



UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

CCITT

COMITÉ CONSULTATIF
INTERNATIONAL
TÉLÉGRAPHIQUE ET TÉLÉPHONIQUE

G.728

(09/92)

**ASPECTS GÉNÉRAUX DES SYSTÈMES
DE TRANSMISSION NUMÉRIQUES;
ÉQUIPEMENTS TERMINAUX**

**CODAGE DE LA PAROLE À 16 kbits
EN UTILISANT LA PRÉDICTION LINÉAIRE
À FAIBLE DÉLAI AVEC EXCITATION
PAR CODE**

Recommandation G.728



Genève, 1992

AVANT-PROPOS

Le CCITT (Comité consultatif international télégraphique et téléphonique) est un organe permanent de l'Union internationale des télécommunications (UIT). Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

L'Assemblée plénière du CCITT, qui se réunit tous les quatre ans, détermine les thèmes d'études et approuve les Recommandations rédigées par ses Commissions d'études. Entre les Assemblées plénières, l'approbation des Recommandations par les membres du CCITT s'effectue selon la procédure définie dans la Résolution n° 2 du CCITT (Melbourne, 1988).

La Recommandation G.728, que l'on doit à la Commission d'études XV, a été approuvée le 1^{er} septembre 1992 selon la procédure définie dans la Résolution n° 2.

NOTES DU CCITT

- 1) Dans cette Recommandation, l'expression «Administration» est utilisée pour désigner de façon abrégée aussi bien une Administration de télécommunications qu'une exploitation privée reconnue de télécommunications.
- 2) La liste des abréviations utilisées dans cette Recommandation se trouve dans l'annexe F.

© UIT 1992

Droits de reproduction réservés. Aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'éditeur.

Recommandation G.728

CODAGE DE LA PAROLE À 16 kbit/s EN UTILISANT LA PRÉDICTION LINÉAIRE À FAIBLE DÉLAI AVEC EXCITATION PAR CODE

(1992)

1 Introduction

La présente Recommandation contient la description d'un algorithme destiné au codage des signaux de parole à 16 kbit/s en utilisant la prédiction linéaire à faible délai avec excitation par code (LD-CELP) (*low-delay code excited linear prediction*). La présente Recommandation est organisée comme suit.

Le § 2 donne une présentation succincte de l'algorithme LD-CELP. Les § 3 et 4 traitent respectivement des principes du codeur LD-CELP et du décodeur LD-CELP. Le § 5 définit les détails de calcul applicables à chaque bloc fonctionnel de l'algorithme. Les annexes A, B, C et D contiennent les tables de constantes utilisés par l'algorithme LD-CELP. L'annexe E donne l'ordre des opérations d'adaptation et d'utilisation des variables. Finalement l'appendice I fournit des informations relatives aux procédures applicables à la vérification de mise en œuvre de l'algorithme.

Pour étude complémentaire, on envisage d'ajouter ultérieurement trois appendices additionnels (qui seront publiés séparément) portant respectivement sur les aspects réseau de l'algorithme LD-CELP, sur la description d'une mise en œuvre de l'algorithme LD-CELP en virgule fixe et sur les procédures de vérification de l'algorithme LD-CELP en virgule fixe.

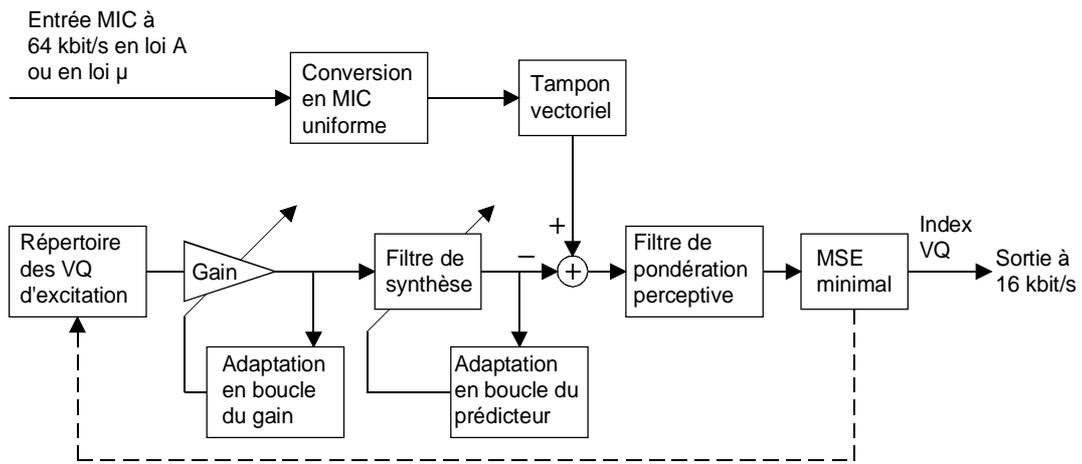
2 Présentation succincte de l'algorithme LD-CELP

L'algorithme LD-CELP se compose d'un codeur et d'un décodeur, qui sont décrits respectivement dans les § 2.1 et 2.2 et illustrés par la figure 1/G.728.

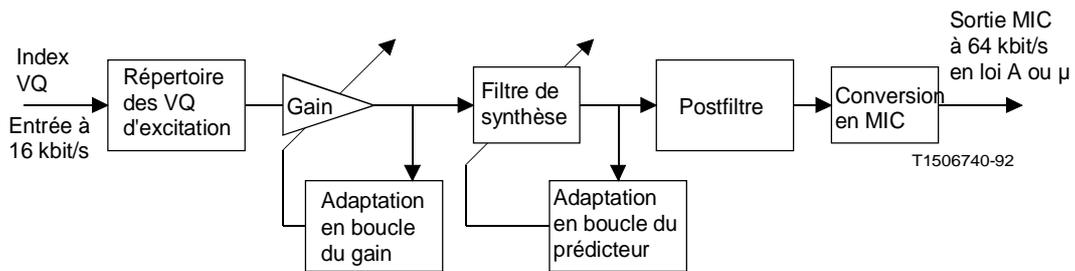
Le principe des techniques de prédiction linéaire avec excitation par code (CELP) (*code excited linear prediction*) qui consistent à effectuer une recherche par analyse et synthèse dans un répertoire de séquences codées, a été retenu dans l'algorithme LD-CELP. Celui-ci utilise toutefois une adaptation séquentielle des prédicteurs et du gain afin d'obtenir un délai d'algorithme de 0,625 ms. Seul l'index du répertoire des séquences codées d'excitation est transmis. Les coefficients des prédicteurs sont mis à jour par analyse en codage prédictif linéaire (LPC) (*linear prediction coding*) des signaux d'excitation est préalablement quantifiés. Le gain mis à jour au moyen des informations de gain incluses dans les séquences d'excitation préalablement quantifiée. La taille du bloc correspondant au vecteur d'excitation et à l'adaptation du gain ne dépassera pas cinq échantillons. Un filtre pondérateur de perception est mis à jour par analyse LPC des signaux de parole non quantifiés.

2.1 Codeur LD-CELP

Le signal entrant après conversion d'un code MIC loi A ou loi μ à un code MIC uniforme est découpé en blocs de cinq échantillons successifs. Pour chaque bloc entrant, le codeur fait passer chacun des 1024 vecteurs possibles du répertoire de séquences codées (stockés dans un répertoire d'excitation) par un module de normalisation du gain et par un filtre de synthèse. Parmi les 1024 vecteurs de signal quantifiés possibles obtenus, le codeur choisit celui qui minimise, par rapport au vecteur du signal entrant, l'erreur quadratique moyenne pondérée en fréquence. Il transmet ensuite au décodeur l'index du répertoire, sur 10 bits, du meilleur vecteur du répertoire d'excitation (ou «vecteurcode») correspondant au meilleur vecteur de signal quantifié possible. Ce meilleur «vecteurcode» traverse ensuite le module de normalisation du gain et le filtre de synthèse afin de mettre à jour la mémoire du filtre pour préparer le codage du vecteur de signal suivant. Les coefficients du filtre de synthèse et le gain sont mis à jour périodiquement par adaptation en boucle basée sur le signal précédent quantifié et l'excitation de gain normalisée.



a) Codeur LD-CELP



b) Décodeur LD-CELP

FIGURE 1/G.728

Schéma simplifié du codec LD-CELP

2.2 Décodeur LD-CELP.

L'opération de décodage est également effectuée bloc par bloc. Dès réception de chaque index de 10 bits, le décodeur consulte le répertoire d'excitation afin d'en extraire le vecteurcode correspondant. Ce vecteurcode extrait traverse ensuite un module de normalisation du gain et dans un filtre de synthèse et afin de produire le vecteur de signal décodé actuel. Les coefficients du filtre de synthèse et le gain sont ensuite mis à jour de la même manière que dans le codeur. Le vecteur de signal décodé traverse ensuite un postfiltre adaptatif afin d'augmenter la qualité perçue. Les coefficients de ce postfiltre sont mis à jour périodiquement au moyen des informations disponibles dans le décodeur. Les cinq échantillons du vecteur de signal issu du postfiltre sont ensuite convertis en cinq échantillons de sortie MIC, loi A ou loi μ .

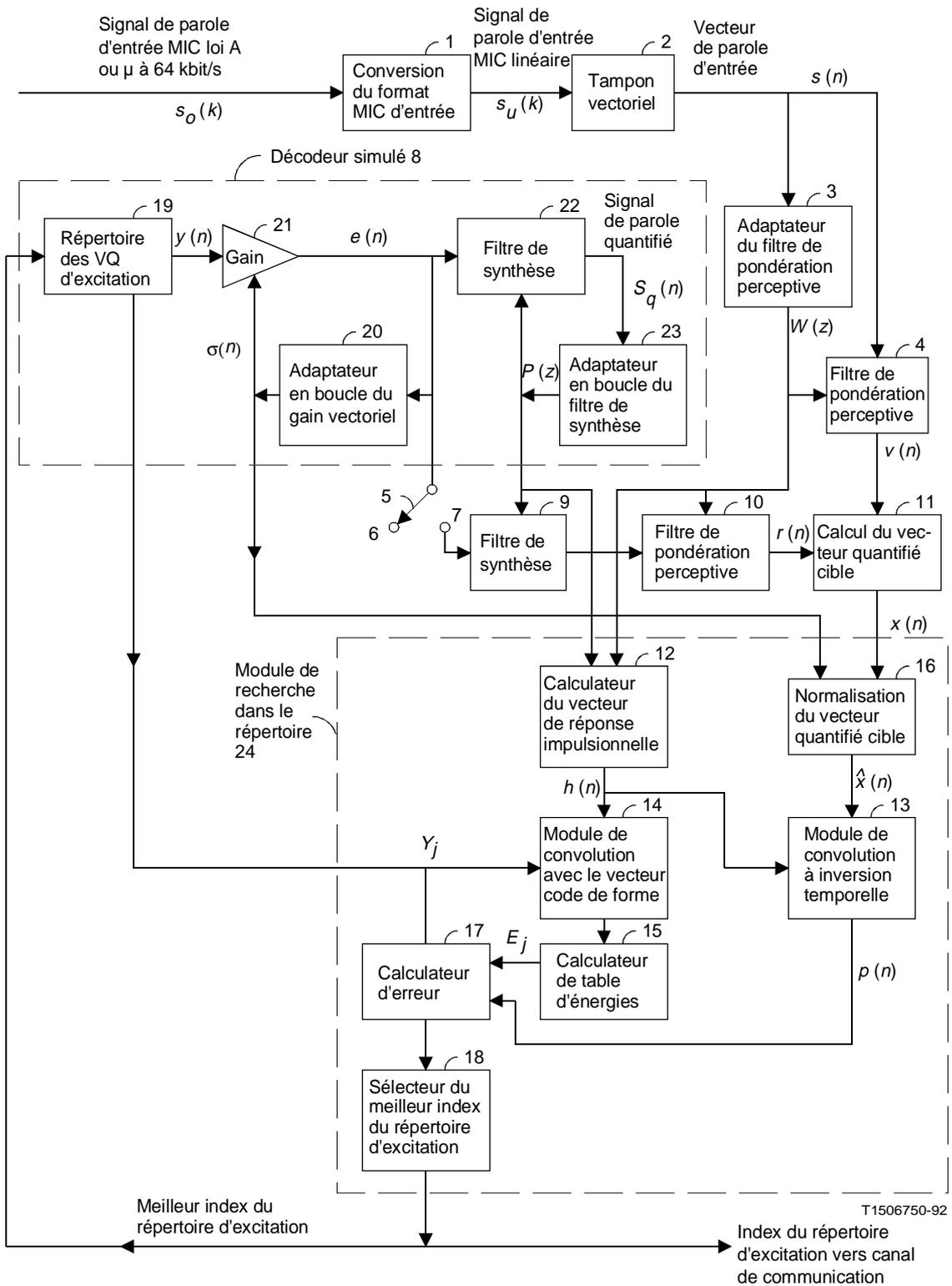


FIGURE 2/G.728

Organigramme du codeur LD-CELP

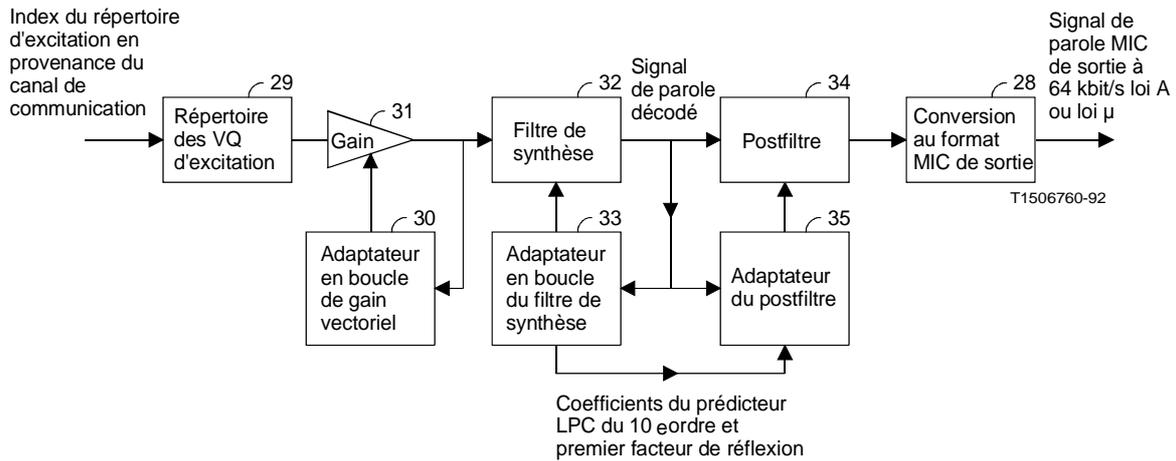


FIGURE 3/G.728

Organigramme du décodeur LD-CELP

3 Principes du codeur LD-CELP

La figure 2/G.728 donne un diagramme fonctionnel détaillé du codeur LD-CELP. Le codeur représenté à la figure 2/G.728 est mathématiquement équivalent au codeur déjà indiqué sur la figure 1/G.728 mais sa mise en œuvre est plus efficace du point de vue du calcul.

Dans la description ci-après:

- pour chaque variable à décrire, k est l'index d'échantillonnage et les échantillons sont prélevés à des intervalles de 125 μ s;
- un groupe de cinq échantillons consécutifs d'un signal donné est appelé *vecteur* de ce signal. Par exemple, cinq échantillons de parole consécutifs forment un vecteur de parole; cinq échantillons d'excitation forment un vecteur d'excitation, etc.;
- la lettre n sera utilisée pour indiquer l'index vectoriel, qui est différent de l'index d'échantillonnage k ;
- quatre vecteurs consécutifs forment un *cycle d'adaptation*. Dans un paragraphe ultérieur, ces cycles seront également appelés *trames*. Les deux termes sont synonymes.

L'index du répertoire d'excitation [codage à quantification vectorielle (VQ) (*vector quantization*)] est la seule information explicitement transmise du codeur au décodeur. Trois autres types de paramètres seront périodiquement mis à jour: le gain d'excitation, les coefficients du filtre de synthèse et les coefficients du filtre de pondération perceptive. Ces paramètres sont déduits par adaptation en boucle à partir des signaux qui précèdent le vecteur du signal en cours. Le gain d'excitation est mis à jour à chaque vecteur tandis que les coefficients du filtre de synthèse et ceux du filtre de pondération perceptive sont mis à jour tous les quatre vecteurs (c'est-à-dire avec une période de vingt échantillons ou 2,5 ms). On observe que bien que la séquence de traitement de l'algorithme ait un cycle d'adaptation de quatre vecteurs (20 échantillons), la capacité du tampon vectoriel de base n'est que d'un vecteur (5 échantillons). La petite taille de ce tampon permet d'obtenir un délai dans un seul sens inférieur à 2 ms.

Une description de chaque bloc du codeur est donnée ci-dessous. Etant donné que le codeur LD-CELP est surtout utilisé pour coder des signaux de parole, nous supposons, pour faciliter la description que le signal d'entrée est vocal bien qu'en pratique il puisse aussi s'agir de signaux non vocaux.

3.1 Conversion du format de signal d'entrée MIC

Ce bloc convertit le signal d'entrée MIC loi A ou loi μ , $s_o(k)$, en signal MIC uniforme $s_u(k)$.

3.1.1 Niveaux internes du MIC linéaire

Lors de la conversion de la loi A ou μ à la modulation MIC linéaire, différentes représentations internes sont possibles, selon le dispositif utilisé. Par exemple, les tables normalisées pour la modulation MIC en loi μ définissent une gamme linéaire de $-4015,5$ à $+4015,5$. La gamme correspondante en loi A est de -2016 à $+2016$. Les deux tables comprennent quelques valeurs de sortie ayant une partie fractionnaire égale à $0,5$. Ces parties fractionnaires ne peuvent pas être représentées dans des entiers, à moins que l'ensemble de la table ne soit multiplié par deux de manière à n'obtenir que des valeurs entières. C'est en fait ce qui est réalisé le plus souvent dans les puces de traitement numérique des signaux (DSP) (*digital signal processing*) en virgule fixe. Par ailleurs, les puces DSP en virgule flottante peuvent représenter des valeurs identiques à celles qui sont énumérées dans les tables. Dans la présente Recommandation, on supposera que le signal d'entrée a une gamme maximale de -4095 à $+4095$, ce qui englobe la loi μ comme la loi A. Dans le cas de la loi A, cela implique que, lorsque la conversion linéaire aboutit à une étendue de -2016 à $+2016$, ces valeurs soient multipliées par 2 avant de commencer le codage du signal. Dans le cas d'une entrée en loi μ dans un processeur à virgule fixe qui convertit la gamme d'entrée de -8031 à $+8031$, cela implique que les valeurs soient divisées par 2 avant de commencer le processus de codage. Une autre possibilité est de traiter ces valeurs comme étant en format Q1, c'est-à-dire avec 1 bit à droite de la virgule. Tous les calculs mettant en jeu ces données devront ensuite tenir compte de ce bit.

Dans le cas de signaux d'entrée en modulation MIC linéaire sur 16 bits possédant la gamme dynamique complète de $-32\ 768$ à $+32\ 767$, on devra considérer que les valeurs d'entrée sont de format Q3, c'est-à-dire qu'il faudra les diviser par 8. A la sortie du décodeur, on multipliera le résultat par 8 pour reconstituer ces signaux.

3.2 Tampon vectoriel

Ce bloc stocke cinq échantillons de parole consécutif $s_u(5n)$, $s_u(5n + 1)$, ..., $s_u(5n + 4)$ pour former un vecteur de parole à cinq dimensions $s(n) = [s_u(5n), s_u(5n + 1), \dots, s_u(5n + 4)]$.

3.3 Adaptateur du filtre de pondération perceptive

La figure 4/G.728 montre le fonctionnement détaillé de l'adaptateur du filtre de pondération perceptive (bloc 3 de la figure 2/G.728). Cet adaptateur calcule les coefficients du filtre de pondération perceptive tous les quatre vecteurs de parole, sur la base d'une analyse par prédiction linéaire (souvent dénommée analyse LPC) des signaux de parole non quantifiés. Les mises à jour de ces coefficients interviennent au troisième vecteur de parole de chaque cycle d'adaptation de 4 vecteurs. Les coefficients sont maintenus constants entre deux mises à jour successives.

Voir la figure 4a)/G.728. Le calcul est effectué comme suit. Tout d'abord, le vecteur d'entrée de parole (non quantifié) traverse un module de fenêtrage hybride (bloc 36) qui place une fenêtre sur les précédents vecteurs de parole et calcule les 11 premiers coefficients d'autocorrélation du signal de parole encadré (fenêtré) par le module. Le module de récurrence de Levinson-Durbin (bloc 37) convertit ensuite ces coefficients d'autocorrélation en coefficients de prédiction. Sur la base de ces coefficients de prédiction, le calculateur de coefficients du filtre de pondération (bloc 38) détermine les coefficients désirés du filtre de pondération. Ces trois blocs sont étudiés plus en détail ci-dessous.

Décrivons tout d'abord les principes du fenêtrage hybride. Comme cette technique sera utilisée dans trois types différents d'analyse LPC, on en donnera une description assez générale avant d'en examiner différents cas particuliers. Supposons que l'analyse LPC doive être exécutée tous les L échantillons de signal. Pour rester dans le cas général, admettons que les échantillons de signal correspondant au cycle d'adaptation LD-CELP actuel soient $s_u(m)$, $s_u(m + 1)$, $s_u(m + 2)$, ..., $s_u(m + L - 1)$. Alors, pour une analyse LPC à adaptation par boucle, le fenêtrage hybride est appliqué à tous les échantillons de signal précédents ayant un index d'échantillon inférieur à m (comme indiqué dans la figure 4 b)/G.728). Supposons que la fonction de fenêtrage hybride encadre N échantillons non récurrents. Alors, tous les échantillons de signal $s_u(m - 1)$, $s_u(m - 2)$, ..., $s_u(m - N)$ seront alors pondérés par la portion non récurrente de la fenêtre. Tous les échantillons de signal situés à gauche de l'échantillon $s_u(m - N - 1)$ (ainsi que ce dernier) seront pondérés par la portion récurrente de la fenêtre, qui a les valeurs b , $b\alpha$, $b\alpha^2$, ..., où $0 < b < 1$ et $0 < \alpha < 1$.

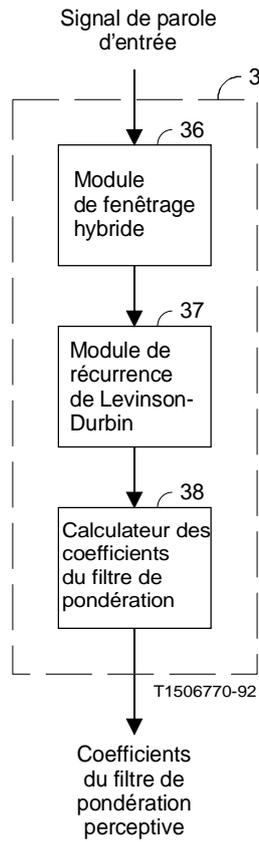


FIGURE 4a)/G.728

Adaptateur du filtre de pondération perceptive

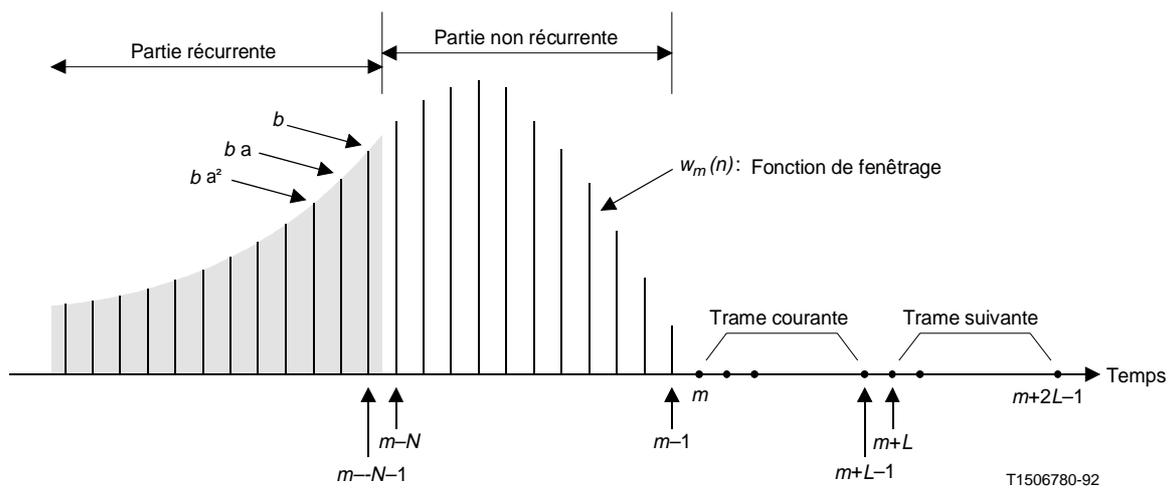


FIGURE 4b)/G.728

Illustration d'une fenêtre hybride

A l'instant m , la fonction de fenêtrage hybride $w_m(k)$ est définie par:

$$w_m(k) = \begin{cases} f_m(k) = b\alpha^{-[k-(m-N-1)]}, & \text{si } k \leq m-N-1 \\ g_m(k) = -\sin [c(k-m)], & \text{si } m-N \leq k \leq m-1 \\ 0, & \text{si } k \geq m \end{cases} \quad (3-1a)$$

et le signal pondéré par la fenêtre est

$$s_m(k) = s_u(k) w_m(k) = \begin{cases} s_u(k) f_m(k) = s_u(k) b\alpha^{-[k-(m-N-1)]}, & \text{si } k \leq m-N-1 \\ s_u(k) g_m(k) = -s_u(k) \sin [c(k-m)], & \text{si } m-N \leq k \leq m-1 \\ 0, & \text{si } k \geq m \end{cases} \quad (3-1b)$$

Les échantillons de la partie non récurrente $g_m(k)$ et la section initiale de la partie récurrente $f_m(k)$ pour différentes fenêtres hybrides sont spécifiés dans l'annexe A. Pour une analyse LPC du $M^{\text{ième}}$ ordre, il est nécessaire de calculer $M+1$ coefficients d'autocorrélation $R_m(i)$ pour $i = 0, 1, 2, \dots, M$. Le $i^{\text{ème}}$ coefficient d'autocorrélation pour le cycle d'adaptation actuel peut être exprimé par:

$$R_m(i) = \sum_{k=-\infty}^{m-1} s_m(k) s_m(k-i) = r_m(i) + \sum_{k=m-N}^{m-1} s_m(k) s_m(k-i) \quad (3-1c)$$

où

$$r_m(i) = \sum_{k=-\infty}^{m-N-1} s_m(k) s_m(k-i) = \sum_{k=-\infty}^{m-N-1} s_u(k) s_u(k-i) f_m(k) f_m(k-i) \quad (3-1d)$$

Dans le membre droit de l'équation (3-1c), le premier terme $r_m(i)$ est la «composante récurrente» de $R_m(i)$, tandis que le deuxième terme en est la «composante non récurrente». La sommation finie des composantes non récurrentes est calculée pour chaque cycle d'adaptation. D'autre part, la composante récurrente est calculée par itérations successives. Les paragraphes suivants en expliquent la méthode de calcul.

