15 Codage et décodage LPC de la parole

15.1 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.

15.2 Prédiction linéaire

15.2.1 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 15.1).

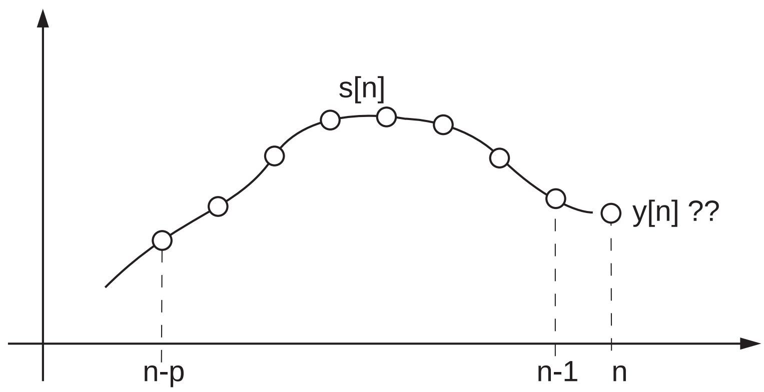

Fig. 15.1: Les échantillons $s[n-p]$ à $s[n-1]$ sont utilisés pour estimer la valeur à venir

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 :

$$y[n] = -\left(a_1 s[n-1] + a_2 s[n-2] + \cdots + a_p s[n-p]\right) = -\sum_{k=1}^{p} a_k s[n-k] \qquad (15.1)$$

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 $y[n]$:

$$e[n] = s[n] - y[n] = s[n] + \sum_{k=1}^{p} a_k s[n-k] \tag{15.2}$$

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 :

$$\sigma_e^2(a_k) = \frac{1}{N}\sum_{n=0}^{N-1} e^2[n] = \frac{1}{N}\sum_{n=0}^{N-1}\left(s[n] + \sum_{k=1}^{p} a_k s[n-k]\right)^2 \tag{15.3}$$

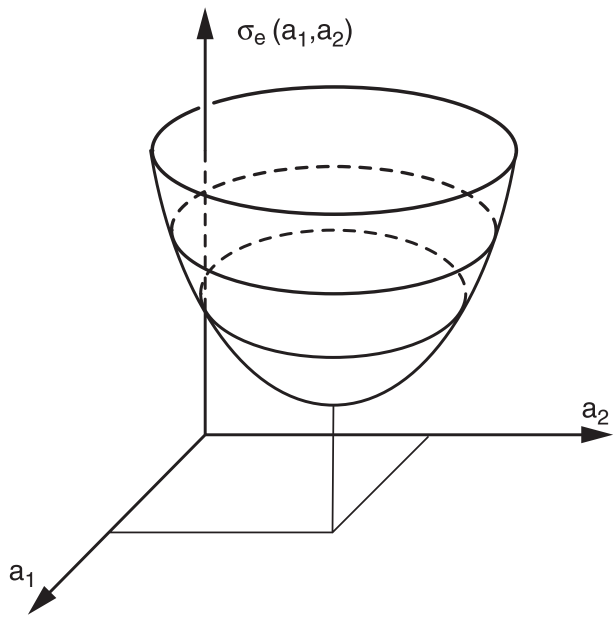

Fig. 15.2: Variance de l'erreur de prédiction

15.2.2 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 15.3.

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

$$\sigma_e^2 = \sigma_e^2(a_1, a_2, \ldots, a_p) = \sigma_e^2(a_k) \tag{15.4}$$

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

$$\sigma_{e,min}^2 \ \Rightarrow\ \frac{\delta\sigma_e^2(a_k)}{\delta a_k} = 0,\ k = 1, \ldots, p \tag{15.5}$$

