



UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

# UIT-T

SECTEUR DE LA NORMALISATION  
DES TÉLÉCOMMUNICATIONS  
DE L'UIT

# G.728

**Annexe J**  
(09/99)

SÉRIE G: SYSTÈMES ET SUPPORTS DE  
TRANSMISSION, SYSTÈMES ET RÉSEAUX  
NUMÉRIQUES

Systèmes de transmission numériques – Equipements  
terminaux – Codage des signaux analogiques par des  
méthodes autres que la MIC

---

Codage de la parole à 16 kbit/s en utilisant la  
prédiction linéaire à faible délai avec excitation  
par code

**Annexe J: Fonctionnement de l'algorithme  
LD-CELP à débit variable principalement pour  
les applications de transmission de données  
en bande vocale sur les DCME**

Recommandation UIT-T G.728 – Annexe J

(Antérieurement Recommandation du CCITT)

---

RECOMMANDATIONS UIT-T DE LA SÉRIE G  
**SYSTÈMES ET SUPPORTS DE TRANSMISSION, SYSTÈMES ET RÉSEAUX NUMÉRIQUES**

CONNEXIONS ET CIRCUITS TÉLÉPHONIQUES INTERNATIONAUX	G.100–G.199
<b>SYSTÈMES INTERNATIONAUX ANALOGIQUES À COURANTS PORTEURS</b>	
CARACTÉRISTIQUES GÉNÉRALES COMMUNES À TOUS LES SYSTÈMES ANALOGIQUES À COURANTS PORTEURS	G.200–G.299
CARACTÉRISTIQUES INDIVIDUELLES DES SYSTÈMES TÉLÉPHONIQUES INTERNATIONAUX À COURANTS PORTEURS SUR LIGNES MÉTALLIQUES	G.300–G.399
CARACTÉRISTIQUES GÉNÉRALES DES SYSTÈMES TÉLÉPHONIQUES INTERNATIONAUX HERTZIENS OU À SATELLITES ET INTERCONNEXION AVEC LES SYSTÈMES SUR LIGNES MÉTALLIQUES	G.400–G.449
COORDINATION DE LA RADIODÉLÉPHONIE ET DE LA TÉLÉPHONIE SUR LIGNES	G.450–G.499
<b>EQUIPEMENTS DE TEST</b>	
<b>CARACTÉRISTIQUES DES SUPPORTS DE TRANSMISSION</b>	<b>G.600–G.699</b>
<b>SYSTÈMES DE TRANSMISSION NUMÉRIQUES</b>	
EQUIPEMENTS TERMINAUX	G.700–G.799
Généralités	G.700–G.709
Codage des signaux analogiques en modulation par impulsions et codage	G.710–G.719
<b>Codage des signaux analogiques par des méthodes autres que la MIC</b>	<b>G.720–G.729</b>
Principales caractéristiques des équipements de multiplexage primaires	G.730–G.739
Principales caractéristiques des équipements de multiplexage de deuxième ordre	G.740–G.749
Caractéristiques principales des équipements de multiplexage d'ordre plus élevé	G.750–G.759
Caractéristiques principales des équipements de transcodage et de multiplication numérique	G.760–G.769
Fonctionnalités de gestion, d'exploitation et de maintenance des équipements de transmission	G.770–G.779
Caractéristiques principales des équipements de multiplexage en hiérarchie numérique synchrone	G.780–G.789
Autres équipements terminaux	G.790–G.799
RÉSEAUX NUMÉRIQUES	G.800–G.899
SECTION NUMÉRIQUES ET SYSTÈMES DE LIGNES NUMÉRIQUES	G.900–G.999

*Pour plus de détails, voir la Liste des Recommandations de l'UIT-T.*

## RECOMMANDATION UIT-T G.728

### CODAGE DE LA PAROLE A 16 kbit/s EN UTILISANT LA PREDICTION LINEAIRE A FAIBLE DELAI AVEC EXCITATION PAR CODE

#### ANNEXE J

#### Fonctionnement de l'algorithme LD-CELP à débit variable principalement pour les applications de transmission de données en bande vocale sur les DCME

##### Résumé

L'Annexe J à la Recommandation G.728 définit une extension à 40 kbit/s, optimisée pour les signaux de données dans la bande vocale (VBD, *voiceband data*), de la spécification relative au débit à 16 kbit/s, avec virgule fixe, de l'actuelle Annexe G/G.728. La principale différence entre le codec décrit dans la présente annexe et le codec décrit dans l'Annexe G/G.728 réside dans l'application de la quantification à codage en treillis (TCQ, *trellis-coded quantization*) à la recherche dans le répertoire codé. La technique faisant appel à la TCQ remplace la méthode d'analyse par synthèse pour la recherche dans un répertoire de séquences codées définie dans la Recommandation G.728, seulement pour la transmission des données dans la bande vocale (VBD).

L'adaptation en boucle du prédicteur réalisée pour la transmission VBD est presque identique à l'adaptation en boucle effectuée en mode de transmission de la parole (Recommandation G.728). De plus, le même cycle d'adaptation est utilisé pour la parole (Recommandation G.728) et pour les données dans la bande vocale. Pour la parole, le débit de 40 kbit/s retourne à la prédiction LD-CELP de la Recommandation G.728.

La présente annexe comporte un fichier électronique contenant les vecteurs tests pour la vérification des implémentations de l'Annexe J/G.728.

##### Source

L'Annexe J à la Recommandation G.728, élaborée par la Commission d'études 16 (1997-2000) de l'UIT-T, a été approuvée le 30 septembre 1999 selon la procédure définie dans la Résolution n° 1 de la CMNT.

## AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'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.

La Conférence mondiale de normalisation des télécommunications (CMNT), qui se réunit tous les quatre ans, détermine les thèmes d'études à traiter par les Commissions d'études de l'UIT-T, lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution n° 1 de la CMNT.