Supposons que nous ayons calculé et mis en mémoire tous les $r_m(i)$ pour le cycle d'adaptation actuel et que nous souhaitons passer au cycle suivant, qui commence à l'échantillon $s_u(m+L)$. Après avoir décalé la fenêtre hybride de L échantillons vers la droite, le nouveau signal pondéré par fenêtrage devient, pour le cycle d'adaptation suivant

$$s_{m+L}(k) = s_u(k) w_{m+L}(k) = \begin{cases} s_u(k) f_{m+L}(k) = s_u(k) f_m(k) \alpha^L, & \text{si } k \leq m+L-N-1 \\ s_u(k) g_{m+L}(k) = -s_u(k) \sin [c(k-m-L)], & \text{si } m+L-N \leq k \leq m+L-1 \\ 0, & \text{si } k \geq m+L \end{cases} \quad (3-1e)$$

La composante récurrente de $R_{m+L}(i)$ peut s'écrire

$$\begin{aligned} r_{m+L}(i) &= \sum_{k=-\infty}^{m+L-N-1} s_{m+L}(k) s_{m+L}(k-i) \\ &= \sum_{k=-\infty}^{m-N-1} s_{m+L}(k) s_{m+L}(k-i) + \sum_{k=m-N}^{m+L-N-1} s_{m+L}(k) s_{m+L}(k-i) \\ &= \sum_{k=-\infty}^{m-N-1} s_u(k) f_m(k) \alpha^L s_u(k-i) f_m(k-i) \alpha^L + \sum_{k=m-N}^{m+L-N-1} s_{m+L}(k) s_{m+L}(k-i) \end{aligned} \quad (3-1f)$$

ou sous la forme

$$r_{m+L}(i) = \alpha^{2L} r_m(i) + \sum_{k=m-N}^{m+L-N-1} s_{m+L}(k) s_{m+L}(k-i) \quad (3-1g)$$

On peut donc calculer $r_{m+L}(i)$ par récurrence à partir de $r_m(i)$ dans l'équation (3-1g). Ce nouveau coefficient $r_{m+L}(i)$ est remis en mémoire pour utilisation lors du cycle d'adaptation suivant. Le coefficient d'autocorrélation $R_{m+L}(i)$ est alors calculé comme suit

$$R_{m+L}(i) = r_{m+L}(i) + \sum_{k=m+L-N}^{m+L-1} s_{m+L}(k) s_{m+L}(k-i) \quad (3-1h)$$

Nous avons jusqu'ici décrit de manière générale les principes de la procédure de calcul par fenêtrage hybride. Les valeurs paramétriques du module de fenêtrage hybride 36 (voir la figure 4 a)/G.728) sont

$$M = 10, L = 20, N = 30 \text{ et } \alpha = \left(\frac{1}{2}\right)^{\frac{1}{40}} = 0,982820598 \left(\text{de telle sorte que } \alpha^{2L} = \frac{1}{2} \right)$$

Une fois calculés les 11 coefficients d'autocorrélation $R(i)$, $i = 0, 1, \dots, 10$ par la procédure de fenêtrage hybride décrite ci-dessus, on applique une procédure de «correction par bruit blanc». Cela s'effectue en augmentant légèrement l'énergie $R(0)$:

$$R(0) \leftarrow \left(\frac{257}{256}\right)R(0) \quad (3-1i)$$

Cela a pour effet de combler les vallées spectrales avec du bruit blanc de manière à réduire la gamme dynamique spectrale et de compenser la mauvaise préparation de la récurrence de Levinson-Durbin. Le facteur de correction par bruit blanc (WNCF) (*white noise correction factor*) de 257/256 correspondant à un niveau de bruit blanc de 24 dB environ au-dessous de la puissance vocale moyenne.

Le module 37 de récurrence de Levinson-Durbin calcule ensuite, au moyen des coefficients d'autocorrélation corrigés par le bruit blanc, les coefficients des prédicteurs d'ordres 1 à 10. Soient $a_j^{(i)}$ les $j^{\text{ièmes}}$ coefficients du prédicteur du $i^{\text{ème}}$ ordre. La procédure de récurrence peut alors être spécifiée comme suit:

$$E(0) = R(0) \quad (3-2a)$$

$$k_i = - \frac{R(i) + \sum_{j=1}^{i-1} a_j^{(i-1)} R(i-j)}{E(i-1)} \quad (3-2b)$$

$$a_i^{(i)} = k_i \quad (3-2c)$$

$$a_j^{(i)} = a_j^{(i-1)} + k_i a_{i-j}^{(i-1)}, \quad 1 \leq j \leq i-1 \quad (3-2d)$$

$$E(i) = (1 - k_i^2) E(i-1) \quad (3-2e)$$

Les équations (3-2b) à (3-2e) sont évaluées par récurrence pour $i = 1, 2, \dots, 10$ et la solution finale est donnée par

$$q_i = a_i^{(10)}, \quad 1 \leq i \leq 10 \quad (3-2f)$$

Si nous définissons $q_0 = 1$, le «filtre d'erreur de prédiction» du 10^e ordre (parfois appelé «filtre d'analyse») a la fonction de transfert suivante

$$\tilde{Q}(z) = \sum_{i=0}^{10} q_i z^{-i} \quad (3-3a)$$

et le prédicteur linéaire du 10^e ordre correspondant est défini par la fonction de transfert suivante

$$Q(z) = -\sum_{i=1}^{10} q_i z^{-i} \quad (3-3b)$$

Le module calculateur du coefficient de filtrage pondérateur (bloc 38) calcule les coefficients du filtre de pondération perceptive conformément aux équations ci-après

$$Q(z / \gamma_1) = -\sum_{i=1}^{10} (q_i \gamma_1^i) z^{-i} \quad (3-4b)$$

et

$$Q(z / \gamma_2) = -\sum_{i=1}^{10} (q_i \gamma_2^i) z^{-i} \quad (3-4c)$$

Le filtre de pondération perceptive est un filtre à pôles et zéros du 10^e ordre défini par la fonction de transfert $W(z)$ de l'équation (3-4a). Les valeurs de γ_1 et de γ_2 sont respectivement de 0,9 et de 0,6

Se reporter maintenant à la figure 2/G.728. L'adaptateur du filtre de pondération perceptive (bloc 3) met à jour périodiquement les coefficients de $W(z)$ conformément aux équations (3-2) à (3-4) et présente ces coefficients au module calculateurs de vecteur de réponse impulsionnelle (bloc 12) et aux filtres de pondération perceptive (blocs 4 et 10).

3.4 Filtre de pondération perceptive

Dans la figure 2/G.728, le vecteur de parole entrant actuel $s(n)$, traverse le filtre de pondération perceptive (bloc 4) et donne le vecteur de parole pondéré $v(n)$.

Remarque – Sauf pendant l'initialisation, la mémoire du filtre (c'est-à-dire les variables d'état internes ou les valeurs contenues dans les unités de délai du filtre) ne devrait jamais être remise à zéro. Par ailleurs, la mémoire du filtre de pondération perceptive (bloc 10) devra faire l'objet d'un traitement spécial, comme indiqué plus loin.

3.4.1 Fonctionnement pour signaux non vocaux

Pour les signaux de modem ou autres signaux non vocaux, les résultats des essais du CCITT indiquent qu'il est souhaitable de neutraliser le filtre de pondération perceptive. Ceci équivaut à poser $W(z) = 1$. Ceci est plus facile à réaliser en donnant la valeur zéro à γ_1 et à γ_2 dans l'équation (3-4a). Les valeurs nominales de ces variables, pour les signaux de parole, sont de 0,9 et de 0,6 respectivement.

3.5 Filtre de synthèse

Dans la figure 2/G.728, il y a deux filtres de synthèse (blocs 9 et 22) aux coefficients identiques. Ces deux filtres sont mis à jour par l'adaptateur du filtre de synthèse en boucle (bloc 23). Chaque filtre de synthèse est un filtre du 50^e ordre tout pôle composé d'une boucle de rétroaction avec un prédicteur LPC du 50^{ème} ordre dans la branche de rétroaction. La fonction de transfert du filtre de synthèse est $F(z) = 1/[1 - P(z)]$, où $P(z)$ est la fonction de transfert du prédicteur LPC du 50^e ordre.

Une fois obtenu le vecteur de parole pondéré $v(n)$, un vecteur de réponse à entrée nulle $r(n)$ est construit au moyen du filtre de synthèse (bloc 9) et du filtre de pondération perceptive (bloc 10). Pour cela, l'interrupteur 5 est tout d'abord ouvert, c'est-à-dire qu'il pointe vers le nœud 6. Cela implique que le signal passant du nœud 7 au filtre de synthèse 9 sera nul. On laisse ensuite le filtre de synthèse 9 et le filtre de pondération perceptive 10 «sonner» pour cinq échantillons (un vecteur). Cela signifie que l'opération de filtrage continue pendant cinq échantillons avec un signal appliqué au nœud 7 nul. Le signal sortant du filtre de pondération perceptive 10 est le vecteur de réponse à entrée nulle recherché, $r(n)$.

Remarque – Sauf pour le vecteur qui suit immédiatement l'initialisation, la mémoire des filtres 9 et 10 est en général non nulle; le vecteur de sortie $r(n)$ sera donc généralement non nul également, même si l'entrée du filtre, issue du nœud 7, est nulle. Ce vecteur $r(n)$ est en fait la réponse des deux filtres aux vecteurs d'excitation de gain normalisé précédents $e(n - 1)$, $e(n - 2)$, ... Ce vecteur représente en réalité l'effet dû à la mémoire du filtre jusqu'à l'instant $(n - 1)$.

3.6 Calcul du vecteur cible VQ

Ce bloc soustrait le vecteur de réponse à entrée nulle $r(n)$ du vecteur de parole pondéré $v(n)$ pour donner le vecteur cible $x(n)$ de recherche dans le répertoire de vecteurs d'excitation quantifiés.

3.7 Adaptateur en boucle du filtre de synthèse

Cet adaptateur (bloc 23) met à jour les coefficients des filtres de synthèse 9 et 22. Il reçoit en entrée le signal de parole quantifié (synthétisé) et produit en sortie un ensemble de coefficients pour le filtre de synthèse. Son fonctionnement est très proche de celui de l'adaptateur 3 du filtre de pondération perceptive.

Une version éclatée de cet adaptateur est représentée à la figure 5/G.728. Le fonctionnement du module de fenêtrage hybride 49 et du module de récurrence de Levinson-Durbin 50 est exactement le même que celui de leurs contreparties (36 et 37) dans la figure 4 a)/G.728, à l'exception des trois différences suivantes:

- le signal d'entrée est maintenant la parole quantifiée au lieu de la parole non quantifiée;
- le prédicteur est d'ordre 50 au lieu de 10;
- les paramètres de fenêtrage hybride sont différents: $N = 35$, $\alpha = \left(\frac{3}{4}\right)^{\frac{1}{40}} = 0,992833749$.

Remarque – La période de mise à jour reste $L = 20$ et le facteur de correction par bruit blanc est toujours de $257/256 = 1,00390625$.

Soit $\hat{P}(z)$ la fonction de transfert du prédicteur LPC du 50^e ordre, sa forme est alors

$$\hat{P}(z) = -\sum_{i=1}^{50} \hat{a}_i z^{-i} \quad (3-5)$$

où les \hat{a}_i sont les coefficients du prédicteur. Afin d'améliorer leur résistance aux erreurs des canaux de communication, ces coefficients sont modifiés de manière que les pics du spectre LPC résultants aient une largeur un peu plus grande. Le module 51 d'extension de largeur exécute cette procédure d'extension de la largeur de bande comme suit. Etant donné les coefficients de prédicteur \hat{a}_i , un nouvel ensemble de coefficients a_i est calculé conformément à l'équation

$$a_i = \lambda^i \hat{a}_i; \quad i = 1, 2, \dots, 50 \quad (3-6)$$

où λ est donné par

$$\lambda = \frac{253}{256} = 0,98828125 \quad (3-7)$$

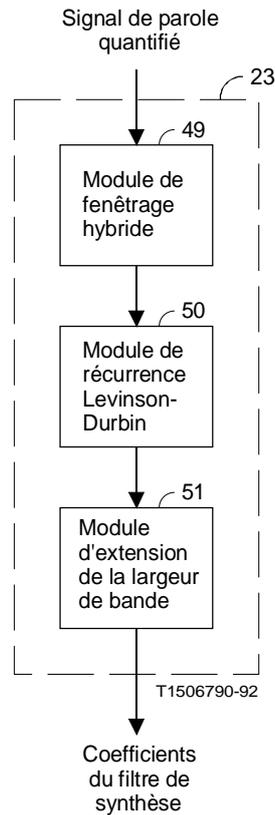


FIGURE 5/G.728

Adaptateur en boucle du filtre de synthèse

Cela a pour effet de déplacer radialement d'un facteur λ tous les pôles du filtre de synthèse vers l'origine. Etant donné que les pôles s'éloignent du cercle unité, les pics de la fonction réponse en fréquence sont élargis.

Après cette extension de la largeur de bande, le prédicteur LPC modifié a la fonction de transfert

$$P(z) = -\sum_{i=1}^{50} a_i z^{-i} \quad (3-8)$$

Les coefficients modifiés sont alors soumis aux filtres 9 et 22. Ces coefficients sont également soumis au calculateur 12 du vecteur de réponse impulsionnelle.

Les filtres de synthèse 9 et 22 ont tous deux la fonction de transfert

$$F(z) = \frac{1}{1 - P(z)} \quad (3-9)$$

Comme le filtre de pondération perceptuelle, les filtres de synthèse 9 et 22 sont mis à jour tous les quatre vecteurs. Les mises à jour interviennent également au troisième vecteur de parole du cycle d'adaptation de quatre vecteurs. Les mises à jour sont cependant fondées sur le signal de parole quantifié jusqu'au dernier vecteur du précédent cycle d'adaptation. En d'autres termes, un retard de deux vecteurs est introduit avant que les mises à jour ne soient effectuées. Cela s'explique du fait que le module 50 de récurrence Levinson-Durbin et le calculateur de table énergétique 15 (décrit plus loin) nécessitent beaucoup de calculs. En conséquence, bien que l'autocorrélation des

signaux de parole déjà numérisés soit disponible au premier vecteur de chaque cycle de quatre vecteurs, les calculs peuvent nécessiter une durée supérieure à celle d'un seul vecteur. Afin de conserver une dimension élémentaire du tampon vectoriel d'un vecteur (de manière à minimiser le délai de codage) et afin de maintenir le fonctionnement en temps réel, on introduit dans les mises à jour du filtre un délai de deux vecteurs pour faciliter la mise en œuvre en temps réel.

3.8 Adaptateur en boucle du gain vectoriel

Cet adaptateur met à jour le gain d'excitation $\sigma(n)$ pour chaque index n de temps vectoriel. Le gain d'excitation $\sigma(n)$ est un facteur de pondération utilisé pour mettre à l'échelle le vecteur d'excitation $y(n)$ choisi. L'adaptateur 20 reçoit en entrée le vecteur d'excitation normalisé en gain $e(n)$ et produit en sortie un gain d'excitation $\sigma(n)$. Il essaye surtout de «prédire» le gain de $e(n)$ d'après les gains de $e(n-1)$, $e(n-2)$, ..., par prédiction linéaire adaptative dans le domaine des gains logarithmiques. Cet adaptateur en boucle du gain vectoriel 20 est représenté plus en détail sur la figure 6/G.728.

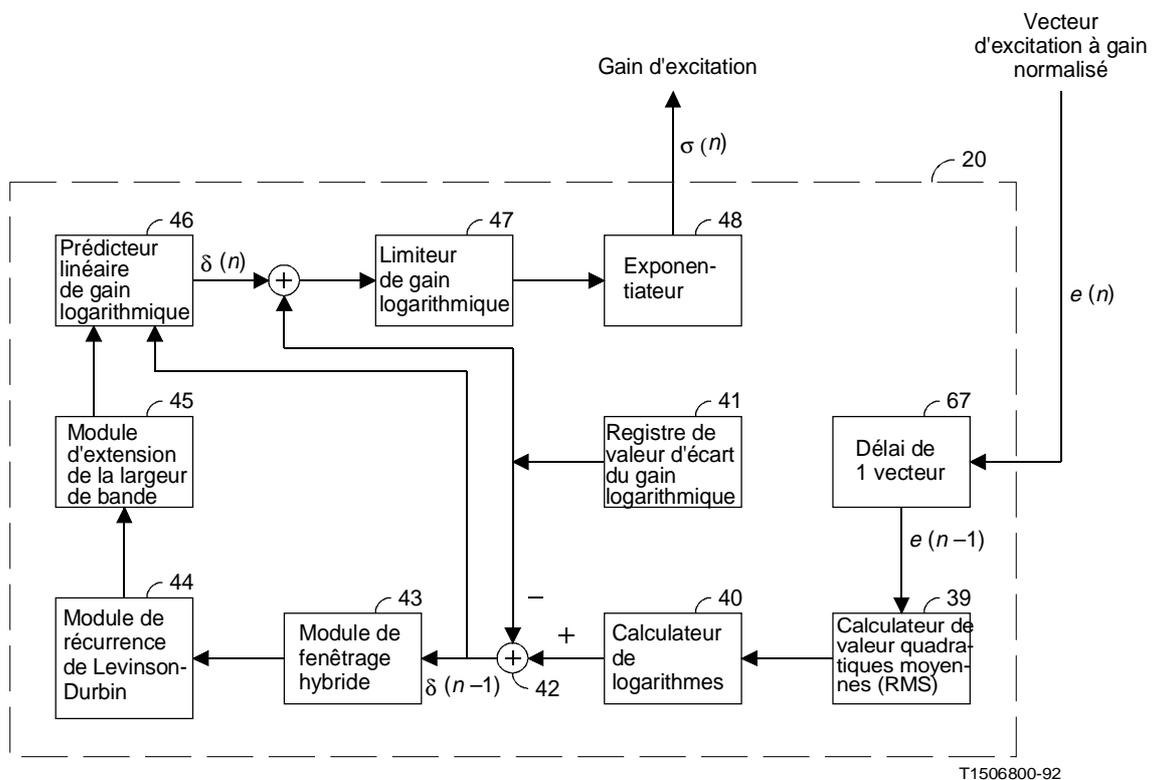


FIGURE 6/G.728

Adaptateur en boucle de gain vectoriel

Selon la figure 6/G.728, l'adaptateur de gain fonctionne comme suit. Le module 67, de délai d'un vecteur fournit le précédent vecteur $e(n-1)$ d'excitation normalisé en gain. Le calculateur 39 de valeur quadratique moyenne (RMS) (*root-mean-square*) calcule alors la valeur RMS du vecteur $e(n-1)$. Ensuite, le calculateur logarithmique 40 calcule la valeur en décibels de la RMS de $e(n-1)$, en calculant d'abord son logarithme décimal puis en multipliant le résultat par 20.

Dans la figure 6/G.728, une valeur d'écart de 32 dB pour le gain logarithmique est stockée dans le registre 41 de valeurs d'écart pour le gain logarithmique. Cette valeur est censée être à peu près égale au niveau (en décibels) du gain d'excitation moyen mesuré pendant le signal de parole. L'additionneur 42 soustrait cette valeur d'écart pour le

gain logarithmique du gain logarithmique produit par le calculateur de logarithmes 40. Le gain logarithmique à écart soustrait qui en résulte, $\delta(n - 1)$, est ensuite utilisé par le module 43 de fenêtrage hybride et par le module 44 de récurrence Levinson-Durbin. Ici encore, les blocs 43 et 44 fonctionnent exactement comme les blocs 36 et 37 du module adaptateur de filtre de pondération perceptives (figure 4 a)/G.728), sauf que les paramètres du fenêtrage hybride sont différents et que le signal analysé est maintenant le gain logarithmique à écart soustrait au lieu du signal de parole d'entrée.

Remarque – Une seule valeur de gain est produite tous les cinq échantillons de parole.

Les paramètres de fenêtrage hybride du bloc 43 sont

$$M = 10, N = 20, L = 4, \alpha = \left(\frac{3}{4}\right)^{\frac{1}{8}} = 0,96467863$$

Le module de récurrence Levinson-Durbin 44 fournit en sortie les coefficients d'un prédicteur linéaire du 10^e ordre avec pour fonction de transfert

$$\hat{R}(z) = -\sum_{i=1}^{10} \hat{\alpha}_i z^{-i} \quad (3-10)$$

Le module d'extension de largeur de bande 45 déplace radialement les racines de ce polynôme vers le plan- z origine de manière semblable au module 51 dans la figure 5/G.728. Le prédicteur de gain à bande élargie qui en résulte a la fonction de transfert

$$R(z) = -\sum_{i=1}^{10} \alpha_i z^{-i} \quad (3-11)$$

dont les coefficients α_i sont calculés comme suit

$$\alpha_i = \left(\frac{29}{32}\right)^i \hat{\alpha}_i = (0,90625)^i \hat{\alpha}_i \quad (3-12)$$

Une telle extension de largeur de bande rend l'adaptateur de gain (bloc 20 dans la figure 2/G.728) plus résistant aux erreurs des canaux de communication. Ces coefficients α_i sont ensuite utilisés comme coefficients du prédicteur linéaire de gain logarithmique (bloc 46 de la figure 6/G.728).

Ce prédicteur 46 est mis à jour tous les quatre vecteurs de parole et les mises à jour interviennent au deuxième vecteur de parole de chaque cycle d'adaptation de quatre vecteurs. Le prédicteur essaye de prédire $\delta(n)$ sur la base d'une combinaison linéaire de gain $\delta(n - 1)$, $\delta(n - 2)$, ..., $\delta(n - 10)$. La version prédite de $\delta(n)$ est notée $\hat{\delta}(n)$ et est donnée par

$$\hat{\delta}(n) = -\sum_{i=1}^{10} \alpha_i \delta(n - i) \quad (3-13)$$

Une fois que $\hat{\delta}(n)$ a été produit par le prédicteur linéaire 46 de gain logarithmique, on lui ajoute la valeur d'écart pour le gain logarithmique de 32 dB, mise en mémoire dans le bloc 41. Le limiteur de gain logarithmique 47 vérifie ensuite la valeur de gain ainsi obtenue et l'écrite si elle est trop grande ou trop petite. Les limites inférieure et supérieure sont fixées à 0 dB et 60 dB respectivement. Le signal de sortie du limiteur de gain est ensuite injecté dans le module d'exponentiation 48, qui effectue l'opération inverse du calculateur de logarithmes 40 et convertit la valeur en décibels du gain en valeur linéaire. Le limiteur de gain fait en sorte que le gain, ainsi rapporté dans le domaine linéaire, soit compris entre 1 et 1000.

3.9 Module de recherche dans le répertoire d'excitation

Dans la figure 2/G.728, les blocs 12 à 18 forment le module 24 de recherche dans le répertoire d'excitation. Ce module passe en revue les 1024 vecteurcodes candidats dans le répertoire 19 d'excitation à quantification vectorielle (VQ) et identifie l'index du meilleur vecteurcode qui produise un vecteur de parole quantifié qui soit le plus proche du vecteur de parole d'entrée.

Afin de réduire la complexité de la recherche dans le répertoire, le répertoire de 1024 entrées codées sur 10 bits est décomposé en deux répertoires plus petits: un répertoire de forme codé sur 7 bits contenant 128 vecteurcodes indépendants et un répertoire de gain codé sur 3 bits, contenant huit valeurs scalaires qui sont symétriques par rapport à zéro (c'est-à-dire un bit de signe et deux bits pour l'amplitude). Le vecteurcode obtenu finalement en sortie est le résultat du meilleur vecteurcode de forme (issu du répertoire de forme à 7 bits) et du meilleur niveau de gain (issu du répertoire de gain à 3 bits). La table du répertoire de forme à 7 bits et la table du répertoire de gain à 3 bits sont données dans l'annexe B.

3.9.1 Principe de la recherche des codes d'excitation

En principe, le module 24 de recherche des codes d'excitation pondère chacun des 1024 vecteurcodes candidats en fonction du gain d'excitation $\sigma(n)$ actuel puis les fait passer un par un dans un filtre monté en cascade, comportant le filtre de synthèse $F(z)$ et le filtre de pondération perceptible $W(z)$. La mémoire du filtre est remise à zéro chaque fois que le module de recherche injecte un nouveau vecteurcode dans le filtre en cascade, de fonction de transfert $H(z) = F(z)W(z)$.