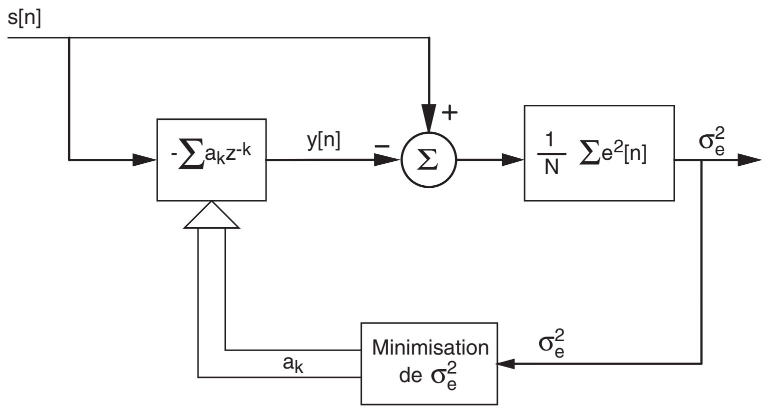

FIG. 15.3: Schéma fonctionnel de la prédiction linéaire

Le calcul de ces p dérivées partielles conduisent à p équations pour les p paramètres inconnus a_k (voir annexe 15.9) :

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

avec :

$$r_{ss}[m] = \sum_{n=0}^{N-1} s[n]s[n-m], \; m = 1, \ldots, p \tag{15.6}$$

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 :

$$\mathbf{R}_{ss}\, \mathbf{a} = -\mathbf{r}_{ss} \tag{15.7}$$

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 $\mathbf{R}_{ss}$:

$$\mathbf{a} = -\mathbf{R}_{ss}^{-1}\, \mathbf{r}_{ss} \tag{15.8}$$

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].

15.2.3 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 :

$$e[n] = s[n] - y[n] = s[n] + \sum_{k=1}^{p} a_k s[n-k] \tag{15.9}$$

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

$$E(z) = S(z)\left(1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_p z^{-p}\right) = S(z)A(z) \tag{15.10}$$

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,

$$S(z) = E(z)\frac{1}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_p z^{-p}} = E(z)\frac{1}{A(z)} \tag{15.11}$$

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.

15.3 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 15.4) :
- 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.

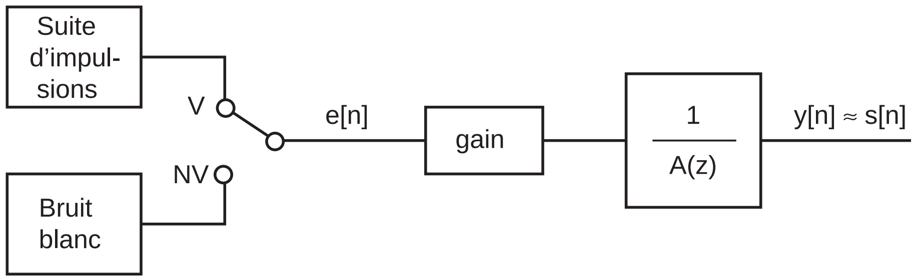

FIG. 15.4: Modèle du conduit vocal

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 15.5). Généralement, ce décalage est de 10 msec (80 échantillons).

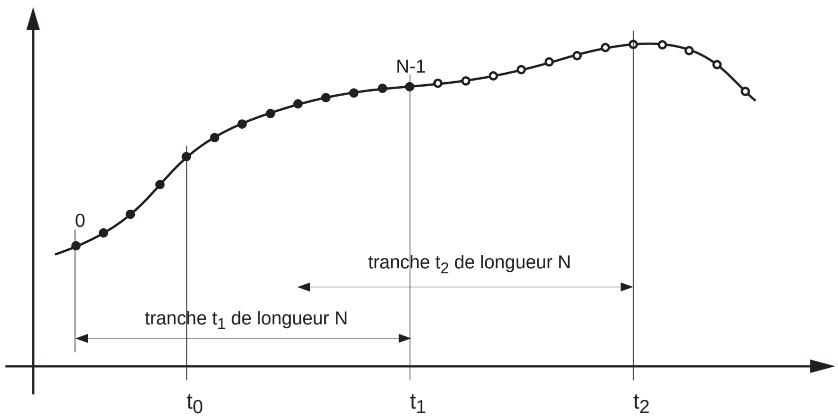

FIG. 15.5: Découpage en tranches t_k du signal $s[n]$

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.

