fait à l'aide de la FFT. Idéalement, le nombre de points de la tranche analysée devrait être une puissance de 2. Si cela n'est pas possible, il faudra être critique par rapport aux résultats obtenus.

Afin d'éviter les effets de bords de la tranche qui peuvent conduire à un étalement spectral, il est nécessaire d'effectuer préalablement un fenêtrage de la tranche. Ces opérations sont réalisées à l'aide des commandes suivantes :

```
stHm = st.*Hamming(Npoints) ;
spectre = fft(stHm) ;
spectre = spectre(1 :Ndemi) % limitation à fe/2
module = abs(spectre) ; phase = angle(spectre) ;
plot(frequence,20*log10(abs(spectre)) ;
```

Une illustration de sons voisé et non voisé est donnée dans les figures 7.6 et 7.7.

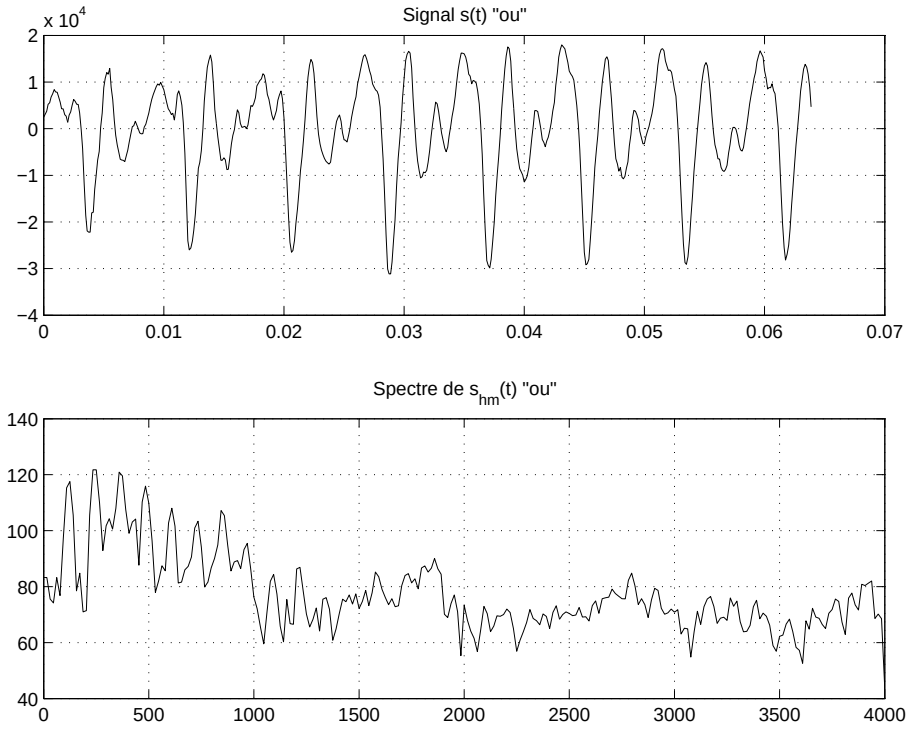

FIG. 7.6.: Signal voisé et son spectre

7.5. Analyse LPC

Celle-ci fournit les coefficients de prédiction linéaire et la variance (la puissance) de l'erreur de prédiction :

```
NbCoeff = 12 ;
[coeff Perreur] = lpc(st, NbCoeff) ;
```

7.5.1. Valeur efficace et gain

Pour évaluer l'amplitude des signaux synthétisés, on calcule la valeur efficace de la tranche considérée

```
Seff = std(st) ; % valeur efficace du signal
```

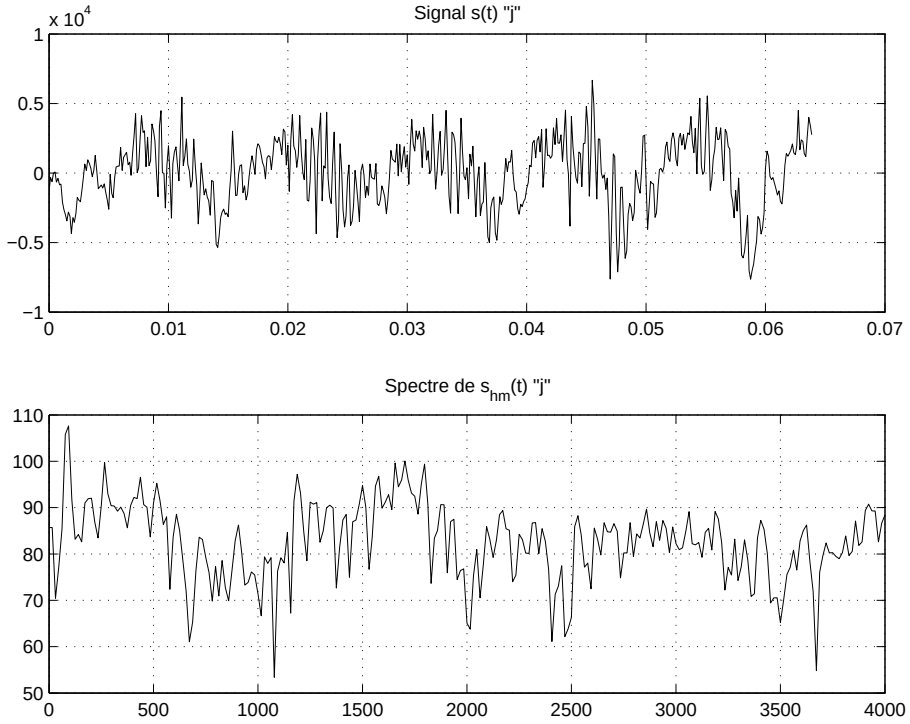

Fig. 7.7.: Signal non voisé et son spectre

Cette valeur efficace peut représenter le gain de la fonction de transfert du conduit vocal :

```
gain = Seff ;
```

7.5.2. Fonction de transfert $H(z)$ du conduit vocal

Les coefficients LPC représentent un polynôme en z^{-1} qui n'est autre que le dénominateur de la fonction de transfert du conduit vocal :

$$H(z) = \frac{gain}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_p z^{-p}} \tag{7.12}$$

Comme le conduit vocal est par essence stable, les pôles doivent se trouver à l'intérieur du cercle de rayon unité.

La donnée de la fonction de transfert sous la forme d'un produit de fonctions biquadratiques et le tracé des zéros et des pôles dans le plan complexe (figure 7.8) s'obtiennent avec les commandes suivantes :

```
Hz = tf(gain, coeff) ;
zpk(Hz) ;
zplane(roots(gain), roots(coeff)) ;
```

7.5.3. Réponse fréquentielle du conduit vocal

La réponse fréquentielle du conduit vocal s'obtient avec :

```
[Hf ff] = freqz(gain, coeff, Ndemi,fe) ;
```

Au tracé de cette réponse fréquentielle, on peut superposer le spectre du signal (figure 7.9) :

```
plot(ff,20*log10(abs(Hf)),frequence,20*log10(abs(spectre))) ;
```

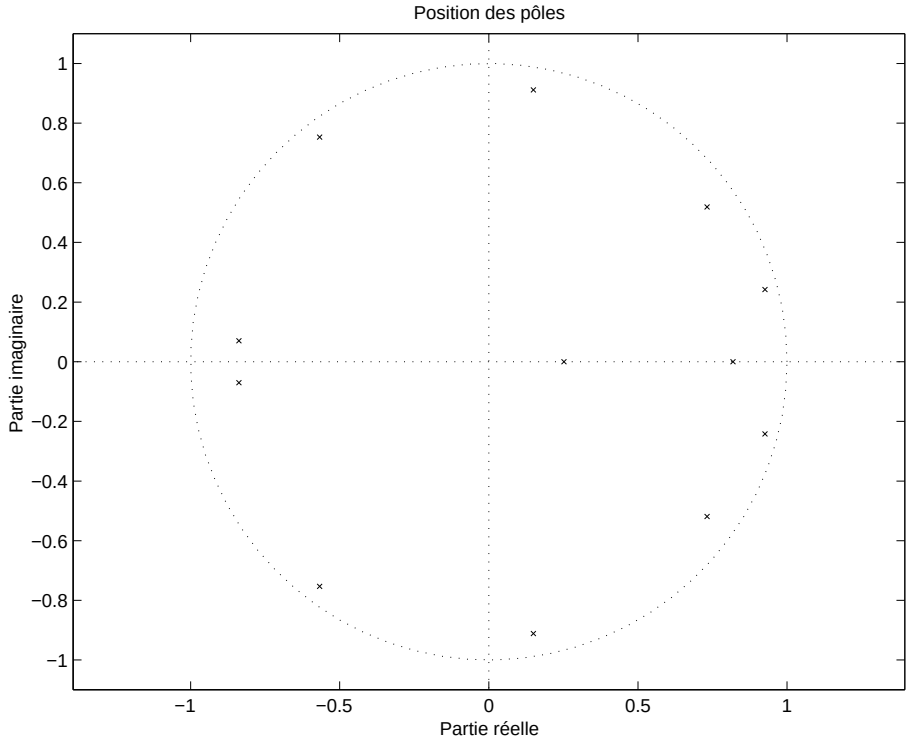

FIG. 7.8.: Pôles de la fonction de transfert représentant le conduit vocal

7.6. Recherche du pitch

7.6.1. Filtrage du signal

Comme on l'a dit plus haut, la période du pitch est comprise entre 2 et 20 msec. Le domaine spectral qui nous préoccupe ici est donc inférieur à 500 Hz. Il est ainsi préférable, avant de poursuivre l'analyse, de commencer par éliminer les fréquences supérieures à 500 Hz à l'aide d'un filtre passe-bas de Butterworth, généralement d'ordre 8.

```
fc = 500 ; fn = fe/2 ; ordre = 8 ;
[nbtw dbtw] = butter(ordre, fc/fn) ;
stf = filter(nbtw, dbtw, st) ;
```

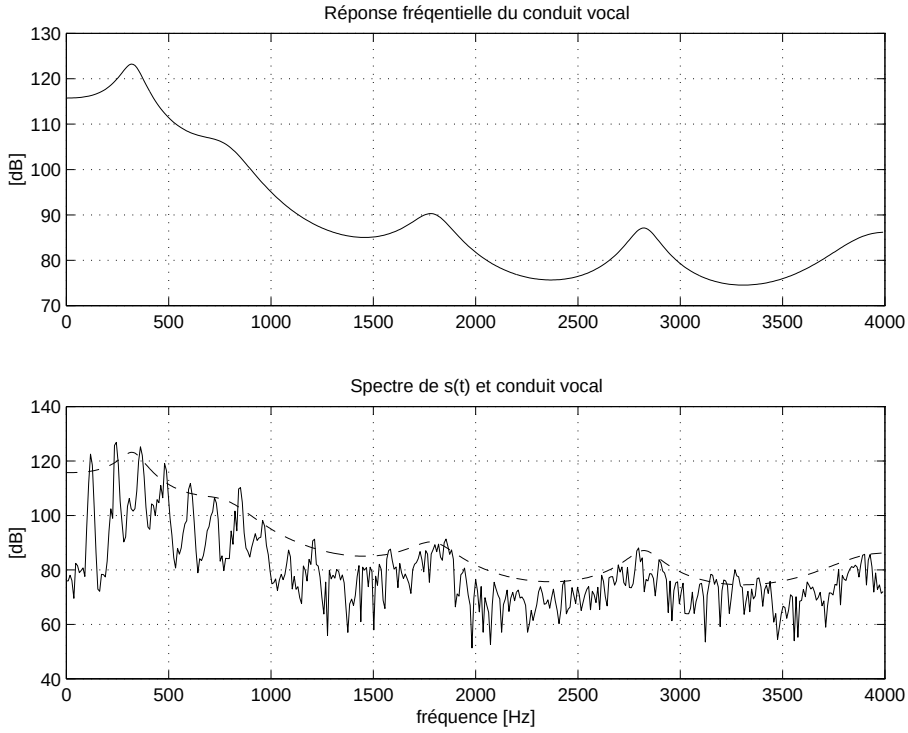

Fig. 7.9.: Réponse fréquentielle du conduit vocal et spectre du signal

7.6.2. Recherche du signal d'excitation $e[n]$

Nous avons vu dans la section 7.2.3 que le résidu $e[n]$ de la prédiction linéaire peut être considéré comme le signal d'excitation servant à créer le signal $s[n]$ en passant à travers le filtre récursif :

$$H(z) \equiv \frac{S(z)}{E(z)} = \frac{1}{A(z)} \tag{7.13}$$

Puisque, dans notre cas, le signal $s[n]$ est connu, on peut par filtrage inverse obtenir le résidu $e[n]$:

$$E(z) = A(z)S(z) \tag{7.14}$$

ce qui revient à convoluer les coefficients $a_k \equiv a[n]$ avec le signal $s[n]$:

$$e[n] = a[n] \otimes s[n] \tag{7.15}$$

Dans MatLab, cela s'écrit simplement

```
en = conv(coeff, stf);
```

7.6.3. Autocorrélation de $e[n]$

On a vu que le signal d'excitation est périodique si le son est voisé et aléatoire dans le cas contraire. Comme le signal est passablement bruité, la recherche de la période est grandement facilitée si on l'effectue sur la fonction d'autocorrélation de $e[n]$ plutôt que sur le signal lui-même (figure 7.10).

Le résultat de l'autocorrélation est un vecteur de longueur $2N$ avec un maximum en son milieu. Si le signal est périodique, d'autres pics distants de la valeur du pitch seront présents. Pour trouver ce dernier, il suffit donc de mesurer cette distance.

