CODEUR MULTI-IMPULSIONNEL AVEC PREDICTION VECTORIELLE A LONG TERME: UN ALGORITHME, UNE PROCEDURE DE CODAGE, L'APPORT DU LANGAGE ADA

N.MOREAU*, P.DYMARSKI**, J.G.FRITSCH***

* ENST 46 rue Barrault 75634 PARIS Cedex 13
** Ecole Polytechnique de Varsovie, 15 rue Nowowiejska VARSOVIE
*** TELIC 206 route de Colmar 67023 STRASBOURG, CRIN BP 239 54506 Vandoeuvre les Nancy Cedex

RESUME

Le principe du codeur MP dans une boucle de prédiction à long terme est similaire à celui du codeur APC mais le traitement s'effectue par bloc: l'algorithme MP joue le rôle de quantificateur et le prédicteur a un caractère vectoriel.
On propose trois modifications de l'algorithme de base: on décale la fenêtre d'analyse MP par rapport à la prédiction à long terme pour placer proprement les impulsions, on place des impulsions en dehors de la fenêtre sans en tenir compte lors du codage dans un but d'homogénéité, on minimise la distance entre les signaux perceptuels originaux et reconstruits pour évaluer les paramètres du prédicteur à long terme.
On propose deux techniques de codage des paramètres du filtre: un codage scalaire calculé qui s'affranchit des tables de codage, une combinaison de codages scalaires et vectoriels.
On montre l'intérêt du langage ADA pour valider tous les choix liés à l'implantation sur machine.

SUMMARY

The principle of a MPLP coder with pulse estimation within the prediction loop is very similar to an AP coder except for a blockwise treatment in MPLPC vs a sample by sample treatment in APC.
With this design of MPLPC, the quantifier is the MP algorithm and the predictor is a vectorial one.
Three improvements to the MPLPC algorithms are proposed.
Two techniques for coding the filter parameters are described.
The interest of ADA as a programming language for such applications is shown.

1. INTRODUCTION

Coder de la parole à moyen débit, avec une qualité de reconstruction quasi téléphonique et avec un coût calcul pas trop élevé, est un problème qui est l'objet de nombreuses études car cela a des applications variées en transmission, en communication avec des radio-mobiles, en messagerie vocale, ... Il existe de nombreuses méthodes. On peut faire, par exemple, de la quantification vectorielle directement sur le signal ou sur le résiduel, du codage prédictif adaptatif, du codage à excitation vocale, à bande de base, en sous-bandes, du codage par excitation multi-impulsionnelle, du codage stochastique, du codage harmonique.

La détermination d'un algorithme de codage est le résultat d'un grand nombre de compromis entre:
- le débit (on veut, bien entendu, qu'il soit le plus faible possible),
- la qualité (on désire une qualité quasi téléphonique avec toute l'ambiguité que représente cette proposition),
- la complexité du traitement en terme de nombre de multiplications/ accumulations par seconde (difficile de dépasser, avec la technologie disponible actuellement, quelques millions de multiplications/ accumulations par seconde), en terme de degré de parallélisme si l'optique retenue est l'intégration de cet algorithme dans un circuit spécialisé, en terme de mémorisations RAM et ROM nécessaires.

En fonction de l'application visée, d'autres contraintes peuvent intervenir:
- la robustesse vis-à-vis des erreurs de transmission,
- les retards introduits par les codeurs par blocs si la synthèse est faite simultanément avec l'analyse.

Ramenées à ces critères, les méthodes précédentes ne sont pas équivalentes. Celles qui sont issues des techniques paramétriques de traitement du signal sont particulièrement intéressantes. Elles se donnent pour objectif d'obtenir des modèles de représentation, c'est à dire de représenter un signal quelconque (ici un signal de parole) comme la sortie d'un système soumis à une entrée convenable. Si les méthodes produisant un modèle du système sont maintenant bien établies (modélisation par prédiction linéaire), il n'en est pas de même pour le choix de l'entrée. Il s'agit là d'un problème, dit de déconvolution, dont la solution est mal conditionnée.