15.4 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.

15.4.1 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
```

15.4.2 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 15.6 et 15.7.

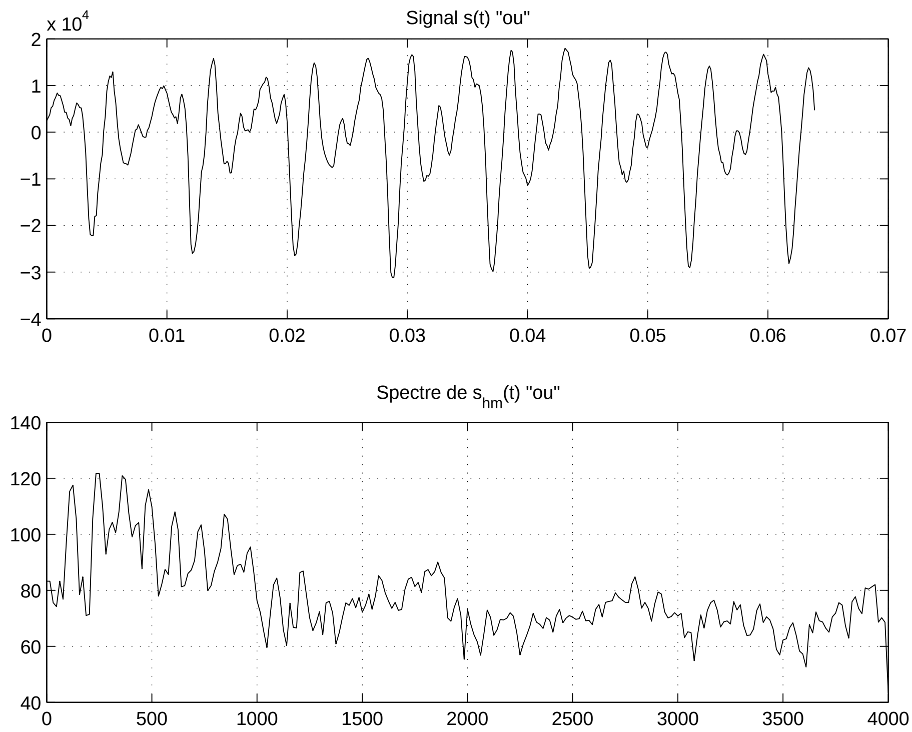

FIG. 15.6: Signal voisé et son spectre

15.5 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) ;
```

15.5.1 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
```

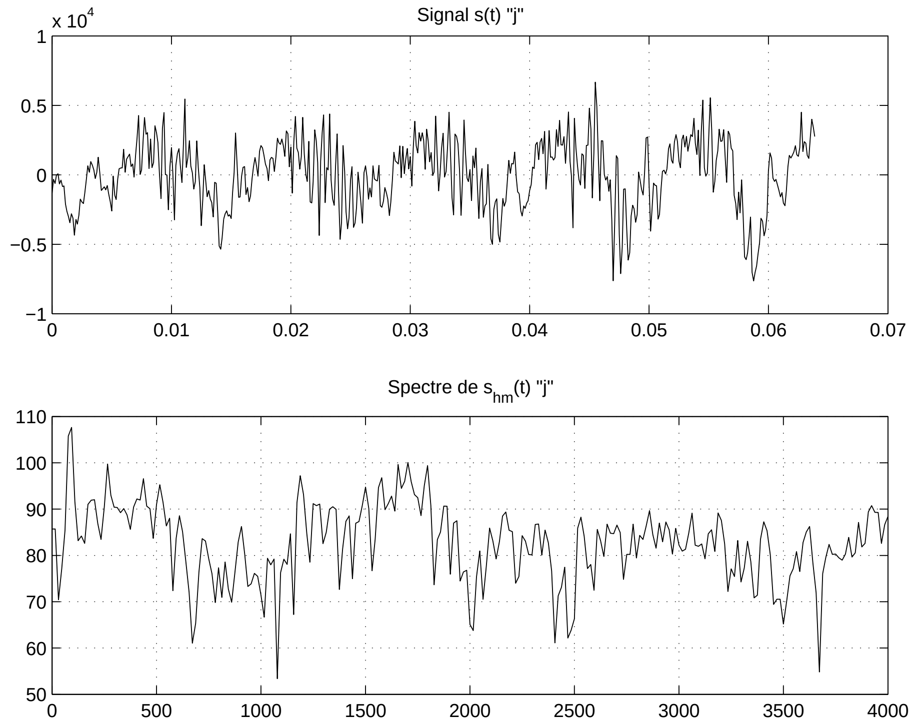

FIG. 15.7: Signal non voisé et son spectre

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