Les commandes sont les suivantes :

```
% autocorrélation des résidus
ree = real(xcorr(en))/Npoints;
[reemax k0] = max(ree);       % maximum central
ree = ree(k0 :length(ree));    % partie droite de ree

% le 1er pic latéral doit se trouver entre Tpmin et Tpmax
fpmax = 500; fpmin = 50;          % fréquences min/max du pitch
Tpmax = 1/fpmin; Tpmin = 1/fpmax; % périodes min et max du pitch
kpmin = round(Tpmin/Te); kpmax = round(Tpmax/Te);

% recherche du premier pic
ree = ree(kpmin :kpmax);  % domaine temporel limité par Tpmin et Tpmax
[reemax1 k1] = max(ree); % k1 = position du 1er max latéral

% entier correspondant à la période du pitch Tp = Kp * Te
Kp = kpmin + k1;
```

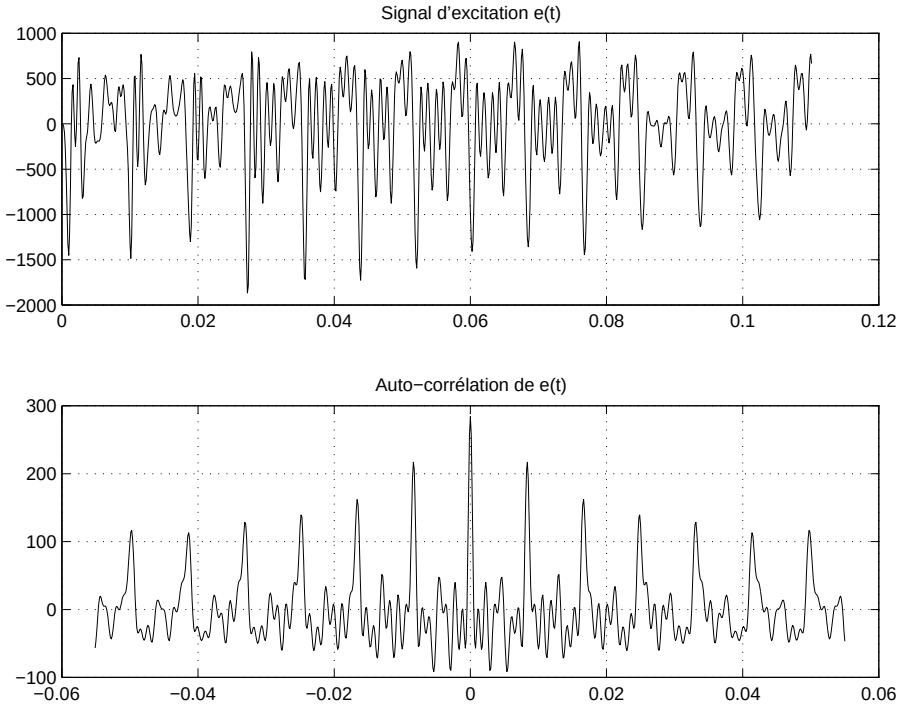

FIG. 7.10.: Résidus de la prédicion linéaire et autocorrélation (son voisé)

7.6.4. Critères de décision

Les éléments pouvant contribuer à la décision voisé/non voisé sont la valeur efficace de la tranche, son taux de passages par zéro et l'amplitude du premier maximum latéral de la fonction d'autocorrélation des résidus.

Pour les ***sons voisés***, on a en effet observé qu'en moyenne

1. l'amplitude du signal est élevée ;
2. le taux de passages par zéros est faible ;
3. le premier pic latéral de la fonction d'autocorrélation des résidus est bien marqué.

Taux de passages par zéro Le taux de passages par zéro est défini comme le rapport entre le nombre de passages par zéro et le nombre d'échantillons considérés

$$n_{zx} = \frac{N_{zx}}{N_{ech}}$$

Le nombre passages par zéro peut se calculer comme suit :

```
function [Nzx] = zcross(xt)
  xz = xt - mean(xt) ;
  xz = (1+sign(xz))/2 ;   % transformation en un signal binaire 0 / 1
  xz = diff(xz) ;         % derivee du signal binaire = +/- 1
  Nzx = sum(abs(xz)) ;    % nombre de passages par 0
```

Fonction de corrélation Pour les sons voisés, l'amplitude du premier pic latéral est souvent supérieure au tiers de celle du pic central. Il est moins marqué pour les sons non voisés.

Choix des seuils Sur la base d'analyses statistiques, on fixe les seuils approximatifs suivants

```
SeuilXeff = 0.05 ;  SeuilZcross = 0.3 ;  SeuilCorrel = 0.3 ;
```

Période du pitch Tenant compte des éléments ci-dessus, le calcul du pitch se fait comme suit

```
voise = (Seff > SeuilXeff) & (Nzx < SeuilZcross) & (reemax1 > SeuilCorrel*reemax) ;
if ~voise
  Kp = 0 ; % le son n'est pas voisé
end ;
```

7.7. Synthèse d'un son

La synthèse d'un son consiste à utiliser les paramètres obtenus ci-dessus pour synthétiser une tranche après l'autre. Il faudra donc créer le signal d'excitation (périodique ou bruité selon que le signal est voisé ou non), le filtrer à travers la fonction de transfert $H(z)$ et adapter son amplitude. Les commandes sont les suivantes :

```
% génération des impulsions de période Kp = Tp / Te
if Kp ~= 0
  for k=1 :Npoints
    if mod(k,Kp) == 0
      xt(k) = -1 ;
    else
      xt(k) = 0 ;
    end ; % if
  end ; % for
  xt(1) = -1 ; % 1ère impulsion non-nulle
end ; % if Kp ...

% génération d'un bruit de longueur Npoints
if Kp == 0
  xt=2*(rand(Npoints,1)-0.5) ;
end ; % if

% synthèse du son
yt = filter(1, coeff, xt) ;

% adaptation de l'amplitude
Seff = std(st) ; % valeur efficace du signal original
Yeff = std(yt) ; %valeur efficace du signal synthétisé
gain = Seff/Yeff ;
yt = yt*gain ;
```

On notera qu'avec MatLab il est également possible d'écouter directement des sons grâce à la fonction sound ou soundsc. Par exemple :

```
soundsc (yt, fe) ;
```

7.7.1. Signaux réel et synthétique

La figure 7.11 montre les résultats de la synthèse d'un phonème voisé. Ces différences visuelles nous paraissent difficilement acceptables. Il ne faut cependant pas oublier que la parole est un message très redondant et que seule l'écoute de la phrase synthétisée permet de juger de la qualité du codage LPC.

7.7.2. Mise en valeur des résultats

Afin de faciliter l'analyse critique des résultats obtenus, il est pratique de réunir sur une ou deux figures les graphes significatifs de la synthèse d'une phrase. Les figures 7.12 et 7.13 illustrent les résultats du codage et décodage de la phrase "*Comment allez-vous ?*".

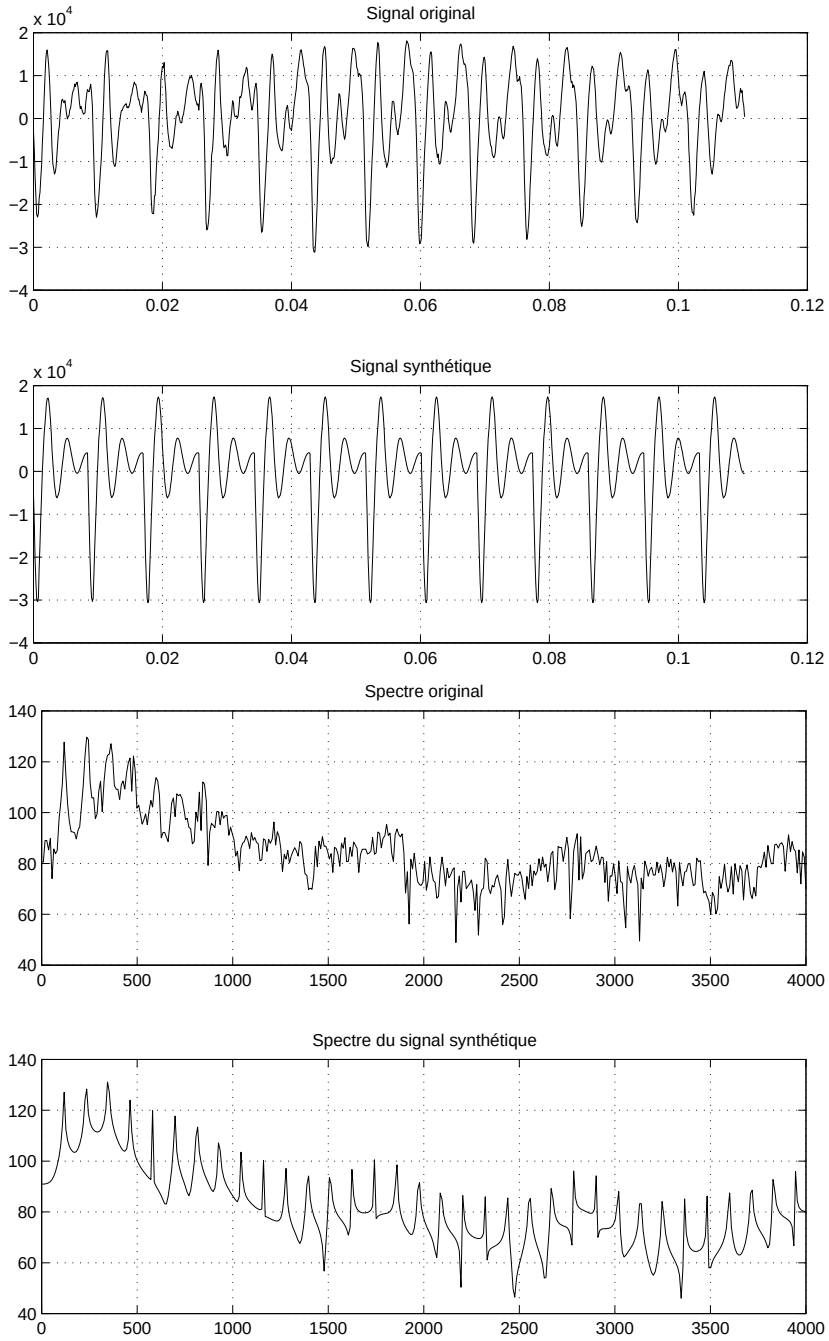

Fig. 7.11.: Signaux réel et synthétique avec leurs spectres respectifs

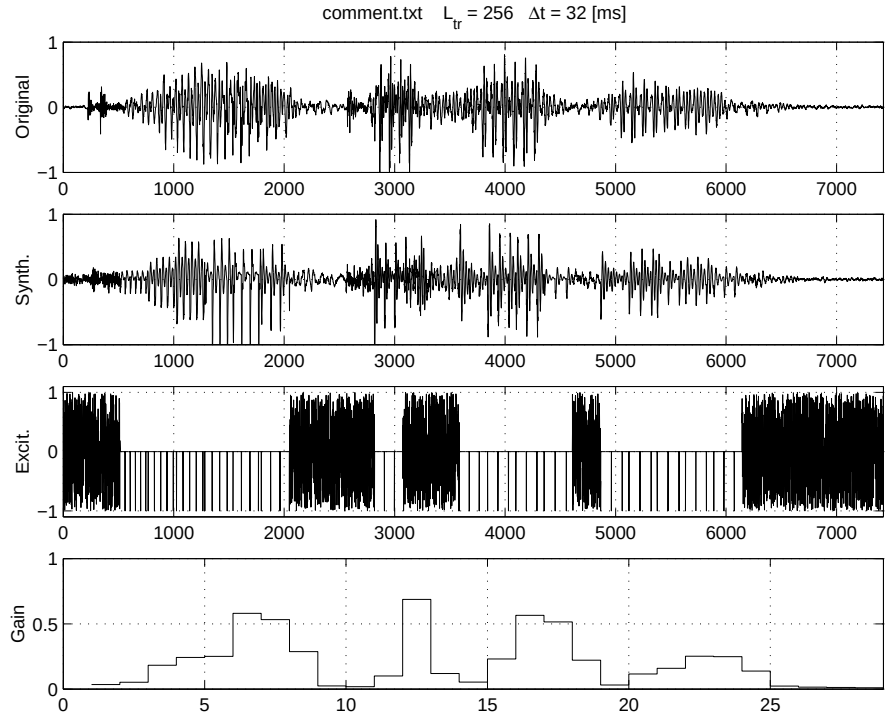

FIG. 7.12.: Signaux, excitation et gain

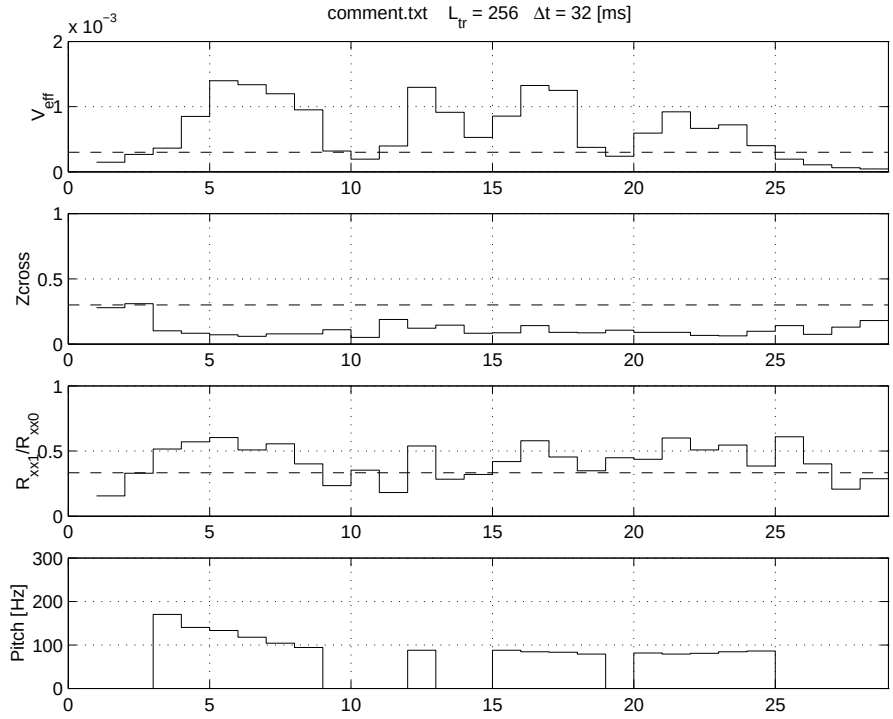

FIG. 7.13.: Évolution des paramètres

7.8. Travail pratique

Le codage de la parole consiste en l'extraction des paramètres étudiés ci-dessus, à savoir les coefficients de prédiction linéaire, la période du pitch et le gain du filtre. Ces paramètres sont ensuite transmis vers le récepteur qui en effectuera le décodage en reconstituant le signal sonore. Afin d'y parvenir, il faut reprendre une partie des éléments présentés plus haut et les placer dans un programme séquentiel dont la structure vous est donnée ci-dessous.