Parmi ces techniques paramétriques, celle qui consiste à approximer l'entrée du filtre de synthèse par un nombre faible d'impulsions judicieusement placées joue un rôle important. Etant donnée toute la littérature consacrée à cette technique de codage depuis 1982, on ne rappelle ici ni son principe, ni sa terminologie propre, ni les notations conventionnelles. On esquisse simplement une classification des différents algorithmes.

La 1er distinction à faire est l'existence ou non d'une prédiction à long terme. Les algorithmes multi-impulsionnels sans prédiction de pitch se séparent eux-mêmes en deux groupes selon que l'on fait apparaître explicitement dans le critère à minimiser soit le signal perceptuel (ref 1) soit le signal résiduel (ref 2). Dans le 1er cas, on est amené à calculer l'inter-corrélation entre le perceptuel et la réponse impulsionnelle de $1/A(z/\gamma)$, dans le 2eme cas, l'inter-corrélation entre le résiduel et l'auto-corrélation normalisée de la réponse impulsionnelle. Pour chacun de ces deux groupes, une 3eme distinction est possible selon le mode de réactualisation du critère. Dans la procédure itérative de détermination des impulsions, on peut enlever la contribution de l'impulsion au signal perceptuel (résiduel) ou directement à l'inter-corrélation. Pour toutes ces variantes des algorithmes sans prédiction de pitch, le résultat est strictement identique. Seule la complexité du traitement est modifiée. Une 4eme distinction est encore possible. Si le critère à minimiser est construit strictement sur l'intervalle d'analyse LPC (0,N-1), on peut parler alors, en analogie avec la terminologie propre à l'analyse LPC, de méthode de covariance. Si l'intervalle est $(0,+\infty)$, en pratique l'intervalle (0,N+M-2) où M représente le nombre d'échantillons non nuls de la réponse impulsionnelle, on peut parler de méthode d'auto-corrélation.
La succession des minimisations partielles ne donnant pas la solution optimale globale, de nombreuses variantes sont encore possibles. La plus classique consiste à réaliser une

minimisation globale uniquement sur les amplitudes, une fois les positions déterminées. On peut même inclure cette optimisation globale à chaque itération. Le coût en complexité de calcul est important sans résultat significatif sur la qualité.
On ne parlera pas ici des nombreuses variantes supplémentaires liées essentiellemnt au codage des positions des impulsions qui entraine un débit prohibitif si on ne s'impose pas certaines contraintes. On peut citer, par exemple, la référence 3.

2. ALGORITHMES MULTI-IMPULSIONNELS AVEC PREDICTION DE PITCH

2.1. Notations

A(z) est le filtre d'analyse, h_n est la réponse impulsionnelle du filtre 1/A(z), les minuscules dénotent les signaux, les majuscules leurs transformés en Z. Pour faciliter l'analyse dans la 1ere partie, admettons que le facteur perceptuel γ=1.

2.2. Principe

Pour la parole voisée, il y a une corrélation significative de l'excitation multi-impulsionnelle. Le signal d'excitation est quasi-périodique et contient des impulsions de grande amplitude. En appliquant une prédiction à long terme, il semble possible de réduire l'amplitude de ces impulsions et leur nombre.

L'algorithme MP sans prédiction de pitch (fig 1) consiste à approximer le signal résiduel $r_n \leftrightarrow R$ par le signal multi-impulsionnel $e_n \leftrightarrow E$ en minimisant la distance entre les signaux $r_n * h_n \leftrightarrow R/A$ et $e_n * h_n \leftrightarrow E/A$. On place des impulsions en minimisant l'énergie du signal:

$$\frac{R}{A} - \frac{E}{A} = \frac{S \cdot A}{A} - \hat{S} = S - \hat{S}$$

i.e. la distance entre le signal original S et le signal synthétique $\hat{S}$.