Le filtrage des vecteurcodes VQ peut être exprimé en termes de multiplication matricielle vectorielle. Soit y_j le $j^{\text{ème}}$ vecteurcode dans le répertoire de forme à 7 bits et g_i les $i^{\text{èmes}}$ niveaux dans le répertoire de gain à 3 bits. Posons $\{h(n)\}$ la séquence de réponse impulsionnelle du filtre en cascade. Dans ce cas, lorsque le vecteurcode spécifié par les index i et j des répertoires est injecté dans le filtre en cascade $H(z)$, la sortie du filtre pourra être exprimée par

$$\tilde{x}_{ij} = \mathbf{H}\sigma(n)g_i y_j \quad (3-14)$$

où

$$\mathbf{H} = \begin{bmatrix} h(0) & 0 & 0 & 0 & 0 \\ h(1) & h(0) & 0 & 0 & 0 \\ h(2) & h(1) & h(0) & 0 & 0 \\ h(3) & h(2) & h(1) & h(0) & 0 \\ h(4) & h(3) & h(2) & h(1) & h(0) \end{bmatrix} \quad (3-15)$$

Le module 24 de recherche des codes d'excitation cherche la meilleure combinaison des index i et j qui minimisera la distorsion mesurée par la variance (MSE) (*mean-squared error*) ci-après

$$D = \|x(n) - \tilde{x}_{ij}\|^2 = \sigma^2(n) \|\hat{x}(n) - g_i \mathbf{H}y_j\|^2 \quad (3-16)$$

où $\hat{x}(n) = x(n)/\sigma(n)$ est le vecteur cible VQ pondéré de gain normalisé. En développant ces termes nous obtenons

$$D = \sigma^2(n) \left[\|\hat{x}(n)\|^2 - 2g_i \hat{x}^T(n) \mathbf{H}y_j + g_i^2 \|\mathbf{H}y_j\|^2 \right] \quad (3-17)$$

Comme le terme $\|\hat{x}(n)\|^2$ et la valeur de $\sigma^2(n)$ sont fixes pendant la recherche de codes d'excitation, minimiser D revient à minimiser

$$\hat{D} = -2g_i p^T(n) y_j + g_i^2 E_j \quad (3-18)$$

où

$$p(n) = \mathbf{H}^T \hat{x}(n) \quad (3-19)$$

et

$$E_j = \|\mathbf{H}y_j\|^2 \quad (3-20)$$

Remarque qu' E_j est en fait l'énergie des $j^{\text{èmes}}$ vecteurcodes de forme filtrés et ne dépend pas du vecteur VQ cible $\hat{x}(n)$. Noter également que le vecteurcode de forme y_j est fixe et que la matrice \mathbf{H} ne dépend que du filtre de synthèse et du filtre de pondération, qui restent fixes sur une période de quatre vecteurs de parole. En conséquence, E_j est également fixe sur une période de quatre vecteurs de parole. Sur la base de cette observation, on peut calculer et mettre en mémoire lorsque les deux filtres sont mis à jour, les 128 termes énergétiques E_j possibles, $j = 0, 1, 2, \dots, 127$ (correspondant aux 128 vecteurcodes de forme). On utilise ensuite ces termes énergétiques de manière répétitive pour la recherche de codes d'excitation durant la période suivante de quatre vecteurs de parole. Cette solution permet de réduire la complexité de la recherche dans le répertoire d'excitation.

Pour diminuer encore les calculs, on peut calculer d'avance et mettre en mémoire les deux tables suivantes:

$$b_i = 2g_i \quad (3-21)$$

et

$$c_i = g_i^2 \quad (3-22)$$

pour $i = 0, 1, \dots, 7$.

Ces deux tables sont fixes puisque les g_i le sont. On peut maintenant exprimer \hat{D} sous la forme suivante

$$\hat{D} = -b_i P_j + c_i E_j \quad (3-23)$$

où $P_j = p^T(n)y_j$.

Noter qu'une fois les tables des E_j , des b_i et des c_i précalculées et mémorisées, le multiplicateur interne $P_j = p^T(n)y_j$, qui ne dépend que de j , accapare la plus grande partie des calculs pour la détermination de \hat{D} . La procédure de recherche de codes d'excitation passe donc en revue le répertoire de forme et repère le meilleur index de gain i pour chacun des vecteurcode de forme y_j .

Il existe plusieurs moyens de trouver le meilleur index de gain i pour un vecteur code de forme donné, y_j :

- le premier et le plus évident moyen consiste à évaluer les huit valeurs possibles de \hat{D} qui correspondent aux huit valeurs possibles de i puis à relever l'index i qui correspond au plus petit \hat{D} . Mais cette méthode exige deux multiplications pour chaque i ;
- une deuxième solution consiste à calculer d'abord le gain optimal $\hat{g} = P_j/E_j$ puis à quantifier ce gain \hat{g} selon l'un des huit niveaux de gain $\{g_0, \dots, g_7\}$ contenus dans le répertoire de gain à 3 bits. Le meilleur index i sera celui du niveau de gain g_i qui sera le plus proche de \hat{g} . Mais cette méthode nécessite une opération de division pour chacun des 128 vecteurcodes de forme, ce qui est considéré comme très peu efficace pour réalisations utilisant des plate-formes DSP;
- une troisième solution, qui est une variante légèrement modifiée de la deuxième, est particulièrement efficace pour les mises en œuvre de DSP. La quantification de \hat{g} peut être vue comme une série de comparaisons entre \hat{g} et les «frontières de cellule du quantificateur», qui sont les points médians entre niveaux de gain adjacents. Soit d_i le point médian entre les niveaux de gain g_i et g_{i+1} ayant le même signe. Dans ce cas, le test « $\hat{g} < d_i$?» est équivalent au test « $P_j < d_i E_j$?». En utilisant ce dernier test, nous pourrions donc éviter l'opération de division tout en n'exigeant qu'une seule multiplication pour chaque index i . C'est l'approche utilisée pour la recherche des codes d'excitation. Les frontières des cellules du quantificateur de gain d_i sont fixes et peuvent être calculées d'avance et mémorisées dans une table. Pour les huit niveaux de gain, on n'utilise en fait que six valeurs de frontière: d_0, d_1, d_2, d_4 et d_6 .

Une fois identifiés les meilleurs index i et j sont concaténés pour former la sortie du module de recherche de codes d'excitation, c'est-à-dire un unique index à 10 bits du meilleur code d'excitation.

3.9.2 Fonctionnement du module de recherche de codes d'excitation

Le principe de la recherche dans le répertoire d'excitation étant posé, on décrit ci-dessous le fonctionnement du module 24 de recherche de codes d'excitation. Voir la figure 2/G.728. Chaque fois que le filtre de synthèse 9 et le filtre de pondération perceptuelle 10 sont mis à jour, le calculateur 12 du vecteur de réponse impulsionnelle calcule les cinq premiers échantillons de la réponse impulsionnelle du filtre en cascade $F(z)W(z)$. Pour calculer le vecteur de

réponse impulsionnelle, on remet d'abord à zéro la mémoire du filtre en cascade; puis on excite ce filtre avec une séquence d'entrée $\{1, 0, 0, 0, 0\}$. Les cinq échantillons correspondants à la sortie du filtre sont $h(0), h(1), \dots, h(4)$ et constituent le vecteur de réponse impulsionnelle recherché. Une fois ce vecteur calculé, il est maintenu constant et utilisé lors de la recherche de codes d'excitation dans la période des quatre vecteurs de parole suivants, jusqu'à ce que les filtres 9 et 10 soient mis à jour.

Le module 14 de convolution des vecteurcodes de forme calcule ensuite les 128 vecteurs $\mathbf{H}y_j$, $j = 0, 1, 2, \dots, 127$. En d'autres termes, ce module convolue chaque vecteurcode de forme y_j , $j = 0, 1, 2, \dots, 127$ avec la séquence du vecteur de réponse impulsionnelle $h(0), h(1), \dots, h(4)$, la convolution n'étant appliquée qu'aux cinq premiers échantillons. Les énergies des 128 vecteurs résultants sont ensuite calculées et mises en mémoire par le calculateur 15 de la table d'énergie, conformément à l'équation (3-20). L'énergie d'un vecteur est définie comme étant la somme des carrés de chacune de ses composantes.

Noter que les calculs dans les blocs 12, 14 et 15 ne sont effectués que tous les quatre vecteurs de parole alors que les autres blocs du module de recherche de codes d'excitation effectuent leurs calculs pour chaque vecteur de parole. Noter également que les mises à jour de la table des E_j sont synchronisées avec celles des coefficients du filtre de synthèse. En d'autres termes, la nouvelle table des E_j sera utilisée à partir du troisième vecteur de parole de chaque cycle d'adaptation. (Voir à ce sujet le § 3.7.)

Le module 16 de normalisation du vecteur cible VQ calcule le vecteur cible VQ pondéré par le gain $\hat{\lambda}(n) = x(n)/\sigma(n)$. Dans les réalisations DSP, il est plus efficace de calculer d'abord $1/\sigma(n)$ puis de multiplier chaque composante du vecteur $x(n)$ par $1/\sigma(n)$.

Le module 13 de convolution à inversion temporelle calcule ensuite le vecteur $p(n) = \mathbf{H}^T \hat{\lambda}(n)$. Cette opération consiste à inverser d'abord l'ordre des composantes du vecteur $\hat{\lambda}(n)$, puis à convoluer le vecteur résultant avec le vecteur de réponse impulsionnelle et enfin à inverser de nouveau l'ordre des composantes à la sortie (d'où l'appellation «convolution à inversion temporelle»).

Une fois les tables des E_j , des b_i et des c_i précalculées et mises en mémoire puis le vecteur $p(n)$ calculé, le calculateur d'erreur 17 et le sélecteur de meilleur index d'excitation 18 coopèrent pour réaliser l'algorithme suivant, de recherche efficace dans le répertoire d'excitation;

- a) initialiser la distorsion \hat{D}_{\min} avec un nombre plus grand que la plus grande valeur possible de la distorsion \hat{D} (ou utiliser le nombre le plus grand possible du système de représentation numérique du processeur DSP);
- b) poser l'index du répertoire de forme $j = 0$;
- c) calculer le produit interne $P_j = p^T(n)y_j$;
- d) si $P_j < 0$, passer à l'étape h) pour passer en revue les gains négatifs; sinon, passer à l'étape e) pour passer en revue les gains positifs;
- e) si $P_j < d_0 E_j$, poser $i = 0$ et passer à l'étape k); sinon, passer à l'étape f);
- f) si $P_j < d_1 E_j$, poser $i = 1$ et passer à l'étape k); sinon, passer à l'étape g);
- g) si $P_j < d_2 E_j$, poser $i = 2$ et passer à l'étape k); sinon, poser $i = 3$ et passer à l'étape k);
- h) si $P_j < d_4 E_j$, poser $i = 4$ et passer à l'étape k); sinon, passer à l'étape i);
- i) si $P_j < d_5 E_j$, poser $i = 5$ et passer à l'étape k); sinon, passer à l'étape j);
- j) si $P_j < d_6 E_j$, poser $i = 6$ sinon, poser $i = 7$;
- k) calculer $\hat{D} = -b_i P_j + c_i E_j$;
- l) si $\hat{D} < \hat{D}_{\min}$, poser $\hat{D}_{\min} = \hat{D}$, $i_{\min} = i$, et $j_{\min} = j$;
- m) si $j < 127$, poser $j = j + 1$ et passer à l'étape c); sinon, passer à l'étape n);
- n) lorsque l'algorithme arrive à ce point, les 1024 combinaisons possibles de gains et de formes ont été passées en revue. Les i_{\min} et j_{\min} résultants sont respectivement les index de canal recherchés pour le gain et la forme. En sortie, le meilleur index de codage (sur 10 bits) sera la concaténation de ces deux index et le meilleur vecteurcode d'excitation correspondant sera $y(n) = g_{i_{\min}} y_{j_{\min}}$. L'index de codage sur 10 bit choisi sera transmis au décodeur par le canal de communication.

3.10 *Décodeur simulé*

Bien que le décodeur ait identifié et transmis le meilleur index de codage trouvé jusque-là, certaines tâches additionnelles doivent être exécutées pour préparer le codage des vecteurs de parole suivants. Tout d'abord, le meilleur index de codage est envoyé vers le répertoire d'excitation VQ afin d'en extraire le meilleur vecteurcode correspondant $y(n) = g_{i_{\min}} y_{j_{\min}}$. Ce dernier est ensuite normalisé par le gain d'excitation actuel $\sigma(n)$ dans l'étage de gain 21. Le vecteur d'excitation normalisé en gain qui en résulte est $e(n) = \sigma(n) y(n)$.

Ce vecteur $e(n)$ traverse ensuite le filtre de synthèse 22 afin d'obtenir le vecteur de parole quantifié actuel $s_q(n)$. Noter que les blocs 19 à 23 forment un décodeur simulé 8. Le vecteur de parole quantifié $s_q(n)$ simule donc en fait le vecteur de parole décodé lorsqu'il n'y a pas d'erreurs de transmission. Dans la figure 2/G.728, l'adaptateur 23 du filtre de synthèse en boucle a besoin de ce vecteur de parole quantifié $s_q(n)$ pour mettre à jour les coefficients du filtre de synthèse. De même, l'adaptateur 20 de gain vectoriel en boucle a besoin du vecteur d'excitation normalisé en gain $e(n)$ pour mettre à jour les coefficients du prédicteur linéaire de gain logarithmique.

Une dernière tâche à exécuter avant de coder le vecteur de parole suivant consiste à mettre à jour la mémoire du filtre de synthèse 9 et le filtre de pondération perceptive 10. A cette fin, on sauvegarde d'abord la mémoire des filtres 9 et 10, qui avait été conservée après l'exécution du calcul de réponse à entrée nulle, décrit au § 3.5. On met ensuite à zéro la mémoire des filtres 9 et 10 et on termine l'interrupteur 5 pour le connecter au nœud 7. Le vecteur d'excitation normalisé en gain $e(n)$ traverse ensuite les deux filtres 9 et 10, à mémoire remise à zéro. Etant donné que la longueur du vecteur $e(n)$ est limitée à cinq échantillons et que les filtres ont zéro en mémoire, on notera que le nombre d'opérations de multiplication-addition ne passe que de 0 à 4 pendant la période de cinq échantillons, ce qui procure une notable économie en temps de calcul car il y aurait 70 opérations de multiplication-addition par échantillon si la mémoire des filtres n'était pas à zéro. On ajoute ensuite le contenu original (sauvegardé) de la mémoire du filtre au nouveau contenu établi à la suite du filtrage du vecteur $e(n)$. Cela revient à ajouter les réponses en entrée nulle aux réponses en état zéro des filtres 9 et 10, ce qui donne l'ensemble souhaité de mémoire de filtrage qui servira à calculer la réponse en entrée nulle pendant le codage du prochain vecteur de parole.

Noter qu'après la mise à jour de la mémoire du filtre, les cinq éléments situés à la partie supérieure de la mémoire du filtre de synthèse 9 sont exactement les mêmes que les composantes du vecteur de parole quantifié recherché, $s_q(n)$. On peut donc omettre en pratique le filtre de synthèse 22 et obtenir $s_q(n)$ d'après la mémoire mise à jour du filtre de synthèse 9. cela entraîne une économie supplémentaire de 50 multiplications-additions par échantillon.

Le fonctionnement du codeur qui a été décrit ci-dessus spécifie la façon de coder un seul vecteur de parole d'entrée. Le codage du signal de parole total sera réalisé en répétant l'opération ci-dessus pour chaque vecteur de parole.

3.11 *Synchronisation et signalisation dans la bande*

Dans la description du codeur qui précède, on suppose que le codeur connaît les bornes des index de 10 bits du répertoire d'excitation reçus et qu'il connaît également le moment où il y a lieu de mettre à jour le filtre de synthèse et le prédicteur de gain logarithmique (rappelons que ces mises à jour interviennent tous les quatre vecteurs). En pratique, une telle information de synchronisation peut être communiquée au décodeur par adjonction de bits de synchronisation supplémentaire en plus du flot binaire transmis à 16 kbit/s. Dans de nombreuses applications cependant, il sera nécessaire d'insérer des bits de synchronisation ou de signalisation dans la bande dans le flot binaire de 16 kbit/s. On pourra y parvenir de la façon suivante. Supposons qu'un bit de synchronisation soit à insérer tous les N vecteurs de parole; dans ce cas, tous les N vecteurs de parole d'entrée, on peut ne passer en revue que la moitié du répertoire de forme pour obtenir un index de forme codé sur 6 bits. De cette manière, on dérobe un bit tous les N index du répertoire d'excitation transmis et on le remplace par un bit de synchronisation ou de signalisation.

Il importe de remarquer que l'on ne peut pas subtiliser arbitrairement un bit d'un index de forme codé sur 7 bits déjà sélectionné; en revanche, le codeur doit savoir lesquels des vecteurs de parole seront dépouillés d'un bit. Il doit ensuite ne passer en revue que la moitié du répertoire d'excitation qui correspond à ces vecteurs de parole. Sinon, le décodeur n'aura pas les mêmes vecteurcodes d'excitation décodés pour ces vecteurs de parole.

Etant donné que l'algorithme de codage possède un cycle d'adaptation de base de quatre vecteurs, il est raisonnable que N soit un multiple de 4, de manière que le décodeur puisse déterminer facilement les limites des cycles d'adaptation du codeur. Avec une valeur raisonnable de N (comme 16, qui correspond à une période de subtilisation de bit de 10 ms), la dégradation de qualité de la parole est pratiquement négligeable. On constate en particulier qu'une valeur de $N = 16$ donne lieu à une très faible distorsion additionnelle. Le débit de cette subtilisation de bit n'est que de 100 bits.

Si l'on suit la procédure ci-dessus, nous recommandons de ne consulter que la première moitié du répertoire de forme lorsque le bit souhaité doit être un 0; c'est-à-dire que l'on ne recherchera que les vecteurs d'index compris entre 0 et 63. Lorsque le bit souhaité est un 1, la deuxième moitié du répertoire sera consultée et l'index résultant sera compris entre 64 et 127. La raison de ce choix est que le bit souhaité sera le bit situé le plus à gauche dans le mot-code puisque les 7 bits du vecteurcode de forme précèdent les 3 bits du signe et du répertoire de gain. Nous recommandons également de subtiliser le bit de synchronisation sur le dernier vecteur d'un cycle de quatre vecteurs. Une fois qu'il a été détecté, le mot-code reçu ensuite peut commencer le nouveau cycle de vecteurcodes.

Bien que nous ayons affirmé ci-dessus que la synchronisation n'apporte que peu de distorsion, il nous faut relever qu'aucun essai systématique n'a été effectué sur le matériel mettant en œuvre cette stratégie de synchronisation. Par conséquent, le niveau de dégradation n'a donc pas été mesuré.

Nous formulons cependant une mise en garde expresse contre l'utilisation du bit de synchronisation pour synchroniser des systèmes dans lesquels le codeur est constamment activé et désactivé. Un système pourrait par exemple utiliser un détecteur d'activité de la parole pour désactiver le codeur en l'absence de parole. Chaque fois que le codeur est activé, le décodeur aura besoin de localiser la séquence de synchronisation. A 100 bit/s, cela prendra sans doute plusieurs centaines de millisecondes. De plus, il est indispensable de prévoir un intervalle de temps pour que l'état du décodeur s'adapte à celui du codeur. Le résultat d'ensemble sera un phénomène appelé mutilation de la parole faisant que le début de la prononciation des paroles sera perdu. Si le codeur et le décodeur sont tous les deux activés au même instant que le début de la parole, aucun signal de parole ne sera perdu. Cela ne sera possible que dans des systèmes faisant appel à une signalisation externe pour les instants de mise en service et à une synchronisation externe.

4 Principes du décodeur LD-CELP

La figure 3/G.728 donne un diagramme par blocs du décodeur LD-CELP. Une description fonctionnelle de chaque bloc est présentée dans les paragraphes qui suivent.

4.1 *Répertoire des vecteurs quantifiés d'excitation*

Ce bloc contient un répertoire des vecteurs quantifiés (VQ) d'excitation (incluant les répertoires de forme et de gain) identique au répertoire 19 du codeur LD-CELP. Il utilise le meilleur index du répertoire d'excitation reçu pour extraire le meilleur vecteurcode $y(n)$ choisi dans le codeur LD-CELP.

4.2 *Module de normalisation du gain*

Ce bloc calcule le vecteur d'excitation $e(n)$ en multipliant chaque composante de $y(n)$ par le gain $\sigma(n)$.

4.3 *Filtre de synthèse*

Ce filtre a la même fonction de transfert que le filtre de synthèse contenu dans le codeur LD-CELP (en admettant une transmission exempte d'erreur). Il filtre le vecteur d'excitation normalisé en gain $e(n)$ afin de produire le vecteur de parole décodé $s_d(n)$. Noter que pour éviter toute accumulation possible d'erreurs d'arrondi pendant le décodage, il est parfois souhaitable de répéter exactement les procédures utilisées dans le codeur afin d'obtenir le vecteur $s_q(n)$. Si tel est le cas et si le codeur obtient le vecteur $s_q(n)$ à partir de la mémoire mise à jour du filtre de synthèse 9, il convient que le décodeur calcule également le vecteur $s_d(n)$ comme la somme de la réponse à entrée nulle et de la réponse en état zéro du filtre de synthèse 32, comme cela se passe dans le codeur.

4.4 *Adaptateur en boucle du gain vectoriel*

La fonction de ce bloc est décrite au § 3.8.

4.5 *Adaptateur en boucle du filtre de synthèse*

La fonction de ce bloc est décrite au § 3.7.

4.6 *Postfiltre*

Ce bloc filtre le signal de parole décodé afin d'augmenter la qualité perçue. Dans la figure 7/G.728, ce bloc est agrandi pour en montrer de plus amples détails. Voir ladite figure. Le postfiltre est en principe composé de trois parties principales: un postfiltre à long terme 71, un postfiltre à court terme 72 et un module de normalisation du gain de sortie 77. Les quatre autres blocs de la figure 7/G.728 servent seulement à calculer le facteur de pondération à utiliser dans le module de normalisation du gain de sortie 77.

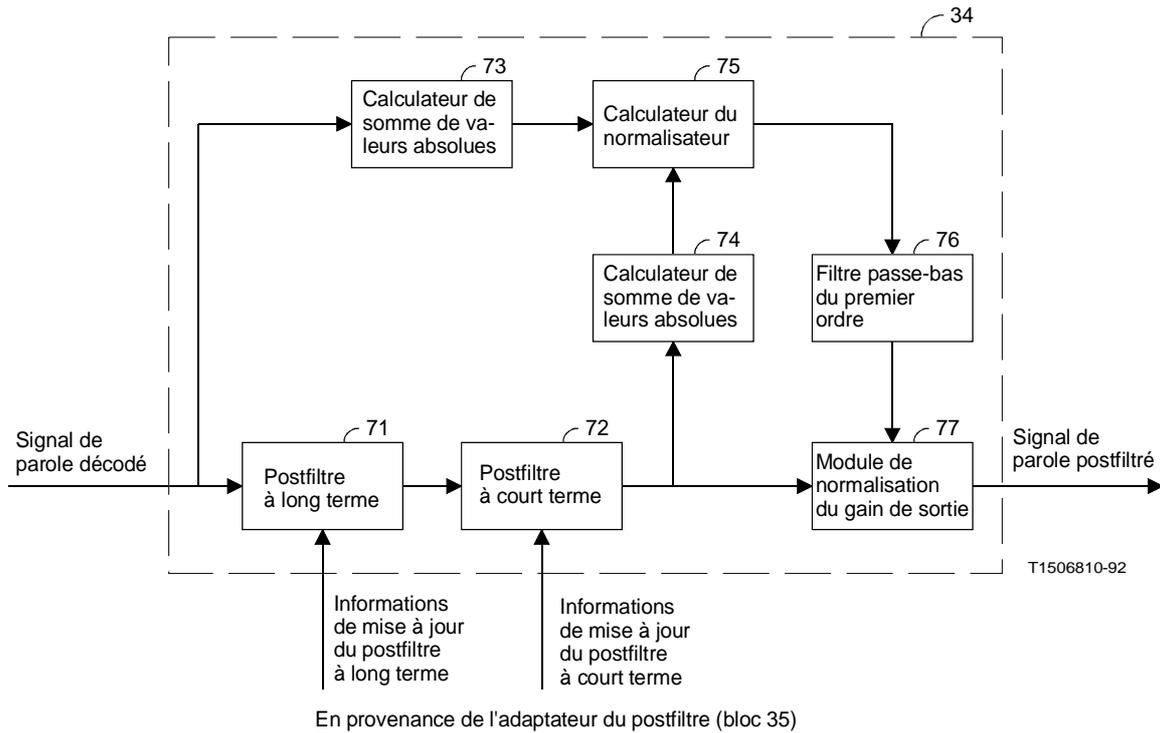


FIGURE 7/G.728

Organigramme du postfiltre

Le *postfiltre à long terme* 71, quelquefois appelé *postfiltre fondamentale* est un filtre en peigne dont les raies spectrales se situent à des multiples de la fréquence fondamentale des signaux de parole à postfiltrer. L'inverse de la *fréquence fondamentale* est appelé *période fondamentale*. La période fondamentale peut être extraite des signaux de parole décodés au moyen d'un détecteur de tonie (ou extracteur de tonie). Soit p la période fondamentale en échantillon obtenue par un détecteur de tonie la fonction de transfert du postfiltre (à long terme) pourra alors être exprimée sous la forme

$$H_l(z) = g_l(1 + b z^{-p}) \quad (4-1)$$

où les coefficients g_l , b et la période fondamentale p sont mis à jour tous les quatre vecteurs de parole (un cycle d'adaptation). Les mises à jour proprement dites interviennent au troisième vecteur de parole de chaque cycle d'adaptation. Par souci de simplification, nous désignerons désormais les cycles d'adaptation par le terme trame. Le calcul de g_l , de b et de p sera décrit au § 4.7.

Le postfiltre à court terme 72 est formé d'un filtre de 10^e ordre monté en cascade avec un filtre tout zéro de premier ordre. Le filtre pôle-zéro de 10^e ordre atténue les composantes fréquentielles entre les raies des formants, alors que le filtre tout zéro de premier ordre tente de compenser les écarts de spectre dans la réponse en fréquence du filtre pôle-zéro de 10^e ordre.