Dans certains secteurs des technologies de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI.

## NOTE

Dans la présente Recommandation, le terme *exploitation reconnue (ER)* désigne tout particulier, toute entreprise, toute société ou tout organisme public qui exploite un service de correspondance publique. Les termes *Administration*, *ER* et *correspondance publique* sont définis dans la *Constitution de l'UIT (Genève, 1992)*.

## DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente Recommandation puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un Membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des Recommandations.

A la date d'approbation de la présente Recommandation, l'UIT n'avait pas été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

© UIT 2000

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'UIT.

## TABLE DES MATIÈRES

	<b>Page</b>
J.1	Domaine d'application ..... 1
J.2	Références normatives ..... 1
J.3	Aperçu ..... 2
J.4	Description de l'algorithme ..... 2
J.4.1	Structure du codec ..... 2
J.4.2	Structure du décodeur ..... 6
J.4.3	Description détaillée du codeur ..... 7
J.4.4	Description détaillée du décodeur ..... 21
J.4.5	Description détaillée des modules commutateurs de mode ..... 22
J.4.6	Tables de transition en treillis ..... 26
J.4.7	Coefficients du polynôme de calcul logarithmique ..... 28
J.4.8	Coefficients d'élargissement de la largeur de bande pour le mode de données ..... 29
J.4.9	Blocs internes ..... 29
J.4.10	Variables et constantes de traitement interne ..... 30
J.4.11	Valeurs initiales ..... 34
J.5	Bibliographie ..... 35
Fichier électronique	
–	Vecteurs tests



## Recommandation G.728

### CODAGE DE LA PAROLE A 16 kbit/s EN UTILISANT LA PREDICTION LINEAIRE A FAIBLE DELAI AVEC EXCITATION PAR CODE

#### ANNEXE J

#### Fonctionnement de l'algorithme LD-CELP à débit variable principalement pour les applications de transmission de données en bande vocale sur les DCME<sup>1</sup>

(Genève, 1999)

##### J.1 Domaine d'application

La présente annexe fournit l'information requise pour la mise en œuvre du codec ainsi que les modifications qui doivent être apportées à l'Annexe G/G.728 pour permettre la commutation de mode dans un dispositif arithmétique à virgule fixe.

L'Annexe J porte sur un débit de transmission de 40 kbit/s. Le retard algorithmique est de 5 échantillons (0,625 ms). Il est donc exactement identique à celui de l'Annexe G et de tous les algorithmes de prédiction linéaire à faible délai à excitation par code (LD-CELP, *low-delay code excited linear prediction*) recommandés dans la Recommandation G.728. L'algorithme VBD à 40 kbit/s de l'Annexe J est destiné aux transmissions avec compression du signal VBD, dans des applications comme les équipements de multiplication de circuit numérique (DCME, *digital circuit multiplication equipment*). Cet algorithme permet une transition progressive à destination et en provenance de l'algorithme LD-CELP (Recommandation G.728) et il est aussi conçu en vue de maintenir une qualité de parole de type circuit interurbain. L'Annexe J vise à remplacer le mode MICDA à 40 kbit/s (Recommandation G.726) dans les systèmes DCME faisant appel à la LD-CELP (Recommandation G.728).

##### J.2 Références normatives

La présente Recommandation se réfère à certaines dispositions des Recommandations UIT-T et textes suivants qui de ce fait en sont partie intégrante. Les versions indiquées étaient en vigueur au moment de la publication de la présente Recommandation. Toute Recommandation ou tout texte étant sujet à révision, les utilisateurs de la présente Recommandation sont invités à se reporter, si possible, aux versions les plus récentes des références normatives suivantes. La liste des Recommandations de l'UIT-T en vigueur est régulièrement publiée.

- [1] Recommandation CCITT G.728 (1992), *Codage de la parole à 16 kbit/s en utilisant la prédiction linéaire à faible délai avec excitation par code.*
- [2] Recommandation UIT-T G.728 Annexe G (1994), *Spécification d'un dispositif à virgule fixe fonctionnant à 16 kbit/s.*
- [3] Recommandation UIT-T G.763 (1998), *Equipements de multiplication de circuit numérique utilisant la modulation par impulsions et codage différentiel adaptatif (Recommandation G.726) et la concentration numérique de la parole.*

---

<sup>1</sup> La présente annexe comporte un fichier électronique contenant les vecteurs tests pour la vérification des implémentations.

### J.3 Aperçu

Le codec utilise un débit de transmission de 40 kbit/s. Le retard algorithmique est de 5 échantillons, pour un total de 0,625 ms. Le codec peut effectuer une commutation de mode à chaque "cycle d'adaptation" (2,5 ms).

La principale différence entre le codec décrit dans la présente annexe et le codec décrit dans l'Annexe G/G.728 réside dans l'application de la quantification à codage en treillis (TCQ) à la recherche dans le répertoire codé. La technique faisant appel à la TCQ remplace la méthode d'analyse par synthèse pour la recherche dans un répertoire de séquences codées définie dans la Recommandation G.728, seulement pour la transmission des données dans la bande vocale (VBD).

L'adaptation en boucle du prédicteur réalisée pour la transmission VBD est presque identique à l'adaptation en boucle effectuée en mode de transmission de la parole (Recommandation G.728). De plus, le même cycle d'adaptation est utilisé pour la parole (Recommandation G.728) et pour les données dans la bande vocale. Pour la parole, le débit de 40 kbit/s retourne à la prédiction LD-CELP de la Recommandation G.728.