7.8.1. Codage et décodage d'une phrase

Créez un fichier Matlab vous permettant de coder puis décoder la phrase du fichier "colibri.txt" en vous inspirant de la structure suivante :

Codage

```
% préparation de l'espace Matlab
% lecture de la phrase à coder et normalisation entre +/- 1
% initialisation de la longueur et du nombre de tranches
% initialisation des constantes et vecteurs nécessaires aux calculs

% répéter jusqu'à la dernière tranche
  % choix de la trancher à coder
  % analyse lpc de la tranche et calcul du gain
  % recherche du signal d'excitation
     %% suppression des fréquences supérieures à fmax du pitch
     %% filtrage inverse pour obtenir l'excitation e(t)
  % recherche de la période d'excitation Kp
     %% autocorrélation de e(t) => ree
     %% limitation de ree entre Tpmin et Tpmax
     %% décision : voisé ou non
  % concaténation dans une matrice des coefficients, gains et périodes
% fin de la boucle répéter
```

Décodage

```
% lecture de la matrice contenant tous les paramètres
% extraction du vecteur contenant les gains
% extraction du vecteur contenant les périodes Kp

% répéter jusqu'à la dernière tranche
  % si Kp > 0 : génération des impulsions de période Kp
  % si Kp = 0 : génération du bruit entre +1 et -1
  % synthèse du son y(t)
  % ajustage de l'amplitude à Veff
  % concaténation des tranches
```

```
% fin de la boucle répéter

% limitation de l'amplitude à +/- 1
% traçage des signaux intéressants
```

Pour comparer le signal original avec le signal synthétisé, il peut être intéressant de les transférer simultanément vers CoolEdit dans un fichier `*.txt` de type ASCII. On peut, par exemple, envoyer le signal original vers le canal gauche et le signal synthétisé vers la droite :

```
% fichier CoolEdit : gauche = original, droite = synthèse
signaux = 32e3*[original synthese] ;
save fichier.txt signaux -ascii -tabs
```

Lors du chargement du fichier avec CoolEdit, on précisera que l'on souhaite travailler en mode stéréophonique.

7.8.2. Analyse des résultats

Débit d'informations Ayant terminé le codage et apprécié la qualité des résultats obtenus, il est intéressant de calculer le taux de compression ainsi obtenu. Pour ce faire, considérant que les informations transmises sont codées 8 bits,

1. faites la liste des informations transmises après codage LPC ;
2. à quel rythme sont-elles transmises ?
3. calculez le nombre de bits nécessaires pour chacune d'entre elles ;
4. calculez le débit d'informations et le taux de compression obtenu par rapport à un codage 8 bits de la parole non codée.

Modifications du codage

1. Qu'est-ce qui change si l'excitation est purement aléatoire ($K_p = 0$ pour toutes les tranches) ?
2. Plutôt que de rechercher et transmettre le pitch de chaque tranche, on peut, à chaque instant d'échantillonnage, transmettre le signe du résidu $e(n)$ et l'utiliser comme signal d'excitation. Que pensez-vous du résultat de cette synthèse très simple ?

7.8.3. Analyse et amélioration de la synthèse

Vous ne serez sûrement pas satisfait des résultats obtenus au premier essai. Essayez d'apporter des modifications à votre algorithme. Par exemple :

– Avez-vous pensé au fait que les conditions finales d'une tranche doivent être les conditions initiales de la suivante ?
– Le compteur de pitch doit-il être propre à chaque tranche ou global ?

– Faut-il augmenter le nombre de paramètres ?
– Essayez de varier les seuils de détection du pitch.

Afin de mieux saisir les possibilités d'amélioration, sélectionnez une tranche qui vous paraît intéressante. Puis sur celle-ci, tentez de comprendre où se situent les difficultés et observez les effets de vos modifications.

Malgré tous vos essais, il est peu probable que vous atteigniez la qualité de la téléphonie mobile. En effet, bien que celle-ci soit basée sur le codage LPC, on a dû, pour des raisons de qualité du son, amélioré l'analyse et la restitution du signal d'excitation. Vous trouverez plus d'informations dans la référence [2].

7.9. Minimisation de l'écart quadratique

Nous avons vu que la puissance ou variance de l'écart de l'ensemble des N échantillons $e[n]$ à disposition avec $0 \leq n \leq N$-1 dépend du choix des coefficients de prédiction a_k et qu'elle vaut :

$$\sigma_e^2(a_k) = \frac{1}{N}\sum_{n=0}^{N-1} e^2[n] = \frac{1}{N}\sum_{n=0}^{N-1}\left(s[n] + \sum_{k=1}^{p} a_k s[n-k]\right)^2 \tag{7.16}$$

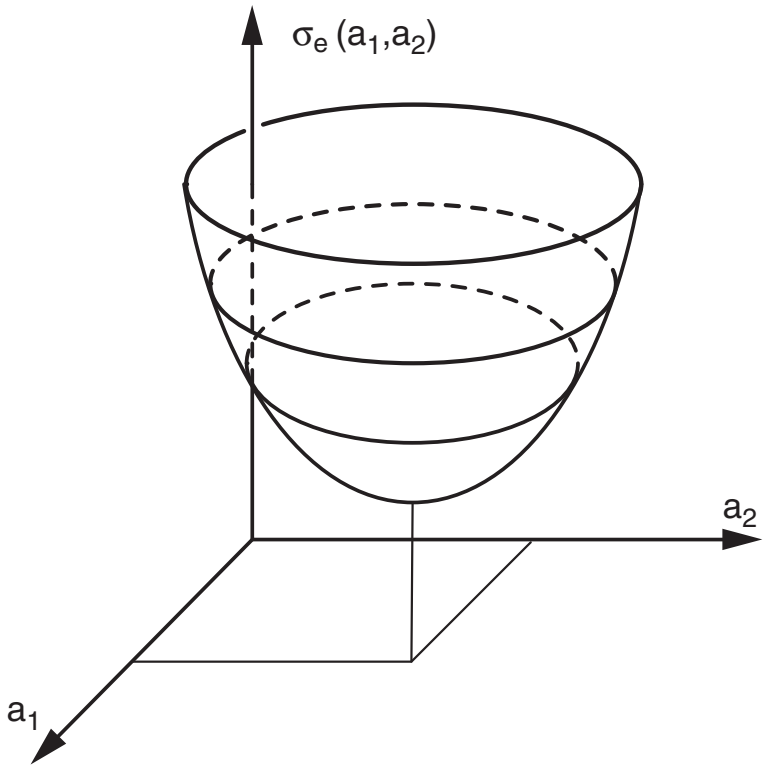

Fig. 7.14.: Variance de l'erreur de prédiction

Mathématiquement, la variance est donc une fonction des paramètres de prédiction a_k :

$$\sigma_e^2 = \sigma_e^2(a_1, a_2, \ldots, a_p) = \sigma_e^2(a_k) \tag{7.17}$$

et les valeurs des paramètres correspondant à sa valeur minimum s'obtiennent en annulant l'ensemble des dérivées partielles de σ_e^2 par rapport aux paramètres a_k :

$$\frac{\delta\sigma_e^2(a_k)}{\delta a_k} = 0,\ k = 1, \ldots, p \tag{7.18}$$

La dérivation de l'équation (7.16) s'effectue comme suit :

$$\begin{aligned}
\frac{1}{2}\frac{\delta\sigma_e^2(a_k)}{\delta a_k} &= \frac{1}{2}\frac{\delta}{\delta a_k}\left(\frac{1}{N}\sum_{n=0}^{N-1}\left(s[n] + \sum_{m=1}^{p} a_m s[n-m]\right)^2\right) \\
&= \frac{1}{2}\frac{1}{N}\sum_{n=0}^{N-1}\left(\frac{\delta}{\delta a_k}\left(s[n] + \sum_{m=1}^{p} a_m s[n-m]\right)^2\right) \\
&= \frac{1}{N}\sum_{n=0}^{N-1}\left(\left(s[n] + \sum_{m=1}^{p} a_m s[n-m]\right)\frac{\delta}{\delta a_k}\left(s[n] + \sum_{m=1}^{p} a_m s[n-m]\right)\right) \\
&= \frac{1}{N}\sum_{n=0}^{N-1}\left(\left(s[n] + \sum_{m=1}^{p} a_m s[n-m]\right)(0 + s[n-k])\right) \\
&= \frac{1}{N}\sum_{n=0}^{N-1}\left(\left(s[n]\,s[n-k] + \sum_{m=1}^{p} a_m s[n-m]\,s[n-k]\right)\right) \\
&= \frac{1}{N}\sum_{n=0}^{N-1} s[n]\,s[n-k] + \sum_{m=1}^{p} a_m \frac{2}{N}\sum_{n=0}^{N-1} s[n-m]\,s[n-k] \\
&= r_{ss}[k] + \sum_{m=1}^{p} a_m\, r_{ss}[k-m]
\end{aligned}$$

L'annulation de ce résultat donne

$$\sum_{m=1}^{p} a_m\, r_{ss}[k-m] = -r_{ss}[k], \qquad 1 \le k \le p$$

qui représente p équations pour les p paramètres inconnus a_m :

$$\begin{aligned}
a_1 r_{ss}[0] + a_2 r_{ss}[-1] + \cdots + a_p r_{ss}[1-p] &= -r_{ss}[1] \\
a_1 r_{ss}[1] + a_2 r_{ss}[0] + \cdots + a_p r_{ss}[2-p] &= -r_{ss}[2] \\
\vdots &= \vdots \\
a_1 r_{ss}[p-1] + a_2 r_{ss}[p-2] + \cdots + a_p r_{ss}[0] &= -r_{ss}[p]
\end{aligned}$$

avec :

$$r_{ss}[k] = \sum_{n=0}^{N-1} s[n]s[n-k],\ k = 1, \ldots, p \tag{7.19}$$

On voit ainsi que les coefficients des inconnues a_k sont les p premières valeurs de la fonction d'autocorrélation $r_{ss}[k]$ du signal $s[n]$ comportant N échantillons.

Cet ensemble d'équations linéaires peut s'écrire sous forme matricielle :

$$\mathbf{R}_{ss}\,\mathbf{a} = -\mathbf{r}_{ss} \tag{7.20}$$

où $\mathbf{R}_{ss}$ est la matrice $p \times p$ d'autocorrélation, $\mathbf{r}_{ss}$ le vecteur $p \times 1$ d'autocorrélation et $\mathbf{a}$ le vecteur $p \times 1$ des paramètres de prédiction. On notera que, la fonction d'autocorrélation étant paire, la matrice $\mathbf{R}_{ss}$ est symétrique.

On voit donc que les coefficients de prédiction linéaire peuvent s'obtenir par inversion de la matrice $\mathbf{R}_{ss}$:

$$\mathbf{a} = -\mathbf{R}_{ss}^{-1}\,\mathbf{r}_{ss} \tag{7.21}$$

Bibliographie

[1] R. Boite et al., *Traitement de la parole*, PPUR, 2000

[2] B. Porat, *A Course in Digital Signal Processing*, John Wiley, 1997

[3] J.R. Deller, J.G. Proakis,J.L Hansen, *Discrete-Time Processing of Speech Signals*, Macmillan, 1993

[4] V.K. Ingle, J.G. Proakis, *Digital Signal Processing Using MatLab*, PWS, 1997

[5] C.S. Burrus et all, *Computer-Based Exercises for Signal Processing Using MatLab*, Prentice Hall, 1994

8. Introduction au filtrage adaptatif

Le filtrage adaptatif est basé sur la recherche de paramètres optimaux par minimisation d'un critère de performance. Fréquemment, cette minimisation se fait en recherchant les moindres carrés. Étant donné le cadre dans lequel cette présentation est faite, il est nécessaire d'introduire quelques éléments préalables.

On commencera donc par rappeler quelques définitions d'estimateurs statistiques puis on montrera ce que sont la régression linéaire et le filtrage de Wiener avant de parler du filtrage adaptatif proprement dit.

8.1. Notions de probabilités

8.1.1. Définitions de quelques estimateurs statistiques

Considérant une variable aléatoire réelle z, on la caractérise généralement à l'aide des grandeurs suivantes :

1. Sa **valeur moyenne** :
$$\mu_z = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} z(n) \tag{8.1}$$
On notera que la valeur moyenne μ_z représente la composante continue du signal autour de laquelle prennent place les fluctuations.
2. Sa **puissance moyenne** :
$$P_z = \mu_{z^2} = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} z^2(n) \tag{8.2}$$
3. Sa **variance** qui mesure la puissance des fluctuations autour de la valeur moyenne
$$\begin{aligned} \sigma_z^2 &= \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} \left(z(n) - \mu_z\right)^2 \\ &= \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} \left(z^2(n) - 2\mu_z\, z(n) + \mu_z^2\right) \\ &= \mu_{z^2} - 2\mu_z\,\mu_z + \mu_z^2 = \mu_{z^2} - 2\mu_z^2 + \mu_z^2 \end{aligned}$$

qui vaut finalement

$$\sigma_z^2 = \mu_{z^2} - \mu_z^2 \tag{8.3}$$

4. Son **écart-type** (ou déviation standard) défini comme la racine carrée de la variance :

$$\sigma_z = \sqrt{\mu_{z^2} - \mu_z^2} \tag{8.4}$$

Sa valeur est égale à la valeur efficace des variations du signal autour de la valeur moyenne.