Soient $\tilde{a}_i, i = 1, 2, \dots, 10$ les coefficients du prédicteur à codage linéaire (LPC) de 10^e ordre, obtenus par analyse LPC rétroactive du signal de parole décodé et soit k_1 le premier coefficient de réflexion obtenu par cette analyse LPC. Dans ce cas, on peut obtenir aussi bien les \tilde{a}_i que k_1 sous forme de sous-produits de l'analyse LPC rétroactive de 50^e ordre (bloc 50 de la figure 5/G.728). Tout ce que nous avons à faire est d'arrêter la récurrence Levinson-Durbin au 10^e ordre, de copier k_1 et les $\tilde{a}_1, \tilde{a}_2, \dots, \tilde{a}_{10}$ puis de reprendre la récurrence Levinson-Durbin du 11^e au 50^e ordre. La fonction de transfert du postfiltre à court terme est

$$H_s(z) = \frac{1 - \sum_{i=1}^{10} \bar{b}_i z^{-i}}{1 - \sum_{i=1}^{10} \bar{a}_i z^{-i}} [1 + \mu z^{-1}] \quad (4-2)$$

$$\bar{b}_i = \tilde{a}_i (0,65)^i, i = 1, 2, \dots, 10 \quad (4-3)$$

$$\bar{a}_i = \tilde{a}_i (0,75)^i, i = 1, 2, \dots, 10 \quad (4-4)$$

$$\mu = (0,15) k_1 \quad (4-5)$$

Les coefficients \bar{a}_i, \bar{b}_i et μ sont également mis à jour une fois par trame mais ces mises à jour interviennent au premier vecteur de chaque trame (c'est-à-dire dès que les \tilde{a}_i deviennent accessibles).

En général, après que les signaux de parole décodés aient traversé le postfiltre à long terme et le postfiltre à court terme, le signal de parole filtré n'aura pas le même niveau de puissance que le signal de parole décodé (non filtré). Pour éviter d'éventuelles importantes excursions de gain, il est nécessaire de contrôler automatiquement le gain pour forcer les signaux de parole postfiltrés à avoir à peu près la même puissance que les signaux de parole non filtrés. C'est ce qui est effectué par les blocs 73 à 77.

Le calculateur de somme de valeurs absolues 73 opère vecteur par vecteur. Il prend le vecteur de parole décodé $s_d(n)$ actuel et calcule la somme des valeurs absolues de ses cinq composantes vectorielles. De même, le calculateur de somme de valeurs absolues 74 effectue le même type de calcul mais sur le vecteur de sortie $s_f(n)$ actuel du postfiltre à court terme. Le calculateur de facteur de pondération 75 divise ensuite la valeur de sortie du bloc 73 par la valeur de sortie du bloc 74 pour obtenir un facteur de pondération applicable au vecteur $s_f(n)$ actuel. Ce facteur de sortie est alors filtré par le filtre passe-bas de premier ordre de pondération 76 pour donner un facteur de pondération distinct pour chacune des cinq composantes du vecteur $s_f(n)$. Ce filtre passe-bas de premier ordre 76 a une fonction de transfert de $0,01/(1 - 0,99 z^{-1})$. Le facteur de pondération filtré en passe-bas est ensuite utilisé par le module de normalisation du gain de sortie 77 pour pondérer échantillon par échantillon à la sortie du postfiltre à court terme. Noter que du fait que le calculateur du facteur de pondération 75 ne produit qu'un seul facteur par vecteur, il y aurait un effet d'escalier sur l'opération de pondération échantillon par échantillon du module 77 si le filtre passe-bas 76 n'était pas présent. Le filtre passe-bas 76 assure le lissage d'un tel effet d'escalier.

4.6.1 *Fonctionnement non vocal*

Les résultats d'essai objectifs du CCITT indiquent que, pour certains signaux non vocaux, la qualité de fonctionnement du codeur est améliorée si le postfiltre d'adaptation est désactivé. Comme le signal d'entrée du postfiltre d'adaptation est le signal de sortie du filtre de synthèse, ce signal est toujours disponible. Dans une mise en œuvre réelle, ce signal non filtré doit être prélevé lorsque l'interrupteur a été positionné de manière à désactiver le postfiltre.

Ce bloc calcule et met à jour les coefficients du postfiltre une fois par trame. Cet adaptateur de postfiltre est agrandi sur la figure 8/G.728.

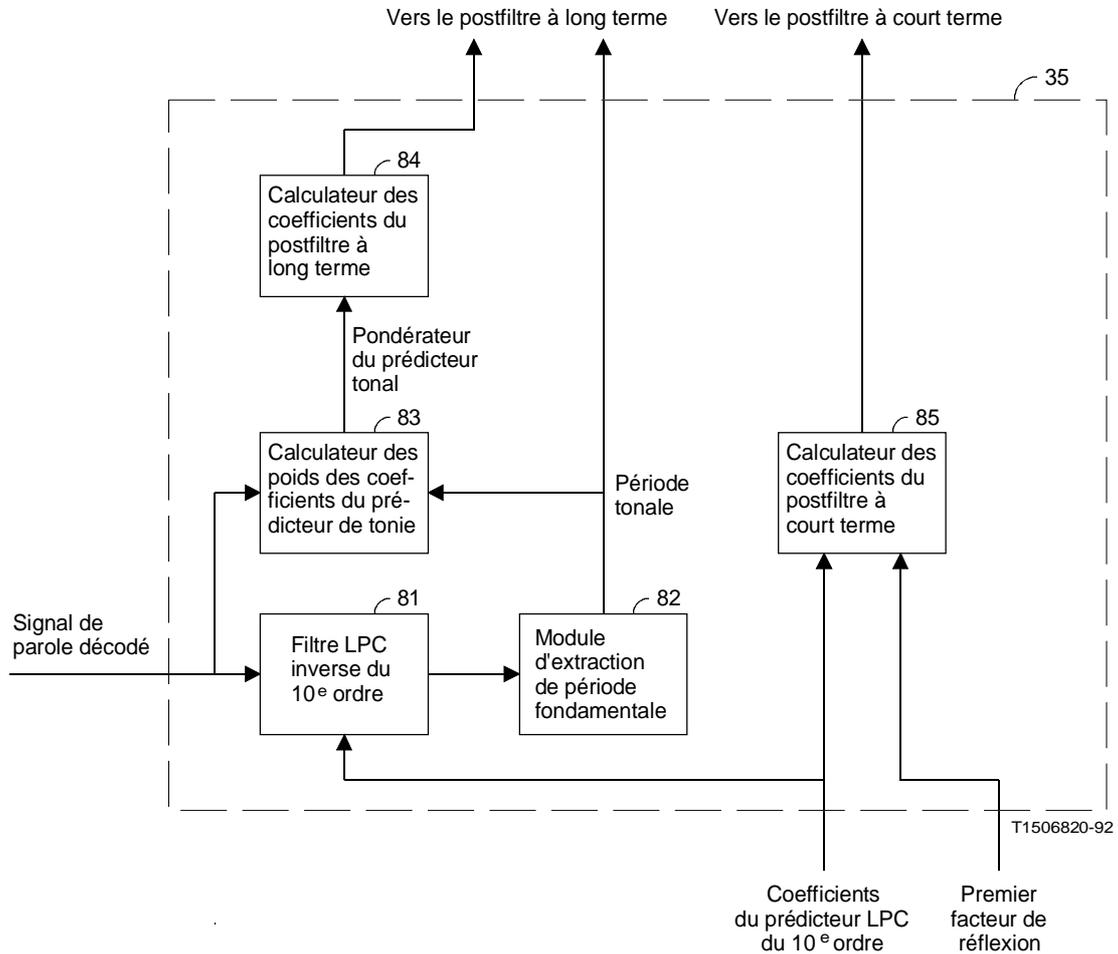


FIGURE 8/G.728

Organigramme de l'adaptateur du postfiltre

Se référer à la figure 8/G.728. Le filtre LPC inverse de 10^e ordre 81 et le module d'extraction de la période fondamentale 82 fonctionnent ensemble pour extraire la période fondamentale du signal de parole décodé. On pourra en fait utiliser ici tout extracteur de tonie de performance acceptable (sans introduire un retard supplémentaire). Le dispositif que nous décrivons ici n'est qu'une des manières possibles de mettre en œuvre un extracteur de tonie.

Le filtre LPC inverse du 10^e ordre 81 a la fonction de transfert

$$\tilde{A}(z) = 1 - \sum_{i=1}^{10} \tilde{a}_i z^{-i} \tag{4-6}$$

dont les coefficients \tilde{a}_i sont fournis par le module de récurrence Levinson-Durbin (bloc 50 de la figure 5/G.728) et sont mis à jour au premier vecteur de chaque trame. Ce filtre LPC inverse prend en entrée le signal de parole décodé et produit en sortie la séquence résiduelle de prédiction LPC $\{d(k)\}$. Nous utilisons une fenêtre d'analyse fondamentale

présentant une dimension de 100 échantillons et une période fondamentale s'étendant de 20 à 140 échantillons. Le module d'extraction de la période fondamentale 82 maintient un tampon de capacité suffisante pour contenir les 240 derniers échantillons des résidus de prédiction LPC. Pour faciliter l'indexation, ces 240 échantillons résiduels LPC, stockés dans le tampon, sont indexés sous la forme $d(-139), d(-138), \dots, d(100)$.

Le module d'extraction de la période fondamentale 82 prélève celle-ci une fois par trame, au troisième vecteur de chaque trame. Il convient donc que les vecteurs issus du filtre LPC inverse soient stockés dans un ordre spécial dans le tampon de résidus LPC: le vecteur LPC résiduel correspondant au quatrième vecteur de la dernière trame est stocké sous la forme $d(81), d(82), \dots, d(85)$; le vecteur LPC résiduel correspondant au premier vecteur de la trame actuelle est stocké sous la forme $d(86), d(87), \dots, d(90)$; le vecteur LPC résiduel correspondant au deuxième vecteur de la trame actuelle est enregistré sous la forme $d(91), d(92), \dots, d(95)$; et le vecteur LPC résiduel correspondant au troisième vecteur est stocké sous la forme $d(96), d(97), \dots, d(100)$. Les échantillons $d(-139), d(-138), \dots, d(80)$ sont simplement les précédents échantillons de résidus LPC rangés dans l'ordre chronologique correct.

Une fois le tampon de résidus LPC prêt, le module d'extraction de la période fondamentale 82 fonctionne comme suit. Tout d'abord, les 20 derniers échantillons du tampon de résidus LPC [$d(81)$ à $d(100)$] sont filtrés en passe-bas à 1 kHz par un filtre à fonction elliptique du troisième ordre (les coefficients sont indiqués dans l'annexe C) puis décimés dans le rapport 4:1 (c'est-à-dire échantillonnés 4 fois moins souvent). Cela donne cinq échantillons de résidus LPC filtrés en passe-bas et décimés, notés $\bar{d}(21), \bar{d}(22), \dots, \bar{d}(25)$, qui sont stockés en tant que les cinq derniers échantillons dans un tampon de résidus LPC décimés. En plus de ces cinq échantillons, les 55 autres échantillons $\bar{d}(-34), \bar{d}(-33), \dots, \bar{d}(20)$ contenus dans le tampon de résidus LPC décimés sont obtenus en décalant les précédentes trames d'échantillons de résidus LPC décimés. Les $i^{\text{èmes}}$ corrélations entre échantillons de résidus LPC décimés sont alors calculées comme suit

$$\rho(i) = \sum_{n=1}^{25} \bar{d}(n) \bar{d}(n-i) \quad (4-7)$$

pour des décalages temporels $i = 5, 6, 7, \dots, 35$ (correspondant à des périodes fondamentales de 20 à 140 échantillons). On recherche ensuite le décalage τ qui donne la plus grande des 31 valeurs de corrélation calculées. Comme ce décalage τ est le retard dans le domaine résiduel décimé à 4:1, le décalage correspondant, qui donnera la corrélation maximale dans le domaine résiduel non décimé original, devrait être compris entre $4\tau - 3$ et $4\tau + 3$. Pour obtenir la résolution temporelle initiale, nous utilisons ensuite le tampon de résidus LPC non décimés afin de calculer la corrélation entre résidus LPC non décimés.

$$C(i) = \sum_{k=1}^{100} d(k) d(k-i) \quad (4-8)$$

pour 7 décalages temporels $i = 4\tau - 3, 4\tau - 2, \dots, 4\tau + 3$. Parmi ces 7 décalages, on repère le décalage p_0 qui donne la meilleure corrélation.

La décalage p_0 ainsi déterminé pourra se révéler être un multiple de la vraie période fondamentale. Ce dont nous avons besoin dans le postfiltre à long terme, c'est de la période fondamentale vraie et non d'un de ses multiples. Il nous faut donc poursuivre le traitement pour la trouver. Nous pouvons tirer parti du fait que nous effectuons de très fréquentes estimations de la période fondamentale – tous les 20 échantillons de parole. Comme la période fondamentale varie normalement entre 20 et 140 échantillons, cette fréquence d'estimation de la période fondamentale implique qu'au début de chaque salve de parole nous obtiendrons la période fondamentale avant que ses multiples aient eu l'occasion d'apparaître dans le processus d'échantillonnage des pics de corrélation décrit ci-dessus. A partir de là, nous aurons la possibilité de nous verrouiller sur la période fondamentale en vérifiant la présence d'un quelconque pic de corrélation au voisinage de la période fondamentale de la trame précédente.

Soit \hat{p} la période fondamentale de la trame précédente. Si le décalage temporel p_0 obtenu ci-dessus n'est pas dans le voisinage de \hat{p} , nous devons également évaluer l'équation (4-8) pour $i = \hat{p} - 6, \hat{p} - 5, \dots, \hat{p} + 5, \hat{p} + 6$. Parmi ces 13 décalages possibles, on repérera le décalage p_1 qui donne la meilleure corrélation. On effectuera ensuite un essai

pour voir si ce nouveau décalage p_1 doit être utilisé en tant que période fondamentale en sortie de la trame actuelle. Nous calculerons d'abord

$$\beta_0 = \frac{\sum_{k=1}^{100} d(k) d(k-p_0)}{\sum_{k=1}^{100} d(k-p_0) d(k-p_0)} \quad (4-9)$$

qui est le gain optimal d'un prédicteur de tonie à un coefficient ayant un décalage temporel de p_0 échantillons. La valeur de β_0 sera ensuite limitée à l'intervalle $[0,1]$. Nous calculerons ensuite

$$\beta_1 = \frac{\sum_{k=1}^{100} d(k) d(k-p_1)}{\sum_{k=1}^{100} d(k-p_1) d(k-p_1)} \quad (4-10)$$

qui est le gain optimal d'un prédicteur de tonie à un coefficient ayant un décalage temporel de p_1 échantillons. La valeur de β_1 sera également limitée à l'intervalle $[0,1]$. La période fondamentale p à la sortie du bloc 82 sera donnée par

$$p = \begin{cases} p_0 & \text{si } \beta_1 \leq 0,4 \beta_0 \\ p_1 & \text{si } \beta_1 > 0,4 \beta_0 \end{cases} \quad (4-11)$$

Une fois que le module d'extraction de période fondamentale 82 a extrait la période fondamentale p , le calculateur de gains de prédicteur de tonie 83 calcule le gain optimal d'un prédicteur de tonie à un coefficient pour le signal de parole décodé. Le calculateur de gains de prédicteur de tonie 83 et le postfiltre à long terme 71 partagent un tampon contenant un grand nombre d'échantillons de parole décodés: $s_d(-239)$, $s_d(-238)$, $s_d(-237)$, ..., $s_d(4)$, $s_d(5)$ où $s_d(1)$ à $s_d(5)$ correspondent au vecteur courant de parole décodé. Le postfiltre à long terme 71 utilisera ce tampon comme ligne à retard du filtre. Par ailleurs, le calculateur de gains de prédicteur de tonie 83 utilisera ce tampon pour calculer

$$\beta = \frac{\sum_{k=-99}^0 s_d(k) s_d(k-p)}{\sum_{k=-99}^0 s_d(k-p) s_d(k-p)} \quad (4-12)$$

Le calculateur de coefficients de filtre à long terme 84 prélève ensuite la période fondamentale et le gain de prédicteur de tonie β pour calculer les coefficients du filtre à long terme, b et g_l , comme suit

$$b = \begin{cases} 0 & \text{si } \beta < 0,6 \\ 0,15 \beta & \text{si } 0,6 \leq \beta \leq 1 \\ 0,15 & \text{si } \beta > 1 \end{cases} \quad (4-13)$$

$$g_l = \frac{1}{1 + b}$$

En général, plus β est proche de l'unité, plus le signal de parole est périodique. Comme on peut le voir dans les équations (4-13) et (4-14), si $\beta < 0,6$, ce qui correspond en gros aux régions non voisées ou transitoires de la parole, on a $b = 0$ et $g_l = 1$. La fonction de transfert du postfiltre à long terme devient alors $H_l(z) = 1$, c'est-à-dire que l'opération de filtrage par le postfiltre à long terme est totalement inhibée. Par ailleurs, si $0,6 \leq \beta \leq 1$, le postfiltre est activé et le degré de filtrage (en peigne) est déterminé par β . Plus le signal de parole est périodique, plus le filtrage en peigne intervient. Finalement, si $\beta > 1$, b sera limité à 0,15 ce qui évitera de trop utiliser le filtre en peigne. Le coefficient g_l est un facteur de normalisation du postfiltre à long terme qui fait en sorte que les régions voisées des signaux de parole ne subissent pas d'amplification par rapport aux régions non voisées ou transitoires. (Si g_l était maintenu constant à l'unité, les régions voisées seraient, à l'issue du filtrage à long terme, amplifiées d'un facteur de $1 + b$ environ. Cela rendrait certaines consonnes qui correspondent à des régions non voisées ou transitoires, peu distinctes ou trop affaiblies.)

Le calculateur de coefficients du postfiltre à court terme 85 calcule les coefficients du postfiltre à court terme \bar{a}_i , \bar{b}_i et μ au premier vecteur de chaque trame, selon les équations (4-3), (4-4) et (4-5).

4.8 Conversion au format MIC de sortie

Ce bloc convertit les cinq composantes du vecteur de parole décodé en cinq échantillons MIC loi A ou loi μ puis sort ces cinq échantillons MIC séquentiellement à des intervalles de 125 μ s. Noter que si le format linéaire MIC interne a été normalisé comme indiqué au § 3.1.1, il est indispensable d'effectuer une normalisation inverse avant la conversion en MIC loi A ou loi μ .

5 Détails relatifs aux calculs

Ce paragraphe traite des détails relatifs aux calculs pour chacun des éléments du codeur et du décodeur LD-CELP. Les § 5.1 et 5.2 énumèrent les noms des paramètres du codeur et les variables de traitement interne auxquels il sera fait référence dans les paragraphes ultérieurs. La spécification particulière de chaque bloc des figures 2/G.728 à 6/G.728 est donnée dans les § 5.3 et suivants. Pour coder et décoder un vecteur de parole d'entrée, les divers blocs du codeur et du décodeur sont mis en fonction dans un ordre qui suit à peu près celui des § 5.3 et suivants.

5.1 Description des paramètres de base du codeur

Les noms des paramètres de base du codeur sont définis dans le tableau 1/G.728. Dans le tableau 1/G.728, la première colonne donne le nom des paramètres de codeur qui seront utilisés dans une description, exposée plus loin, de l'algorithme LD-CELP. Si un paramètre a été cité dans les paragraphes 3 ou 4 mais a été représenté par un symbole différent, ce symbole équivalent sera indiqué dans la deuxième colonne pour faciliter les références. Chaque paramètre de codeur a une valeur fixe qui est déterminée dans la phase de conception du codeur. La troisième colonne montre ces valeurs paramétriques fixes et la quatrième colonne donne une brève description des paramètres du codeur.

5.2 Description des variables internes

Les variables de traitement interne de l'algorithme LD-CELP sont énumérées au tableau 2/G.728, qui a une présentation analogue à celle du tableau 1/G.728. La deuxième colonne montre la plage d'index dans chaque tableau de variables. La quatrième colonne donne les valeurs initiales recommandées des variables. Les valeurs initiales de certains tableaux sont données dans les annexes A, B ou C. Il est recommandé (quoique non exigé) que les variables internes soient réglées à leur valeur initiale lorsque le codeur ou décodeur commence juste à fonctionner ou chaque fois qu'il est nécessaire de réinitialiser les états du codeur (comme dans les applications des équipements de multiplication de circuit numérique (DCME) (*digital circuit multiplication equipment*)). Ces valeurs initiales font en sorte qu'il n'y ait pas d'aléas parasites immédiatement après les démarrages ou les réinitialisations.

Noter que certaines tables de variables, bien que celles-ci aient reçu des noms différents selon chaque tableau pour plus de clarté, peuvent partager les mêmes emplacements physiques de mémoire afin d'économiser la capacité mémoire.

TABLEAU 1/G.728

Paramètres de base du codeur LD-CELP

| Nom | Symbole équivalent | Valeur | Description |
|---------|--------------------|---------|---|
| AGCFAC | | 0,99 | Facteur de commande de la rapidité d'adaptation du contrôle automatique de gain |
| FAC | λ | 253/256 | Facteur d'élargissement de bande du filtre de synthèse |
| FACGP | λ_g | 29/32 | Facteur d'élargissement de bande du prédicteur de gain logarithmique |
| DIMINV | | 0,2 | Inverse de dimension vectorielle |
| IDIM | | 5 | Dimension vectorielle (longueur du bloc d'excitation) |
| GOFF | | 32 | Valeur d'écart du gain logarithmique |
| KPDELTA | | 6 | Ecart admissible par rapport à la période fondamentale précédente |
| KPMIN | | 20 | Période fondamentale minimale (échantillons) |
| KPMAX | | 140 | Période fondamentale maximale (échantillons) |
| LPC | | 50 | Ordre du filtre de synthèse |
| LPCLG | | 10 | Ordre du prédicteur de gain logarithmique |
| LPCW | | 10 | Ordre du filtre de pondération perceptive |
| NCWD | | 128 | Taille du répertoire de forme (nombre de «vecteurcode») |
| NFRSZ | | 20 | Taille de la trame (dimension du cycle d'adaptation, en échantillons) |
| NG | | 8 | Taille du répertoire de gain (nombre de niveaux de gain) |
| NONR | | 35 | Nombre d'échantillons fenêtrés non récursifs pour le filtre de synthèse |
| NONRLG | | 20 | Nombre d'échantillons fenêtrés non récursifs pour le prédicteur de gain logarithmique |
| NONRW | | 30 | Nombre d'échantillons fenêtrés non récursifs pour le filtre de pondération |
| NPWSZ | | 100 | Taille (en échantillons) de la fenêtre d'analyse fondamentale |
| NUPDATE | | 4 | Période (en termes de vecteurs) de mise à jour du prédicteur |
| PPFTH | | 0,6 | Seuil pour désactiver le postfiltre fondamental |
| PPFZCF | | 0,15 | Facteur de commande des zéros du postfiltre fondamental |
| SPFPCF | | 0,75 | Facteur de commande des pôles du postfiltre à court terme |
| SPFZCF | | 0,65 | Facteur de commande des zéros du postfiltre à court terme |
| TAPTH | | 0,4 | Seuil de recalage de la période fondamentale |
| TILTF | | 0,15 | Facteur de commande de compensation du niveau spectral |
| WNCF | | 257/256 | Facteur de correction par bruit blanc |
| WPCF | γ_2 | 0,6 | Facteur de commande des pôles du filtre de pondération perceptive |
| WZCF | γ_1 | 0,9 | Facteur de commande des zéros du filtre de pondération perceptive |

Comme cela a été signalé plus haut, la séquence de traitement possède un cycle d'adaptation de base de quatre vecteurs de parole. La variable ICOUNT est utilisée comme index vectoriel. En d'autres termes, ICOUNT = n lorsque le codeur ou décodeur est en train de traiter le $n^{\text{ième}}$ vecteur de parole d'un cycle d'adaptation.

Il y a lieu de remarquer que, pour les besoins de la récurrence de Levinson-Durbin, le premier élément des tables A, ATMP, AWP, AWZ ainsi que GP est toujours 1 et n'est jamais modifié; et que, pour $i \geq 2$, les $i^{\text{ièmes}}$ éléments sont les $(i - 1)^{\text{èmes}}$ éléments des symboles correspondants du § 3.

Dans les paragraphes qui suivent, un astérisque * représentera une multiplication arithmétique.