Le paragraphe J.4, Description de l'algorithme, explique en détail le fonctionnement du codec. Une description détaillée de la mise en œuvre commence au J.4.3 – Codeur; le sous-paragraphe J.4.4 donne des détails supplémentaires au sujet du décodeur et le J.4.5 donne les détails de la commutation de mode.

### J.4 Description de l'algorithme

#### J.4.1 Structure du codec

L'algorithme est basé sur une structure de codage prédictif linéaire à excitation par résidus (RELP, *residual excited linear predictive coding*) dont les éléments sont répartis comme suit (Figure J.1):

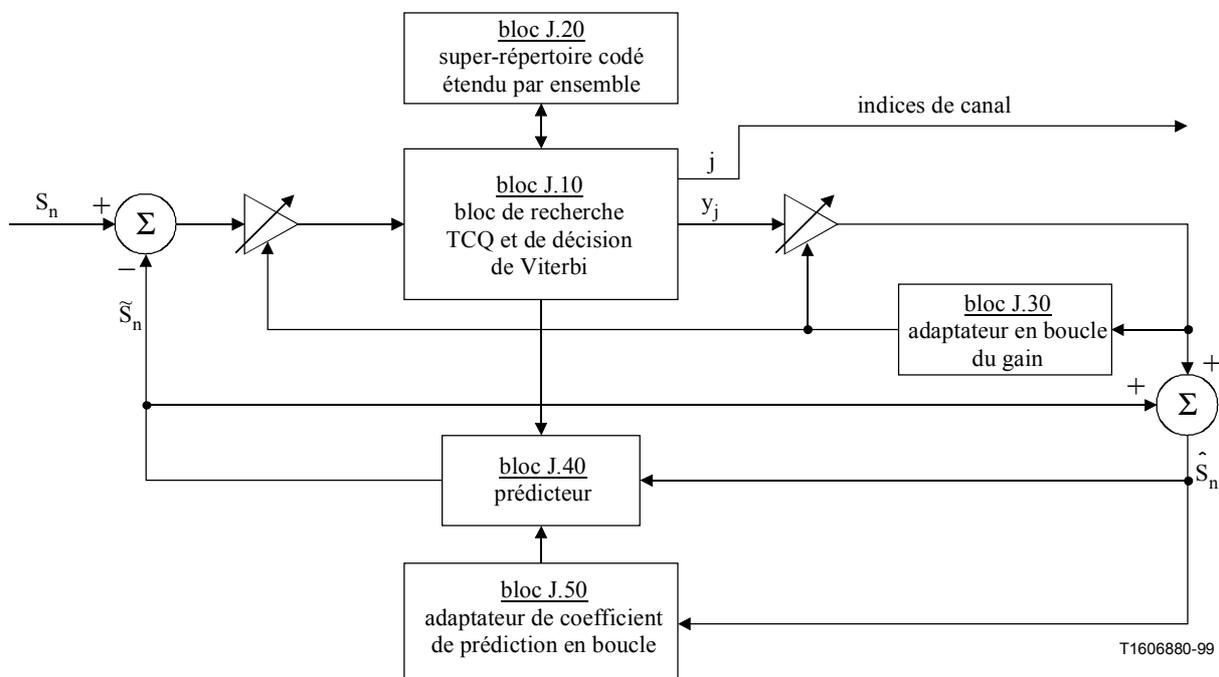


Figure J.1/G.728 – Structure du codec

### J.4.1.1 Bloc J.10 – Bloc de recherche TCQ et de décision de Viterbi

Ce bloc effectue toutes les opérations requises pour l'algorithme de quantification à codage en treillis (TCQ, *trellis-coded quantization*), comme la gestion des éléments subsistants du codage en treillis et des valeurs de reproduction spécifiées, le calcul et la comparaison des matrices ainsi que la détermination des décisions de Viterbi. Le codeur produit 5 symboles de canal sélectionnés par l'algorithme de Viterbi à partir du meilleur élément subsistant des 5 échantillons source.

La Figure J.2 montre l'automate d'états qui génère le diagramme en treillis et la Figure J.3 montre le diagramme en treillis. Les tables des J.4.6.1, J.4.6.2, J.4.6.3 et J.4.6.4 sont dérivées des figures mentionnées ci-dessus, tel qu'il est décrit ci-dessous.

Le sous-paragraphe J.4.6.1 énonce le parcours permis vers les nœuds **précédents** à travers le treillis, pour chaque nœud. Par exemple, les nœuds précédents permis pour le premier nœud ( $s[0]$ ) sont le nœud 0 sous la branche 0 ( $b[0]$ ) et le nœud 2 sous la branche 1 ( $b[1]$ ).

Le sous-paragraphe J.4.6.2 énonce le parcours permis vers les nœuds **suyvants** à travers le treillis, pour chaque nœud. Par exemple, les nœuds suivants permis pour le premier nœud ( $s[0]$ ) sont le nœud 0 sous la branche 0 ( $b[0]$ ) et le nœud 2 sous la branche 1 ( $b[1]$ ).

Le sous-paragraphe J.4.6.3 donne le sous-ensemble de quantification  $\{D0, D1, D2, D3\}$  associé à chaque parcours du treillis. Par exemple, la transition de  $s[0]$  à  $s[0]$  est associée au sous-ensemble D0. La transition de  $s[0]$  à  $s[1]$  est associée au sous-ensemble D2, et les transitions à  $s[2]$  et  $s[3]$  ne sont pas permises et sont donc marquées d'un X.

Le sous-paragraphe J.4.6.4 donne le bit d'indice avec lequel chaque transition est étiquetée et elle identifie les deux branches provenant de chaque nœud. Par exemple, la transition de  $s[0]$  à  $s[0]$  est associée à 0. La transition de  $s[0]$  à  $s[1]$  est associée à 1 (il est à noter que le bit 5 est utilisé et que 0x10 donne 10h en C) et les transitions à  $s[2]$  et  $s[3]$  ne sont pas permises et sont donc marquées d'un X.