```
gain = Seff ;
```

15.5.2 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 :

$$H(z) = \frac{gain}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_p z^{-p}} \tag{15.12}$$

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 15.8) s'obtiennent avec les commandes suivantes :

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

15.5.3 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 15.9) :

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

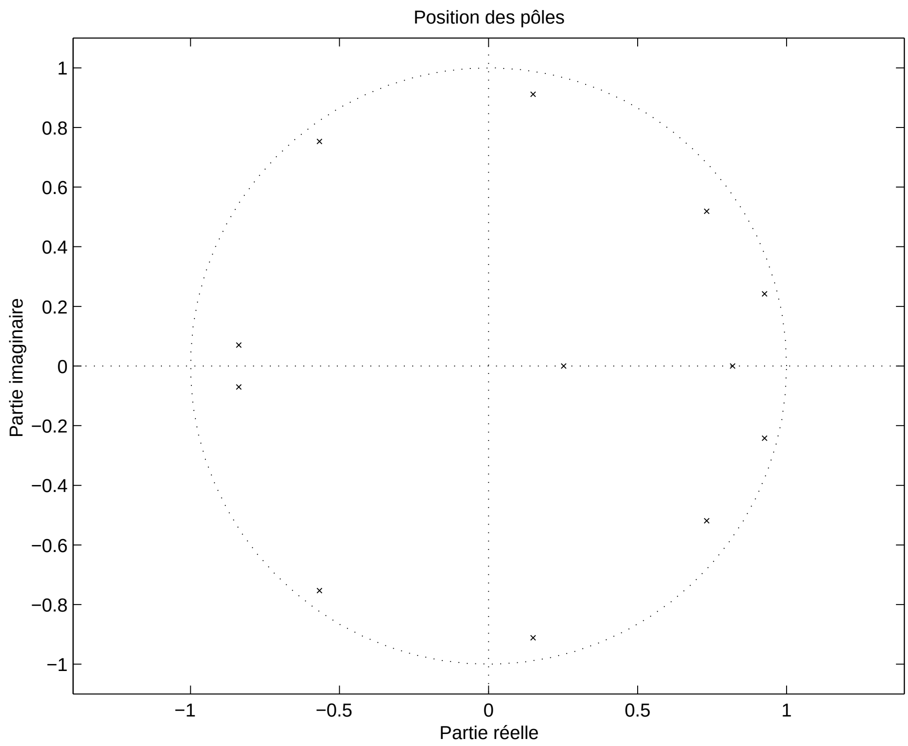

FIG. 15.8: Pôles de la fonction de transfert représentant le conduit vocal

15.6 Recherche du pitch

15.6.1 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) ;
```