TABLEAU 2/G.728
Variables de traitement interne LD-CELP

| Nom | Plage d'index de table | Symbole équivalent | Valeur initiale | Description |
|--------|------------------------|--------------------|-----------------|--|
| A | 1 à LPC+1 | $-a_{i-1}$ | 1,0,0,... | Coefficients du filtre de synthèse |
| AL | 1 à 3 | | Annexe D | Coefficients du dénominateur de filtre passe-bas à 1 kHz |
| AP | 1 à 11 | $-\bar{a}_{i-1}$ | 1,0,0,... | Coefficients du dénominateur de postfiltre à court terme |
| APF | 1 à 11 | $-\tilde{a}_{i-1}$ | 1,0,0,... | Coefficients de filtre LPC du 10 ^e ordre |
| ATMP | 1 à LPC+1 | $-a_{i-1}$ | | Tampon temporaire pour les coefficients du filtre de synthèse |
| AWP | 1 à LPCW+1 | | 1,0,0,... | Coefficients du dénominateur de filtre de pondération perceptive |
| AWZ | 1 à LPCW+1 | | 1,0,0,... | Coefficients du numérateur de filtre de pondération perceptive |
| AWZTMP | 1 à LPCW+1 | | 1,0,0,... | Tampon temporaire pour les coefficients du filtre de pondération |
| AZ | 1 à 11 | $-\bar{b}_{i-1}$ | 1,0,0,... | Coefficients du numérateur de postfiltre à court terme |
| B | 1 | b | 0 | Coefficients du postfiltre à long terme |
| BL | 1 à 4 | | Annexe D | Coefficients du numérateur de filtre passe-bas à 1 kHz |
| DEC | -34 à 25 | $\bar{d}(n)$ | 0,0,...,0 | Résidu de prédiction LPC, décimé à 4:1 |
| D | -139 à 100 | $d(k)$ | 0,0,...,0 | Résidu de prédiction LPC |
| ET | 1 à IDIM | $e(n)$ | 0,0,...,0 | Vecteur d'excitation normalisé en gain |
| FACV | 1 à LPC+1 | λ^{i-1} | Annexe C | Vecteur d'extension de la largeur de bande du filtre de synthèse |
| FACGPV | 1 à LPCLG+1 | λ_g^{i-1} | Annexe C | Vecteur d'extension de la largeur de bande du prédicteur de gain |
| G2 | 1 à NG | b_i | Annexe B | Deux fois les niveaux de gain dans le répertoire de gain |
| GAIN | 1 | $\sigma(n)$ | | Gain d'excitation |
| GB | 1 à NG-1 | d_i | Annexe B | Point médian entre niveaux de gain adjacents |
| GL | 1 | g_l | 1 | Facteur de normalisation du postfiltre à long terme |
| GP | 1 à LPCLG+1 | $-\alpha_{i-1}$ | 1,-1,0,0,... | Coefficient du prédicteur linéaire de gain logarithmique |
| GPTMP | 1 à LPCLG+1 | $-\alpha_{i-1}$ | | Table temporaire de coefficient du prédicteur linéaire de gain logarithmique |
| GQ | 1 à NG | g_i | Annexe B | Niveaux de gain dans le répertoire de gain |
| GSQ | 1 à NG | c_i | Annexe B | Carrés des niveaux de gain dans le répertoire de gain |

TABLEAU 2/G.728 (suite)

| Nom | Plage d'index de table | Symbole équivalent | Valeur initiale | Description |
|---------|------------------------|--------------------|----------------------|---|
| GSTATE | 1 à LPCLG | $\delta(n)$ | -32,-32,...,-32 | Registre du prédicteur linéaire de gain logarithmique |
| GTMP | 1 à 4 | | -32,-32,-32,-32 | Tampon temporaire de gain logarithmique |
| H | 1 à IDIM | $h(n)$ | 1,0,0,0,0 | Vecteur de réponse impulsionnelle de $F(z)W(z)$ |
| ICHAN | 1 | | | Meilleur index de répertoire d'excitation à transmettre |
| ICOUNT | 1 | | | Compteur de vecteurs de parole (indexé de 1 à 4) |
| IG | 1 | i | | Meilleur index à 3 bit du répertoire de gain |
| IP | 1 | | IPINIT ^{b)} | Pointeur d'adresse vers le résidu de prédiction LPC |
| IS | 1 | j | | Meilleur index à 7 bits du répertoire de forme |
| KP | 1 | p | | Période fondamentale de la trame courante |
| KP1 | 1 | \hat{p} | 50 | Période fondamentale de la trame précédente |
| PN | 1 à IDIM | $p(n)$ | | Vecteur de corrélation pour la recherche dans le répertoire |
| PTAP | 1 | β | | Pondérateur de prédicteur tonal calculé par le bloc 83 |
| R | 1 à NR+1 ^{a)} | | | Coefficients d'autocorrélation |
| RC | 1 à NR ^{a)} | | | Facteur de réflexion, servant également de tableau de travail |
| RCTMP | 1 à LPC | | | Tampon temporaire du facteur de réflexion |
| REXP | 1 à LPC+1 | | 0,0,...,0 | Partie récurrente de l'autocorrélation – Filtre de synthèse |
| REXP LG | 1 à LPCLG+1 | | 0,0,...,0 | Partie récurrente de l'autocorrélation – Prédicteur de gain logarithmique |
| REXPW | 1 à LPCW+1 | | 0,0,...,0 | Partie récurrente de l'autocorrélation – Filtre de pondération |
| RTMP | 1 à LPC+1 | | | Tampon temporaire pour coefficients d'autocorrélation |
| S | 1 à IDIM | $s(n)$ | 0,0,...,0 | Vecteur de parole d'entrée MIC uniforme |
| SB | 1 à 105 | | 0,0,...,0 | Tampon pour signaux de parole préalablement quantifiés |
| SBLG | 1 à 34 | | 0,0,...,0 | Tampon pour le gain logarithmique précédent |
| SBW | 1 à 60 | | 0,0,...,0 | Tampon pour le signal de parole d'entrée précédent |
| SCALE | 1 | | | Facteur de normalisation de postfiltre filtré en passe-bas |

TABLEAU 2/G.728 (suite)

| Nom | Plage d'index de table | Symbole équivalent | Valeur initiale | Description |
|----------|------------------------|--------------------|-----------------|---|
| SCALEFIL | 1 | | 1 | Facteur de normalisation de postfiltre filtré en passe-bas |
| SD | 1 à IDIM | $s_d(k)$ | | Tampon de parole décodé |
| SPF | 1 à IDIM | | | Vecteur de parole postfiltré |
| SPFPCFV | 1 à 11 | $SPFPCF^{i-1}$ | Annexe C | Vecteur de commande des pôles du postfiltre à court terme |
| SPFZCFV | 1 à 11 | $SPFZCF^{i-1}$ | Annexe C | Vecteur de commande des zéros du postfiltre à court terme |
| SO | 1 | $s_o(k)$ | | Echantillon de parole d'entrée MIC loi A ou loi μ |
| SU | 1 | $s_u(k)$ | | Echantillon de parole d'entrée MIC uniforme |
| ST | -239 à IDIM | $s_q(n)$ | 0,0,...,0 | Vecteur de parole quantifié |
| STATELPC | 1 à LPC | | 0,0,...,0 | Mémoire du filtre de synthèse |
| STLPCI | 1 à 10 | | 0,0,...,0 | Mémoire du filtre inverse LPC |
| STLPF | 1 à 3 | | 0,0,0 | Mémoire du filtre passe-bas à 1 kHz |
| STMP | 1 à 4*IDIM | | 0,0,...,0 | Tampon pour fenêtre hybride du filtre de pondération perceptive |
| STPFIR | 1 à 10 | | 0,0,...,0 | Mémoire du postfiltre à court terme – section tous zéros |
| STPFIIR | 10 | | 0,0,...,0 | Mémoire du postfiltre à court terme – section tous pôles |
| SUMFIL | 1 | | | Somme des valeurs absolues du signal de parole postfiltré |
| SUMUNFIL | 1 | | | Somme des valeurs absolues du signal de parole décodé |
| SW | 1 à IDIM | $v(n)$ | | Vecteur de parole issu du filtre de pondération perceptive |
| TARGET | 1 à IDIM | $\hat{x}(n); x(n)$ | | Vecteur VQ cible (normalisé en gain) |
| TEMP | 1 à IDIM | | | Table de travail pour espace de travail temporaire |
| TILTZ | 1 | μ | 0 | Coefficient de compensation d'écart de niveau du postfiltre à court terme |
| WFIR | 1 à LPCW | | 0,0,...,0 | Mémoire du filtre de pondération 4 – section tous zéros |
| WIIR | 1 à LPCW | | 0,0,...,0 | Mémoire du filtre de pondération 4 – section tous pôles |
| WNR | 1 à 105 | $w_m(k)$ | Annexe A | Fonction de fenêtrage pour filtre de synthèse |
| WNRLG | 1 à 34 | $w_m(k)$ | Annexe A | Fonction de fenêtrage pour prédicteur de gain logarithmique |
| WNRW | 1 à 60 | $w_m(k)$ | Annexe A | Fonction de fenêtrage pour filtre de pondération |

TABLEAU 2/G.728 (suite)

| Nom | Plage d'index de table | Symbole équivalent | Valeur initiale | Description |
|---------|------------------------|--------------------|-------------------|---|
| WPCFV | 1 à LPCW+1 | γ_2^{i-1} | Annexe C | Vecteur de commande des pôles du filtre de pondération perceptive |
| WS | 1 à 105 | | | Table d'espace de travail pour variables intermédiaires |
| WZCFV | 1 à LPCW+1 | γ_1^{i-1} | Annexe C | Vecteur de commande des zéros du filtre de pondération perceptive |
| Y | 1 à IDIM*NCWD | y_j | Annexe B | Table du répertoire de forme |
| Y2 | 1 à NCWD | E_j | Energie des y_j | Energie du vecteurcode de forme convolué |
| YN | 1 à IDIM | $y(n)$ | | Vecteur d'excitation quantifié |
| ZIRWFIR | 1 à LPCW | | 0,0,...,0 | Mémoire du filtre de pondération 10 –section tous zéros |
| ZIRWIIR | 1 à LPCW | | 0,0,...,0 | Mémoire du filtre de pondération 10 –section tous pôles |

a) $NR = \text{Max}(LPCW, LPCLG) > IDIM$.

b) $IPINIT = NPWSZ - NFRSZ + IDIM$.

Remarque – Dans le présent tableau, un astérisque * représente une multiplication arithmétique.

5.3 Conversion du format MIC d'entrée (bloc 1)

Entrée: SO

Sortie: SU

Fonction: Convertir en échantillon MIC uniforme l'échantillon d'entrée loi A, loi μ ou MIC linéaire de 16 bits.

Etant donné que le fonctionnement de ce bloc est complètement défini dans la Recommandation G.721 ou G.711, nous ne le reprendrons pas ici. Il convient cependant de rappeler que, selon le § 3.1.1, une certaine normalisation peut être nécessaire afin de satisfaire à la spécification d'une plage d'entrée de -4095 à $+4095$ conformément à la présente description.

5.4 Tampon vectoriel (bloc 2)

Entrée: SU

Sortie: S

Fonction: Met en mémoire cinq échantillons de parole MIC uniformes consécutifs pour former un unique vecteur de parole à cinq dimensions.

5.5 *Adaptateur pour filtre de pondération perceptive (bloc 3, figure 4a)/G.728*

Les trois blocs (36, 37 et 38) de la figure 4a)/G.728 sont spécifiés en détail ci-dessous.

MODULE DE FENÊTRAGE HYBRIDE (bloc 36)

Entrée: STMP

Sortie: R

Fonction: Appliquer la fenêtre hybride sur le signal de parole d'entrée et calculer des coefficients d'autocorrélation.

Le fonctionnement de ce module est décrit ci-dessous dans un style de type Fortran, les frontières de boucle étant indiquées par des retraits et les commentaires étant précédés d'une barre verticale «|». L'algorithme ci-après sera utilisé à chaque cycle d'adaptation (tous les 20 échantillons). La table STMP contient quatre vecteurs de parole d'entrée consécutifs, jusqu'au deuxième vecteur de parole du cycle d'adaptation courant. C'est-à-dire que les valeurs STMP(1) à STMP(5) correspondent au troisième vecteur de parole d'entrée du précédent cycle d'adaptation (initialement à zéro); STMP(6) à STMP(10) correspondent au quatrième vecteur de parole d'entrée du cycle d'adaptation précédent (initialement à zéro); STMP(11) à STMP(15) correspondent au premier vecteur de parole d'entrée du cycle d'adaptation courant et STMP(16) à STMP(20) correspondent au deuxième vecteur de parole d'entrée du cycle d'adaptation courant.

```

N1 = LPCW + NFRSZ           | Calculer quelques constantes
N2 = LPCW + NONRW          | (qui pourront être précalculées
N3 = LPCW + NFRSZ + NONRW  | et mises en mémoire)

For N = 1,2,...,N2, do the next line
  SBW(N) = SBW(N + NFRSZ)   | Décaler le tampon de l'ancien signal

For N = 1,2,...,NFRSZ, do the next line
  SBW(N2 + N) = STMP(N)    | Introduire par décalage le nouveau signal;
                           | SBW(N3) est le plus récent échantillon

K = 1
For N = N3,N3 - 1,...,3,2,1, do the next two lines
  WS(N) = SBW(N) * WNRW(K) | Multiplier par la fonction de fenêtrage
  K = K + 1

For I = 1,2,...,LPCW + 1, do the next four lines
  TMP = 0
  For N = LPCW + 1,LPCW + 2,...,N1, do the next line
    TMP = TMP + WS(N) * WS(N + 1 - I)
  REXPW(I) = (1/2) * REXPW(I) + TMP | Mettre à jour la composante récurrente

For I = 1,2,...,LPCW + 1, do the next three lines
  R(I) = REXPW(I)
  For N = N1 + 1,N1 + 2,...,N3, do the next line
    R(I) = R(I) + WS(N) * WS(N + 1 - I) | Ajouter la composante non récurrente

R(1) = R(1) * WNCF         | Correction par bruit blanc

```

MODULE DE RÉCURRENCE DE LEVINSON-DURBIN (bloc 37)

Entrée: R (sortie du block 36)

Sortie: AWZTMP

Fonction: Convertir les coefficients d'autocorrélation en coefficients du prédicteur linéaire.

Ce bloc est exécuté à chaque cycle d'adaptation de quatre vecteurs. Il est fait à ICOUNT = 3, une fois que le traitement du bloc 36 est terminé. Comme la récurrence de Levinson-Durbin est une méthode déjà bien connue, l'algorithme est reproduit ci-dessous sans explications.

| | |
|--|---|
| If R(LPCW + 1) = 0, go to LABEL | Sauter à LABEL si valeur nulle |
| If R(1) ≤ 0, go to LABEL | Sauter à LABEL si signal nul |
| RC(1) = R(2)/R(1) | |
| AWZTMP(1) = 1 | |
| AWZTMP(2) = RC(1) | Prédicteur du premier ordre |
| ALPHA = R(1) + R(2) * RC(1) | |
| If ALPHA ≤ 0, go to LABEL | Sauter à LABEL si défaut de conditionnement |
| For MINC = 2,3,4,...,LPCW, do the following: | |
| SUM = 0 | |
| For IP = 1,2,3,...,MINC, do the next two lines | |
| N1 = MINC - IP + 2 | |
| SUM = SUM + R(N1) * AWZTMP(IP) | |
| | |
| RC(MINC) = -SUM/ALPHA | Facteur de réflexion |
| MH = MINC/2 + 1 | |
| For IP = 2,3,4,...,MH, do the next four lines | |
| IB = MINC - IP + 2 | |
| AT = AWZTMP(IP) + RC(MINC) * AWZTMP(IB) | |
| AWZTMP(IB) = AWZTMP(IB) + RC(MINC) AWZTMP(IP) | Coefficient du prédicteur |
| AWZTMP(IP) = AT | |
| | |
| AWZTMP(MINC + 1) = RC(MINC) | |
| ALPHA = ALPHA + RC(MINC) * SUM | Energie résiduelle de prédiction |
| If Alpha ≤ 0, go to LABEL | Abandonner si défaut conditionnement |
| | |
| Repeat the above for the next MINC | |
| Exit this program | Fin normale du programme si |
| | l'exécution parvient à ce point |

LABEL: Si le programme parvient à ce point, un défaut de conditionnement s'est produit. Dans ce cas sauter le bloc 38, ne pas mettre à jour les coefficients du filtre de pondération (c'est-à-dire utiliser les coefficients de filtre de pondération du précédent cycle d'adaptation).

CALCULATEUR DE COEFFICIENTS DE FILTRE DE PONDÉRATION (bloc 38)

Entrée: AWZTMP

Sorties: AWZ, AWP

Fonction: Calculer les coefficients du filtre de pondération perceptive d'après les coefficients du prédicteur linéaire pour les signaux de parole d'entrée.

Ce bloc est exécuté une fois par cycle d'adaptation. Il est fait à ICOUNT = 3, une fois que le traitement du bloc 37 est terminé.

```
For I = 2,3,...,LPCW + 1, do the next line          |  
  AWP(I) = WPCFV(I) * AWZTMP(I)                   | Coefficients du dénominateur  
  
For I = 2,3,...,LPCW + 1, do the next line          |  
  AWZ(I) = WZCFV(I) * AWZTMP(I)                   | Coefficients du numérateur
```

5.6 *Adaptateur en boucle du filtre de synthèse (bloc 23, figure 5/G.728)*

Les trois blocs (49, 50 et 51) de la figure 5/G.728 sont spécifiés ci-dessous.

MODULE DE FENÊTRAGE HYBRIDE (bloc 49)

Entrée: STTMP

Sortie: RTMP

Fonction: Appliquer la fenêtre hybride aux signaux de parole quantifiés et calculer les coefficients d'autocorrélation.

Le fonctionnement de ce bloc est pratiquement le même que celui du bloc 36, à l'exception de quelques substitutions de paramètres et de variables ainsi que de l'instant de l'échantillonnage où les coefficients d'autocorrélation sont obtenus. Comme cela est décrit au § 3, les coefficients d'autocorrélation sont calculés sur la base des vecteurs de parole quantifiés jusqu'au dernier vecteur du cycle d'adaptation quadrivectoriel précédent. En d'autres termes, les coefficients d'autocorrélation utilisés dans le cycle d'adaptation courant sont fondés sur les informations contenues dans le signal de parole quantifié jusqu'au dernier (20^e) échantillon du cycle d'adaptation précédent. (C'est en fait de cette manière que l'on définit le cycle d'adaptation.) La table STTMP contient les quatre vecteurs de parole quantifiés du précédent cycle d'adaptation.

| | |
|--|--|
| N1 = LPC + NFRSZ | Calculer quelques constantes |
| N2 = LPC + NONR | (qui pourront être précalculées |
| N3 = LPC + NFRSZ + NONR | et mises en mémoire) |
| For N = 1,2,...,N2, do the next line | |
| SB(N) = SB(N + NFRSZ) | Décaler le tampon de l'ancien signal |
| For N = 1,2,...,NFRSZ, do the next line | |
| SB(N2 + N) = STTMP(N) | Introduire par décalage le nouveau signal; |
| | SB(N3) est le plus récent échantillon |
| K = 1 | |
| For N = N3,N3 - 1,...,3,2,1, do the next two lines | |
| WS(N) = SB(N) * WNR(K) | Multiplier par la fonction de fenêtrage |
| K = K + 1 | |
| For I = 1,2,...,LPC + 1, do the next four lines | |
| TMP = 0 | |
| For N = LPC + 1,LPC + 2,...,N1, do the next line | |
| TMP = TMP + WS(N) * WS(N + 1 - I) | |
| REXP(I) = (3/4) * REXP(I) + TMP | Mettre à jour la composante récurrente |
| For I = 1,2,...,LPC + 1, do the next three lines | |
| RTMP(I) = REXP(I) | |
| For N = N1 + 1,N1 + 2,...,N3, do the next line | |
| RTMP(I) = RTMP(I) + WS(N) * WS(N + 1 - I) | Ajouter la composante non récurrente |
| RTMP(1) = RTMP(1) * WNCF | Correction par bruit blanc |

MODULE DE RÉCURRENCE DE LEVINSON-DURBIN (bloc 50)

Entrée: RTMP

Sortie: ATMP

Fonction: Convertir les coefficients d'autocorrélation en coefficients du filtre de synthèse.

Le fonctionnement de ce bloc est exactement le même que celui du bloc 37, à l'exception de quelques substitutions de paramètres et de variables. Il convient cependant de prendre des précautions particulières lors de la mise en œuvre de ce bloc. Comme cela est décrit au § 3, bien que la table RTMP d'autocorrélation soit disponible au premier vecteur de chaque cycle d'adaptation, les mises à jour réelles des coefficients du filtre de synthèse n'interviendront pas avant le troisième vecteur. Ce délai de mise à jour intentionnel permet au matériel temps réel de répartir le calcul de ce module sur les trois premiers vecteurs de chaque cycle d'adaptation. Pendant l'exécution de ce module sur les deux premiers vecteurs de chaque cycle, l'ancien ensemble de coefficients de filtre de synthèse (la table «A») obtenu lors du cycle précédent continue à être utilisé. C'est pourquoi il est nécessaire de maintenir une table ATMP distincte, permettant d'éviter d'écraser l'ancienne table «A». De même, on utilisera RTMP, RCTMP, ALPHATMP, etc. pour éviter des brouillages avec d'autres modules de récurrence de Levinson-Durbin (blocs 37 et 44).

| | |
|--|--|
| If RTMP(LPC + 1) = 0, go to LABEL | Sauter à LABEL si valeur nulle |
| | |
| If RTMP(1) ≤ 0, go to LABEL | Sauter à LABEL si signal nul |
| | |
| RCTMP(1) = -RTMP(2)/RTMP(1) | |
| ATMP(1) = 1 | |
| ATMP(2) = RCTMP(1) | Prédicteur du premier ordre |
| ALPHATMP = RTMP(1) + RTMP(2) * RCTMP(1) | |
| If ALPHATMP ≤ 0, go to LABEL | Abandonner si défaut de conditionnement |
| | |
| For MINC = 2,3,4,...,LPC, do the following: | |
| SUM = 0 | |
| For IP = 1,2,3,...,MINC, do the next two lines | |
| N1 = MINC - IP + 2 | |
| SUM = SUM + RTMP(N1) * ATMP(IP) | |
| | |
| RCTMP(MINC) = -SUM/ALPHATMP | |
| MH = MINC/2 + 1 | Facteur de réflexion |
| For IP = 2,3,4,...,MH, do the next four lines | |
| IB = MINC - IP + 2 | |
| AT = ATMP(IP) + RCTMP(MINC) * ATMP(IB) | |
| ATMP(IB) = ATMP(IB) + RCTMP(MINC) * ATMP(IP) | |
| ATMP(IP) = AT | Mise à jour du coefficient du prédicteur |
| | |
| ATMP(MINC + 1) = RCTMP(MINC) | |
| ALPHATMP = ALPHATMP + RCTMP(MINC) * SUM | Energie résiduelle de prédiction |
| If ALPHATMP ≤ 0, go to LABEL | Abandonner si défaut de conditionnement |
| | |
| | |
| Repeat the above for the next MINC | |
| Exit this program | Récurrence effectuée normalement |
| | si l'exécution parvient à ce point |

LABEL: Si le programme parvient à ce point, un défaut de conditionnement s'est produit. Dans ce cas sauter le bloc 51, ne pas mettre à jour les coefficients du filtre de pondération (c'est-à-dire utiliser les coefficients de filtre de pondération du précédent cycle d'adaptation).

MODULE D'EXTENSION DE LA LARGEUR DE BANDE (bloc 51)

Entrée: ATMP

Sortie: A

Fonction: Normaliser les coefficients du filtre de synthèse pour élargir la bande des pics spectraux.

Ce bloc n'est exécuté qu'une fois par cycle d'adaptation, après la fin de traitement du bloc 50 et avant l'exécution des blocs 9 et 10 à ICOUNT = 3. Lorsque l'exécution de ce module est terminée et que ICOUNT = 3, la table ATMP est copiée dans la table «A» afin de mettre à jour les coefficients du filtre.

| | | |
|--|--|--|
| For I = 1,3,...,LPC + 1, do the next line ATMP(I) = FACV(I) * ATMP(I) | | |
| | | Normalisation des coefficients |
| Wait until ICOUNT = 3, then For I = 2,3,...,LPC + 1, do the next line A(I) = ATMP(I) | | |
| | | Mise à jour des coefficients au troisième vecteur chaque cycle |

5.7 Adaptateur en boucle de gain vectoriel (bloc 20, figure 6/G.728)

Les blocs de la figure 6/G.728 sont spécifiés ci-dessous. Pour améliorer l'efficacité de la mise en œuvre, certains blocs sont décrits ensemble, sous la forme d'un bloc unique (ils ne sont représentés séparément dans la figure 6/G.728 que pour expliquer le principe). Tous les blocs de la figure 6/G.728 sont exécutés une fois par vecteur de parole, sauf pour les blocs 43, 44 et 45, qui ne sont exécutés que lorsque ICOUNT = 2.

DÉLAI DE 1 VECTEUR, CALCULATEUR DE VALEUR QUADRATIQUE MOYENNE ET CALCULATEUR DE LOGARITHMES (blocs 67, 39 et 40)

Entrée: ET

Sortie: ETRMS

Fonction: Calculer le niveau en dB de la valeur quadratique moyenne (RMS) du précédent vecteur d'excitation normalisé en gain.

Lorsque ces trois blocs sont exécutés (ce qui intervient avant la recherche dans le répertoire d'excitation VQ), la table ET contient le vecteur d'excitation normalisé en gain qui a été déterminé pour le précédent vecteur de parole. Le module de délai d'un vecteur (bloc 67) est automatiquement exécuté. (Ce bloc n'apparaît sur la figure 6/G.728 que pour plus de clarté.) Comme le calculateur de logarithmes suit immédiatement le calculateur de valeur quadratique moyenne, l'opération d'extraction de la racine carrée dans le calculateur RMS peut prendre la forme d'une «division par deux» à la sortie du calculateur de logarithmes. La sortie de celui-ci (la valeur en dB) est donc $10 * \log_{10}$ (énergie de ET/IDIM). Pour éviter un débordement de valeur logarithmique lorsque ET = 0 (après initialisation ou remise à zéro du système), l'argument de l'opération logarithmique est forcé à 1 si la valeur est trop petite. De même, nous noterons que ETRMS est habituellement conservé dans un accumulateur car il s'agit d'une valeur temporaire immédiatement traitée dans le bloc 42.

| | | |
|---|--|---------------------------------------|
| ETRMS = ET(1) * ET(1) | | |
| For K = 2,3,...,IDIM, do the next line ETRMS = ETRMS + ET(K) * ET(K) | | Calculer l'énergie de ET |
| ETRMS = ETRMS * DIMINV | | Diviser par IDIM |
| If ETRMS < 1, set ETRMS = 1 | | Forcer à 1 pour éviter un débordement |
| ETRMS = $10 * \log_{10}$ (ETRMS) | | logarithmique |
| | | Calculer la valeur en décibels |

SOUSTRACTEUR D'ÉCART DE GAIN LOGARITHMIQUE (bloc 42)

Entrées: ETRMS, GOFF
Sortie: GSTATE(1)
Fonction: Soustraire la valeur d'écart de gain logarithmique contenue dans le bloc 41 à partir de la sortie du bloc 40 (niveau du gain en dB).

$GSTATE(1) = ETRMS - GOFF$

MODULE DE FENÊTRAGE HYBRIDE (bloc 43)

Entrée: GTMP
Sortie: R
Fonction: Appliquer la fenêtre hybride à la séquence de gain logarithmique à écart soustrait et calculer les coefficients d'autocorrélation.

Le fonctionnement de ce bloc est très semblable à celui du bloc 36, à l'exception de quelques substitutions de paramètres et de variables ainsi que de l'instant de l'échantillonnage où les coefficients d'autocorrélation sont obtenus.

Une importante différence entre le bloc 36 et ce bloc 43 est que, chaque fois que le bloc 43 est exécuté, seuls quatre échantillons de gain (et non 20) traversent ce bloc.