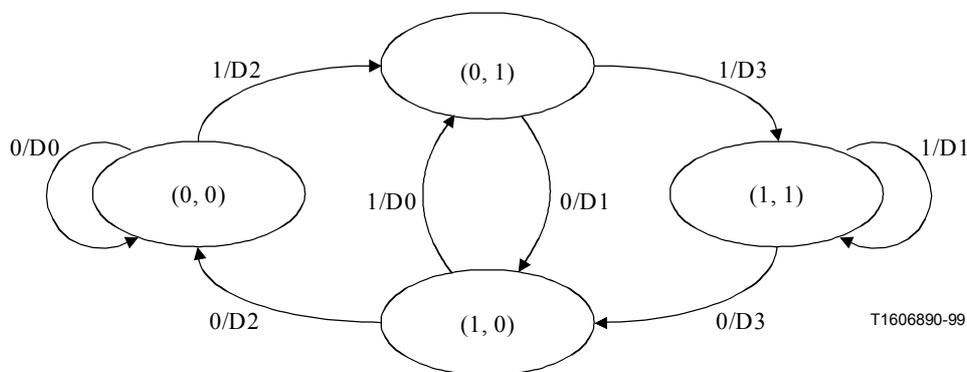


Figure J.2/G.728 – Automate d'états de la TCQ

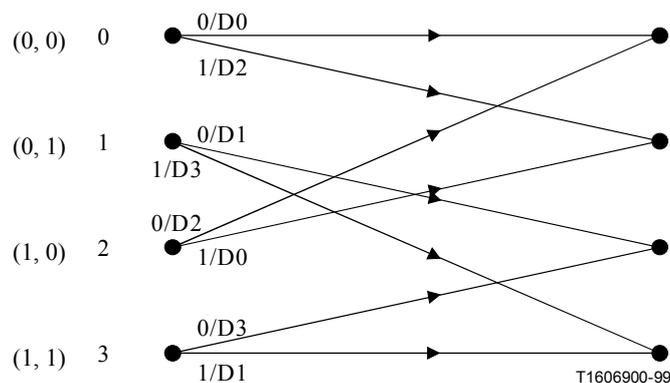


Figure J.3/G.728 – Diagramme en treillis de la TCQ

#### J.4.1.2 Bloc J.20 – Super-répertoire codé étendu par ensemble

Le super-répertoire codé est un quantificateur scalaire de Lloyd-Max étendu par ensemble. Les 64 niveaux de sortie sont divisés en quatre sous-ensembles, à partir du point le plus négatif jusqu'au point le plus positif, en étiquetant les points consécutifs  $\{D0, D1, D2, D3, \dots, D0, D1, D2, D3\}$ . Les niveaux de quantification sont donnés au J.4.6.6 et les limites des intervalles sont données au J.4.6.5. Les niveaux qui appartiennent au sous-ensemble D0 figurent dans la colonne marquée  $s[0]$ . Les niveaux D1 figurent sous  $s[1]$ , ceux de D2 figurent sous  $s[2]$  et ceux du sous-ensemble D3 figurent sous  $s[3]$ .

#### J.4.1.3 Bloc J.30 – Adaptateur en boucle du gain

La technique d'adaptation du gain du mode VBD est presque identique à celle énoncée dans la Recommandation G.728, pour la parole. Voici les principales différences:

- 1) En mode VBD, la valeur quadratique moyenne des valeurs de sortie du répertoire codé est calculée sur une séquence de niveaux de sortie (résidus quantifiés) spécifiée par le parcours de l'élément subsistant. La valeur quadratique moyenne est calculée sur une séquence de 8 échantillons. Par contre, à l'opposé de la situation de l'Annexe G où des tables précalculées contiennent les moyennes quadratiques logarithmiques, pour la transmission VBD, il est nécessaire de calculer la valeur logarithmique de la valeur quadratique moyenne. L'équation J.4-1 donne l'approximation logarithmique. Les coefficients  $d_0, d_1, d_2, d_3, d_4$  sont spécifiés au J.4.7 et la description détaillée du calculateur logarithmique est donnée au J.4.3.17.

$$2 * \log_{10}(x) = d_0 * (x-1) + d_1 * (x-1)^2 + d_2 * (x-1)^3 + d_3 * (x-1)^4 + d_4 * (x-1)^5 \quad (\text{J.4-1})$$

pour  $1 \leq x < 2$

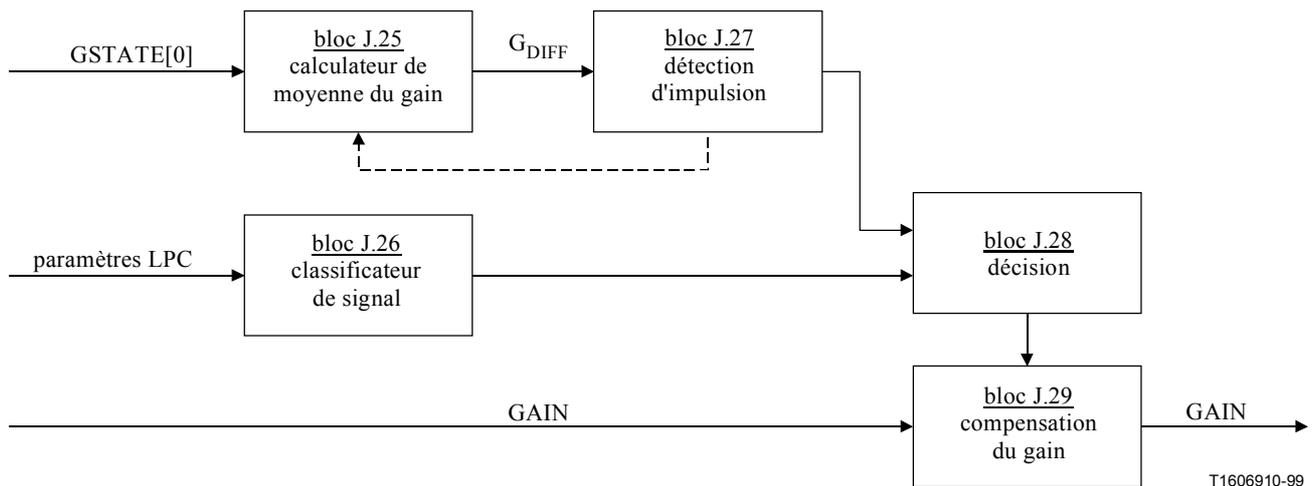
La valeur quadratique moyenne logarithmique remplace la sortie du répertoire codé de forme et de gain, blocs de tables de gain logarithmique G.93 et G.94 (les deux derniers termes de l'équation G-14).