Il est intéressant de noter que la puissance moyenne (μ_{z^2}) de la variable z peut également s'écrire sous la forme

$$\mu_{z^2} = \mu_z^2 + \sigma_z^2 \tag{8.5}$$

On voit ainsi que la puissance de la variable μ_{z^2} est égale à la puissance de sa valeur moyenne μ_z^2 plus la puissance de ses fluctuations σ_z^2.

8.1.2. Remarques

1. Il est intéressant de relever que si l'on considère une notation vectorielle du type :

$$z = [z(0), \ldots, z(N-1)], \qquad z^T = \begin{bmatrix} z(0) \\ \vdots \\ z(N-1) \end{bmatrix}$$

la puissance s'écrit simplement sous la forme d'un produit scalaire :

$$P_z = \frac{1}{N} \sum_{n=0}^{N-1} z^2(n) = \frac{1}{N}\, z\, z^T \tag{8.6}$$

2. De même, l'indépendance (ou la non correspondance) de deux signaux ou vecteurs peut se mesurer avec le produit scalaire :

$$r = x\, y^T = \sum_{n=0}^{N-1} x(n)\, y(n) \tag{8.7}$$

Dans le cas où les signaux sont orthogonaux (c'est à dire indépendants), $x\, y^T$ sera nul alors que si les signaux sont fortement dépendants (ou ressemblants), la valeur du produit $x\, y^T$ sera proche de son maximum.

8.1.3. Fonction de répartition et densité de probabilités

Dans le cas d'une variable aléatoire continue x, on définit la probabilité d'avoir la valeur mesurée x' inférieure à une valeur x donnée

$$P(x) \equiv \text{prob}(x' < x) \tag{8.8}$$

Cette fonction porte le nom de fonction de répartition la variable x.

La probabilité d'avoir la valeur mesurée x' comprise entre deux valeurs x et $x + \Delta x$ vaut donc

$$P(x < x' < x + \Delta x) = P(x' < x + \Delta x) - P(x' < x) \tag{8.9}$$

Cette relation permet de définir la densité de probabilité

$$p(x) \equiv \lim_{\Delta x \to 0} \frac{P(x < x' < x + \Delta x)}{\Delta x} = \frac{dP(x)}{dx} \tag{8.10}$$

et d'en tirer la fonction de répartition

$$P(x) = \int_{-\infty}^{x} p(x)\, dx \tag{8.11}$$

avec comme propriété évidente

$$P(-\infty < x' < +\infty) = P(\infty) = 1 \tag{8.12}$$

C'est la densité de probabilité qui est généralement utilisée comme modèle de pour décrire la répartion des valeurs d'une variable aléatoire. À partir de celle-ci, on peut calculer la valeur moyenne, la variance et la puissance d'une variable x à l'aide de

$$\mu_x = \int_{-\infty}^{+\infty} x\, p(x)\, dx \tag{8.13}$$

$$\sigma_x^2 = \int_{-\infty}^{+\infty} (x - \mu_x)^2\, p(x)\, dx \tag{8.14}$$

$$\mu_{x^2} = \int_{-\infty}^{+\infty} x^2\, p(x)\, dx \tag{8.15}$$

8.1.4. Modèles statistiques

Les modèles les plus fréquemment utilisés sont

1. La **répartition uniform**e entre deux valeurs extrêmes x_{min} et x_{max}

   $$p(x) = \text{constante} = \frac{1}{x_{max} - x_{min}} = \frac{1}{\Delta x} \tag{8.16}$$

   On montre que dans ce cas, la variance vaut

   $$\sigma_x^2 = \frac{(x_{max} - x_{min})^2}{12} = \frac{\Delta x^2}{12} \tag{8.17}$$

2. La **répartition gaussienne** entre $-\infty$ et $+\infty$

   $$p(x) = \frac{1}{\sqrt{2\pi}\,\sigma_x} \exp\left(-\frac{(x - \mu_x)^2}{2\,\sigma_x^2}\right) \tag{8.18}$$

   Il est intéressant de relever que que la probabilité de trouver à l'intérieur des domaines $\pm\sigma_x$ et $\pm 3\sigma_x$ par rapport à la valeur moyenne valent respectivement

   $$P\left(|x - \mu_x| < \sigma_x\right) = 68\% \tag{8.19}$$

   $$P\left(|x - \mu_x| < 3\sigma_x\right) = 99.7\% \tag{8.20}$$

Suivant les applications, on peut imaginer d'autres distributions comme par exemple la **répartition exponentielle** décrite par

$$p(x) = \frac{1}{\sqrt{2}\,\sigma_x} \exp\left(-\sqrt{2}\,\frac{|x-\mu_x|}{\sigma_x}\right) \tag{8.21}$$

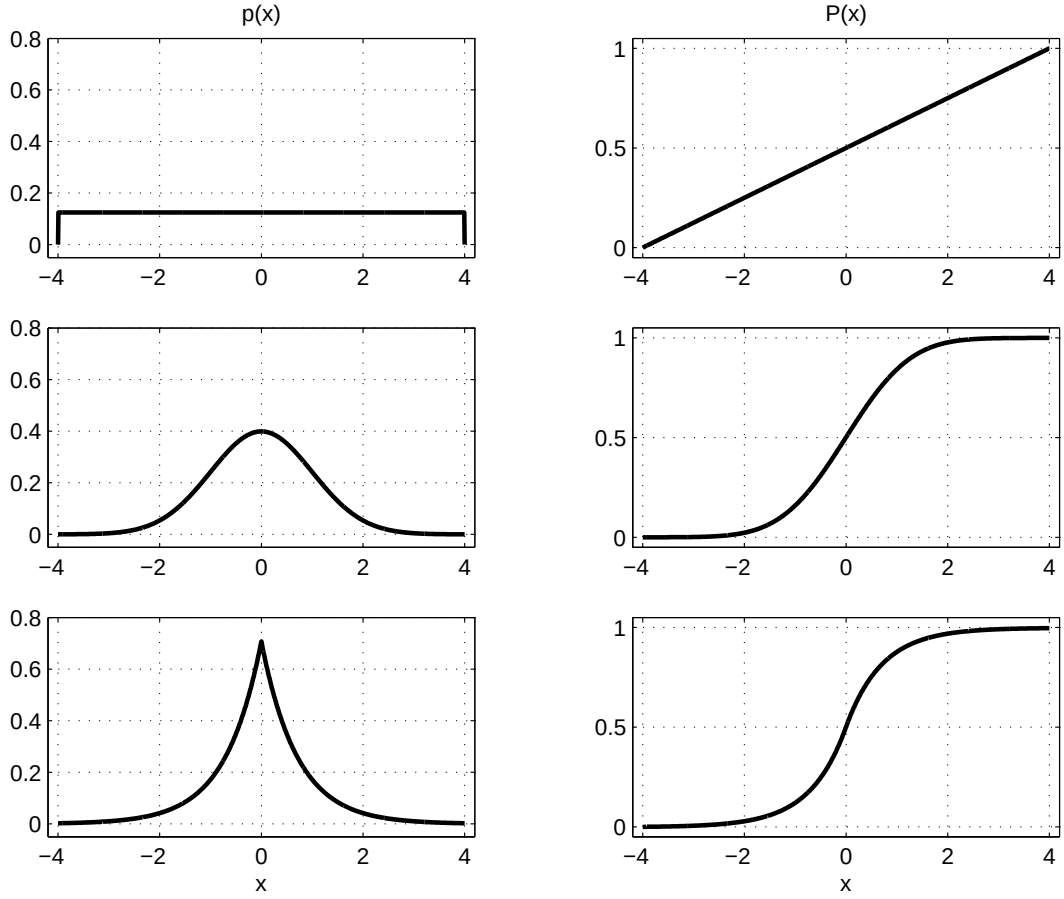

Fig. 8.1.: Illustration des distributions uniforme, gaussienne et exponentielle

8.2. Régression linéaire

La régression linéaire consiste en la recherche de la droite passant au mieux parmi un ensemble de points mesurés (figure 8.2). Le critère conduisant à cet optimum est la minimisation des distances quadratiques entre les points mesurés et la droite optimum.

On notera que la régression linéaire s'applique aux systèmes statiques alors que l'approche de Wiener (que l'on verra dans la section suivante) sert à optimiser des systèmes évoluant au cours du temps.

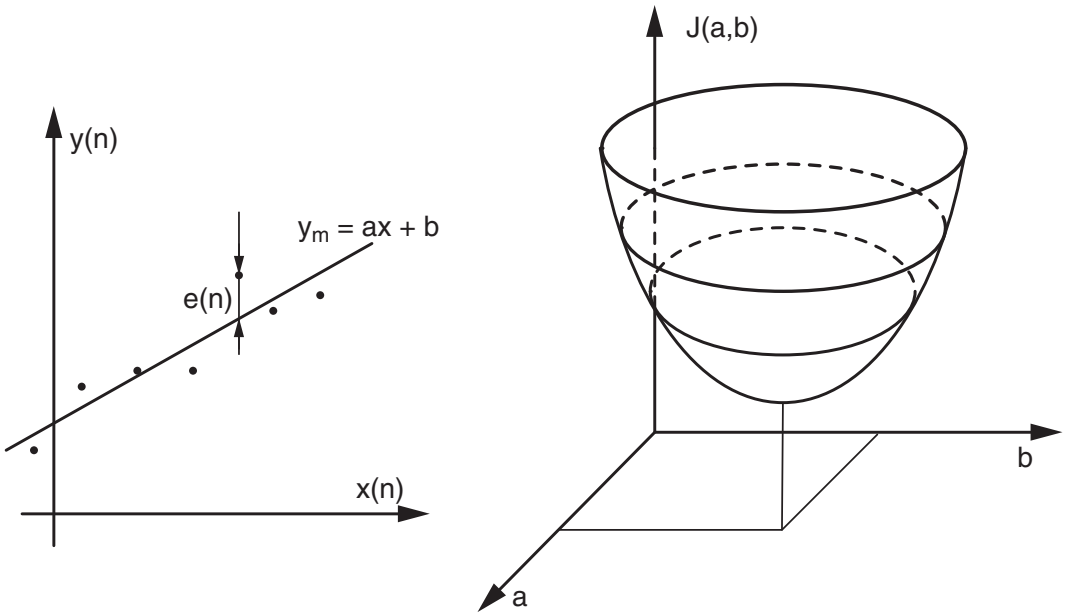

FIG. 8.2.: Régression linéaire

8.2.1. Mesure, modèle et écart

Comme on souhaite faire passer une droite parmi un ensemble de points,on se donne un modèle dont l'équation est :

$$y_m = ax + b \tag{8.22}$$

L'écart de $y(n)$ par rapport au modèle s'écrit donc :

$$e(n) = y(n) - y_m(n)$$

$$e(n) = y(n) - (a\,x(n) + b) \tag{8.23}$$

Si l'on décrit la mesure $y(n)$ par rapport au modèle $y_m(n)$, on a évidemment :

$$y(n) = y_m(n) + e(n) \tag{8.24}$$

On associe généralement deux grandeurs à l'écart $e(n)$:

1. sa valeur moyenne μ_e qui doit tendre vers 0 si le modèle n'est pas biaisé ;
2. sa puissance σ_e^2 qui doit diminuer lorsque a et b se rapprochent des "vraies" valeurs liant y à x.

On notera que pour le calcul d'une régression linéaire, on fait l'hypothèse qu'il n'y a pas de bruit sur la valeur de la variable indépendante $x(n)$.

8.2.2. Minimisation de l'écart quadratique

L'obtention de la droite passant au mieux parmi les points mesurés nécessite la recherche du minimum d'une fonction dépendant des paramètres recherchés a et b. Pour cela, on définit un critère d'optimisation qui mesure la puissance ou la variance de l'écart :

$$J(a,b) = \frac{1}{N}\sum_{n=0}^{N-1} e^2(n) = \frac{1}{N}\sum_{n=0}^{N-1} \left(y(n) - (ax(n)+b)\right)^2 \tag{8.25}$$

Lorsque l'écart quadratique est minimum, on a :

$$\frac{\partial J(a,b)}{\partial a} = 0 \qquad\qquad \frac{\partial J(a,b)}{\partial b} = 0 \tag{8.26}$$

avec

$$\begin{aligned}
\frac{\partial J(a,b)}{\partial a} &= \frac{1}{N}\sum_{n=0}^{N-1} 2\ \left(y(n) - (ax(n)+b)\right)\ (0 - x(n) - 0) \\
&= \frac{2}{N}\sum_{n=0}^{N-1} \left(-x(n)\,y(n) + a\,x^2(n) + b\,x(n)\right) \\
&= \frac{2}{N}\left(-\sum_{n=0}^{N-1} x(n)\,y(n) + a\sum_{n=0}^{N-1} x^2(n) + b\sum_{n=0}^{N-1} x(n)\right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial J(a,b)}{\partial b} &= \frac{1}{N}\sum_{n=0}^{N-1} 2\ \left(y(n) - (ax(n)+b)\right)\ (0 - 0 - 1) \\
&= \frac{2}{N}\sum_{n=0}^{N-1} \left(-y(n) + ax(n) + b\right) \\
&= \frac{2}{N}\left(-\sum_{n=0}^{N-1} y(n) + a\sum_{n=0}^{N-1} x(n) + \sum_{n=0}^{N-1} b\right)
\end{aligned}$$

On en tire 2 équations dont les inconnues sont a et b :

$$a\,\frac{1}{N}\sum_{n=0}^{N-1} x^2(n) + b\,\frac{1}{N}\sum_{n=0}^{N-1} x(n) \;=\; \frac{1}{N}\sum_{n=0}^{N-1} x(n)\,y(n) \tag{8.27}$$

$$a\,\frac{1}{N}\sum_{n=0}^{N-1} x(n) + \frac{1}{N}\sum_{n=0}^{N-1} b \;=\; \frac{1}{N}\sum_{n=0}^{N-1} y(n) \tag{8.28}$$

8.2.3. Équations de la régression linéaire

Se souvenant de la définition d'une valeur moyenne, on voit que les équations (8.28) et (8.27) s'écrivent plus simplement sous la forme :

$$a\,\mu_x + b = \mu_y \tag{8.29}$$

$$a\,\mu_{x^2} + b\,\mu_x = \mu_{xy} \tag{8.30}$$

Sous forme matricielle, cela donne :

$$\begin{pmatrix} \mu_x & 1 \\ \mu_{x^2} & \mu_x \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \mu_y \\ \mu_{xy} \end{pmatrix}$$

dont la solution est

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \mu_x & 1 \\ \mu_{x^2} & \mu_x \end{pmatrix}^{-1} \begin{pmatrix} \mu_y \\ \mu_{xy} \end{pmatrix} \tag{8.31}$$

L'inversion de la matrice et le calcul explcite de a et b donnent alors

$$a = \frac{\mu_x\,\mu_y - \mu_{xy}}{\mu_x^2 - \mu_{x^2}} \tag{8.32}$$

$$b = \frac{\mu_x\,\mu_{xy} - \mu_y\,\mu_{x^2}}{\mu_x^2 - \mu_{x^2}} \tag{8.33}$$

Dans le cas particulier où la droite passe par l'origine, les valeurs moyennes μ_x et μ_y sont nulles et on a :

$$a = \frac{\mu_{xy}}{\mu_{x^2}} = \frac{\sum x(n)\,y(n)}{\sum x^2(n)} = \frac{x^T\,y}{x^T\,x}, \qquad b = 0 \tag{8.34}$$

8.3. Filtrage de Wiener

Dans de nombreuses applications, les signaux temporels sont entachées d'une interférence ou d'un bruit non désirés. Il faut alors trouver une solution permettant de supprimer ou tout au moins réduire ces composantes perturbatrices. Dans le cas où le spectre du signal désiré et celui du signal perturbateur se superposent, il n'est pas possible de recourir au filtrage classique. Le filtre de Wiener apporte une solution à ce problème lorsque le processus est stationnaire.