Les principales structures des codeurs avec prédiction de pitch sont données fig 2 et 3. La structure donnée fig 2 est classique. Par contre la structure donnée fig 3 est plus originale. Elle a été proposée par Lançon (ref 4) et par Ozawa, Araseki (ref 5). Le prédicteur à long terme est caractérisé par un coefficient b et un décalage P (pour la parole voisée, P est égal à la période du fondamental ou à un multiple).
Si le prédicteur est placé en cascade (fig 2), on obtient le résiduel du pitch R' par filtrage:

$$R' = R \cdot (1-B) = S \cdot A \cdot (1-B) \quad \text{avec} \quad B = b \cdot z^{-P}$$

On applique l'algorithme MP au signal R' ce qui entraine le signal multi-impulsionnel V. Au niveau de la synthèse on obtient le signal de sortie:

$$\hat{S} = \frac{E}{A} = \frac{V}{(1-B) \cdot A}$$

On construit le signal V en minimisant l'énergie du signal:

$$\frac{R'}{A} - \frac{V}{A} = S \cdot (1-B) - \hat{S} \cdot (1-B) = (1-B) \cdot (S - \hat{S})$$

Cette formule montre l'inconvénient principal de la méthode. On voudrait minimiser la distance entre les signaux S et $\hat{S}$; en fait on minimise la distance entre les signaux pondérés $S \cdot (1-B)$ et $\hat{S} \cdot (1-B)$. C'est pourquoi on n'obtient pas de bons résultats suivant un critère objectif: le rapport signal sur bruit est généralement inférieur à celui fourni par le codeur sans prédiction de pitch. Toutefois, suivant un critère subjectif, la qualité de la parole synthétique est jugée supérieure. En effet le signal d'entrée R' du bloc MP est moins redondant que le signal résiduel R et plus facile à remplacer par le signal multi-impulsionnel V. Le signal résiduel reconstitué à la synthèse E contient beaucoup plus d'impulsions que le signal transmis V et ressemble d'avantage au "vrai" signal résiduel R. C'est pourquoi, malgré le mauvais RSB, la qualité du signal $\hat{S}$ est jugée satisfaisante.

Etudions maintenant le codeur MP placé dans une boucle de prédiction à long terme (fig 3). Le signal d'entrée du codeur MP est le signal résiduel R auquel on a enlevé sa prédiction X:

$$R'' = R - X = R - V \frac{B}{1-B}$$

On applique l'algorithme MP, i.e. on remplace R'' par $V = \hat{S} \cdot (1-B) \cdot A$, en minimisant l'énergie du signal:

$$\frac{R''}{A} - \frac{V}{A} = \frac{R}{A} - \frac{V}{A} \frac{1}{1-B} = S - \hat{S}$$

i.e. la distance entre le signal original et le signal synthétique. On obtient la même expression que dans le cas du codeur sans prédiction de pitch mais on a réduit le niveau d'erreur. En effet, pour le codeur sans prédiction de pitch, $S - \hat{S} = (R - E)/A$. Pour le codeur avec prédiction de pitch $S - \hat{S} = (R'' - V)/A$. On profite du fait que le niveau de R'' - V est plus bas que le niveau de R-E. Le RSB est donc amélioré.

2.3. Algorithme de base

Le signal de parole est divisé en fenêtres de N échantillons. Chaque fenêtre longue Θ, utilisée pour l'analyse LPC, est divisée en L sous-fenêtres ϕ de N' échantillons.
Le codeur (fig 3) peut être regardé comme une espèce de codeur prédictif-adaptif vectoriel. L'algorithme MP joue le rôle d'un quantificateur vectoriel qui remplace le vecteur du signal d'entrée r'' par le vecteur du signal impulsionnel v. Le prédicteur de pitch calcule le vecteur x à partir des vecteurs précédents du signal y.