- 2) Un filtre de lissage est inséré dans la boucle de gain logarithmique afin de réduire les oscillations permanentes des signaux présentant une variance stationnaire, comme les formes d'onde des données dans la bande vocale. Afin de surmonter les signaux de parole et de données, un algorithme de quantification à verrouillage dynamique (DLQ, *dynamic locking quantizer*) (5) réalise une adaptation à vitesse variable. Le DLQ est semblable au bloc DLQ de la Recommandation G.726.

L'entrant appliqué au DLQ est le gain logarithmique  $d(n)$  dont le décalage a été supprimé. La moyenne de cet entrant est calculée par le filtre de pondération (J.4.3.18, bloc J.14) afin de produire le gain verrouillé  $G_L$ .

Le quantificateur est en état purement verrouillé si  $a_1 = 0$  et en état purement déverrouillé si  $a_1 = 1$ .  $a_1$  est calculée en comparant l'énergie à long terme et l'énergie à court terme des résidus quantifiés  $ET(n)$  (J.4.3.10, bloc J.12). La comparaison donne la mesure de la constance de la variance des résidus quantifiés.

$$G = G_U * \alpha_1 + G_L * (1 - \alpha_1) \quad (J.4-2)$$



**Figure J.4/G.728 – Compensation du gain**

- 3) Des impulsions d'erreur de prédiction peuvent entraîner la saturation du quantificateur. Afin d'éviter cette situation, un groupe de cinq blocs additionnels (voir Figure J.4) apporte un changement temporel dans le gain de quantification. Ces blocs sont les suivants :

*Bloc J.25 – Calculateur de moyenne du gain*

Un filtre de lissage calcule la moyenne de l'estimation du gain,  $G_{ave}$ , à partir de la valeur de gain vectoriel la plus récente,  $GSTATE[0]$  (J.4.3.12, bloc J.25 et équation J.4-3). La différence entre  $GSTATE[0]$  et  $G_{ave}$  est calculée ( $G_{diff}$ ) et transmise au bloc de détection d'impulsion.

*Bloc J.27 – Bloc de détection d'impulsion*

Ce bloc détecte les changements soudains de gain après une période prédéterminée de gain constant (voir J.4.3.14, bloc J.27).  $G_{diff}$  est comparée à un seuil fixe. Si  $G_{diff}$  est inférieure au seuil pendant une période plus longue qu'une période prédéfinie, le signal est considéré comme étant "constant". Une impulsion d'erreur est détectée si  $G_{diff}$  est supérieure au seuil et que le signal précédent a été "constant".

*Bloc J.26 – Classificateur de signal*

Pendant certaines transmissions VBD, des impulsions d'erreur sont plus susceptibles de se produire. Par conséquent, lorsqu'elles sont détectées, la compensation du gain est maximisée. Le bloc classificateur de signal détecte ces transmissions au moyen du coefficient LP (voir J.4.3.13, bloc J.26).

### *Bloc J.28 – Décision*

Le bloc de décision reçoit la sortie du bloc classificateur de signal et la sortie du bloc de détection d'impulsion, et il active le bloc de compensation du gain (voir J.4.3.15, bloc J.28).

### *Bloc J.29 – Compensation du gain*

Ce bloc augmente le facteur du gain pendant une période fixe (à moins qu'un niveau crête de facteur du gain soit atteint, auquel cas la période est prolongée).

$$G_{ave} = G_{const} \times G_{ave} + (1 - G_{const}) \times GSTATE[0] \quad (J.4-3)$$

(voir J.4.3.16, bloc J.29).

#### **J.4.1.4 Bloc J.40 – Bloc prédicteur**

Le prédicteur est une version écourtée du filtre de synthèse de la Recommandation G.728 (bloc G.22). L'ordre du LPC est de 10 pondérateurs, au lieu des 50 pondérateurs habituellement utilisés dans le filtre de synthèse. La prédiction est basée sur le parcours de l'élément subsistant (voir J.4.3.4, bloc J.7) de la manière suivante: au temps  $n$ , une prédiction de l'échantillon courant est formée pour chaque nœud (J.4.3.5, bloc J.8) en utilisant la séquence de reproductions spécifiée par l'élément subsistant sélectionné au temps  $n - 1$ . Avec cette méthode, seule une prédiction scalaire à un échelon est exécutée et il n'est pas nécessaire que la prédiction soit prolongée loin dans le futur. La prédiction est ainsi plus "localisée" que dans beaucoup d'autres techniques de quantification vectorielle prédictives.