8.3.1. Définition du problème

On considère ici le schéma de la figure 8.3 dans lequel on trouve :

1. le signal d'excitation $x(n)$ connu ou mesuré ;
2. le signal de sortie du processus $y_p(n)$ inatteignable ;

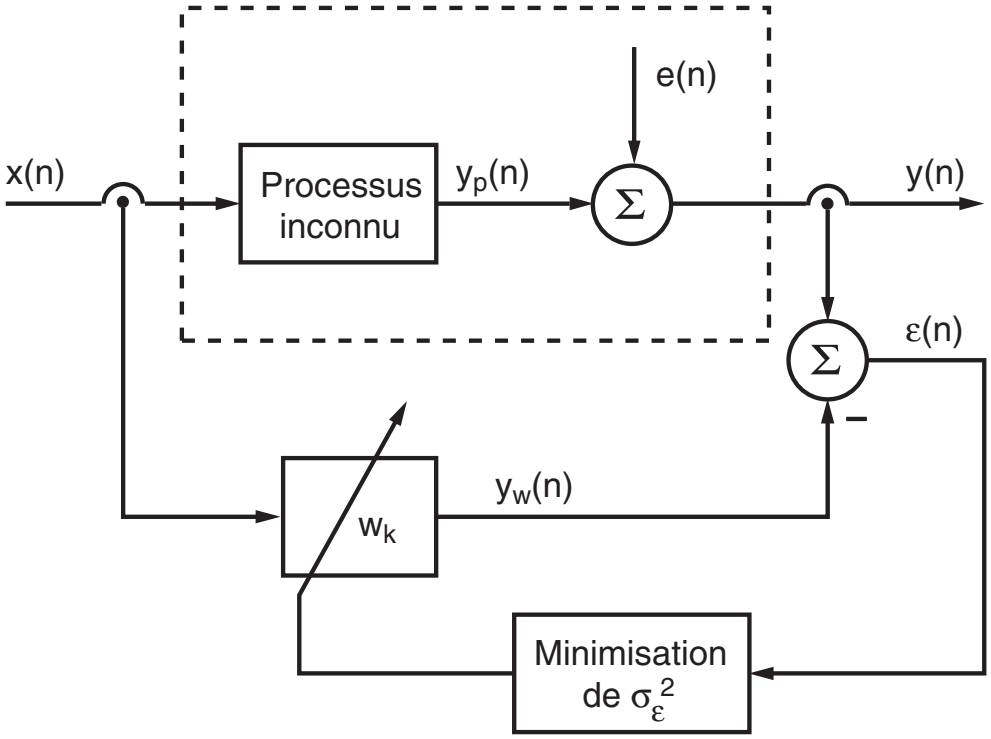

Fig. 8.3.: Filtrage de Wiener

3. le signal de sortie mesuré $y(n)$ entâché d'une perturbation inconnue $e(n)$;
4. le signal modélisé $y_w(n)$ à l'aide des paramètres w_k ;
5. le signal d'écart $\epsilon(n)$ entre le modèle $y_w(n)$ et la mesure $y(n)$.

On admet que le signal mesuré $y(n)$, causé par l'excitation $x(n)$, peut être modélisé à l'aide d'un modèle MA (*Moving Average* = moyenne glissante) d'ordre p représentant un processus stationnaire inconnu :

$$y_p(n) = \sum_{k=0}^{p-1} w_k \, x(n-k)$$

Le but poursuivi est de trouver les coefficients w_k du modèle MA à partir de la mesure des signaux d'entrée $x(n)$ et de sortie $y(n)$.

La recherche d'une solution consiste à rendre $y_w(n)$ aussi proche que possible du signal $y_p(n)$ en minimisant l'erreur quadratique moyenne (*Mean Square Error* = MSE) par ajustage des coefficients w_k. Il est important de bien comprendre que

> ***si la solution exacte est trouvée, le signal d'écart*** $\varepsilon(n)$ ***n'est pas nul, mais égal à la peturbation*** $e(n)$ ***de la mesure.***

Afin d'alléger l'écriture de ce qui suit, on se contentera de traiter le cas particulier où le processus est décrit par 3 paramètres (l'extension à une dimension plus grande se fait sans difficulté)

$$W = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} \tag{8.35}$$

Dans ce cas, l'estimateur $y_w(n)$ du signal $y_p(n)$ vaut :

$$y_w(n) = w_0\, x(n) + w_1\, x(n-1) + w_2\, x(n-2) \qquad 0 \leq n \leq N-1 \tag{8.36}$$

8.3.2. Résolution au sens des moindres carrés

Le problème ainsi posé est proche de celui de la régression linéaire que l'on a étudié pour les systèmes statiques (ou sans mémoire). Dans le cas des systèmes dynamiques, les signaux évoluent temporellement. L'erreur est alors une fonction du temps que l'on cherche à réduire en minimisant sa valeur quadratique moyenne ; cela se fait en variant les coefficients inconnus w_k. On pose donc :

$$\varepsilon(n) = y(n) - y_w(n) \tag{8.37}$$

$$J = \frac{1}{N} \sum_{n=0}^{N-1} \left(y(n) - y_w(n)\right)^2 \tag{8.38}$$

Tenant compte de l'équation (8.36), il vient

$$J\left(w_0, w_1, w_2\right) = \frac{1}{N} \sum_{n=0}^{N-1} \left(y(n) - w_0\, x(n) - w_1\, x(n-1) - w_2\, x(n-2)\right)^2 \tag{8.39}$$

Le calcul des dérivées partielles de $J\left(w_0, w_1, w_2\right)$ par rapport à chacun des coefficients inconnus w_k donne

$$\begin{aligned}
\frac{\partial J}{\partial w_0} &= \frac{2}{N} \sum_{n=0}^{N-1} \left(y(n) - w_0\, x(n) - w_1\, x(n-1) - w_2\, x(n-2)\right)\, \left(-x(n)\right) \\
&= -\frac{2}{N} \sum_{n=0}^{N-1} \left(x(n)y(n) - w_0\, x(n)x(n) - w_1\, x(n)x(n-1) - w_2\, x(n)x(n-2)\right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial J}{\partial w_1} &= \frac{2}{N} \sum_{n=0}^{N-1} \left(y(n) - w_0\, x(n) - w_1\, x(n-1) - w_2\, x(n-2)\right)\, \left(-x(n-1)\right) \\
&= -\frac{2}{N} \sum_{n=0}^{N-1} \left(x(n-1)y(n) - w_0\, x(n-1)x(n) - w_1\, x(n-1)x(n-1) - w_2\, x(n-1)x(n-2)\right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial J}{\partial w_2} &= \frac{2}{N} \sum_{n=0}^{N-1} \left(y(n) - w_0\, x(n) - w_1\, x(n-1) - w_2\, x(n-2)\right)\, \left(-x(n-2)\right) \\
&= -\frac{2}{N} \sum_{n=0}^{N-1} \left(x(n-2)y(n) - w_0\, x(n-2)x(n) - w_1\, x(n-2)x(n-1) - w_2\, x(n-2)x(n-2)\right)
\end{aligned}$$

Tenant compte de la définition de la fonction de corrélation

$$r_{xy}(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)\, y(n+k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n-k)\, y(n) = r_{yx}(-k) \tag{8.40}$$

on voit que ces trois dérivées s'écrivent plus simplement sous la forme

$$\begin{aligned}
\frac{\partial J}{\partial w_0} &= -2\left(r_{xy}(0) - w_0 r_{xx}(0) - w_1 r_{xx}(-1) - w_2 r_{xx}(-2)\right) \\
\frac{\partial J}{\partial w_1} &= -2\left(r_{xy}(+1) - w_0 r_{xx}(+1) - w_1 r_{xx}(0) - w_2 r_{xx}(-1)\right) \\
\frac{\partial J}{\partial w_2} &= -2\left(r_{xy}(+2) - w_0 r_{xx}(+2) - w_1 r_{xx}(+1) - w_2 r_{xx}(0)\right)
\end{aligned}$$

Comme l'erreur quadratique obtenue est minimum lorsque ces dérivées s'annullent, on obtient finalement un ensemble de 3 équations à 3 inconnues

$$\begin{aligned}
w_0\, r_{xx}(0) + w_1\, r_{xx}(-1) + w_2\, r_{xx}(-2) &= r_{xy}(0) \\
w_0\, r_{xx}(+1) + w_1\, r_{xx}(0) + w_2\, r_{xx}(-1) &= r_{xy}(+1) \\
w_0\, r_{xx}(+2) + w_1\, r_{xx}(+1) + w_2\, r_{xx}(0) &= r_{xy}(+2)
\end{aligned}$$

que l'on écrit sous la forme matricielle suivante

$$\begin{pmatrix} r_{xx}(0) & r_{xx}(-1) & r_{xx}(-2) \\ r_{xx}(+1) & r_{xx}(0) & r_{xx}(-1) \\ r_{xx}(+2) & r_{xx}(+1) & r_{xx}(0) \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} r_{xy}(0) \\ r_{xy}(1) \\ r_{xy}(2) \end{pmatrix} \tag{8.41}$$

Cette matrice d'autocorrélation est obligatoirement symétrique car la fonction d'autocorrélation est paire.

En représentant la matrice d'autocorrélation par le symbole R_{xx}, le vecteur des paramètres par W et le vecteur d'intercorrélation par r_{xy}, ce résultat s'écrit plus succinctement sous la forme :

$$R_{xx}\, W = r_{xy} \tag{8.42}$$

dont la solution est

$$W = R_{xx}^{-1}\, r_{xy} \tag{8.43}$$

Cette équation porte le nom d'équation normale ou de Wiener-Hopf.

8.3.3. Description matricielle

Les calculs que l'on vient d'exposer peuvent être présentés dans une écriture plus concise fréquemment utilisée. Définissant tout d'abord les vecteurs colonnes suivants :

$$W = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{p-1} \end{bmatrix} \qquad X(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-p+1) \end{bmatrix} \tag{8.44}$$

on obtient :

– le signal estimé $y_w(n)$

$$y_w(n) = \sum_{i=0}^{p-1} w_i\, x(n-i) = W^T\, X(n) = X(n)^T\, W \tag{8.45}$$

– l'erreur d'estimation $\varepsilon(n)$

$$\varepsilon(n) \;=\; y(n) - y_w(n) = y(n) - X(n)^T W \tag{8.46}$$

– l'erreur quadratique $\varepsilon^2(n)$

$$\varepsilon^2(n) \;=\; \big(y(n) - X(n)^T\, W\big)^2 \tag{8.47}$$

$$\varepsilon^2(n) \;= y^2(n) - 2\, y(n)\, X(n)^T\, W + W^T\, X(n)\, X(n)^T\, W \tag{8.48}$$

– l'erreur quadratique moyenne $J(W)$ fonction des paramètres W

$$\begin{aligned} J(W) \;&=\; E\{\varepsilon^2(n)\} = E\{\big(y(n) - X(n)^T\, W\big)^2\} \\ &=\; E\{y^2(n)\} - 2\, E\{y(n)\, X(n)^T\, W\} + E\{W^T\, X(n)\, X(n)^T\, W\} \end{aligned}$$

d'où

$$J(W) = r_{yy}(0) - 2\, r_{xy}^T\, W + W^T R_{xx} W \tag{8.49}$$

– le gradient de $J(W)$ par rapport au vecteur W des coefficients w_k

$$\frac{dJ}{dW} = -2\, r_{xy} + 2\, R_{xx}\, W \tag{8.50}$$

– le vecteur des paramètres optimaux qui annule le gradient

$$W = R_{xx}^{-1}\, r_{xy} \tag{8.51}$$

8.3.4. Applications du filtrage de Wiener

Les applications du filtrage de Wiener diffèrent par la manière dont est extraite la réponse désirée. Dans ce contexte, on peut distinguer quatre classes fondamentales utilisant le filtrage de Wiener :

1. l'identification de processus ; dans ce cas, on souhaite trouver la réponse impulsionnelle $w(n)$ représentant au mieux le processus inconnu ;
2. la modélisation inverse avec laquelle on tente de reconstruire un signal ;
3. la prédiction linéaire qui, sur la base des échantillons précédents, permet d'estimer une valeur à venir (codage LPC de la parole) ;
4. la suppression d'un signal perturbateur.