Les coefficients du prédicteur de gain logarithmique sont mis à jour au deuxième vecteur de chaque cycle d'adaptation. La table GTMP ci-dessous contient quatre valeurs de gain logarithmique à écart soustrait, en commençant par le gain logarithmique du deuxième vecteur du précédent cycle d'adaptation pour arriver au gain logarithmique du premier vecteur du cycle d'adaptation courant, qui est GTMP(1). GTMP(4) est la valeur de gain logarithmique à écart soustrait du premier vecteur du cycle d'adaptation courant: il s'agit donc de la plus récente valeur.

| | |
|---|---|
| N1 = LPCLG + NUPDATE | Calculer quelques constantes |
| N2 = LPCLG + NONRLG | (qui peuvent être précalculées et |
| N3 = LPCLG + NUPDATE + NONRLG | mises en mémoire) |
| For N = 1,2,...,N2, do the next line | Décaler le tampon de l'ancien signal |
| SBLG(N) = SBLG(N + NUPDATE) | |
| For N = 1,2,...,NUPDATE, do the next line | Introduire par décalage le nouveau signal |
| SBLG(N2 + N) = GTMP(N) | SBLG(N3) est le plus récent échantillon |
| K = 1 | |
| For N = N3, N3 - 1, ..., 3, 2, 1, do the next two lines | |
| WS(N) = SBLG(N) * WNRLG(K) | Multiplier par la fonction de fenêtrage |
| K = K + 1 | |
| For I = 1,2,...,LPCLG + 1, do the next four lines | |
| TMP = 0 | |
| For N = LPCLG + 1, LPCLG + 2, ..., N1, do the next line | |
| TMP = TMP + WS(N) * WS(N + 1 - I) | |
| REXPLOG(I) = (3/4) * REXPLOG(I) + TMP | Mettre à jour la composante récurrente |
| For I = 1,2,...,LPCLG + 1, do the next three lines | |
| R(I) = REXPLOG(I) | |
| For N = N1 + 1, N1 + 2, ..., N3, do the next line | |
| R(I) = R(I) + WS(N) * WS(N + 1 - I) | Ajouter la composante non récurrente |
| R(1) = R(1) * WNCF | Correction par bruit blanc |

MODULE DE RÉCURRENCE DE LEVINSON-DURBIN (bloc 44)

Entrée: R (sortie du bloc 43)

Sortie: GPTMP

Fonction: Convertir les coefficients d'autocorrélation en coefficients de prédicteur de gain logarithmique.

Le fonctionnement de ce bloc est exactement semblable à celui du bloc 37, à l'exception des substitutions de paramètres et de variables indiquées ci-dessous: remplacer LPCW par LPCLG et AWZ par GP. Ce bloc n'est exécuté que lorsque ICOUNT = 2, après exécution du bloc 43. Noter que dans un premier temps, la valeur de R(LPCLG + 1) sera vérifiée. Si elle est égale à zéro, l'on sautera les blocs 44 et 45 sans mettre à jour les coefficients du prédicteur de gain logarithmique. (C'est-à-dire que l'on continuera à utiliser les anciens coefficients de prédicteur de gain logarithmique, déterminés lors du précédent cycle d'adaptation.) Cette procédure spéciale est conçue de manière à éviter le très faible aléa parasite qui se serait autrement produit immédiatement après l'initialisation ou la remise à zéro du système. Si la matrice présente un défaut de conditionnement, l'on sautera également le bloc 45 et l'on utilisera les anciennes valeurs.

MODULE D'EXTENSION DE LA LARGEUR DE BANDE (bloc 45)

Entrée: GPTMP

Sortie: GP

Fonction: Normaliser les coefficients du prédicteur de gain logarithmique afin d'augmenter la largeur de bande des pics spectraux.

Ce bloc n'est exécuté que lorsque ICOUNT = 2, après exécution du bloc 44.

```
For I = 2,3,...,LPCLG + 1, do the next line          |  
  GP(I) = FACGPV(I) * GPTMP(I)                    | Normalisation des coefficients
```

PRÉDICTEUR LINÉAIRE DE GAIN LOGARITHMIQUE (bloc 46)

Entrées: GP, GSTATE

Sortie: GAIN

Fonction: Prédire la valeur courante du gain logarithmique à écart soustrait.

GAIN = 0

For I = LGLPC, LPCLG - 1, ..., 3, 2, do the next two lines

GAIN = GAIN - GP(I + 1) * GSTATE(I)

GSTATE(I) = GSTATE(I - 1)

GAIN = GAIN - GP(2) * GSTATE(1)

ADDITIONNEUR D'ÉCART DE GAIN LOGARITHMIQUE (entre blocs 46 et 47)

Entrées: GAIN, GOFF

Sortie: GAIN

Fonction: Rajouter la valeur d'écart de gain logarithmique à la sortie du prédicteur de gain logarithmique.

GAIN = GAIN + GOFF

LIMITEUR DE GAIN LOGARITHMIQUE (bloc 47)

Entrée: GAIN
Sortie: GAIN
Fonction: Limiter la plage du gain logarithmique prédit.

If GAIN < 0, set GAIN = 0 | Correspond à un gain linéaire de 1
If GAIN > 60, set GAIN = 60 | Correspond à un gain linéaire de 1000

MODULE D'EXPONENTIATION (bloc 48)

Entrée: GAIN
Sortie: GAIN
Fonction: Reconvertir le gain logarithmique prédit (en dB) en valeurs arithmétiques (domaine linéaire).

$GAIN = 10^{(GAIN/20)}$

5.8 *Filtre de pondération perceptive*

FILTRE DE PONDÉRATION PERCEPTIVE (bloc 4)

Entrées: S, AWZ, AWP
Sortie: SW
Fonction: Filtrer le vecteur de parole d'entrée pour réaliser la pondération perceptive.

For K = 1,2,...,IDIM, do the following:
SW(K) = S(K)
For J = LPCW,LPCW - 1,...,3,2, do the next two lines
SW(K) = SW(K) + WFIR(J) * AWZ(J + 1) |
WFIR(J) = WFIR(J - 1) | Section tous zéros du filtre

SW(K) = SW(K) + WFIR(1) * AWZ(2) | Traiter le dernier vecteur
WFIR(1) = S(K) | différemment

For J = LPCW,LPCW - 1,...,3,2, do the next two lines |
SW(K) = SW(K) - WIIR(J) * AWP(J + 1) | Section tous pôles du filtre
WIIR(J) = WIIR(J - 1)

SW(K) = SW(K) - WIIR(1) * AWP(2) |
WIIR(1) = SW(K) | Traiter le dernier vecteur différemment

Repeat the above for the next K

5.9 Calcul du vecteur de réponse à entrée nulle

Le § 3.5 explique comment un «vecteur de réponse à entrée nulle» est calculé par les blocs 9 et 10. Le fonctionnement de ces deux blocs pendant cette phase $r(n)$ est spécifié ci-dessous. Leur fonctionnement pendant la «phase de mise à jour de la mémoire» sera décrit ultérieurement.

FILTRE DE SYNTHÈSE (bloc 9) PENDANT LE CALCUL DE LA RÉPONSE À ENTRÉE NULLE

Entrées: A, STATELPC

Sortie: TEMP

Fonction: Calculer le vecteur de réponse à entrée nulle du filtre de synthèse.

For K = 1,2,...,IDIM, do the following:

TEMP(K) = 0

For J = LPC,LPC - 1,...,3,2, do the next two lines

TEMP(K) = TEMP(K) - STATELPC(J) * A(J + 1)

| Multiplier-ajouter

STATELPC(J) = STATELPC(J - 1)

| Décalage mémoire

TEMP(K) = TEMP(K) - STATELPC(1) * A(2)

|

STATELPC(1) = TEMP(K)

| Traiter le dernier vecteur différemment

Repeat the above for the next K

FILTRE DE PONDÉRATION PERCEPTIVE (bloc 10) PENDANT LE CALCUL DE LA RÉPONSE À ENTRÉE NULLE

Entrées: AWZ, AWP, ZIRWFIR, ZIRWIIR, TEMP calculé ci-dessus

Sortie: ZIR

Fonction: Calculer le vecteur de réponse à entrée nulle du filtre de pondération perceptive.

For K = 1,2,...,IDIM, do the following:

TMP = TEMP(K)

For J = LPCW,LPCW - 1,...,3,2, do the next two lines

TEMP(K) = TEMP(K) + ZIRWFIR(J) * AWZ(J + 1)

| Section tous

ZIRWFIR(J) = ZIRWFIR(J - 1)

| zéros du filtre

TEMP(K) = TEMP(K) + ZIRWFIR(1) * AWZ(2)

|

ZIRWFIR(1) = TMP

| Traiter le dernier vecteur différemment

For J = LPCW,LPCW - 1,...,3,2, do the next two lines

|

TEMP(K) = TEMP(K) - ZIRWIIR(J) * AWP(J + 1)

| Section tous pôles du filtre

ZIRWIIR(J) = ZIRWIIR(J - 1)

ZIR(K) = TEMP(K) - ZIRWIIR(1) * AWP(2)

|

ZIRWIIR(1) = ZIR(K)

| Traiter le dernier vecteur différemment

Repeat the above for the next K

5.10 *Calcul du vecteur cible VQ (de quantification vectorielle)*

CALCUL DU VECTEUR CIBLE VQ (bloc 11)

Entrées: SW, ZIR

Sortie: TARGET

Fonction: Soustraire le vecteur de réponse à entrée nulle du vecteur de parole pondéré.

Remarque – D'après le bloc 10 ci-dessus, $ZIR(K) = ZIRWIIR(IDIM + 1 - K)$: il n'y a donc pas besoin de prévoir un emplacement mémoire distinct.

For K = 1,2,...,IDIM, do the next line
TARGET(K) = SW(K) - ZIR(K)

5.11 *Module de recherche dans le répertoire d'excitation (bloc 24)*

Les sept blocs contenus dans le module de recherche dans le répertoire d'excitation (bloc 24) sont spécifiés ci-dessous. Ici encore, certains blocs sont décrits sous la forme d'un bloc unique par commodité et pour faciliter la mise en œuvre. Les blocs 12, 14 et 15 sont exécutés une fois par cycle d'adaptation, lorsque ICOUNT = 3, tandis que les autres blocs sont exécutés une fois par vecteur de parole.

CALCULATEUR DE VECTEUR DE RÉPONSE IMPULSIONNELLE (bloc 12)

Entrées: A, AWZ, AWP

Sortie: H

Fonction: Calculer le vecteur de réponse impulsionnelle du filtre de synthèse et du filtre de pondération perceptive, montés en cascade.

Ce bloc est exécuté lorsque ICOUNT = 3 et après exécution des blocs 23 et 3 (c'est-à-dire lorsque les nouveaux ensembles de coefficients A, AWZ et AWP sont prêts).

```
TEMP(1) = 1 | TEMP = mémoire du filtre de synthèse
RC(1) = 1 | RC = mémoire de la section tous pôles de W(z)
For K = 2,3,...,IDIM, do the following:
  A0 = 0
  A1 = 0
  A2 = 0
  For I = K,K - 1,...,3,2, do the next five lines
    TEMP(I) = TEMP(I - 1)
    RC(I) = RC(I - 1)
    A0 = A0 - A(I) * TEMP(I) |
    A1 = A1 + AWZ(I) * TEMP(I) | Filtrage
    A2 = A2 - AWP(I) * RC(I) |
  TEMP(1) = A0
  RC(1) = A0 + A1 + A2
Repeat the above indented section for the next K

ITMP = IDIM + 1 | Obtenir h(n) en inversant
For K = 1,2,...,IDIM, do the next line | l'ordre de la mémoire de la section
  H(K) = RC(ITMP - K) | tous pôles de W(z)
```

MODULE DE CONVOLUTION DES VECTEURCODES DE FORME
ET CALCULATEUR DE TABLE D'ÉNERGIE (blocs 14 et 15)

Entrées: H, Y

Sortie: Y2

Fonction: Convoluer chaque vecteurcode de forme avec la réponse impulsionnelle obtenue dans le bloc 12, puis calculer et mémoriser l'énergie du vecteur résultant.

Ce bloc est également exécuté lorsque ICOUNT = 3, une fois que le bloc 12 a été exécuté.

```
For J = 1,2,...,NCWD, do the following: | Un seul vecteurcode par boucle
  J1 = (J - 1) * IDIM
  For K = 1,2,...,IDIM, do the next four lines
    K1 = J1 + K + 1
    TEMP(K) = 0
    For I = 1,2,...,K, do the next line |
      TEMP(K) = TEMP(K) + H(I) * Y(K1 - I) | Convolution
  Repeat the above 4 lines for the next K
Y2(J) = 0
For K = 1,2,...,IDIM, do the next line |
  Y2(J) = Y2(J) + TEMP(K) * TEMP(K) | Calcul de l'énergie
Repeat the above for the next J
```

NORMALISATION DU VECTEUR CIBLE VQ (bloc 16)

Entrées: TARGET, GAIN

Sortie: TARGET

Fonction: Normaliser le vecteur cible VQ en utilisant le gain d'excitation prédit.

```
TMP = 1 / GAIN
For K = 1,2,...,IDIM, do the next line
  TARGET(K) = TARGET(K) * TMP
```

MODULE DE CONVOLUTION À INVERSION TEMPORELLE (bloc 13)

Entrées: H, TARGET (sortie du bloc 16)

Sortie: PN

Fonction: Effectuer la convolution en inversion temporelle du vecteur de réponse impulsionnelle et du vecteur cible VQ normalisé [pour obtenir le vecteur $p(n)$].

Remarque – Le vecteur PN peut être conservé dans une mémoire temporaire.

```
For K = 1,2,...,IDIM, do the following:
  K1 = K - 1
  PN(K) = 0
  For J = K, K + 1, ..., IDIM, do the next line
    PN(K) = PN(K) + TARGET(J) * H(J - K1)
```

Repeat the above for the next K

CALCULATEUR D'ERREUR ET SÉLECTEUR DU MEILLEUR
INDEX DU RÉPERTOIRE D'EXCITATION (blocs 17 et 18)

Entrées: PN, Y, Y2, GB, G2, GSQ

Sorties: IG, IS, ICHAN

Fonction: Rechercher dans le répertoire de gain et le répertoire de forme pour identifier la meilleure combinaison d'index de gain et d'index de forme; combiner les deux index pour obtenir sur 10 bits le meilleur index du répertoire d'excitation.

Remarque – La variable COR utilisée ci-dessous est habituellement conservée dans un accumulateur plutôt qu'en mémoire. Les variables IDXG et J peuvent être conservées dans des registres temporaires, tandis que les variables IG et IS peuvent être gardées en mémoire.

Initialize DISTM to the largest number representable in the hardware

$N1 = NG/2$

For $J = 1, 2, \dots, NCWD$, do the following:

$J1 = (J - 1) * IDIM$

COR = 0

For $K = 1, 2, \dots, IDIM$, do the next line

$COR = COR + PN(K) * Y(J1 + K)$

|
| Calculer le produit interne P_j

If $COR > 0$, then do the next five lines

IDXG = N1

For $K = 1, 2, \dots, N1 - 1$, do the next "if" statement

If $COR < GB(K) * Y2(J)$, do the next two lines

IDXG = K

GO TO LABEL

| Meilleur gain positif trouvé

If $COR \leq 0$, then do the next five lines

IDXG = NG

For $K = N1 + 1, N1 + 2, \dots, NG - 1$, do the next "if" statement

If $COR > GB(K) * Y2(J)$, do the next two lines

IDXG = K

GO TO LABEL

| Meilleur gain négatif trouvé

LABEL: $D = -G2(IDXG) * COR + GSQ(IDXG) * Y2(J)$

| Calcul de la distorsion \hat{D}

If $D < DISTM$, do the next three lines

DISTM = D

IG = IDXG

IS = J

| Sauvegarde de la plus faible distorsion
| et des meilleurs index du répertoire d'excitation
| trouvés jusqu'à maintenant

Repeat the above indented section for the next K

ICHAN = $(IS - 1) * NG + (IG - 1)$

| Concaténer les index
| de forme et de gain

Transmit ICHAN through communication channel

Dans le cas d'une transmission en série, le bit de plus fort poids de ICHAN sera transmis en premier. Si ICHAN est représenté par le mot de 10 bits $b_9b_8b_7b_6b_5b_4b_3b_2b_1b_0$, alors l'ordre de transmission des bits sera b_9 , puis b_8 , puis b_7, \dots et finalement b_0 (b_9 est le bit de plus fort poids).

5.12 *Décodeur simulé (bloc 8)*

Les blocs 20 et 23 ont été décrits plus haut. Les blocs 19, 21 et 22 sont spécifiés ci-dessous.

RÉPERTOIRE DES VECTEURS D'EXCITATION QUANTIFIÉS (bloc 19)

Entrées: IG, IS

Sortie: YN

Fonction: Effectuer une consultation de la table pour en extraire le meilleur vecteurcode de forme et le meilleur gain, puis les multiplier pour obtenir le vecteur d'excitation quantifié.

$NN = (IS-1) * IDIM$

For K = 1,2,...,IDIM, do the next line

$YN(K) = CQ(IG) * Y(NN + K)$

MODULE DE NORMALISATION DU GAIN (bloc 21)

Entrées: GAIN, YN

Sortie: ET

Fonction: Multiplier le vecteur d'excitation quantifié par le gain d'excitation.

For K = 1,2,...,IDIM, do the next line

$ET(K) = GAIN * YN(K)$

FILTRE DE SYNTHÈSE (bloc 22)

Entrées: ET, A

Sortie: ST

Fonction: Filtrer le vecteur de détection à gain normalisé pour obtenir le vecteur de parole quantifié.

Comme expliqué au § 3, on peut omettre ce bloc et obtenir le vecteur de parole quantifié sous la forme d'un sous-produit de la procédure de mise à jour de la mémoire, comme décrit ci-dessous. Si cependant l'on souhaite mettre en œuvre ce bloc malgré cela, il convient d'utiliser une section distincte de la mémoire du filtre (plutôt que STATELPC) pour ce filtre de synthèse à configuration tous pôles.

5.13 *Mise à jour de la mémoire du filtre pour les blocs 9 et 10*

La description suivante des procédures de mise à jour de la mémoire du filtre de synthèse (blocs 9 et 10) part de l'hypothèse que le vecteur de parole quantifié ST est un sous-produit des mises à jour de mémoire. Pour protéger contre une éventuelle surcharge des niveaux de signal, un limiteur d'amplitude est intégré dans la procédure, de sorte que la mémoire du filtre écrête à MAX et à MIN, qui sont respectivement les niveaux de saturation positive et de saturation négative du signal MIC loi A ou loi μ , selon la loi utilisée.

MISE À JOUR DE LA MÉMOIRE DU FILTRE (blocs 9 et 10)

Entrées: ET, A, AWZ, AWP, STATELPC, ZIRWFIR, ZIRWIIR

Sorties: ST, STATELPC, ZIRWFIR, ZIRWIIR

Fonction: Mettre à jour la mémoire de filtre dans les blocs 9 et 10 et déterminer le vecteur de parole quantifié.

ZIRWFIR(1) = ET(1) | ZIRWFIR devient une table de travail

TEMP(1) = ET(1)

For K = 2,3,...,IDIM, do the following:

A0 = ET(K)

A1 = 0

A2 = 0

For I = K, K - 1, ..., 2, do the next five lines

ZIRWFIR(I) = ZIRWFIR(I - 1)

TEMP(I) = TEMP(I - 1)

A0 = A0 - A(I) * ZIRWFIR(I)

A1 = A1 + AWZ(I) * ZIRWFIR(I)

A2 = A2 - AWP(I) * TEMP(I)

ZIRWFIR(1) = A0

TEMP(1) = A0 + A1 + A2

| Calculer les réponses d'état
| zéro à divers étages du
| filtre en cascade

Repeat the above indented section for the next K

| Maintenant mettre à jour la mémoire
| du filtre en ajoutant les réponses
| d'état zéro aux réponses à entrée nulle

For K = 1,2,...,IDIM, do the next four lines

STATELPC(K) = STATELPC(K) + ZIRWFIR(K)

If STATELPC(K) > MAX, set STATELPC(K) = MAX

If STATELPC(K) < MIN, set STATELPC(K) = MIN

ZIRWIIR(K) = ZIRWIIR(K) + TEMP(K)

| Limitation de plage

For I = 1,2,...,LPCW, do the next line

ZIRWFIR(I) = STATELPC(I)

| Maintenant positionner ZIRWFIR à la
| valeur qui convient

I = IDIM + 1

For K = 1,2,...,IDIM, do the next line

ST(K) = STATELPC(I - K)

| Obtenir le signal de parole quantifié
| en inversant l'ordre de la mémoire
| du filtre de synthèse

5.14 *Décodeur (figure 3/G.728)*

Les blocs du décodeur (figure 3/G.728) sont décrits ci-dessous. Sauf pour le bloc de conversion en format MIC de sortie, tous les autres blocs sont exactement les mêmes que ceux du décodeur simulé (bloc 8) dans la figure 2/G.728.

Le décodeur n'utilise qu'un sous-ensemble des variables du tableau 2/G.728. Si un décodeur et un codeur doivent être réalisés dans une même puce DSP, il convient de donner des noms différents aux variables du décodeur pour éviter d'écraser les variables utilisées dans le bloc du décodeur simulé dans le codeur. Par exemple, pour nommer les variables du décodeur, on peut ajouter un préfixe «d» aux noms des variables correspondantes du tableau 2/G.728. Si un décodeur est en tant que module autonome, indépendant d'un codeur, il n'est pas nécessaire de modifier les noms des variables.

La description suivante part de l'hypothèse d'un décodeur autonome. Ici encore, les blocs sont exécutés dans l'ordre de leur description ci-dessous.

ADAPTATEUR EN BOUCLE DU FILTRE DE SYNTHÈSE DU DÉCODEUR (bloc 33)

Entrée: ST

Sortie: A

Fonction: Générer périodiquement à partir des signaux de parole préalablement décodés, les coefficients du filtre de synthèse.

Le fonctionnement de ce bloc est exactement le même que celui du bloc 23 du codeur.

ADAPTATEUR EN BOUCLE DE GAIN VECTORIEL DU DÉCODEUR (bloc 30)

Entrée: ET

Sortie: GAIN

Fonction: Générer à partir des vecteurs d'excitation à gain normalisé précédents le gain d'excitation.

Le fonctionnement de ce bloc est exactement le même que celui du bloc 20 du codeur.

RÉPERTOIRE DES VECTEURS D'EXCITATION QUANTIFIÉS DU DÉCODEUR (bloc 29)

Entrée: ICHAN

Sortie: YN

Fonction: Décoder le meilleur index du répertoire d'excitation reçu (index de canal) afin d'obtenir le vecteur d'excitation.

Ce bloc extrait d'abord l'index IG sur 3 bits du répertoire de gain et l'index IS sur 7 bits du répertoire de forme, de l'index de canal sur 10 bits reçu. Le reste de l'opération est exactement le même que dans le bloc 19 du codeur.

ITMP = integer part of (ICHAN / NG)

| Décoder (IS - 1)

IG = ICHAN - ITMP * NG + 1

| Décoder IG

NN = ITMP * IDIM

For K = 1,2,...,IDIM, do the next line

YN(K) = GQ(IG) * Y(NN + K)

MODULE DE NORMALISATION DU GAIN DU DÉCODEUR (bloc 31)

Entrées: GAIN, YN

Sortie: ET

Fonction: Multiplier le vecteur d'excitation par le gain d'excitation.

Le fonctionnement de ce bloc est exactement le même que celui du bloc 21 dans le codeur.

FILTRE DE SYNTHÈSE DU DÉCODEUR (bloc 32)

Entrées: ET, A, STATELPC

Sortie: ST

Fonction: Filtrer le vecteur d'excitation normalisé en gain pour obtenir le vecteur de parole décodé.

Ce bloc peut être réalisé comme un simple filtre tous pôles. Mais, comme mentionné au § 4.3, si le codeur obtient le vecteur de parole quantifié comme sous-produit de la mise à jour de mémoire du filtre (pour économiser des calculs) et si l'éventuel cumul d'erreurs d'arrondi constitue une préoccupation, il y a lieu que ce bloc calcule le vecteur de parole décodé exactement comme dans le bloc de décodeur du codeur. C'est-à-dire qu'il convient que le vecteur de parole décodé soit calculé sous forme de somme du vecteur de réponse à entrée nulle et du vecteur de réponse d'état zéro du filtre de synthèse. Ceci peut être effectué par la procédure suivante.

```
For      K = 1,2,...,IDIM, do the next seven lines
        TEMP(K) = 0
        For  J = LPC,LPC - 1,...,3,2, do the next two lines
            TEMP(K) = TEMP(K) - STATELPC(J) * A(J + 1)      | Réponse à entrée nulle.
            STATELPC(J) = STATELPC(J - 1)
        TEMP(K) = TEMP(K) - STATELPC(1) * A(2)              |
        STATELPC(1) = TEMP(K)                               | Traiter le dernier vecteur différemment

Repeat the above for the next K

TEMP(1) = ET(1)
For      K = 2,3,...,IDIM, do the next five lines
        A0 = ET(K)
        For  I = K,K - 1,...,2, do the next two lines
            TEMP(I) = TEMP(I - 1)
            A0 = A0 - A(I) * TEMP(I)                        | Calculer la réponse d'état zéro
        TEMP(1) = A0

Repeat the above for the next K

                                                | Maintenant mettre à jour la mémoire
                                                | du filtre en ajoutant les réponses d'état
                                                | zéro aux réponses à entrée nulle

For      K = 1,2,...,IDIM, do the next three lines
        STATELPC(K) = STATELPC(K) + TEMP(K)                | ZIR + ZSR
        If STATELPC(K) > MAX, set STATELPC(K) = MAX        | Limitation de la plage
        If STATELPC(K) < MIN, set STATELPC(K) = MIN        |

I = IDM + 1
For      K = 1,2,...,IDIM, do the next line
        ST(K) = STATELPC(I - K)                            | Obtention du vecteur de parole quantifié
                                                         | par inversion de l'ordre de la mémoire
                                                         | du filtre de synthèse
```

FILTRE LPC INVERSE DU 10^e ORDRE (bloc 81)

Ce bloc est exécuté une fois par vecteur et le vecteur de sortie est écrit séquentiellement dans les 20 derniers échantillons du tampon de résidus de prédiction linéaire LPC [soit de D(81) à D(100)]. Nous utilisons un pointeur IP pour pointer vers l'adresse des échantillons de la table D(K) à laquelle il faudra écrire. Ce pointeur IP est initialisé à NPWSZ – NFRSZ + IDIM avant que ce bloc ne commence à traiter le premier vecteur de parole décodé du premier cycle d'adaptation (trame); à partir de là, le pointeur IP est mis à jour de la façon décrite ci-dessous. Les coefficients APF(I) du prédicteur LPC du 10^e ordre sont obtenus dans le bloc 50 au milieu de la récurrence de Levinson-Durbin, comme décrit au § 4.6. On part de l'hypothèse que, avant que ce bloc ne commence à s'exécuter, le filtre de synthèse du décodeur (bloc 32 de la figure 3/G.728) a déjà écrit le vecteur de parole décodé courant dans ST(1) à ST(IDIM).