#### **J.4.1.5 Bloc J.50 – Adaptateur de coefficient de prédiction en boucle**

L'adaptateur de coefficient de prédiction en boucle est presque identique à l'adaptateur de filtre de synthèse en boucle (bloc G.23). Les principales différences sont les suivantes:

- seuls 10 paramètres LPC sont calculés. Le module de fenêtrage hybride (bloc G.49) calcule constamment 51 coefficients d'auto-corrélation, ce qui améliore les transitions données/parole;
- le facteur d'élargissement de largeur de bande du filtre de synthèse est maintenant 240/256. Les coefficients d'élargissement de la largeur de bande sont précisés au J.4.8.

#### **J.4.2 Structure du décodeur**

La Figure J.5 montre le schéma fonctionnel du décodeur. La Figure J.6 montre le module de mappage de signal, le bloc J.60. La séquence de 5 bits par échantillon reçue du décodeur est combinée à partir de deux types de séquences de bits. Le module de mappage de signal, le bloc J.60, divise la séquence combinée en deux séquences  $j_{4n}$ . Une séquence de 1 bit par échantillon est étendue à 2 bits au moyen d'un codeur convolutif. La sortie du codeur convolutif précise le parcours en treillis et sélectionne le sous-ensemble approprié. Le sous-ensemble est sélectionné selon les tables des J.4.6.2 et J.4.6.3. La sélection est décrite en détail au J.4.4.2. Les 4 bits restants par échantillon,  $j_{0n} \dots j_{3n}$ , sont utilisés pour sélectionner le niveau de sortie dans le sous-ensemble sélectionné. Le niveau de sortie est normalisé pendant l'adaptation en boucle du gain. Le niveau de sortie normalisé sur le plan du gain est ensuite appliqué au bloc prédicteur J.40 et il produit les valeurs de reproduction (voir Figure J.5).