Dans ce qui suit, compte tenu du cadre dans lequel est présentée cette note, on se contentera d'illustrer comment on peut supprimer une perturbation grâce au filtrage adaptatif.

8.4. Suppression d'une perturbation

Comme illustration du filtrage de Wiener, imaginons la mesure de l'activité cardiaque d'un foetus à l'aide d'un électrocardiogramme (ECG) pris au niveau de l'abdomen de la mère et qui, naturellement, est perturbé par l'ECG de celle-ci.

Cette mesure nécessite l'utilisation de 2 capteurs. Avec le premier, on mesure le signal de référence $x(n)$ représentant, si possible, uniquement l'ECG de la mère. Avec le deuxième, on mesure le signal $y(n)$ qui est l'ECG du foetus perturbé par l'activité cardiaque de la mère.

Les signaux du schéma de Wiener (figure 8.4) sont alors les suivants :

1. $x(n)$ = l'ECG maternel mesuré près du coeur,
2. $y(n)$ = l'ECG foetal perturbé par celui de la mère,
3. $y_p(n)$ = l'ECG maternel près du foetus,
4. $y_w(n)$ = l'estimation de l'ECG maternel près du foetus,
5. $e(n)$ = l'ECG foetal,
6. $\varepsilon(n)$ = l'estimation de l'ECG foetal.

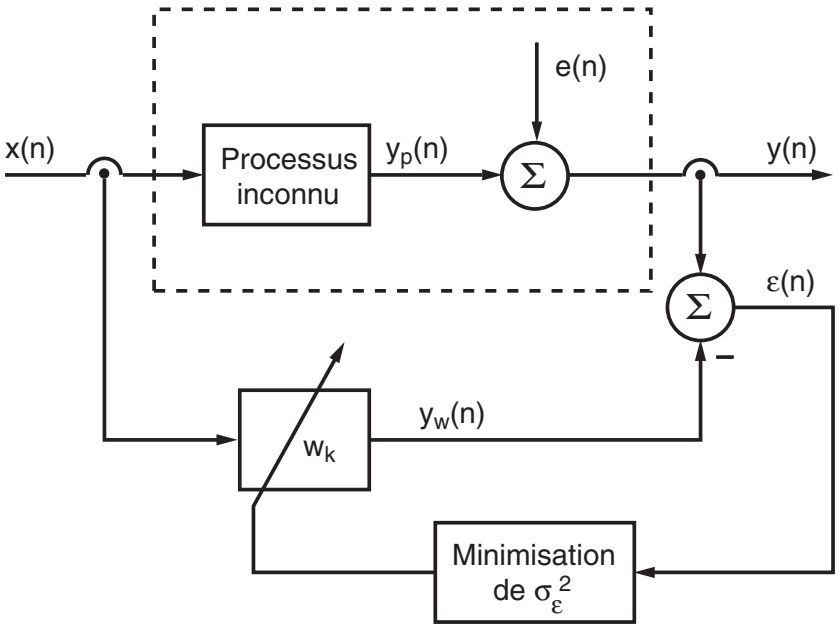

Fig. 8.4.: Suppression de la perturbation $y_p(n)$

On notera que dans ce problème, les rôles sont inversés par rapport à la définition initiale du filtre de Wiener. En effet, le signal $e(n)$ considéré plus haut comme une perturbation du signal recherché $y_p(n)$, est, dans notre cas, le signal ECG que l'on souhaite mesurer et le signal $y_p(n)$ est la perturbation que l'on souhaite rejeter. C'est en recherchant $y_w(n) \simeq y_p(n)$ que l'on obtient une bonne estimation $\varepsilon(n)$ de l'ECG foetal $e(n) \simeq x(n) - y_w(n)$.

Dans cette simulation et dans un but didactique, on a choisi un modèle MA constitué de deux coefficients seulement $W = [w_0, w_1]^T$. Le système à résoudre s'écrit alors :

$$\begin{pmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(1) & r_{xx}(0) \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} = \begin{pmatrix} r_{xy}(0) \\ r_{xy}(1) \end{pmatrix}$$

La recherche des coefficients W peut se faire de deux manières :

1. Dans le cas où l'on considère que le processus générateur de la perturbation est stationnaire, on commence par enregistrer la totalité des signaux $x(n)$ et $y(n)$. Puis, on calcule le vecteur des coefficients W après avoir calculé R_{xx} et r_{xy} pour l'ensemble des points acquis.
2. Si les signaux ne sont pas stationnaires (ce qui est le cas lorsque le processus change au cours du temps), il faut, après chaque échantillonnage, calculer les coefficients $W = R_{xx}^{-1}\, r_{xy}$.

8.4.1. Filtrage de Wiener classique

La figure 8.5 présente les résultats que l'on obtient avec un filtrage de Wiener classique dans lequel l'ensemble des points acquis est analysé en une seule fois. Dans cette approche, on fait l'hypothèse que le processus générateur de la perturbation est stationnaire. Sur la figure 8.5, on a tracé dans l'ordre :

– le signal $x(n)$ correspondant à l'ECG maternel,
– le signal $y(n)$ correspondant à l'ECG foetal perturbé,
– l'estimation $\varepsilon(n)$ de l'ECG foetal et sa valeur exacte $e(n)$ en pointillé.

On peut relever à quel point le résultat obtenu est proche du signal original. Un exemple de codage pour 2 paramètres est donné à la figure 8.6.

8.4.2. Remarque

D'un point de vue pratique, le filtre de Wiener tel qu'il a été présenté ci-dessus souffre de quelques limitations :

– il nécessite le calcul de la matrice d'autocorrélation R_{xx} et du vecteur d'intercorrélation r_{xy}, tous deux coûteux en temps de calcul ;
– il faut inverser la matrice R_{xx}, ce qui peut demander beaucoup de calcul et d'espace mémoire ;
– si les signaux ne sont pas stationnaires (ce qui est fréquent), R_{xx} et r_{xy} évoluent au cours du temps ; il faut donc à chaque instant résoudre l'équation de Wiener-Hopf.

Pour des applications en temps réel, il faut donc trouver un moyen rapide, efficace et robuste pour calculer récursivement la solution $W = R_{xx}^{-1} r_{xy}$. C'est ce que fait le filtrage adaptatif.

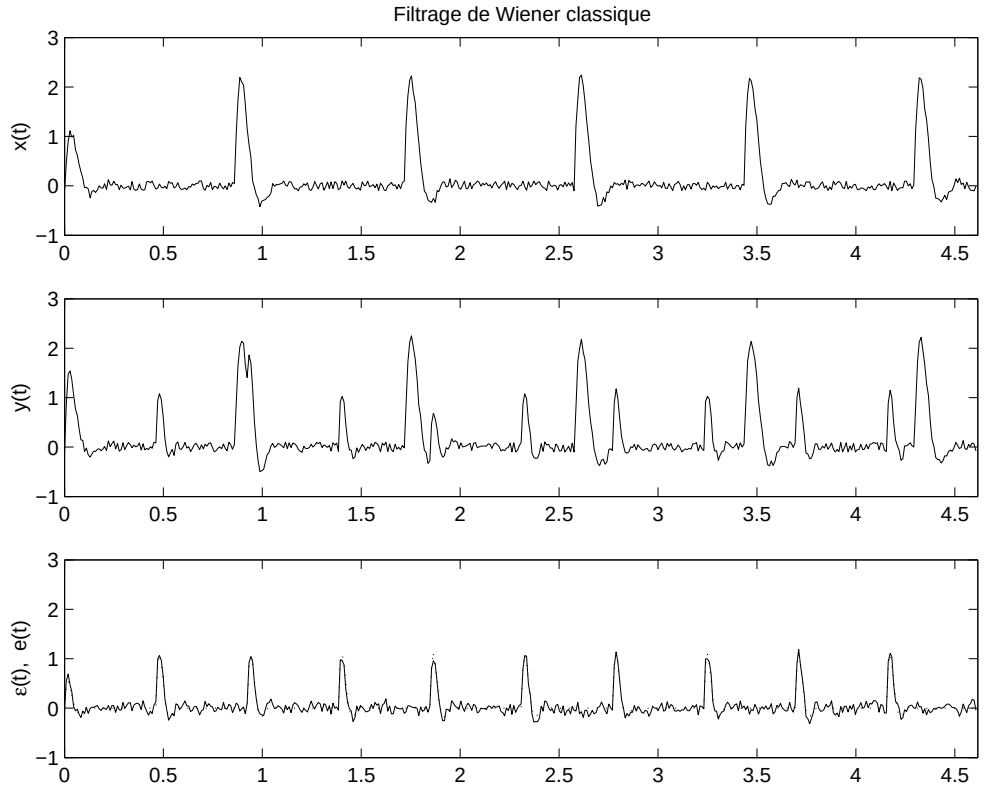

Fig. 8.5.: Suppression d'une perturbation par filtrage de Wiener

8.5. Filtrage adaptatif

Un filtre adaptatif est un système numérique dont les coefficients se modifient eux mêmes en fonction des signaux extérieurs. Il est utilisé chaque fois qu'un environnement est mal connu ou changeant ou pour supprimer des perturbations situées dans le domaine de fréquences du signal utile, ce que les filtres classiques ne peuvent pas faire.

Un filtre adaptatif est constitué de deux parties distinctes :

- un filtre numérique à coefficients ajustables ;
- un algorithme de modification des coefficients basé sur un critère d'optimisation.

8.5.1. Algorithme récursif des moindres carrés (RLMS)

Nous avons vu au paragraphe 8.3.2 que pour trouver les paramètres optimaux, il faut descendre le long d'un paraboloïde afin atteindre le minimum de l'erreur quadratique moyenne. Mathématiquement, cette descente se fait dans le sens opposé à celui du gradient

$$\frac{\partial J}{\partial W} = -2\,r_{xy} + 2\,R_{xx}W$$

```
% les signaux mesures sont stockes dans les vecteurs xt et yt
% initialisation des calculs
  rxx0 = 0 ; rxx1 = 0 ;
  rxy0 = 0 ; rxy1 = 0 ;
% boucle de calculs
  kmax = length(xt)-1 ;
  for n = 1 :kmax
    % lecture des signaux
    xn = xt(n) ;
    yn = yt(n) ;
    xn1 = xt(n+1) ;
    yn1 = yt(n+1) ;
    % correlation
    rxx0 = rxx0 + xn*xn ;
    rxx1 = rxx1 + xn*xn1 ;
    rxy0 = rxy0 + xn*yn ;
    rxy1 = rxy1 + xn*yn1 ;
  end ;
% solution de Wiener-Hopf
  Rxx = [rxx0 rxx1 ;
         rxx1 rxx0]
  rxy = [rxy0 ; rxy1]
  w = inv(Rxx) * rxy
% calcul du signal recherche
  xn1 = 0 ;
  for n = 0 :kmax-1
    xn = xt(n+1) ;
    yn = yt(n+1) ;
    ew(n+1) = yn - [xn, xn1]*w ;
    xn1 = xn ;
  end ;
```

FIG. 8.6.: Exemple de codage d'un filtre de Wiener

et on atteint le point optimum lorsque le gradient s'annule. La valeur des paramètres est alors donnée par la solution

$$W = R_{xx}^{-1}\, r_{xy}$$

De manière heuristique, on imagine bien que cette solution peut être atteinte récursivement en corrigeant les valeurs des coefficients w_k en chaque instant n dans le sens opposé à l'évolution de l'erreur quadratique par rapport au vecteur des coefficients $W(n)$ (figure 8.7) :

$$W(n) = W(n-1) - \frac{\gamma}{2}\left(\frac{\partial \varepsilon^2(n)}{\partial W}\right) \tag{8.52}$$

où γ est un facteur de pondération du gradient.

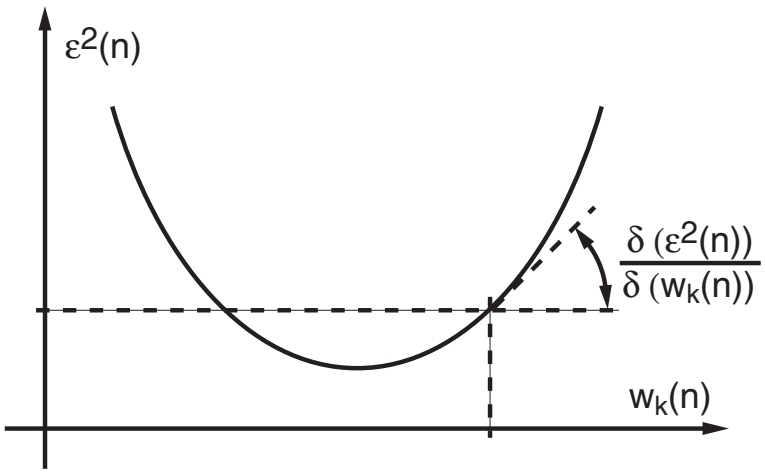

Fig. 8.7.: Erreur quadratique $\varepsilon^2(n)$ en l'instant n et sa dérivée par rapport au coefficient $w_k(n)$

Comme l'erreur quadratique à l'instant n vaut :

$$\varepsilon^2(n) = \left(y(n) - \sum_{i=0}^{p-1} w_i\, x(n-i)\right)^2 = \left(y(n) - X(n)^T\, W\right)^2$$

il vient :

$$\frac{\partial \varepsilon^2(n)}{\partial W} = 2\,\varepsilon(n)\,\frac{\partial \varepsilon(n)}{\partial W} = -2\,\varepsilon(n)\,X(n)$$

On en déduit que la recherche de l'optimum peut se faire avec l'algorithme récursif suivant

$$W(n) = W(n-1) + \gamma\;\varepsilon(n)\,X(n) \tag{8.53}$$

que l'on désigne sous le nom d'algorithme RLMS (*Recursive Least Mean Square*).