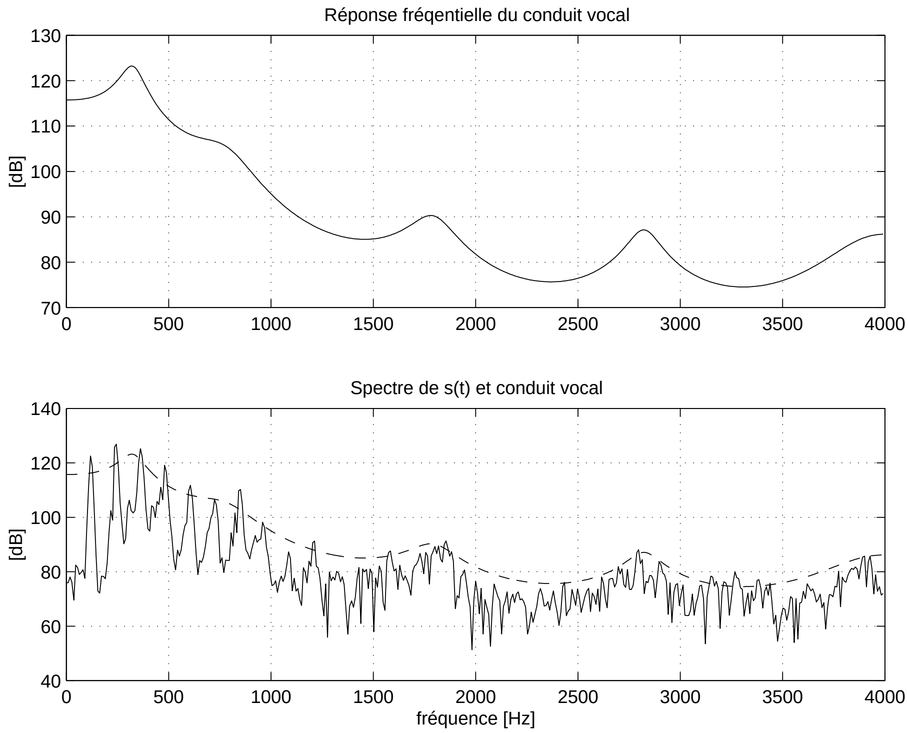

Fig. 15.9: Réponse fréquentielle du conduit vocal et spectre du signal

15.6.2 Recherche du signal d'excitation $e[n]$

Nous avons vu dans la section 15.2.3 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 :

$$H(z) \equiv \frac{S(z)}{E(z)} = \frac{1}{A(z)} \tag{15.13}$$

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

$$E(z) = A(z)S(z) \tag{15.14}$$

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

$$e[n] = a[n] \otimes s[n] \tag{15.15}$$

Dans MatLab, cela s'écrit simplement

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

15.6.3 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 15.10).

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 ;
```

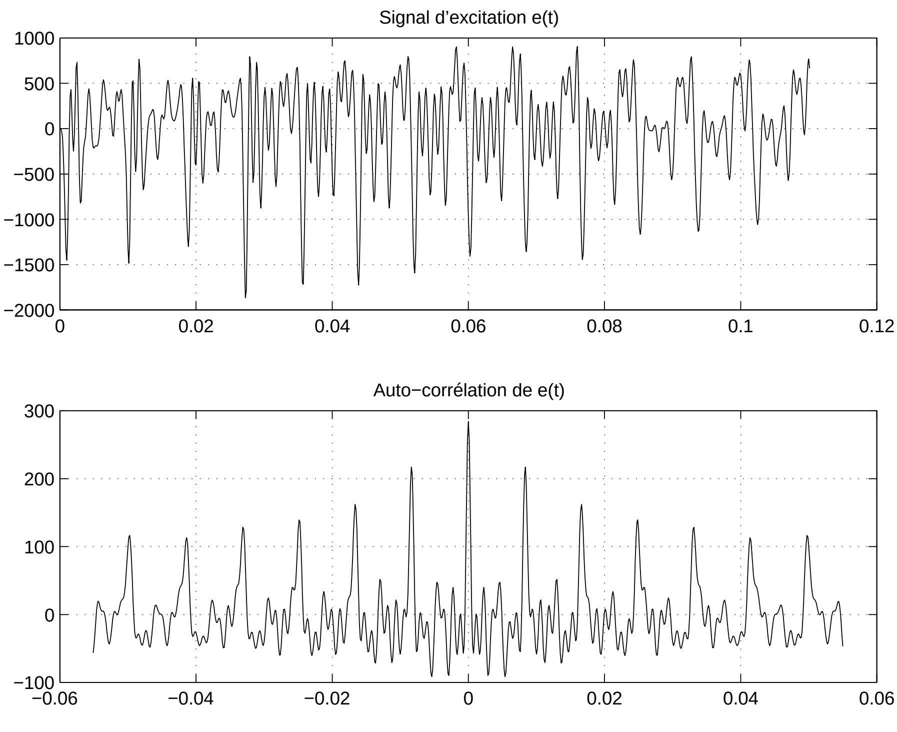

FIG. 15.10: Résidus de la prédicion linéaire et autocorrélation (son voisé)

15.6.4 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 ;
```