Le premier algorithme possible (ref 4) est le suivant. Pour chaque sous-fenêtre ϕ de la fenêtre Θ, pour la dernière sous-fenêtre de la fenêtre Θ-1 et la première sous-fenêtre de la fenêtre Θ+1, les paramètres du prédicteur à long terme (décalage $P(\phi)$ et coefficient $b(\phi)$) sont établis de la façon suivante: pour chaque valeur de l'indice m ($P_{min} \leq m \leq P_{max}$), on calcule l'intercorrélation entre le signal résiduel de la sous-fenêtre ϕ et le signal résiduel décalé de m échantillons:

$$l_m = \sum_{n \in \phi} r_n \, r_{n-m} \qquad (1)$$

Comme valeur de P_{min}, on prend la longueur de la sous-fenêtre. $P(\phi)$ est la valeur de m qui rend maximum $|l_m|$. Le coefficient $b(\phi)$ est donné par:

$$b(\phi) = \frac{l_{P(\phi)}}{\left[\sum_{n \in \phi} r_n^2 \sum_{n \in \phi} r_{n-P(\phi)}^2 \right]^{1/2}} \qquad (2)$$

Pour appliquer l'algorithme MP dans la sous-fenêtre ϕ, il faut tout d'abord calculer le signal d'entrée r'', non seulement pour la sous-fenêtre ϕ, mais aussi pour les sous-fenêtres voisines. On commence par le calcul de la prédiction x pour les sous-fenêtres ϕ-1 et ϕ:

$$x(\phi-1) = B_{P(\phi-1), b(\phi-1)} \left[y(\phi-2), y(\phi-3), \cdots \right]$$

$$x(\phi) = B_{P(\phi), b(\phi)} \left[y(\phi-1), y(\phi-2), \cdots \right]$$

Pour la sous-fenêtre ϕ+1 une formule analogue peut être utilisée seulement si $P(\phi+1) > 2N'$ (on ne connait pas $y(\phi)$ parce que le signal impulsionnel $v(\phi)$ n'est pas encore établi). Sinon on utilise une formule approchée en substituant $x(\phi)$ à $y(\phi)$:

$$x(\phi+1) = B_{P(\phi+1), b(\phi+1)} \left[x(\phi), y(\phi-1), \cdots \right]$$

On calcule ensuite le signal r'' = r - x pour les 3 sous-fenêtres et on applique un algorithme MP.

2.4. Première modification

Il reste quelques problèmes:
- la prédiction pour la sous-fenêtre $\phi+1$ n'est pas juste,
- pour calculer toutes les impulsions de la fenêtre Θ, on doit connaître les paramètres du prédicteur de pitch pour la première sous-fenêtre de la fenêtre $\Theta+1$,
- la modification du signal d'entrée $r''(\phi)$, introduite par l'algorithme MP, n'est pas gardée pour la sous-fenêtre suivante.

La première amélioration proposée est la suivante. Moyennant la contrainte $P_{min} > N' + M$, on crée un décalage de M-1 échantillons entre la sous-fenêtre ϕ utilisée pour la prédiction et la sous-fenêtre ϕ' utilisée pour le calcul des impulsions. A cause de cette contrainte, on calcule la prédiction $x(\phi)$ en utilisant la formule:

$$x(\phi) = B_{P(\phi),b(\phi)} \left[y(\phi'-1), y(\phi'-2), \cdots \right]$$

puis on calcule le signal d'entrée pour le codeur MP:

$$r''(\phi) = r(\phi) - x(\phi)$$

A cause du décalage, on peut calculer maintenant le signal impulsionnel $v(\phi')$, parce que l'on connaît les M-1 premiers échantillons du signal d'entrée r'' pour la sous-fenêtre $\phi'+1$ et les M-1 derniers échantillons (déjà modifiés) de la sous-fenêtre $\phi'-1$:

$$v(\phi') = MP \left[r''(\phi), r''(\phi-1), \cdots \right]$$

L'algorithme MP non seulement calcule le signal multi-impulsionnel $v(\phi')$ mais aussi modifie le signal $r''(\phi')$:

$$r''(\phi') \leftarrow r''(\phi') - v(\phi')$$

Lorsque $v(\phi')$ est connu, on calcule le signal $y(\phi')$:

$$y(\phi') = x(\phi') + v(\phi')$$

Ce décalage de M-1 échantillons demande certaines précautions. Pour placer des impulsions dans les M-1 dernières positions de la fenêtre $\Theta-1$, on doit utiliser l'auto-coorélation normalisée de la réponse impulsionnelle de la fenêtre $\Theta-1$. Il faut donc garder en mémoire un double jeu de coefficients.