Les grandeurs dont on a besoin sont :

– le vecteur des p coefficients à l'instant $n-1$:

$$W(n-1) = [w_0(n-1), w_1(n-1), \cdots, w_{p-1}(n-1)]^T$$

– les p dernières valeurs du signal d'entrée :

$$X(n) = [x(n),\, x(n-1),\, \cdots,\, x(n-p+1)]^T$$

– la valeur du signal de sortie $y(n)$ pour calculer l'écart à l'instant n

$$\varepsilon(n) = y(n) - \sum_{i=0}^{p-1} w_i\, x(n-i) \tag{8.54}$$

– le gain d'adaptation γ de l'algorithme récursif (généralement très inférieur à 1).

La valeur du gain d'adaptation γ est difficile à fixer : si on la choisit trop faible, la convergence vers la valeur optimum est très lente ; si on la choisit trop forte, la convergence se fait en oscillant onguement autour de la valeur optimum ; enfin, si le gain d'adaptation est trop élevé, le processus d'optimisation diverge.

Les avantages de cet algorithme résident dans la simplicité à le déduire, à le programmer, et au peu de calculs à effectuer. Par contre, ses inconvénients sont la lente convergence des paramètres et le risque d'oscillations ou de divergence si le gain d'adaptation est trop grand. Ces inconvénients, associés au fait que les signaux sont généralement non stationnaires, ont nécessité la recherche d'une adaptation automatique du gain.

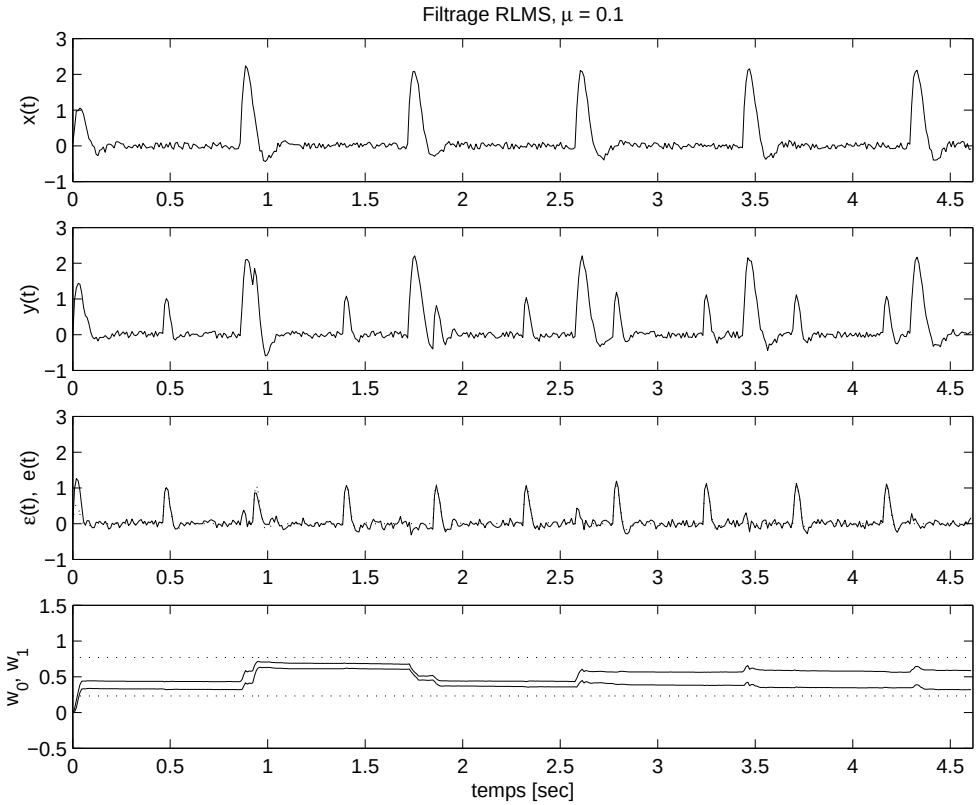

FIG. 8.8.: Filtrage avec l'algorithme RLMS

8.5.2. Algorithme récursif normalisé (NLMS)

En 1975, Widrow et ses coauteurs ont montré qu'un gain d'adaptation stable compris entre 0 et 1 peut être utilisé si on le normalise par le nombre p de paramètres du vecteur W et par la puissance ou variance σ_x^2 du signal d'entrée $x(n)$.

Gain d'adaptation normalisé

Pour la plupart des situations pratiques, on choisit un gain initial $\gamma_0 \simeq 0.1$ qui, après normalisation par le nombre de paramètres et par la variance du signal d'entrée, donne un gain d'adaptation qui évolue en fonction de la puissance du signal d'entrée :

$$\gamma \;=\; \frac{\gamma_0}{p \cdot \sigma_x^2} \tag{8.55}$$

De manière à éviter que le gain n'augmente indéfiniment lorsque la puissance du signal de référence tend vers zéro, on peut corriger le dénominateur du gain en y ajoutant un terme constant $a \ll 1$:

$$\gamma \;=\; \frac{\gamma_0}{a + p \cdot \sigma_x^2} \tag{8.56}$$

L'algorithme s'écrit alors :

$$W(n) = W(n-1) + \frac{\gamma_0}{a + p \cdot \sigma_x^2}\, \varepsilon(n)\, X(n) \tag{8.57}$$

Comme cet algorithme utilise un gain normalisé par la puissance σ_x^2 du signal $x(n)$, il porte le nom d'algorithme NLMS (*Normalised Least Mean Square*).

Puissance moyenne du signal de référence

Dans le cas où le signal $x(n)$ n'est pas stationnaire, on doit évaluer sa puissance $P_x \equiv \sigma_x^2$ en tout instant :

$$P_x(n) = \frac{1}{n+1} \sum_{k=0}^{n} x^2(k)$$

Cette valeur moyenne peut également être évaluée à l'aide d'un filtre passe-bas en oubliant progressivement les anciennes valeurs.

Se souvenant qu'un filtre passe-bas d'ordre 1 et de gain unité, d'entrée $e(n)$ et de sortie $s(n)$ est décrit par sa fonction de transfert

$$H(z) = \frac{S(z)}{E(z)} = \frac{1-\lambda}{1-\lambda\, z^{-1}}, \qquad 0 < \lambda < 1 \tag{8.58}$$

ou, de manière équivalente, par son équation récursive

$$s(n) = (1-\lambda)\, e(n) + \lambda\, s(n-1) \tag{8.59}$$

le calcul de $P_x(n)$ se fait de la manière suivante :

$$P_x(n) = (1 - \lambda)\, x^2(n) + \lambda\, P_x(n-1) \tag{8.60}$$

avec $\lambda = 0.90 \cdots 0.98$ suivant l'horizon de mémoire N désiré

$$N \simeq 3\, K_c = \frac{3}{|\ln(\lambda)|}$$

On montre aisément qu'au-delà de $N = 3/|\ln(\lambda)|$, la contribution des anciennes valeurs est inférieure à 5%. On peut relever que l'horizon de mémoire N vaut 30 pour $\lambda = 0.90$ ou 150 lorsque $\lambda = 0.98$.

Résultats

Les résultats ainsi obtenus sont présentés à la figure 8.9. Ils illustrent à l'évidence la rapidité de la convergence et la qualité des résultats qui est pratiquement aussi bonne que celle obtenue avec le filtrage optimum de Wiener.

La figure 8.10 montre comment le gain change au cours du temps ; on y voit nettement son augmentation lorsque la puissance du signal de référence est faible. Une partie du codage est présenté dans la figure 8.11.

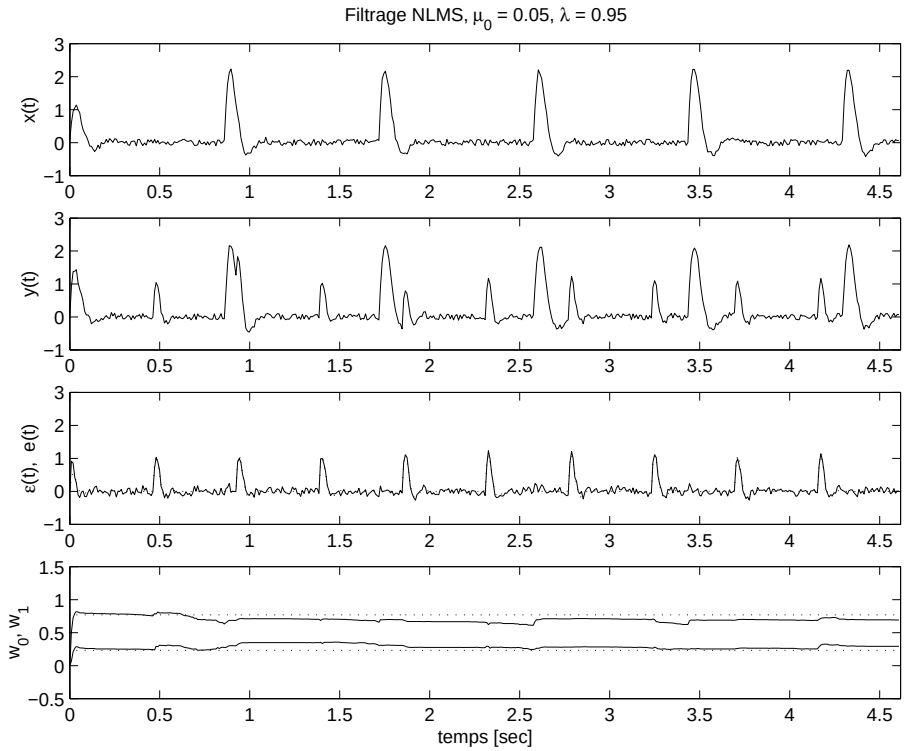

FIG. 8.9.: Filtrage avec l'algorithme NLMS

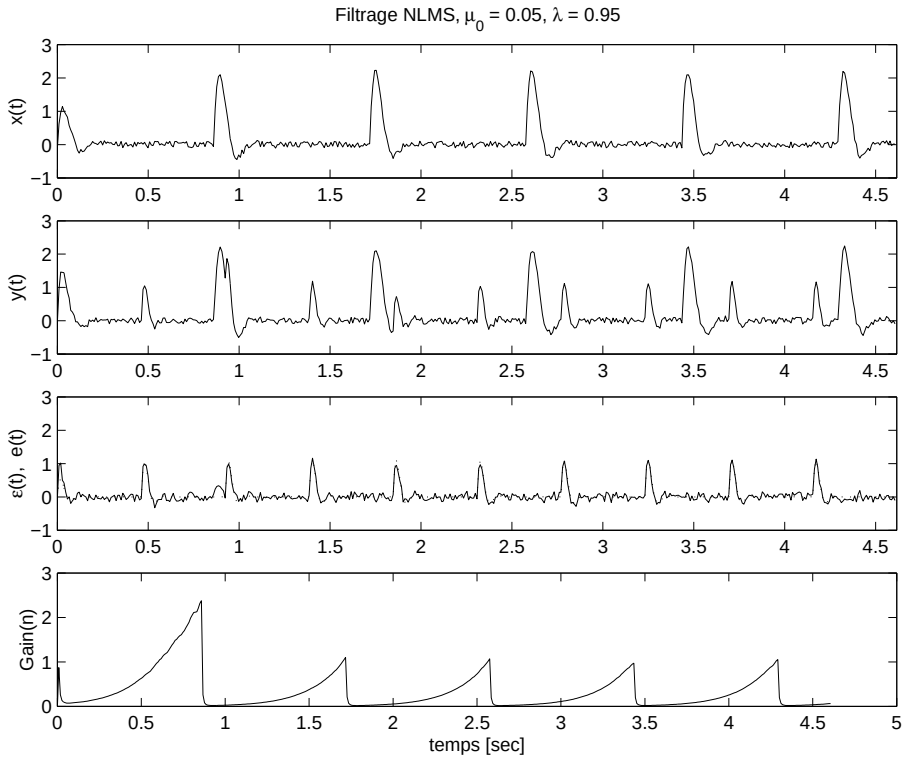

Fig. 8.10.: Évolution du gain de l'algorithme NLMS

```
% constantes
  p = 3 ; % modifier selon les besoins
  gamma0 = 0.1 ;
  lambda = 0.95 ;
% initialisation des calculs
  Wn = zeros(p,1) ;
  Pxx = 0 ;
  a = 1e-3 ;
% boucle de calculs
for n = p-1 :kmax-1
  % signaux a l'instant n
  xn = xt(n+1) ;
  yn = yt(n+1) ;
  % puissance moyenne de x(n)
  Pxx = (1-lambda)*xn^2 + lambda*Pxx ;
  % calcul des paramètres W(n)
  Xn = xt(n+1 :-1 :n-p+2) ;
  en = yn - Xn'*Wn ;
  Wn = Wn + gamma0/(a+p*Pxx) * en*Xn ;
  % memorisation du signal recherche et des parametres
  ew(n+1) = en ;
  wt(n+1, :) = Wn' ;
end ;
```

FIG. 8.11.: Exemple de codage d'un filtre NLMS

8.6. Exercices

RL 1

Considérant l'ensemble des notes n suivantes :

2.8	4.7	3.2	4.2	2.6	4.8	3.5	5.4	5.4	4.4
4.0	5.6	5.3	4.6	5.3	4.6	3.4	3.2	3.4	4.2

1. Calculez la note moyenne et sa déviation standard.
2. Comparez aux résultats obtenus avec les fonctions Matlab `mean`, `var` et `std`.
3. Calculez puis tracez à la main la répartition des notes telles que

$$n < 3, \quad 3 \leq n < 4, \quad 4 \leq n < 5, \quad 5 \leq n \leq 6$$

4. Commentez vos résultats.

RL 2

La mesure de la caractéristique statique d'un amplificateur a donné les résultats suivants :

U_{in} [mV]	-50	-40	-30	-20	-10	0	10	20	30	40	50
U_{out} [V]	-3.69	-3.38	-2.43	-1.68	-0.56	0.17	0.87	1.96	2.45	3.28	3.76

1. Tracez cette caractéristique ; qu'en pensez-vous ?
2. Admettant que l'amplificateur est linéaire, utilisez la régression linéaire (équations de la section 8.2.3) pour calculer à la main son gain et sa tension de décalage. Puis :
   a) Comparez aux résultats fournis par `polyfit`.
   b) Tracez sur un même graphe les points mesurés et la caractéristique linéaire de l'amplificateur avec une abscisse fine ($\Delta U_{in} = 1\,mV$).
   c) Calculez la puissance des écarts entre le modèle et la mesure ?
3. Considérant que l'amplificateur n'est pas parfaitement linéaire, utilisez la fonction `polyfit` pour calculer le polynôme d'ordre 3 (pourquoi 3 et pas 2 ou 4 ?) passant au mieux parmi ces points.
   a) Y a-t-il un sens à donner le gain de l'amplificateur et sa tension de décalage ?
   b) Tracez sur un même graphe les points mesurés et la caractéristique non linéaire de l'amplificateur avec une abscisse fine ($\Delta U_{in} = 1\,mV$).
   c) Calculez la puissance des écarts entre le modèle et la mesure ?
4. Répétez le point 3 avec un polynôme d'ordre 9. Commentez la puissance des écarts et l'allure de ce polynôme par rapport aux modèles précédents.