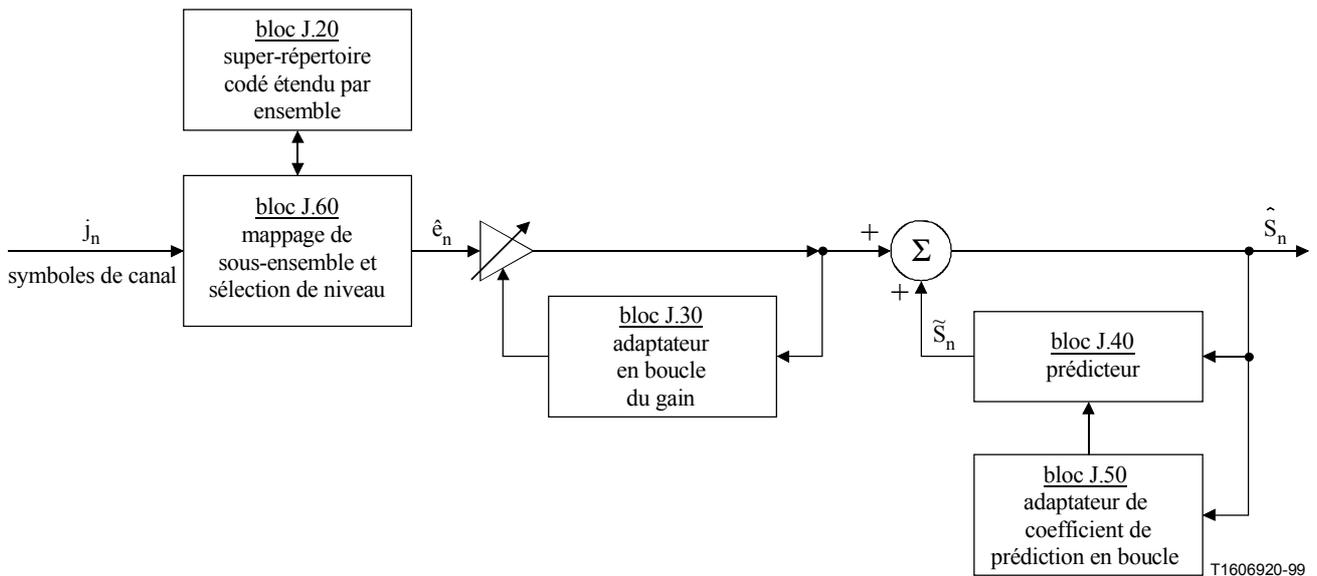


Figure J.5/G.728 – Structure du décodeur

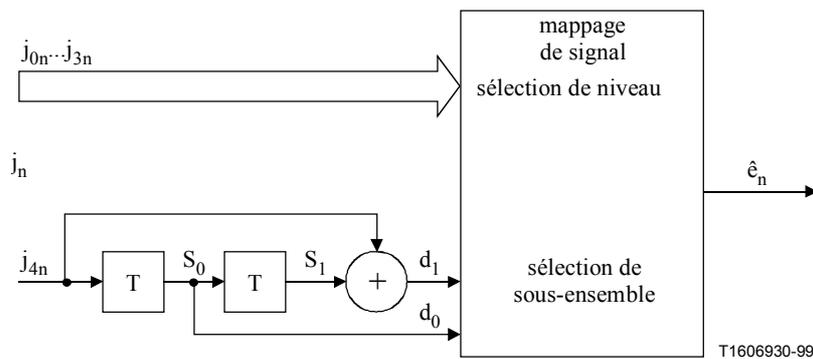
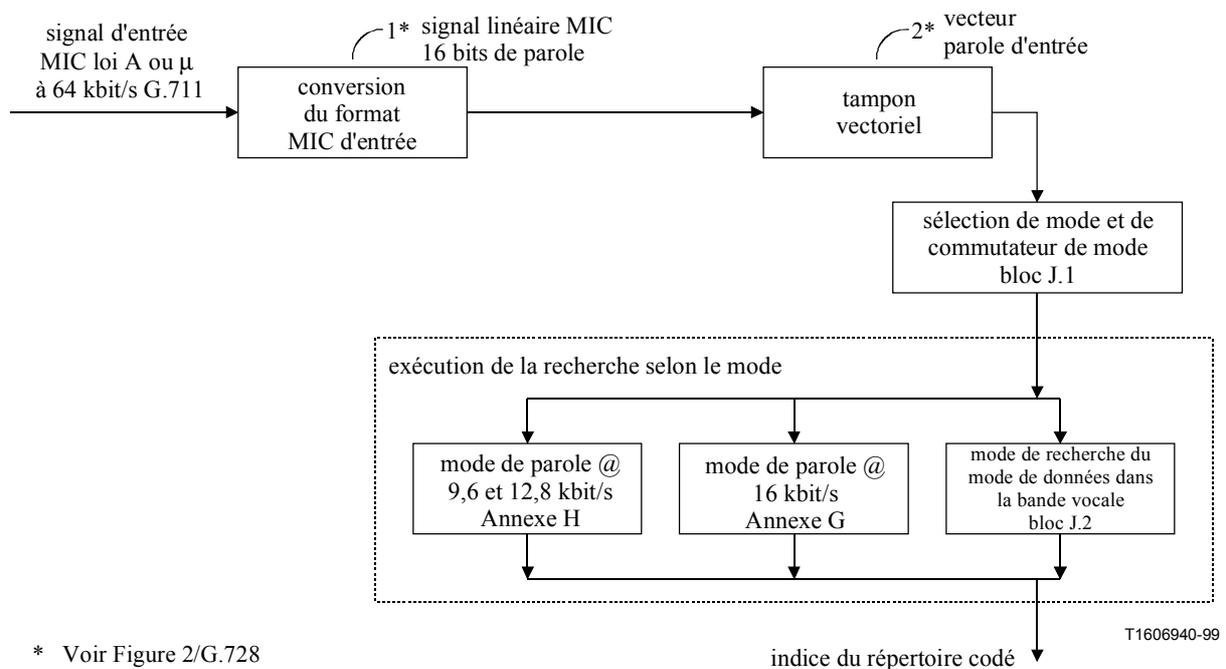


Figure J.6/G.728 – Module de mappage (bloc J.60)

### J.4.3 Description détaillée du codeur

Le présent sous-paragraphe fournit une description détaillée des blocs internes du codeur. Cette description est donnée sous forme de structure descendante commençant à la Figure J.7, qui donne une description schématique du codage de la Recommandation G.728 à divers débits de codage. Les deux blocs – bloc J.1 et bloc J.2 – sont dérivés de la Figure 2/G.728 [1]. Les deux blocs étiquetés respectivement Annexe H et Annexe G représentent les deux annexes à la Recommandation G.728 (spécification de système à virgule fixe et mode de fonctionnement à débit variable). Le bloc J.1 représente le commutateur de mode parole/données et données/parole. Le bloc J.2 représente le mode de données dans la bande vocale.

Plusieurs opérateurs en C (comme  $\&$ ,  $*$ ,  $\gg$ ,  $\ll$ ) sont utilisés dans cette description. Il est supposé que le lecteur est familier avec la définition de ces opérateurs (il est à noter que le mot "OR" remplace l'opérateur OU désigné par le symbole "|" du C au niveau du bit).



**Figure J.7/G.728 – Schéma fonctionnel des divers modes de fonctionnement de la Recommandation G.728**

#### J.4.3.1 Bloc J.2 – Mode de recherche du mode données dans la bande vocale

**Entrée:** S(n)

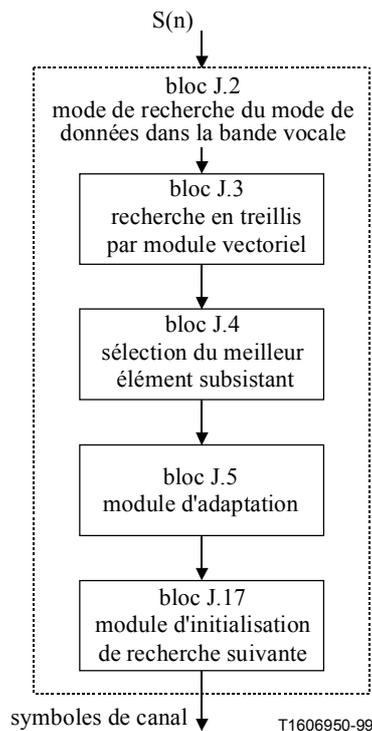
**Sortie:** TCQ\_channel\_symbols

**Fonction:** codage du vecteur d'entrée.

| Pour chaque vecteur (5 échantillons), exécuter les blocs suivants.

CALL BLOCK #J.3 | Recherche en treillis par module vectoriel.  
 CALL BLOCK #J.4 | Sélection du module du meilleur élément subsistant.  
 CALL BLOCK #J.5 | Module d'adaptation.

CALL BLOCK #J.17 | Initialisation du module de recherche suivant.



**Figure J.8/G.728 – Bloc J.2 Mode de données en bande vocale**

#### J.4.3.2 Bloc J.3 – Recherche en treillis par module vectoriel

**Entrée:** block\_depth, next\_stage

**Sortie:** voir la sortie des blocs J.8, J.9, J.10

**Fonction:** passage pas à pas séquentiel des échantillons dans le diagramme en treillis.

```

FOR I = 1, 2, ..., IDIM | Exécution de la ligne suivante
  CALL BLOCK #J.6 | TCQ transition.
  