2.5. Deuxième modification

Des simulations montrent que les algorithmes MP ont tendance à placer des impulsions au bord droit des sous-fenêtres. En effet, dans le critère que l'on cherche à minimiser, dans le cas habituel de la méthode d'auto-corrélation, interviennent des positions $n = N', \ldots, N'+M-2$ interdites pour placer des impulsions. Pour compenser cette dissymétrie, le nombre des impulsions placées au bord droit des sous-fenêtres a tendance à croître. Pour égaliser la densité des impulsions dans le domaine temporel, on propose la solution suivante: on accepte de placer des impulsions pour toutes les positions $n = 0, \ldots, N'+M-2$, jusqu'à ce que le nombre des impulsions propres (placées dans la sous-fenêtre) soit égal au nombre d'impulsions prévues. On oublie ensuite les impulsions qui se trouvent en dehors de la sous-fenêtre. Cela demande l'introduction de deux signaux d'entrée du bloc MP, le premier est actualisé par toutes les impulsions, le second est actualisé seulement par les impulsions propres.

2.6. Troisième modification

Une troisième amélioration plus significative est obtenue au niveau du calcul des paramètres du prédicteur à long terme. Les formules (1) et (2) ne sont pas optimales. Les paramètres optimaux $b_{opt}(\phi)$ et $P_{opt}(\phi)$ sont ceux qui rendent minimum l'énergie du signal $r_n''*h_n'$. On a remarqué, précédemment, que l'erreur de modélisation du signal de parole $s_n-\hat{s}_n$ est égale à $r_n''-v_n$ filtrée par $1/A(z/\gamma)$. Pour minimiser cette erreur, il faut que le signal d'entrée du bloc MP, filtré par $1/A(z/\gamma)$, ait une énergie la plus faible possible. On minimise donc:

$$E(P,b) = \sum_{n\in\phi} \left[(r_n - b \cdot y_{n-P}) * h_n' \right]^2$$

Si on appelle p_n et q_n les signaux perceptuels originaux et reconstruits, i.e. les signaux r_n et y_n filtrés par $1/A(z/\gamma)$, on obtient:
- la valeur optimale du décalage est la valeur de P qui maximise:

$$\frac{\left(\sum_{n\in\phi} p_n \, q_{n-P} \right)^2}{\sum_{n\in\phi} q_{n-P}^2}$$

- le coefficient optimal est donné par:

$$\frac{\sum_{n\in\phi} p_n \, q_{n-P}}{\sum_{n\in\phi} q_{n-P}^2}$$

Ce traitement est symbolisé par des pointillés fig 3.

2.7. Evaluation de la qualité

Cette évaluation a été faite en utilisant un critère objectif, le rapport signal sur bruit segmental, sur 4 fichiers d'environ 4 ou 5 secondes de signal de parole chacun correspondant à plusieurs locuteurs. La prise de son a été réalisée à l'aide d'un microphone linéaire de radio-téléphone, sans bruit. Les fichiers ont été filtrés dans la bande 300-3300 Hz.

Le fait d'introduire un prédicteur à long terme entraine un gain de 2.5 à 3 dB si tous les paramètres restent identiques d'une simulation à l'autre. Le débit augmente puisque ce prédicteur doit être transmis. Si on joue sur le nombre d'impulsions, on constate, qu'à débit constant, le gain n'est plus que légèrement supérieur à 1 db mais que ce gain est pratiquement constant quelque soit le débit entre 9.6 Kb/s et 14 Kb/s.

3. CODAGE DES PARAMETRES DU FILTRE

Les coefficients de réflexion partiel K_i sont les paramètres les plus aptes à la transmission ou au stockage pour les codeurs à prédiction linéaire. Nous proposons ici une transformation englobant les deux transformations non-linéaires avant quantification proposées par Viswanathan Makhoul (ref 6) et Markel Gray (ref 7) pour compenser respectivement la sensibilité non-uniforme et la distribution non-uniforme des K_i.