Prb 1

On s'intéresse ici aux notions de base des statistiques et probabilités. Pour cela :

1. Générez les trois signaux suivants avec $N = 10'000$ et $n = 0 : N-1$,

$$x_u(n) = \text{rand}(size(n)), \quad x_g(n) = 0.5\,\text{randn}(size(n)), \quad x_s(n) = \sin(2\pi\, n\,/N)$$

2. Tracez ces signaux par rapport à n (`subplot(3,1,k)`) ; observez-les avec le zoom ; commentez vos observations.
3. Quels résultats fournissent les fonctions `rand` et `randn` ?
4. Pouvez-vous donner une estimation des valeurs moyennes et variances des trois signaux ?
5. Pour chacun des 3 signaux, calculez leurs valeur moyenne, variance et écart-type ; vérifiez l'équation (8.3) ; commentez.
6. Tracez et comparez les 3 histogrammes (`hist(xn,round(sqrt(N)))`).
7. Commentez ces résultats d'un point de vue statistique ; en particulier, que pensez-vous de la probabilité d'apparition de certaines valeurs ?

Prb 2

Avec l'exercice précédent vous avez compris que l'histogramme permet de compter le nombre de fois n_k où une valeur x se situe dans la case k de largeur

$$\Delta x = \frac{x_{max} - x_{min}}{N_k}, \quad \text{avec} \quad N_k = \text{nombre de cases}$$

La probabilité de se trouver dans une case est donc égale au contenu de la case divisé par le nombre total de points N. On définit ainsi la fonction de probabilité discrète

$$p(k) \equiv p(k\Delta x \leq x < (k+1)\Delta x) = \frac{n_k}{N} \tag{8.61}$$

On peut également considérer la somme cumulée des valeurs de l'histogramme et normaliser son maximum à 1. On obtient alors la fonction de répartition des probabilités définie comme la probabilité que la valeur de x soit inférieure à une valeur donnée $X = K\,\Delta x$. Mathématiquement, cela s'écrit :

$$P(K) \equiv p(-\infty < x \leq K\,\Delta x) = \sum_{k\to-\infty}^{K} p(k) \tag{8.62}$$

Cette approche permet de calculer simplement la probabilité que la valeur de x se situe dans un domaine donné ; on a en effet

$$P(K_1\Delta X < x \leq K_2\Delta X) = \sum_{k=K_1+1}^{K_2} p(k) = P(K_2) - P(K_1) \tag{8.63}$$

Dans le cas d'une variable continue, la fonction de probabilité devient une densité de probabilité définie comme suit

$$p(x) = \lim_{N\to\infty,\,\Delta x\to 0} \left(\frac{n_k}{N\,\Delta x}\right) \tag{8.64}$$

et la fonction de répartition des probabilités s'écrit alors

$$P(x : x' < x) = \int_{-\infty}^{x} p(x)\, dx \tag{8.65}$$

Afin de bien comprendre ce qui précède, je vous propose d'appliquer les points ci-après sur les 3 signaux suivants

- un signal aléatoire $x_u[n]$ à distribution uniforme compris entre -4 et $+4$;
- un signal aléatoire $x_g[n]$ à distribution gaussienne de variance unité ;
- un signal sinusoïdal $x_s[n]$ d'amplitude 4 et de période N ;

où $N = 10'000$ et $n = 0 : N - 1$.

1. Représentez le signal avec `subplot(3,1,1) ; plot(nn,xn)`.
2. Calculez sa valeur moyenne et sa variance ; comparez avec les valeurs théoriques.
3. Calculez son histogramme avec `[nk,xk] = hist(xn,sqrt(N))`.
4. Calculez et représentez la fonction de probabilité $p(k)$ avec `subplot(3,1,2) ; plot(xk,pk)`.
5. Calculez et représentez la répartition des probabilités $P(K)$ avec `PK = cumsum(pk) ; subplot(3,1,3) ; plot(xk,PK)`.
6. Calculez la probabilité de trouver une valeur de x comprise entre -1 et $+1$ (la fonction `find` facilite ce calcul).
7. Analysez et commentez vos graphes et résultats.

Prb 3

Appliquez ce que vous venez de voir à des signaux réels (sons et images). Plus précisément,

1. Chargez le fichier sons : `xt = load('colibri.txt') ;` normalisez sa valeur maximum à 1 : `xt = xt/max(abs(xt)) ;` .
2. Tracez son graphe et écoutez-le avec `wavplay(xt,8000) ;` .
3. Calculez sa valeur moyenne, sa puissance et sa variance.
4. Tracez son histogramme. Quel est le modèle de distribution qui vous paraît approprié pour décrire le signal ?
5. Ajustez les modèles choisis sur la courbe de probabilité réelle. Commentez.
6. Chargez le fichier image : `img = imread('lena256.jpg') ;` visualisez son contenu et transformez la matrice image en un vecteur : `img_vecteur = double(img( :)) ;`
7. Répétez les points 1 à 5 ci-dessus.

Corr 1

Dans le but de vous familiariser avec les résultats de la corrélation, appliquez la fonction Matlab de corrélation `rxy = xcorr(y,x,L,'unbiased')` à chacun des 3 signaux suivants

$$x_c[n] = \sin(2\pi\, n\,/N), \quad x_q[n] = \text{square}(2\pi\, n\,/N), \quad x_g[n] = \text{randn}(size(n))$$

où $n = 0 : 1000$ et $N = 50$. Pour ce faire :

1. calculez puis tracez les 3 signaux dans une fenêtre ;
2. calculez puis tracez les 3 fonctions d'autocorrélation dans une nouvelle fenêtre ;
3. quelle est l'utilité du paramètre L ?
4. vous souvenant que $r_{xx}[0] = P_x$, que doit valoir le maximum de chaque autocorrélation ?
5. calculez et tracez l'intercorrélation entre $x_c[n]$ et $x_q[n]$ (prenez garde à l'ordre des signaux `rcq = xcorr(xq,xc,L,'unbiased')`).

WH 1

Considérant un filtre MA causal décrit par ses coefficients $w[0 \cdots 2] = [3, 2, 1]$,

1. dessinez le schéma fonctionnel de ce filtre en prenant $x[n]$ et $y[n]$ comme signaux d'entrée et de sortie ;
2. montrez que, si on lui applique le signal $x(n) = [+1, -1, +1, -1, +1]$, la sortie vaudra $y[n] = [+3, -1, +2, -2, +2]$;
3. pour faire ce calcul, quelle modification implicite avez-vous appliqué à $x[n]$?
4. calculez $y[n]$ avec Matlab.

WH 2

Ayant appliqué le signal $x[n] = [+1, +1, -1, +1, -1, 0]$ à un système MA décrit par l'équation

$$y[n] = w_0 x[n] + w_1 x[n-1]$$

on a obtenu en sortie le signal $y[n] = [+2, +3, -1, +1, -1, -1]$. Utilisez le filtrage de Wiener pour trouver les paramètres w_0, w_1 sans Matlab. Pour ce faire :

1. Calculez à la main les fonctions de corrélation $r_{xx}([k]$ et $r_{xy}[k]$ pour k compris entre –5 et +5.
2. Quelles sont les valeurs de corrélation dont vous avez besoin pour résoudre ce problème ?
3. Écrivez les vecteurs et matrices r_{xx}, r_{xy}, R_{xx} nécessaires pour le filtrage de Wiener.
4. Résolvez le système $R_{xx}\, W = r_{xy}$.
5. A-t-on besoin de 5 couples de valeurs (x, y) pour trouver w_0 et w_1 ? Justifiez et vérifiez votre réponse.
6. Quel est l'avantage d'avoir un grand nombre de valeurs (x, y) ?

WH 3

Résolvez le problème précédent avec l'aide de Matlab. Pour ce faire, utilisez

1. la fonction `rxy = xcorr(y,x,p-1)` où p est le nombre de paramètres (prenez garde à l'ordre des vecteurs `x` et `y`) ;
2. la fonction `Rxx = toeplitz(rxx)` pour remplir la matrice de corrélation R_{xx}.

WH 4

Dans ce qui suit on souhaite extraire un signal inconnu $s(n)$ fortement perturbé par le réseau électrique de fréquence 50 Hz. Pour ce faire, on a mesuré simultanément le signal du réseau $x(n)$ et le signal bruité $y(n)$ en les échantillonnant à la fréquence f_e de 10 kHz. Pour résoudre ce problème,

1. chargez les signaux contenus dans le fichier `xy50hz.txt` :
```
signaux = load('xy50hz.txt') ;
xn = signaux( :,1) ;
yn = signaux( :,2) ;
N = length(xn) ;
nn = 0 :Te :(N-1)*Te ;
```
2. tracez $x(n)$ et $y(n)$;
3. dessinez le schéma de Wiener ; où se trouve le signal $s[n]$?
4. recherchez $s[n]$ en appliquant l'algorithme de Wiener-Hopf avec 2 paramètres ;
5. augmentez le nombre de paramètres p ; observez leurs valeurs et la puissance de $s[n]$; concluez ;
6. calculez le rapport signal sur bruit S_{eff}/Y_{eff}.

LMS 1

Il est possible d'évaluer en temps réel la puissance moyenne d'un signal en "oubliant" progressivement les valeurs anciennes. Ceci peut se faire de la manière suivante :

$$P_x[n] = (1-\lambda)\, x^2[n] + \lambda\, P_x[n-1]$$

1. Montrez que cet algorithme est l'équivalent d'un filtre passe-bas d'ordre 1. Pour cela :
   a) dessinez son schéma fonctionnel ;
   b) calculez sa fonction de transfert ;
   c) que valent l'instant caractéristique et l'horizon de mémoire à 5% ?
2. Imaginez (sans l'aide de Matlab !) un signal $x(t)$ composé d'une sinusoïde d'amplitude $A_1 = 1\,V$, de fréquence $f_0 = 1000\,Hz$ et de durée $t_{max} = 0.1\,sec$ suivi de la même sinusoïde d'amplitude $A_2 = 2\,V$. Comment évolue la puissance de ce signal ? Que vaut-elle ?
3. Générez le signal $x[n]$; quelle fréquence d'échantillonnage prenez-vous ?

4. Calculez sa puissance moyenne avec l'algorithme ci-dessus en prenant $\lambda = 0.95$ et $\lambda = 0.99$; quel est l'horizon de mémoire à 5% correspondant à ces deux valeurs ?
5. Tracez $P_x[n]$; concluez.

LMS 2

Dans ce problème, on souhaite diminuer le bruit environnant lors d'une conversation téléphonique en utilisant un deuxième microphone placé sur le côté extérieur du téléphone. Le microphone de base capte le message entaché du bruit environnant, alors que le deuxième capte seulement le bruit. Grâce à l'algorithme LMS, il est possible d'améliorer sensiblement la qualité du message.

Pour le vérifier :

1. Dessinez le schéma de Wiener correspondant à ce problème et précisez quels sont les signaux en présence.
2. Le fichier `bjrbruit.dat` constitué de trois colonnes contient trois signaux enregistrés à la fréquence $f_e = 8\,kHz$, à savoir, le bruit (microphone extérieur = $x[n]$), le message perturbé par le bruit (microphone-bouche = $y[n]$) et, dans un but de comparaison, le message non bruité. De ce fichier, extrayez les deux premiers signaux ; tracez-les et écoutez-les avec la fonction `soundsc(signal,fe)`.
3. Appliquez l'algorithme NLMS sur les deux premiers signaux pour diverses longueurs p du vecteur W.
4. Pour $p = 5$ et $p = 20$ par exemple, tracez les valeurs asymptotiques des composantes de $W(n \to \infty)$. Qu'en pensez-vous ?
5. Tracez le signal fourni par NLMS et l'évolution des paramètres $W[n]$.
6. Observez les spectrogrammes des trois signaux (`specgram(signal,128,fe)`) ; commentez vos observations.
7. Écoutez le signal extrait ; qu'en pensez-vous ?
8. Le temps de calcul sur un PC 500 MHz est prohibitif (environ quinze fois la durée du message). Sachant qu'en un cycle d'horloge, un DSP réalise une multiplication et une addition, pensez-vous qu'il soit possible de faire ces calculs en temps réel avec un DSP dont le temps de cycle est de 20 ns ? Justifiez votre réponse.

Bibliographie

[1] B. Widrow, S.D. Stearns : *Adaptive Signal Processing Algorithms*, Prentice Hall, 1985.

[2] S.D. Stearns, R.A. David : *Signal Processing Algorithms in Matlab*, Prentice Hall, 1996.

[3] E.C. Ifeachor, Q.W. Jervis : *Digital Signal Processing, A Practical Approach*, Addison Wesley, 1993.

[4] S. Haykin : *Adaptive Filter Theory*, Prentice Hall, 1991.