Entrées: ST, APF

Sortie: D

Fonction: Calculer le résidu de prédiction linéaire LPC pour le vecteur de parole décodé courant.

```

If IP = NPWSZ, then set IP = NPWSZ – NFRSZ                | Vérifier et mettre IP à jour

  For K = 1,2,...,IDIM, do the next seven lines
    ITMP = IP + K
    D(ITMP) = ST(K)
    For J = 10,9,...,3,2, do the next two lines
      D(ITMP) = D(ITMP) + STLPCI(J) * APF(J + 1)          | Filtrage de FIR
      STLPCI(J) = STLPCI(J – 1)                          | Décalage mémoire
    D(ITMP) = D(ITMP) + STLPCI(1) * APF(2)                | Traiter le dernier
    STLPCI(1) = ST(K)                                     | Introduire le signal d'entrée par décalage

IP = IP + IDIM                                           | Mettre IP à jour
  
```

MODULE D'EXTRACTION DE LA PÉRIODE FONDAMENTALE (TONIE) (bloc 82)

Ce bloc est exécuté une fois par trame, au troisième vecteur de chaque trame, après génération du troisième vecteur de parole décodé.

Entrée: D

Sortie: KP

Fonction: Extraire la période fondamentale du résidu de prédiction linéaire LPC.

```

If ICOUNT ≠ 3, skip the execution of this block,
otherwise, do the following:                               | Filtrage passe-bas et décimation
                                                         | de rapport 4:1

  For K = NPWSZ - NFRSZ + 1,...,NPWSZ, do the next seven lines | Filtre IIR
    TMP = D(K) – STLPF(1) * AL(1) – STLPF(2) *
    AL(2) – STLPF(3) * AL(3)
  If K is divisible by 4, do the next two lines           | N'effectuer le filtrage FIR que si nécessaire
    N = K/4

    DEC(N) = TMP * BL(1) + STLPF(1) * BL(2) + STLPF(2) * BL(3) + STLPF(3) * BL(4)

    STLPF(3) = STLPF(2)
    STLPF(2) = STLPF(1)                                  | Décaler la mémoire du filtre passe-bas
    STLPF(1) = TMP

M1 = KPMIN/4                                             | Commencer la recherche des pics
M2 = KPMAX/4                                             | de corrélation dans le domaine des résidus
CORMAX = most negative number of the machine            | LPC décimés
  
```

| | |
|---|---|
| For J = M1, M1 + 1, ..., M2, do the next six lines | |
| TMP = 0 | |
| For N = 1, 2, ..., NPWSZ/4, do the next line | |
| TMP = TMP + DEC(N) * DEC(N - J) | TMP = corrélation dans le domaine décimé |
| If TMP > CORMAX, do the next two lines | Trouver le coefficient de corrélation |
| CORMAX = TMP | maximal et le décalage temporel correspondant |
| KMAX = J | |
| For N = -M2 + 1, -M2 + 2, ..., (NPWSZ - NFRSZ)/4, do the next line | Décaler le tampon des résidus |
| DEC(N) = DEC(N + IDIM) | LPC décimés |
| M1 = 4 * KMAX - 3 | Commencer la recherche des pics de corrélation |
| M2 = 4 * KMAX + 3 | dans le domaine non décimé |
| If M1 < KPMIN, set M1 = KPMIN | Vérifier si M1 est hors plage |
| If M2 > KPMAX, set M2 = KPMAX | Vérifier si M2 est hors plage |
| CORMAX = most negative number of the machine | |
| For J = M1, M1 + 1, ..., M2, do the next six lines | |
| TMP = 0 | |
| For K = 1, 2, ..., NPWSZ, do the next line | |
| TMP = TMP + D(K) * D(K - J) | Corrélation dans le domaine non décimé |
| If TMP > CORMAX, do the next two lines | Trouver la corrélation maximale |
| CORMAX = TMP | et le décalage temporel correspondant |
| KP = J | |
| M1 = KP1 - KPDELTA | Déterminer la plage de recherche |
| M2 = KP1 + KPDELTA | autour de la période fondamentale |
| | de la trame précédente |
| If KP < M2 + 1, go to LABEL | Si cette relation est vérifiée, KP ne peut pas |
| | être un multiple de la période fondamentale |
| If M1 < KPMIN, set M1 = KPMIN | Vérifier si M1 est hors plage |
| CMAX = most negative number of the machine | |
| For J = M1, M1 + 1, ..., M2, do the next six lines | Corrélation dans le domaine non décimé |
| TMP = 0 | |
| For K = 1, 2, ..., NPWSZ, do the next line | |
| TMP = TMP + D(K) * D(K - J) | Trouver la corrélation maximale |
| If TMP > CMAX, do the next two lines | et le décalage temporel correspondant |
| CMAX = TMP | |
| KPTMP = J | |
| SUM = 0 | |
| TMP = 0 | Commencer à calculer les poids des coefficients |
| | des filtres de prédiction de tonie |
| For K = 1, 2, ..., NPWSZ, do the next two lines | |
| SUM = SUM + D(K - KP) * D(K - KP) | |
| TMP = TMP + D(K - KPTMP) * D(K - KPTMP) | |
| If SUM = 0, set TAP = 0, otherwise, set TAP = CORMAX/SUM | |
| If TMP = 0, set TAP1 = 0, otherwise, set TAP1 = CMAX/TMP | Clamper TAP entre 0 et 1 |
| If TAP > 1, set TAP = 1 | |
| If TAP < 0, set TAP = 0 | Clamper TAP1 entre 0 et 1 |
| If TAP1 > 1, set TAP1 = 1 | |
| If TAP1 < 0, set TAP1 = 0 | Remplacer KP par la période fondamentale |
| | si TAP1 est assez grand |
| If TAP1 > TAPTH * TAP, then set KP = KPTMP | |
| LABEL: KP1 = KP | Mettre à jour la période fondamentale |
| | de la trame précédente |
| For K = KPMAX + 1, -KPMAX + 2, ..., NPWSZ - NFRSZ, do the next line | |
| D(K) = D(K + NFRSZ) | Décaler le tampon des résidus LPC |

CALCULATEUR DES COEFFICIENTS DE PRÉDICTEUR DE TONIE (bloc 83)

Ce bloc est également exécuté une fois par trame, au troisième vecteur de chaque trame, immédiatement après l'exécution du bloc 82. Ce bloc partage le tampon de vecteurs de parole décodés [table des ST(K)] avec le postfiltre à long terme 71, qui veille au décalage de cette table de manière que ST(1) à ST(IDIM) constituent le vecteur de parole décodé courant et que ST(-KPMAX - NPWSZ + 1) à ST(0) soient les vecteurs de parole décodé précédents.

Entrées: ST, KP

Sortie: PTAP

Fonction: Calculer le coefficient optimal du prédicteur de tonie à un coefficient du signal de parole décodé.

If ICOUNT \neq 3, skip the execution of this block,
otherwise, do the following:

SUM = 0

TMP = 0

For K = -NPWSZ + 1, -NPWSZ + 2, ..., 0, do the next two lines

SUM = SUM + ST(K - KP) * ST(K - KP)

TMP = TMP + ST(K) * ST(K - KP)

If SUM = 0, set PTAP = 0, otherwise, set PTAP = TMP/SUM

CALCULATEUR DES COEFFICIENTS DU POSTFILTRE À LONG TERME (bloc 84)

Ce bloc est également exécuté une fois par trame, au troisième vecteur de chaque trame, immédiatement après l'exécution du bloc 83.

Entrée: PTAP

Sorties: B, GL

Fonction: Calcul du coefficient b et du facteur de normalisation g_l du postfiltre à long terme.

If ICOUNT \neq 3, skip the execution of this block,
otherwise, do the following:

If PTAP > 1, set PTAP = 1

If PTAP < PPFTH, set PTAP = 0

| Clamper PTAP à 1

| Désactiver le postfiltre fondamental

| si PTAP est inférieur au seuil

B = PPFZCF * PTAP

GL = 1 / (1 + B)

CALCULATEUR DES COEFFICIENTS DU POSTFILTRE À COURT TERME (bloc 85)

Ce bloc est également exécuté une fois par trame, mais au premier vecteur de chaque trame.

Entrées: APF, RCTMP(1)

Sorties: AP, AZ, TILTZ

Fonction: Calculer les coefficients du postfiltre à court terme.

If ICOUNT \neq 1, skip the execution of this block,
otherwise, do the following:

For I = 2, 3, ..., 11, do the next two lines

AP(I) = SPFPCFV(I) * APF(I)

AZ(I) = SPFZCFV(I) * APF(I)

TILTZ = TILTF * RCTMP(1)

|
| Normaliser les coefficients du dénominateur
| Normaliser les coefficients du numérateur
| Coefficients du filtre de compensation
| d'écart de niveau spectral

POSTFILTRE À LONG TERME (bloc 71)

Ce bloc est exécuté une fois par vecteur.

Entrées: ST, B, GL, KP

Sortie: TEMP

Fonction: Effectuer l'opération de filtrage du filtre à long terme.

```
For K = 1,2,...,IDIM, do the next line
  TEMP(K) = GL * (ST(K) + B * ST(K - KP))          | Postfiltrage à long terme
For K = -NPWSZ - KPMAX + 1,...,-2,-1,0, do the next line
  ST(K) = ST(K + IDIM)                            | Décaler le registre de vecteurs de parole décodés
```

POSTFILTRE À COURT TERME (bloc 72)

Ce bloc est exécuté une fois par vecteur, immédiatement après l'exécution du bloc 71.

Entrées: AP, AZ, TILTZ, STPFIR, STPFIR, TEMP (sortie du bloc 71)

Sortie: TEMP

Fonction: Effectuer l'opération de filtrage du poste filtre à court terme.

```
For K = 1,2,...,IDIM, do the following:
  TMP = TEMP(K)
  For J = 10,9,...,3,2, do the next two lines
    TEMP(K) = TEMP(K) + STPFIR(J) * AZ(J + 1)      |
    STPFIR(J) = STPFIR(J - 1)                    | Section tous zéros du filtre
  TEMP(K) = TEMP(K) + STPFIR(1) * AZ(2)          | Dernier multiplicateur
  STPFIR(1) = TMP

  For J = 10,9,...,3,2, do the next two lines
    TEMP(K) = TEMP(K) - STPFIR(J) * AP(J + 1)      |
    STPFIR(J) = STPFIR(J - 1)                    | Section tous pôles du filtre
  TEMP(K) = TEMP(K) - STPFIR(1) * AP(2)          | Dernier multiplicateur
  STPFIR(1) = TEMP(K)
TEMP(K) = TEMP(K) + STPFIR(2) * TILTZ            | Filtre de compensation
                                                | d'écart de niveau spectral
```

CALCULATEUR DE SOMME DE VALEURS ABSOLUES (bloc 73)

Ce bloc est exécuté une fois par vecteur, après l'exécution du bloc 32.

Entrée: ST

Sortie: SUMUNFIL

Fonction: Calculer la somme des valeurs absolues des composantes du vecteur de parole décodé.

SUMUNFIL = 0

```
For K = 1,2,...,IDIM, do the next line
  SUMUNFIL = SUMUNFIL + absolute value of ST(K)
```

CALCULATEUR DE SOMME DE VALEURS ABSOLUES (bloc 74)

Ce bloc est exécuté une fois par vecteur, après l'exécution du bloc 72.

Entrée: TEMP (sortie du bloc 72)

Sortie: SUMFIL

Fonction: Calculer la somme des valeurs absolues des composantes du vecteur sortant du postfiltre à court terme.

SUMFIL = 0

For K = 1,2,...,IDIM, do the next line

SUMFIL = SUMFIL + absolute value of TEMP(K)

CALCULATEUR DE FACTEUR DE NORMALISATION (bloc 75)

Ce bloc est exécuté une fois par vecteur, après l'exécution des blocs 73 et 74.

Entrées: SUMUNFIL, SUMFIL

Sortie: SCALE

Fonction: Calculer le facteur de normalisation global du postfiltre.

If SUMFIL > 1, set SCALE = SUMUNFIL / SUMFIL,

otherwise, set SCALE = 1

FILTRE PASSE-BAS DU PREMIER ORDRE (bloc 76) ET MODULE DE NORMALISATION DU GAIN DE SORTIE (bloc 77)

Ces deux blocs sont exécutés une fois par vecteur, après l'exécution des blocs 72 et 75. Il est plus pratique de décrire ces deux blocs ensemble.

Entrées: SCALE, TEMP (sortie du bloc 72)

Sortie: SPF

Fonction: Filtrer en passe-bas le facteur de normalisation calculé une fois par vecteur et utiliser le facteur de normalisation filtré pour normaliser le vecteur de sortie du postfiltre à court terme.

For K = 1,2,...,IDIM, do the following:

SCALEFIL = AGCFAC * SCALEFIL +

(1 - AGCFAC) * SCALE

SPF(K) = SCALEFIL * TEMP(K)

| Sortie de normalisation

| par filtrage passe-bas

CONVERSION AU FORMAT MIC DE SORTIE (bloc 28)

Entrée: SPF

Sortie: SD

Fonction: Convertir les cinq composantes du vecteur de parole décodé en cinq échantillons MIC correspondants de loi A ou de loi μ puis les émettre séquentiellement à des intervalles de 125 μ s.

Les règles de conversion du format MIC uniforme au format MIC loi A ou loi μ sont spécifiées dans la Recommandation G.711.

ANNEXE A
(à la Recommandation G.728)

**Fonctions de fenêtrage hybride pour diverses analyses LPC
dans l'algorithme LD-CELP**

Dans le codeur LD-CELP, nous utilisons trois analyses LPC distinctes afin de mettre à jour les coefficients de trois filtres: le filtre de synthèse, le prédicteur de gain logarithmique et le filtre de pondération perceptive. Chacune de ces trois analyses LPC a sa propre fenêtre hybride. Pour chaque fenêtre hybride, nous énumérons les valeurs des échantillons de la fonction de fenêtrage qui seront utilisées dans la procédure de calcul du fenêtrage hybride. Ces fonctions de fenêtrage ont été initialement conçues au moyen d'une arithmétique en virgule flottante puis quantifiées en nombres qui peuvent être exactement représentés par des séquences de 16 bits dont 15 fractionnaires. Pour chaque fenêtre, nous présenterons d'abord une table contenant l'équivalent en virgule flottante de ces nombres à 16 bits puis nous donnerons une table représentant les entiers à 16 bits correspondants.

A.1 *Fenêtre hybride pour le filtre de synthèse*

La table suivante contient les 105 premiers échantillons de la fonction de fenêtrage pour le filtre de synthèse. Les 35 premiers échantillons sont la partie non récurrente et le reste est la partie récurrente. Il y a lieu de lire cette table de gauche à droite à partir de la première rangée puis de gauche à droite pour la deuxième rangée et ainsi de suite (exactement comme des lignes de balayage).

| | | | | |
|-------------|-------------|-------------|-------------|-------------|
| 0,047760010 | 0,095428467 | 0,142852783 | 0,189971924 | 0,236663818 |
| 0,282775879 | 0,328277588 | 0,373016357 | 0,416900635 | 0,459838867 |
| 0,501739502 | 0,542480469 | 0,582000732 | 0,620178223 | 0,656921387 |
| 0,692199707 | 0,725891113 | 0,757904053 | 0,788208008 | 0,816680908 |
| 0,843322754 | 0,868041992 | 0,890747070 | 0,911437988 | 0,930053711 |
| 0,946533203 | 0,960876465 | 0,973022461 | 0,982910156 | 0,990600586 |
| 0,996002197 | 0,999114990 | 0,999969482 | 0,998565674 | 0,994842529 |
| 0,988861084 | 0,981781006 | 0,974731445 | 0,967742920 | 0,960815430 |
| 0,953948975 | 0,947082520 | 0,940307617 | 0,933563232 | 0,926879883 |
| 0,920227051 | 0,913635254 | 0,907104492 | 0,900604248 | 0,894134521 |
| 0,887725830 | 0,881378174 | 0,875061035 | 0,868774414 | 0,862548828 |
| 0,856384277 | 0,850250244 | 0,844146729 | 0,838104248 | 0,832092285 |
| 0,826141357 | 0,820220947 | 0,814331055 | 0,808502197 | 0,802703857 |
| 0,796936035 | 0,791229248 | 0,785583496 | 0,779937744 | 0,774353027 |
| 0,768798828 | 0,763305664 | 0,757812500 | 0,752380371 | 0,747009277 |
| 0,741638184 | 0,736328125 | 0,731048584 | 0,725830078 | 0,720611572 |
| 0,715454102 | 0,710327148 | 0,705230713 | 0,700164795 | 0,695159912 |
| 0,690185547 | 0,685241699 | 0,680328369 | 0,675445557 | 0,670593262 |
| 0,665802002 | 0,661041260 | 0,656280518 | 0,651580811 | 0,646911621 |
| 0,642272949 | 0,637695313 | 0,633117676 | 0,628570557 | 0,624084473 |
| 0,619598389 | 0,615142822 | 0,610748291 | 0,606384277 | 0,602020264 |

La table ci-dessous contient les entiers correspondants, représentés sur 16 bits. La table précédente correspond à la division par $2^{15} = 32\,768$ des valeurs suivantes.

| | | | | |
|--------|--------|--------|--------|--------|
| 1 565 | 3 127 | 4 681 | 6 225 | 7 755 |
| 9 266 | 10 757 | 12 223 | 13 661 | 15 068 |
| 16 441 | 17 776 | 19 071 | 20 322 | 21 526 |
| 22 682 | 23 786 | 24 835 | 25 828 | 26 761 |
| 27 634 | 28 444 | 29 188 | 29 866 | 30 476 |
| 31 016 | 31 486 | 31 884 | 32 208 | 32 460 |
| 32 637 | 32 739 | 32 767 | 32 721 | 32 599 |
| 32 403 | 32 171 | 31 940 | 31 711 | 31 484 |
| 31 259 | 31 034 | 30 812 | 30 591 | 30 372 |
| 30 154 | 29 938 | 29 724 | 29 511 | 29 299 |
| 29 089 | 28 881 | 28 674 | 28 468 | 28 264 |
| 28 062 | 27 861 | 27 661 | 27 463 | 27 266 |
| 27 071 | 26 877 | 26 684 | 26 493 | 26 303 |
| 26 114 | 25 927 | 25 742 | 25 557 | 25 374 |
| 25 192 | 25 012 | 24 832 | 24 654 | 24 478 |
| 24 302 | 24 128 | 23 955 | 23 784 | 23 613 |
| 23 444 | 23 276 | 23 109 | 22 943 | 22 779 |
| 22 616 | 22 454 | 22 293 | 22 133 | 21 974 |
| 21 817 | 21 661 | 21 505 | 21 351 | 21 198 |
| 21 046 | 20 896 | 20 746 | 20 597 | 20 450 |
| 20 303 | 20 157 | 20 013 | 19 870 | 19 727 |

A.2 Fenêtre hybride pour le prédicteur de gain logarithmique

La table suivante contient les 34 premiers échantillons de la fonction de fenêtrage pour le prédicteur de gain logarithmique. Les 20 premiers échantillons sont la partie non récurrente et le reste est la partie récurrente. Il y a lieu de lire cette table comme les deux précédentes.

| | | | | |
|-------------|-------------|-------------|-------------|-------------|
| 0,092346191 | 0,183868408 | 0,273834229 | 0,361480713 | 0,446014404 |
| 0,526763916 | 0,602996826 | 0,674072266 | 0,739379883 | 0,798400879 |
| 0,850585938 | 0,895507813 | 0,932769775 | 0,962066650 | 0,983154297 |
| 0,995819092 | 0,999969482 | 0,995635986 | 0,982757568 | 0,961486816 |
| 0,932006836 | 0,899078369 | 0,867309570 | 0,836669922 | 0,807128906 |
| 0,778625488 | 0,751129150 | 0,724578857 | 0,699005127 | 0,674316406 |
| 0,650482178 | 0,627502441 | 0,605346680 | 0,583953857 | |

La table suivante contient les entiers correspondants, représentés sur 16 bits. La table précédente correspond à la division par $2^{15} = 32\,768$ des valeurs suivantes.

| | | | | |
|--------|--------|--------|--------|--------|
| 3 026 | 6 025 | 8 973 | 11 845 | 14 615 |
| 17 261 | 19 759 | 22 088 | 24 228 | 26 162 |
| 27 872 | 29 344 | 30 565 | 31 525 | 32 216 |
| 32 631 | 32 767 | 32 625 | 32 203 | 31 506 |
| 30 540 | 29 461 | 28 420 | 27 416 | 26 448 |
| 25 514 | 24 613 | 23 743 | 22 905 | 22 096 |
| 21 315 | 20 562 | 19 836 | 19 135 | |

A.3 Fenêtre hybride pour le filtre de pondération perceptive

La table suivante contient les 60 premiers échantillons de la fonction de fenêtrage pour le prédicteur de gain logarithmique. Les 30 premiers échantillons sont la partie non récurrente et le reste est la partie récurrente. Il y a lieu de lire cette table comme les quatre précédentes.

| | | | | |
|-------------|-------------|-------------|-------------|-------------|
| 0,059722900 | 0,119262695 | 0,178375244 | 0,236816406 | 0,294433594 |
| 0,351013184 | 0,406311035 | 0,460174561 | 0,512390137 | 0,562774658 |
| 0,611145020 | 0,657348633 | 0,701171875 | 0,742523193 | 0,781219482 |
| 0,817108154 | 0,850097656 | 0,880035400 | 0,906829834 | 0,930389404 |
| 0,950622559 | 0,967468262 | 0,980865479 | 0,990722656 | 0,997070313 |
| 0,999847412 | 0,999084473 | 0,994720459 | 0,986816406 | 0,975372314 |
| 0,960449219 | 0,943939209 | 0,927734375 | 0,911804199 | 0,896148682 |
| 0,880737305 | 0,865600586 | 0,850738525 | 0,836120605 | 0,821746826 |
| 0,807647705 | 0,793762207 | 0,780120850 | 0,766723633 | 0,753570557 |
| 0,740600586 | 0,727874756 | 0,715393066 | 0,703094482 | 0,691009521 |
| 0,679138184 | 0,667480469 | 0,656005859 | 0,644744873 | 0,633666992 |
| 0,622772217 | 0,612091064 | 0,601562500 | 0,591217041 | 0,581085205 |

La table suivante contient les entiers correspondants, représentés sur 16 bits. La table précédente correspond à la division par $2^{15} = 32\,768$ des valeurs suivantes.

| | | | | |
|--------|--------|--------|--------|--------|
| 1 957 | 3 908 | 5 845 | 7 760 | 9 648 |
| 11 502 | 13 314 | 15 079 | 16 790 | 18 441 |
| 20 026 | 21 540 | 22 976 | 24 331 | 25 599 |
| 26 775 | 27 856 | 28 837 | 29 715 | 30 487 |
| 31 150 | 31 702 | 32 141 | 32 464 | 32 672 |
| 32 763 | 32 738 | 32 595 | 32 336 | 31 961 |
| 31 472 | 30 931 | 30 400 | 29 878 | 29 365 |
| 28 860 | 28 364 | 27 877 | 27 398 | 26 927 |
| 26 465 | 26 010 | 25 563 | 25 124 | 24 693 |
| 24 268 | 23 851 | 23 442 | 23 039 | 22 643 |
| 22 254 | 21 872 | 21 496 | 21 127 | 20 764 |
| 20 407 | 20 057 | 19 712 | 19 373 | 19 041 |

ANNEXE B

(à la Recommandation G.728)

Tables des répertoires d'excitation de gain et de forme

La présente annexe présente d'abord la table du répertoire des vecteurs d'excitation quantifiés de forme sur 7 bits. Chaque rangée de la table spécifie un des 128 vecteurcodes de forme. La première colonne est l'index de canal de communication associé à chaque vecteurcode de forme (et obtenu par un algorithme d'affectation d'index à code Gray). Les deuxième à sixième colonnes contiennent les première à cinquième composantes des 128 vecteurcodes de forme, en représentation à virgule fixe sur 16 bits. Pour obtenir la valeur en virgule flottante à partir de la valeur en entier, diviser celle-ci par 2048, ce qui revient à une multiplication par 2^{-11} ou à décaler la virgule binaire de 11 bits vers la gauche.