3.1. Quantification scalaire calculée

La première fonction de transformation, appelée G, qui compense la sensibilité non-uniforme, est identique pour tous les K_i. La deuxième, que l'on appellera H_i, qui compense la distribution non-uniforme, est spécifique à chaque K_i. Ces fonctions H_i sont issues d'une analyse statistique des coefficients K_i. La fonction de transformation globale proposée $F_i = H_i(G(K_i))$ peut se décomposer sous une forme polynômiale:

$$F_i = A_0 + A_1.K_i + A_2.K_i^2 + \cdots + A_n.K_i^n$$

Les coefficients du polynôme sont déterminés de façon à minimiser l'erreur quadratique par rapport à la fonction F_i évaluée statistiquement. Un ordre peu élevé, de 3 à 5, permet de modéliser ces fonctions.
Les fonctions F_1 et F_2 sont très spécifiques, compte tenu de la particularité de leurs distributions statistiques. Par contre, pour les K_i d'ordre supérieur, les fonctions de transformation F_i sont similaires. Aussi peut-on approximer, par le même polynôme, les fonctions de transformation des K_i d'indice supérieur à 2.
Des mesures objectives de distances cepstrales montrent que la quantification non-uniforme calculée que nous proposons offrent, à même débit, des performances identiques, à 0.05 dB près, au codage classique par table.

3.2. Codage mixte:

Le codage scalaire ne permet pas de réduire le débit à moins de 40 à 35 bits/trame pour un filtre d'ordre 10. Par contre, le codage vectoriel permet de diviser par un facteur 3 à 4 ce débit. Les inconvénients de ce codage sont la taille du dictionnaire qu'il faut stocker et son coût en opérations.
Comme K_1 et K_2 représente 90% de l'inertie, nous proposons de les encoder de manière scalaire calculée. Les K_i restant sont encodés de manière vectorielle. Ce codage mixte permet de réduire d'un facteur 2.5 la taille du dictionnaire et du même facteur la puissance de calcul nécessaire. Quant au débit, il est ramené à 20 bits/trame environ, soit une réduction de 45% par rapport au codage scalaire classique.

4. INTERET DU LANGAGE ADA

ADA est un langage fortement typé. La programmation est structurée. La modularité des programmes est réalisée par l'utilisation de paquetages avec une partie spécification qui constitue l'interface avec les utilisateurs et une partie corps qui peut être améliorée par l'auteur du paquetage sans conséquences visibles pour les utilisateurs. La notion de paquetage générique permet d'augmenter encore la souplesse d'utilisation de ces modules. L'intérêt essentiel de ce langage est de favoriser l'écriture de composants logiciels fonctionnant de manière sûre, l'existence d'une norme pour le compilateur rendant très facile le transport de ces composants d'une machine à l'autre. Toutes ces qualités sont très séduisantes pour un traiteur de signal en collaboration avec des équipes extérieures. Quel est le prix à payer, spécialement en terme de temps d'exécution? Il nous parait négligeable puisque l'algorithme proposé est exécuté en 40 fois le temps réel sur µVAX lorsqu'il est écrit en FORTRAN et en 60 fois le temps réel lorsqu'il est écrit en ADA.

Lorsque l'on cherche à implanter un algorithme de traitement du signal sur une machine, par exemple un µPTS ou un circuit VLSI spécialisé, de nouveaux avantages du langage apparaissent. En effet, il faut alors pouvoir évaluer la dégradation apportée par une implantation en virgule fixe. Le compilateur DEC fournit une solution très simple. On propose:
- de définir un type distinct pour tous les signaux,
- de tester systématiquement la valeur maximale de chacun de ces signaux, chaque fois qu'ils sont réactualisés, la notion de variable locale rémanente permettant de cacher tous ces détails dans le corps du paquetage,
- d'imprimer les valeurs maximales en fin de traitement.