15.7 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) ;
```

15.7.1 Signaux réel et synthétique

La figure 15.11 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.

15.7.2 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 15.12 et 15.13 illustrent les résultats du codage et décodage de la phrase "*Comment allez-vous ?*".

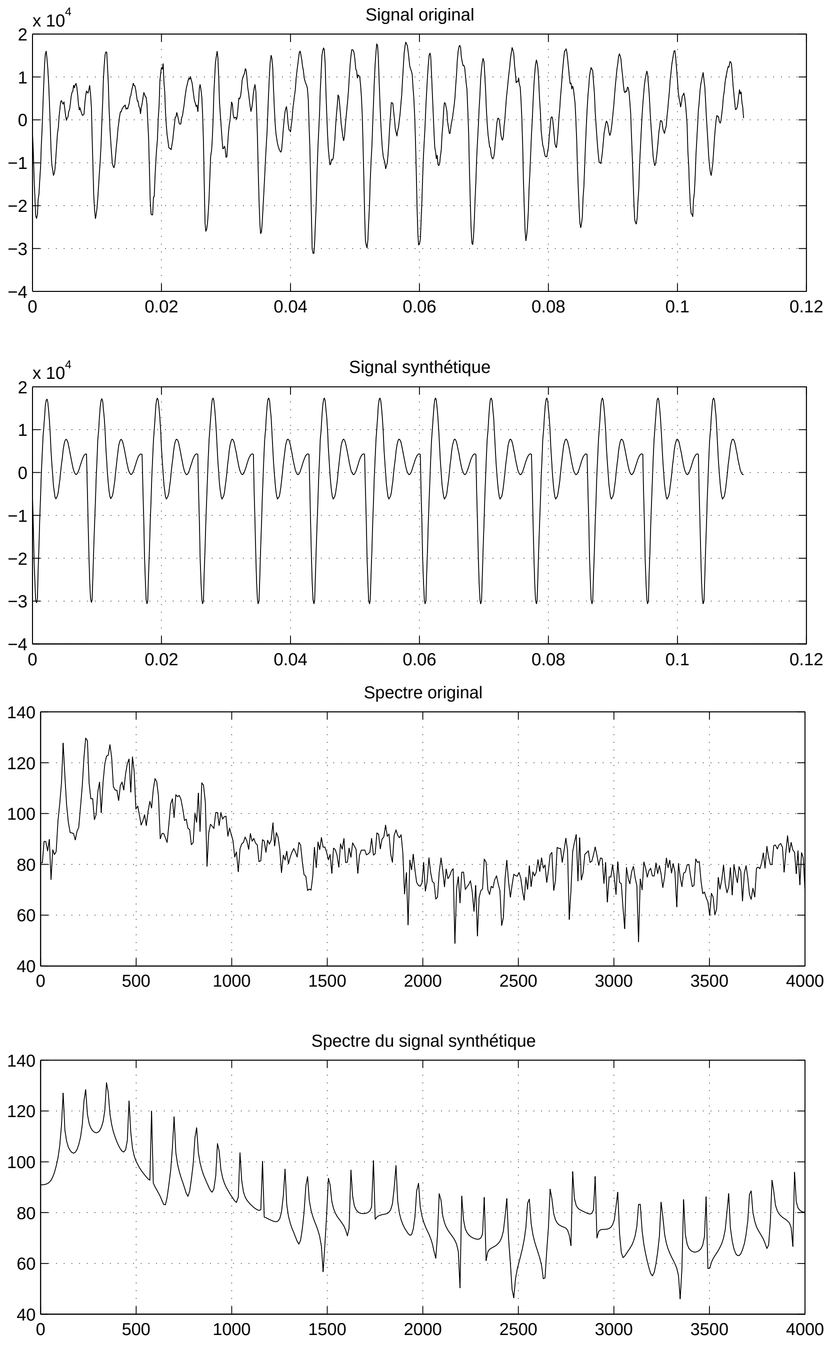

Fig. 15.11: Signaux réel et synthétique avec leurs spectres respectifs

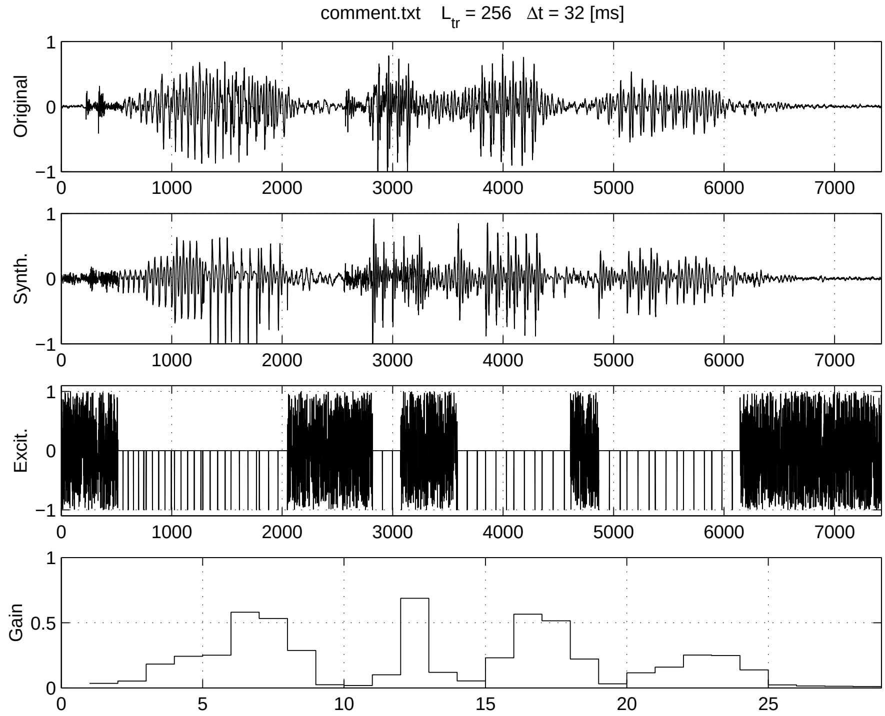

FIG. 15.12: Signaux, excitation et gain

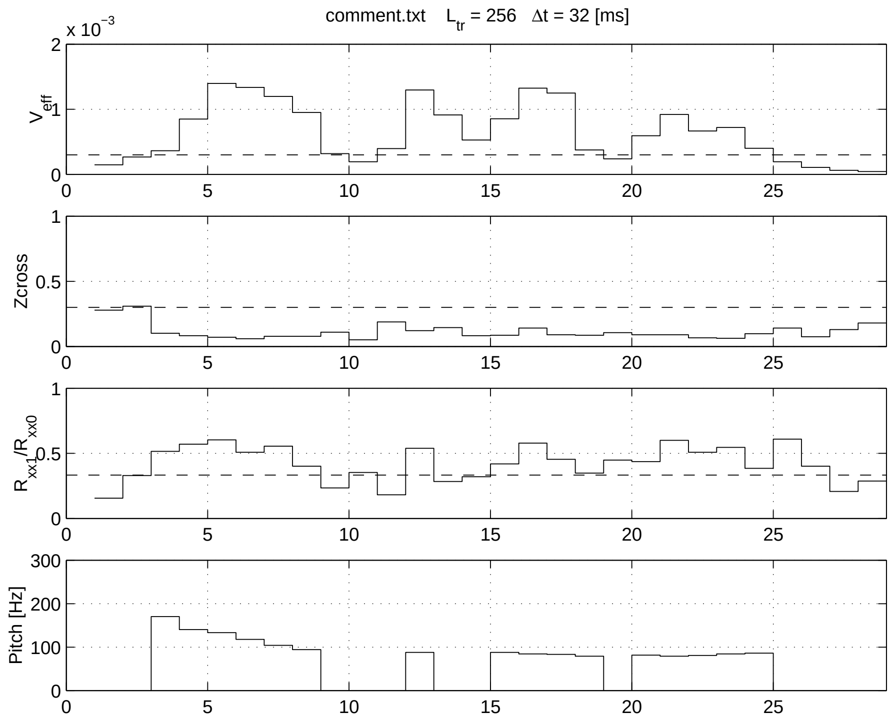

FIG. 15.13: Évolution des paramètres

15.8 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.