| Index de canal | Composantes des vecteurcodes | | | | |
|----------------|------------------------------|--------|---------|---------|---------|
| 0 | 668 | -2 950 | -1 254 | -1 790 | -2 553 |
| 1 | -5 032 | -4 577 | -1 045 | 2 908 | 3 318 |
| 2 | -2 819 | -2 677 | -948 | -2 825 | -4 450 |
| 3 | -6 679 | -340 | 1 482 | -1 276 | 1 262 |
| 4 | -562 | -6 757 | 1 281 | 179 | -1 274 |
| 5 | -2 512 | -7 130 | -4 925 | 6 913 | 2 411 |
| 6 | -2 478 | -156 | 4 683 | -3 873 | 0 |
| 7 | -8 208 | 2 140 | -478 | -2 785 | 533 |
| 8 | 1 889 | 2 759 | 1 381 | -6 955 | -5 913 |
| 9 | 5 082 | -2 460 | -5 778 | 1 797 | 568 |
| 10 | -2 208 | -3 309 | -4 523 | -6 236 | -7 505 |
| 11 | -2 719 | 4 358 | -2 988 | -1 149 | 2 664 |
| 12 | 1 259 | 995 | 2 711 | -2 464 | -10 390 |
| 13 | 1 722 | -7 569 | -2 742 | 2 171 | -2 329 |
| 14 | 1 032 | 747 | -858 | -7 946 | -12 843 |
| 15 | 3 106 | 4 856 | -4 193 | -2 541 | 1 035 |
| 16 | 1 862 | -960 | -6 628 | 410 | 5 882 |
| 17 | -2 493 | -2 628 | -4 000 | -60 | 7 202 |
| 18 | -2 672 | 1 446 | 1 536 | -3 831 | 1 233 |
| 19 | -5 302 | 6 912 | 1 589 | -4 187 | 3 665 |
| 20 | -3 456 | -8 170 | -7 709 | 1 384 | 4 698 |
| 21 | -4 699 | -6 209 | -11 176 | 8 104 | 16 830 |
| 22 | 930 | 7 004 | 1 269 | -8 977 | 2 567 |
| 23 | 4 649 | 11 804 | 3 441 | -5 657 | 1 199 |
| 24 | 2 542 | -183 | -8 859 | -7 976 | 3 230 |
| 25 | -2 872 | -2 011 | -9 713 | -8 385 | 12 983 |
| 26 | 3 086 | 2 140 | -3 680 | -9 643 | -2 896 |
| 27 | -7 609 | 6 515 | -2 283 | -2 522 | 6 332 |
| 28 | -3 333 | -5 620 | -9 130 | -11 131 | 5 543 |
| 29 | -407 | -6 721 | -17 466 | -2 889 | 11 568 |
| 30 | 3 692 | 6 796 | -262 | -10 846 | -1 856 |
| 31 | 7 275 | 13 404 | -2 989 | -10 595 | 4 936 |
| 32 | 244 | -2 219 | 2 656 | 3 776 | -5 412 |
| 33 | -4 043 | -5 934 | 2 131 | 863 | -2 866 |
| 34 | -3 302 | 1 743 | -2 006 | -128 | -2 052 |
| 35 | -6 361 | 3 342 | -1 583 | -21 | 1 142 |
| 36 | -3 837 | -1 831 | 6 397 | 2 545 | -2 848 |

>>

<<

| Index de canal | Composantes des vecteurcodes | | | | |
|----------------|------------------------------|--------|--------|--------|--------|
| 37 | -9 332 | -6 528 | 5 309 | 1 986 | -2 245 |
| 38 | -4 490 | 748 | 1 935 | -3 027 | -493 |
| 39 | -9 255 | 5 366 | 3 193 | -4 493 | 1 784 |
| 40 | 4 784 | -370 | 1 866 | 1 057 | -1 889 |
| 41 | 7 342 | -2 690 | -2 577 | 676 | -611 |
| 42 | -502 | 2 235 | -1 850 | -1 777 | -2 049 |
| 43 | 1 011 | 3 880 | -2 465 | 2 209 | -152 |
| 44 | 2 592 | 2 829 | 5 588 | 2 839 | -7 306 |
| 45 | -3 049 | -4 918 | 5 955 | 9 201 | -4 447 |
| 46 | 697 | 3 908 | 5 798 | -4 451 | -4 644 |
| 47 | -2 121 | 5 444 | -2 570 | 321 | -1 202 |
| 48 | 2 846 | -2 086 | 3 532 | 566 | -708 |
| 49 | -4 279 | 950 | 4 980 | 3 749 | 452 |
| 50 | -2 484 | 3 502 | 1 719 | -170 | 238 |
| 51 | -3 435 | 263 | 2 114 | -2 005 | 2 361 |
| 52 | -7 338 | -1 208 | 9 347 | -1 216 | -4 013 |
| 53 | -13 498 | -439 | 8 028 | -4 232 | 361 |
| 54 | -3 729 | 5 433 | 2 004 | -4 727 | -1 259 |
| 55 | -3 986 | 7 743 | 8 429 | -3 691 | -987 |
| 56 | 5 198 | -423 | 1 150 | -1 281 | 816 |
| 57 | 7 409 | 4 109 | -3 949 | 2 690 | 30 |
| 58 | 1 246 | 3 055 | -35 | -1 370 | -246 |
| 59 | -1 489 | 5 635 | -678 | -2 627 | 3 170 |
| 60 | 4 830 | -4 585 | 2 008 | -1 062 | 799 |
| 61 | -129 | 717 | 4 594 | 14 937 | 10 706 |
| 62 | 417 | 2 759 | 1 850 | -5 057 | -1 153 |
| 63 | -3 887 | 7 361 | -5 768 | 4 285 | 666 |
| 64 | 1 443 | -938 | 20 | -2 119 | -1 697 |
| 65 | -3 712 | -3 402 | -2 212 | 110 | 2 136 |
| 66 | -2 952 | 12 | -1 568 | -3 500 | -1 855 |
| 67 | -1 315 | -1 731 | 1 160 | -558 | 1 709 |
| 68 | 88 | -4 569 | 194 | -454 | -2 957 |
| 69 | -2 839 | -1 666 | -273 | 2 084 | -155 |
| 70 | -189 | -2 376 | 1 663 | -1 040 | -2 449 |
| 71 | -2 842 | -1 369 | 636 | -248 | -2 677 |
| 72 | 1 517 | 79 | -3 013 | -3 669 | -973 |
| 73 | 1 913 | -2 493 | -5 312 | -749 | 1 271 |
| 74 | -2 903 | -3 324 | -3 756 | -3 690 | -1 829 |
| 75 | -2 913 | -1 547 | -2 760 | -1 406 | 1 124 |
| 76 | 1 844 | -1 834 | 456 | 706 | -4 272 |
| 77 | 467 | -4 256 | -1 909 | 1 521 | 1 134 |
| 78 | -127 | -994 | -637 | -1 491 | -6 494 |
| 79 | 873 | -2 045 | -3 828 | -2 792 | -578 |
| 80 | 2 311 | -1 817 | 2 632 | -3 052 | 1 968 |
| 81 | 641 | 1 194 | 1 893 | 4 107 | 6 342 |
| 82 | -45 | 1 198 | 2 160 | -1 449 | 2 203 |

>>

<<

| Index de canal | Composantes des vecteurcodes | | | | |
|----------------|------------------------------|--------|--------|--------|--------|
| 83 | -2 004 | 1 713 | 3 518 | 2 652 | 4 251 |
| 84 | 2 936 | -3 968 | 1 280 | 131 | -1 476 |
| 85 | 2 827 | 8 | -1 928 | 2 658 | 3 513 |
| 86 | 3 199 | -816 | 2 687 | -1 741 | -1 407 |
| 87 | 2 948 | 4 029 | 394 | -253 | 1 298 |
| 88 | 4 286 | 51 | -4 507 | -32 | -659 |
| 89 | 3 903 | 5 646 | -5 588 | -2 592 | 5 707 |
| 90 | -606 | 1 234 | -1 607 | -5 187 | 664 |
| 91 | -525 | 3 620 | -2 192 | -2 527 | 1 707 |
| 92 | 4 297 | -3 251 | -2 283 | 812 | -2 264 |
| 93 | 5 765 | 528 | -3 287 | 1 352 | 1 672 |
| 94 | 2 735 | 1 241 | -1 103 | -3 273 | -3 407 |
| 95 | 4 033 | 1 648 | -2 965 | -1 174 | 1 444 |
| 96 | 74 | 918 | 1 999 | 915 | -1 026 |
| 97 | -2 496 | -1 605 | 2 034 | 2 950 | 229 |
| 98 | -2 168 | 2 037 | 15 | -1 264 | -208 |
| 99 | -3 552 | 1 530 | 581 | 1 491 | 962 |
| 100 | -2 613 | -2 338 | 3 621 | -1 488 | -2 185 |
| 101 | -1 747 | 81 | 5 538 | 1 432 | -2 257 |
| 102 | -1 019 | 867 | 214 | -2 284 | -1 510 |
| 103 | -1 684 | 2 816 | -229 | 2 551 | -1 389 |
| 104 | 2 707 | 504 | 479 | 2 783 | -1 009 |
| 105 | 2 517 | -1 487 | -1 596 | 621 | 1 929 |
| 106 | -148 | 2 206 | -4 288 | 1 292 | -1 401 |
| 107 | -527 | 1 243 | -2 731 | 1 909 | 1 280 |
| 108 | 2 149 | -1 501 | 3 688 | 610 | -4 591 |
| 109 | 3 306 | -3 369 | 1 875 | 3 636 | -1 217 |
| 110 | 2 574 | 2 513 | 1 449 | -3 074 | -4 979 |
| 111 | 814 | 1 826 | -2 497 | 4 234 | -4 077 |
| 112 | 1 664 | -220 | 3 418 | 1 002 | 1 115 |
| 113 | 781 | 1 658 | 3 919 | 6 130 | 3 140 |
| 114 | 1 148 | 4 065 | 1 516 | 815 | 199 |
| 115 | 1 191 | 2 489 | 2 561 | 2 421 | 2 443 |
| 116 | 770 | -5 915 | 5 515 | -368 | -3 199 |
| 117 | 1 190 | 1 047 | 3 742 | 6 927 | -2 089 |
| 118 | 292 | 3 099 | 4 308 | -758 | -2 455 |
| 119 | 523 | 3 921 | 4 044 | 1 386 | 85 |
| 120 | 4 367 | 1 006 | -1 252 | -1 466 | -1 383 |
| 121 | 3 852 | 1 579 | -77 | 2 064 | 868 |
| 122 | 5 109 | 2 919 | -202 | 359 | -509 |
| 123 | 3 650 | 3 206 | 2 303 | 1 693 | 1 296 |
| 124 | 2 905 | -3 907 | 229 | -1 196 | -2 332 |
| 125 | 5 977 | -3 585 | 805 | 3 825 | -3 138 |
| 126 | 3 746 | -606 | 53 | -269 | -3 301 |
| 127 | 606 | 2 018 | -1 316 | 4 064 | 398 |

Nous indiquerons ensuite les valeurs du répertoire de gain. Cette table ne contient pas seulement les valeurs de GQ, mais aussi celles de GB, G2 et GSQ. Les valeurs de GQ, comme celles de GB, peuvent être représentées exactement en arithmétique 16 bits avec le format Q13. La représentation de G2 en virgule fixe est exactement la même que celle de GQ, sauf que le format devient Q12. Une représentation des GSQ arrondie à l'entier le plus proche suffira en virgule fixe.

Valeurs des tables relatives au répertoire de gain

| Index de table | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|------------------|-------------|-------------|-------------|-------------|---------|---------|---------|---------|
| GQ ^{b)} | 0,515625 | 0,90234375 | 1,579101563 | 2,763427734 | - GQ(1) | - GQ(2) | - GQ(3) | - GQ(4) |
| GB | 0,708984375 | 1,240722656 | 2,171264649 | a) | -GB(1) | - GB(2) | - GB(3) | a) |
| G2 | 1,03125 | 1,8046875 | 3,158203126 | 5,526855468 | - G2(1) | - G2(2) | - G2(3) | - G2(4) |
| GSQ | 0,26586914 | 0,814224243 | 2,493561746 | 7,636532841 | GSQ(1) | GSQ(2) | GSQ(3) | GSQ(4) |

a) Peut être une valeur arbitraire (non utilisée).

b) Noter que $GQ(1) = 33/64$ et que $GQ(i) = (7/4) GQ(i - 1)$ pour $i = 2, 3, 4$.

ANNEXE C

(à la Recommandation G.728)

Valeurs utilisées pour l'extension de la largeur de bande

La table ci-dessous donne les valeurs entières pour les vecteurs d'extension de la largeur de bande, de commande des pôles et de commande des zéros énumérés dans le tableau 2/G.728. Pour obtenir la valeur en virgule flottante, diviser la valeur entière par 16 384. Les valeurs de cette table représentent ces valeurs à virgule flottante en format Q14, qui est le plus couramment utilisé pour représenter des nombres inférieurs à 2 en arithmétique 16 bits à virgule fixe.

| <i>i</i> | FACV | FACGPV | WPCFV | WZCFV | SPFPCFV | SPFZCFV |
|----------|--------|--------|--------|--------|---------|---------|
| 1 | 16 384 | 16 384 | 16 384 | 16 384 | 16 384 | 16 384 |
| 2 | 16 192 | 14 848 | 9 830 | 14 746 | 12 288 | 10 650 |
| 3 | 16 002 | 13 456 | 5 898 | 13 271 | 9 216 | 6 922 |
| 4 | 15 815 | 12 195 | 3 539 | 11 944 | 6 912 | 4 499 |
| 5 | 15 629 | 11 051 | 2 123 | 10 750 | 5 184 | 2 925 |
| 6 | 15 446 | 10 015 | 1 274 | 9 675 | 3 888 | 1 901 |
| 7 | 15 265 | 9 076 | 764 | 8 707 | 2 916 | 1 236 |
| 8 | 15 086 | 8 225 | 459 | 7 836 | 2 187 | 803 |
| 9 | 14 910 | 7 454 | 275 | 7 053 | 1 640 | 522 |
| 10 | 14 735 | 6 755 | 165 | 6 347 | 1 230 | 339 |
| 11 | 14 562 | 6 122 | 99 | 5 713 | 923 | 221 |
| 12 | 14 391 | | | | | |
| 13 | 14 223 | | | | | |
| 14 | 14 056 | | | | | |
| 15 | 13 891 | | | | | |

| <i>i</i> | FACV | FACGPV | WPCFV | WZCFV | SPFPCFV | SPFZCFV |
|----------|--------|--------|-------|-------|---------|---------|
| 16 | 13 729 | | | | | |
| 17 | 13 568 | | | | | |
| 18 | 13 409 | | | | | |
| 19 | 13 252 | | | | | |
| 20 | 13 096 | | | | | |
| 21 | 12 943 | | | | | |
| 22 | 12 791 | | | | | |
| 23 | 12 641 | | | | | |
| 24 | 12 493 | | | | | |
| 25 | 12 347 | | | | | |
| 26 | 12 202 | | | | | |
| 27 | 12 059 | | | | | |
| 28 | 11 918 | | | | | |
| 29 | 11 778 | | | | | |
| 30 | 11 640 | | | | | |
| 31 | 11 504 | | | | | |
| 32 | 11 369 | | | | | |
| 33 | 11 236 | | | | | |
| 34 | 11 104 | | | | | |
| 35 | 10 974 | | | | | |
| 36 | 10 845 | | | | | |
| 37 | 10 718 | | | | | |
| 38 | 10 593 | | | | | |
| 39 | 10 468 | | | | | |
| 40 | 10 346 | | | | | |
| 1 | 10 225 | | | | | |
| 42 | 10 105 | | | | | |
| 43 | 9 986 | | | | | |
| 44 | 9 869 | | | | | |
| 45 | 9 754 | | | | | |
| 46 | 9 639 | | | | | |
| 47 | 9 526 | | | | | |
| 48 | 9 415 | | | | | |
| 49 | 9 304 | | | | | |
| 50 | 9 195 | | | | | |
| 51 | 9 088 | | | | | |

ANNEXE D

(à la Recommandation G.728)

Coefficients du filtre à fonction elliptique passe-bas à 1 kHz utilisé dans le module d'extraction de période fondamentale (bloc 82)

Le filtre passe-bas à 1 kHz utilisé dans le module d'extraction de décalage temporel de tonie et de codage (bloc 82) est un filtre du troisième ordre à configuration de pôles et de zéros; sa fonction de transfert est la suivante:

$$L(z) = \frac{\sum_{i=0}^3 b_i z^{-i}}{1 + \sum_{i=1}^3 a_i z^{-i}}$$

où les coefficients a_i et b_i sont donnés dans la table ci-après:

| i | a_i | b_i |
|-----|--------------|---------------|
| 0 | — | 0,0357081667 |
| 1 | -2,34036589 | -0,0069956244 |
| 2 | 2,01190019 | -0,0069956244 |
| 3 | -0,614109218 | 0,0357081667 |

ANNEXE E

(à la Recommandation G.728)

Chronologie de la séquence de calcul

L'ensemble des calculs effectués dans le codeur et dans le décodeur peut être subdivisé en deux catégories. Dans la première s'inscriront les calculs qui interviennent une fois par vecteur. Les § 3 à 5.14 précisent la nature de ces opérations. Il s'agit en général de celles qui impliquent ou précèdent la quantification proprement dite du signal d'excitation et la synthèse du signal de sortie. Si l'on se rapporte plus particulièrement aux numéros des blocs dans la figure 2/G.728, cette catégorie comprendra les blocs 1, 2, 4, 9, 10, 11, 13, 16, 17, 18, 21 et 22. Dans la figure 3/G.728, les blocs 28, 29, 31, 32 et 34 entrent dans cette catégorie. Dans la figure 6/G.728, il s'agira des blocs 39, 40, 41, 42, 46, 47, 48 et 67. (Noter que la figure 6/G.728 est applicable aussi bien au bloc 20 de la figure 2/G.728 qu'au bloc 30 de la figure 3/G.728. Les blocs 43, 44 et 45 de la figure 6/G.728 ne font pas partie de cette catégorie. Les blocs 20 et 30 font donc partie des deux catégories.)

Appartiennent à la deuxième catégorie les calculs qui ne sont effectués qu'une fois tous les quatre vecteurs. Si l'on consulte de nouveau les figures 2/G.728 à 8/G.728, cette catégorie comprend les blocs 3, 12, 14, 15, 23, 33, 35, 36, 37, 38, 43, 44, 45, 49, 50, 51, 81, 82, 83, 84 et 85. Tous les calculs de cette deuxième catégorie sont associés à la mise à jour dans le codeur d'un ou de plusieurs des filtres d'adaptation ou des prédicteurs. Il existe dans le codeur trois structures d'adaptation de ce type: le filtre de synthèse LPC du 50^e ordre, le prédicteur de gain vectoriel et le filtre de pondération perceptive. Il existe dans le décodeur quatre structures de ce type: le filtre de synthèse, le prédicteur de gain et les postfiltres d'adaptation à long et à court terme. Les § 3 à 5.14 donnent la description des repères temporels et des signaux d'entrée de chacune de ces cinq structures d'adaptation. Bien qu'elle soit redondante, la présente annexe énumère précisément tous ces renseignements de synchronisation, regroupés pour en faciliter la consultation. Le tableau E-1/G.728 résume les cinq structures d'adaptation, leurs signaux d'entrée, leurs instants de calcul et le moment où les valeurs mises à jour sont utilisées pour la première fois. A titre de référence pour retrouver ces calculs, la quatrième colonne du tableau E-1/G.728 indique les numéros de blocs utilisés dans les figures et dans les § 3, 4 et 5.

TABLEAU E-1/G.728

| Synchronisation des mises à jour des adaptateurs | | | |
|--|--|--|-------------------------------|
| Adaptateur | Signal (signaux) d'entrée | Première utilisation des paramètres mis à jour | Blocs de référence |
| Adaptateur en boucle du filtre de synthèse | Signal de parole (ST) en sortie du filtre de synthèse jusqu'au vecteur 4 | Codage/décodage du vecteur 3 | 23, 33 (49,50,51) |
| Adaptateur en boucle de gain vectoriel | Gains logarithmiques jusqu'au vecteur 1 | Codage/décodage du vecteur 2 | 20, 30 (43,44,45) |
| Adaptateur du filtre de pondération perceptive et de consultation rapide du répertoire | Signal de parole (S) d'entrée jusqu'au vecteur 2 | Codage du vecteur 3 | 3 (36,37,38) 12, 14, 15 |
| Adaptateur du filtre adaptatif à long terme | Signal de parole (ST) en sortie du filtre de synthèse jusqu'au vecteur 3 | Synthèse du vecteur postfiltré 3 | 35 (81 à 84) |
| Adaptateur du postfiltre adaptatif à court terme | Signal de parole (ST) en sortie du filtre de synthèse jusqu'au vecteur 4 | Synthèse du vecteur postfiltré 1 | 35 (85) |

La plus grande partie des calculs est, de loin, consacrée à mettre à jour le filtre de synthèse du 50^e ordre. Le signal d'entrée requis est le dernier vecteur de parole (ST) issu de ce filtre. Dès que le quatrième vecteur du précédent cycle d'adaptation a été décodé, la méthode de fenêtrage hybride peut commencer à calculer les coefficients d'autocorrélation (bloc 49). Ensuite, la récurrence de Levinson-Durbin (bloc 50) peut commencer à déterminer les coefficients de prédiction. On constate en pratique qu'il faut prolonger ce calcul sur plus d'un cycle vectoriel. On commence le calcul de fenêtrage hybride avant que le vecteur 1 ait été complètement reçu. Il faut interrompre la récurrence de Durbin pour coder le vecteur 1. La récurrence de Durbin ne se termine qu'avec le vecteur 2. Finalement, on applique l'extension de la largeur de bande (bloc 51) aux coefficients du prédicteur. Les résultats de ce calcul ne sont pas utilisés avant le codage ou le décodage du vecteur 3 parce qu'il faut combiner dans le codeur ces valeurs mises à jour avec la mise à jour du filtre de pondération perceptive et des énergies des vecteurcodes. Or ces mises à jour ne sont pas disponibles avant le vecteur 3.

L'adaptation du gain s'effectue en deux temps. Le prédicteur adaptatif est mis à jour une fois tous les quatre vecteurs mais produit, une fois par vecteur, une nouvelle valeur de gain. Dans la présente annexe, nous décrirons la synchronisation de la mise à jour pour le prédicteur. Ce calcul exige d'appliquer d'abord la méthode de fenêtrage hybride aux gains logarithmiques précédents (bloc 43) puis d'effectuer la récurrence de Durbin (bloc 44) et l'extension de la largeur de bande (bloc 45). Tout cela peut être réalisé pendant le vecteur 2 au moyen des gains logarithmiques disponibles jusqu'au vecteur 1. Si le résultat de la récurrence de Durbin ne révèle pas de singularité, le nouveau prédicteur de gain est utilisé immédiatement pour le codage du vecteur 2.

La mise à jour du filtre de pondération perceptive est calculée pendant le vecteur 3. La première partie de cette mise à jour consiste à effectuer l'analyse LPC du signal de parole d'entrée jusqu'au vecteur 2. On peut commencer ce calcul immédiatement après le codage du vecteur 2, sans attendre d'avoir reçu complètement le vecteur 3. Ce calcul consiste à appliquer la méthode de fenêtrage hybride (bloc 36), la récurrence de Durbin (bloc 37) et le calcul du coefficient du filtre de pondération perceptive (bloc 38). Il faut ensuite combiner le filtre de pondération perceptive avec le filtre de synthèse mis à jour afin de calculer le vecteur de réponse impulsionnelle (bloc 12). Il faut également effectuer la convolution de chaque vecteurcode de forme avec cette réponse impulsionnelle pour obtenir les énergies des vecteurcodes (blocs 14 et 15). Dès que ces calculs sont terminés, l'on peut utiliser toutes les valeurs ainsi mises à jour pour coder le vecteur 3.

Remarque – Comme le calcul des énergies des vecteurcodes met assez le processeur à contribution, nous n'avons pas été en mesure de terminer la mise à jour du filtre de pondération perceptive dans le cadre des calculs à effectuer pendant la durée du vecteur 2, même en transférant ailleurs les données de mise à jour du prédicteur de gain. C'est pourquoi la fin de cette mise à jour a été reportée au vecteur 3.

Le postfiltre adaptatif à long terme est mis à jour sur la base d'un algorithme d'extraction rapide de tonie qui fait appel, en entrée, au signal (ST) issu du filtre de synthèse. Comme le postfiltre n'est utilisé que dans le décodeur, la programmation dans le temps pour exécuter ce calcul a été fondée sur les autres charges de calcul imposées dans le décodeur. Celui-ci n'a pas à mettre à jour le filtre de pondération perceptive et les énergies des vecteurcodes, de sorte que le créneau temporel du vecteur 3 est disponible. Le mot-code du vecteur 3 est décodé et son signal de parole en sortie du filtre de synthèse est disponible au même moment que tous les précédents vecteurs sortant du filtre de synthèse. Ces vecteurs sont injectés dans l'adaptateur, qui construira alors la nouvelle période fondamentale (blocs 81 et 82) et le coefficient du postfiltre à long terme (blocs 83 et 84). Ces nouvelles valeurs sont immédiatement utilisées pour calculer le signal de sortie postfiltré pour le vecteur 3.

Le postfiltre adaptatif à court terme est mis à jour comme sous-produit de la mise à jour du filtre de synthèse. La récurrence de Durbin est interrompue à l'ordre 10 et les coefficients de prédiction sont mis en mémoire pour la mise à jour du postfiltre. Comme la récurrence de Durbin commence pendant le vecteur 1, la mise à jour du postfiltre adaptatif à court terme est terminée à temps pour le postfiltrage du vecteur de sortie 1.

ANNEXE F

(à la Recommandation G.728)

Liste alphabétique des abréviations utilisées dans la présente Recommandation

| | |
|---------|---|
| CELP | Prédiction linéaire avec excitation par code (<i>code excited linear prediction</i>) |
| DCME | Equipement de multiplication de circuit numérique (<i>digital circuit multiplication equipment</i>) |
| DSP | Traitement numérique de signal (<i>digital signal processing</i>) |
| LD-CELP | Prédiction linéaire à faible délai avec excitation par code (<i>low-delay code excited linear prediction</i>) |
| LPC | Codage prédictif linéaire (<i>linear prediction coding</i>) |
| MIC | Modulation par impulsions et codage |
| MSE | Variance (<i>mean-squared error</i>) |
| RMS | Valeur quadratique moyenne (<i>root-mean-square</i>) |
| VQ | Quantification vectorielle (<i>vector quantization</i>) |
| WNCF | Facteur de correction par bruit blanc (<i>white noise correction factor</i>) |

APPENDICE I

(à la Recommandation G.728)

Vérification de la mise œuvre

Un ensemble d'outils de vérification a été conçu afin de faciliter la vérification de la conformité des différentes mises en œuvre de l'algorithme défini dans la présente Recommandation. Ces outils de vérification peuvent être demandés à l'UIT sous forme d'un jeu de disquettes de distribution.