Il suffit alors:
- dans un 1er temps, de déclarer tous les types selon le modèle suivant: "type ECH_SIGNAL is new FLOAT;" la simulation sera alors réalisée en virgule flottante,
- dans un 2eme temps de remplacer, par l'éditeur de texte dans le programme source, toutes les (ou quelques unes des) déclarations de type selon le modèle: "type ECH_SIGNAL is delta 2.0**(-10) range -2.0 .. 2.0;" si on désire, par exemple, représenter une variable de type ECH_SIGNAL sur 12 bits avec une précision de 2^{-10} et une dynamique maximale de 2. La simulation sera réalisée complétement (partiellement) en virgule fixe. L'opération réalisée est un arrondi. L'algorithme proposé est alors exécuté en 90 fois le temps réel.

5. REFERENCES

/1/ "Efficient computation and encoding of the multi-pulse excitation for LPC"
M.BEROUTI, H.GARTEN, P.KABAL, P.MERMELSTEIN
ICASSP 84 - 10.1

/2/ "Efficient algorithms for obtaining multipulse excitation for LPC coders"
J.P.LEFEVRE, O.PASSIEN
ICASSP 85 - 25.6

/3/ "Regular excitation reduction for effective and efficient LP-coding of speech"
E.F.DEPRETTERE, P.KROON
ICASSP 85 - 25.8

/4/ "Etude, simulation et mise en oeuvre sur microprocesseur de codeurs predictifs multi-impulsionnels"
E.LANCON
Universite de NICE - Thèse de 3ème cycle - Novembre 1985

/5/ "High quality multi-pulse speech coder with pitch prediction"
K.OZAWA, T.ARASEKI
ICASSP 86 - 33.3

/6/ "Quantization properties of transmission parameters in linear predictive systems"
VISWANATHAN, MAKHOUL
IEEE Trans. on ASSP, June 1975

/7/ "Implementation and comparison of two transformed reflection coefficient scalar quantization methods"
MARKEL, GRAY
IEEE Trans. on ASSP, October 1980

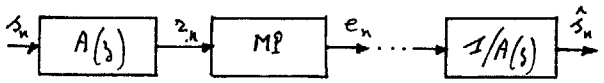

fig 1 but : Min $\sum_n ((z_n - e_n) * h'_n)^2$

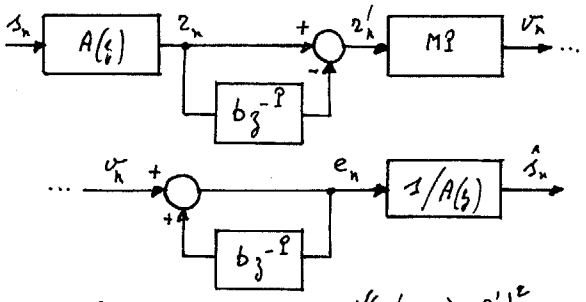

fig 2 but : Min $\sum_n ((z'_n - v_n) * h'_n)^2$

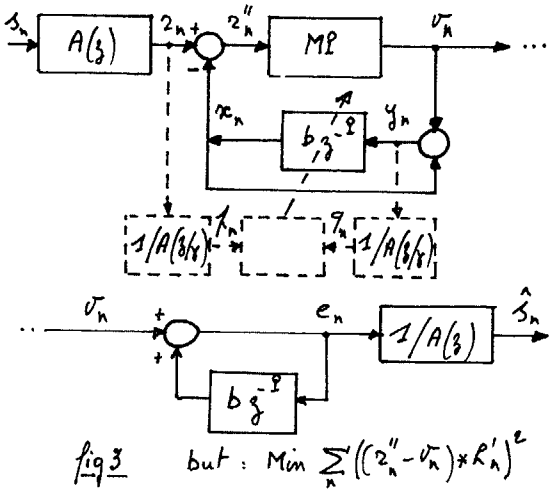

fig 3 but : Min $\sum_n ((z''_n - v_n) * h'_n)^2$