15.8.1 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.

15.8.2 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 ?

15.8.3 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].

15.9 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 :

$$\sigma_e^2(a_k) = \frac{1}{N}\sum_{n=0}^{N-1} e^2[n] = \frac{1}{N}\sum_{n=0}^{N-1}\left(s[n] + \sum_{k=1}^{p} a_k s[n-k]\right)^2 \tag{15.16}$$

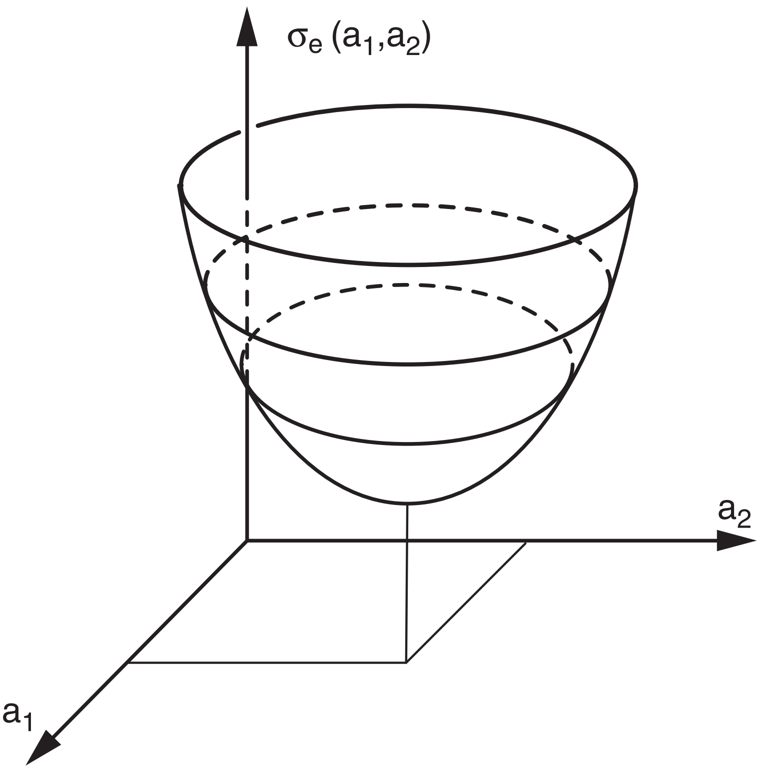

Fig. 15.14: Variance de l'erreur de prédiction

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

$$\sigma_e^2 = \sigma_e^2(a_1, a_2, \ldots, a_p) = \sigma_e^2(a_k) \tag{15.17}$$

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 :

$$\frac{\delta\sigma_e^2(a_k)}{\delta a_k} = 0,\ k = 1, \ldots, p \tag{15.18}$$

La dérivation de l'équation (15.16) 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

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

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 :

$$r_{ss}[k] = \sum_{n=0}^{N-1} s[n]s[n-k],\ k = 1, \ldots, p \tag{15.19}$$

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 :

$$\mathbf{R}_{ss}\,\mathbf{a} = -\mathbf{r}_{ss} \tag{15.20}$$

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 $\mathbf{R}_{ss}$:

$$\mathbf{a} = -\mathbf{R}_{ss}^{-1}\,\mathbf{r}_{ss} \tag{15.21}$$

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