```

```

block_depth = next_stage
next_stage = next_stage+1
IF (BLOCK_LEN <= next_stage)
  next_stage = 0
  
```

#### J.4.3.3 Bloc J.6 – Transition TCQ

**Entrée:** S(n), block\_depth, next\_stage

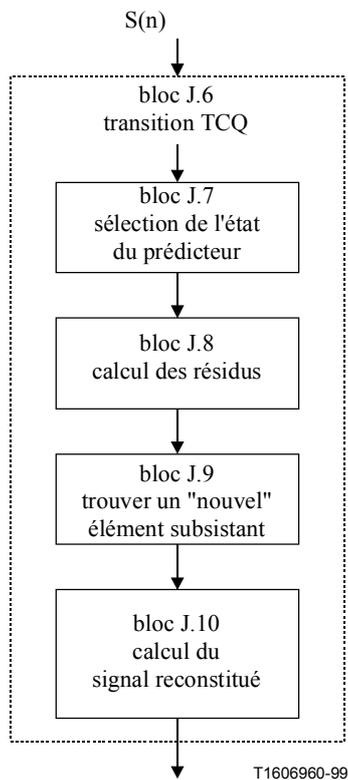
**Sortie:** voir la sortie des blocs J.8, J.9 et J.10

**Fonction:** un seul pas dans le diagramme en treillis.

| Pour chaque échantillon, exécuter les blocs suivants.

```

CALL BLOCK #J.7 | Sélection de l'état du prédicteur.
CALL BLOCK #J.8 | Calcul des résidus.
CALL BLOCK #J.9 | Sélection du "nouvel" élément subsistant.
CALL BLOCK #J.10 | Calcul du signal reconstitué.
  
```



**Figure J.9/G.728 – Bloc J.6 Transition TCQ**

#### J.4.3.4 Bloc J.7 – Sélection de l'état du prédicteur

**Entrée:** block\_depth, TCQ\_reconstructed\_q2

**Sortie:** TCQ\_predictor\_state\_q2

**Fonction:** pour chaque nœud 0, 1, 2, 3 (Figure J.3), constituer l'état du prédicteur en retraçant le parcours des niveaux reconstitués du treillis.

NOTE – L'état du filtre prédictif se compose de deux séquences:

**Séquence 1:** les niveaux reconstitués sélectionnés précédemment (à partir de l'indice: block\_depth+1, block\_depth+2, ..., et ainsi de suite jusqu'à 10) qui avaient été sélectionnés pendant le cycle de recherche précédent sont valides pour tous les états du treillis (nœuds) et ils sont calculés une fois par échantillon.

**Séquence 2:** chaque nœud est associé à la séquence de niveau reconstitué courante (à partir des indices 1, 2, et ainsi de suite jusqu'à block\_depth), qui est en cours de formation dans le treillis des niveaux reconstitués.

Ce module constitue la séquence 2 de l'état du prédicteur de chaque nœud.

```

FOR d = 0, 1, ..., (N_STATES - 1) | Exécution des 6 lignes suivantes
  IF (0 < block_depth)
    I=0
    src = d
    FOR k = block_depth, block_depth - 1, ..., 1 | Exécution des 3 lignes
      suivantes
        TCQ_predictor_state_q2[d][i] = TCQ_reconstructed_q2[k][src]
        src = prev_node [k] [src]
        i=i+1
  
```

### J.4.3.5 Bloc J.8 – Calcul des résidus

**Entrée:** block\_depth, A, TCQ\_STTMP\_q2, DLQ\_inv\_GAIN, DLQ\_inv\_nls\_m, échantillon

**Sortie:** TCQ\_predict\_sample\_q2, TCQ\_resid

**Fonction:** pour chaque nœud 0, 1, 2, 3 (Figure J.3), calculer la valeur prédite associée et les résidus.

NOTE – Chaque nœud de l'état courant est associé à une seule prédiction, alors que les nœuds de l'état suivant sont associés à deux niveaux de prédiction (une pour chaque branche d'arrivée). En conséquence, il est préférable d'associer les prévisions au nœud de l'état courant (temps de traitement plus court) et de les référencer par block\_depth.

```
FOR d = 0, 1, ..., (N_STATES - 1) | Exécuter les lignes jusqu'à END J.8

| Calculer la partie de la valeur prédite qui est valide (commune)
| pour tous les états du treillis (nœuds). Voir note au J.4.3.4.

    IF (d == 0) | Exécuter l'instruction FOR suivante. Faire le calcul une seule
    fois, pendant le premier nœud.
        TCQ_common_part_pred_q2 = 0
        FOR i = 0, 1, ... (PREDICTOR_ORDER - block_depth-1)
            TCQ_common_part_pred_q2=TCQ_common_part_pred_q2-TCQ_STTMP_q2
            [(NFRSZ - 1) - i] * A[i+1+block_depth]
        | Fin de IF (d == 0).
| Pour chaque état de treillis (nœud), calculer la prédiction pour l'échantillon
| courant.
AA0 = TCQ_common_part_pred_q2
FOR i = 0, 1, ..., (block_depth - 1) | Exécution de la ligne suivante
    AA0 = AA0 - TCQ_predictor_state_q2[d][i] * A[i+1]

AA0 = rnd_int (AA0<<2) | Représentation Q2.

TCQ_predict_sample_q2[d] = AA0 | sauvegarder la valeur prédite.
TCQ_resid[d] = sample - AA0

| Normaliser les résidus.

AA0 = TCQ_resid[d] * DLQ_inv_GAIN
AA1 = -1*(DLQ_inv_nls_m-1)
IF (0 <= AA1)
    AA0 = AA0 >> AA1
ELSE
    AA1 = -1*AA1
    AA0 = AA0 << AA1

| Forcer la saturation.

IF (32767L < AA0)
    AA0 = 32767L

IF (AA0 < -32768L)
    AA0 = -32768L
TCQ_resid [d] = AA0 | Sauvegarder le résidu normalisé.

END#J.8:
```

### J.4.3.6 Bloc J.9 – Trouver un "nouvel" élément subsistant

**Entrée:** block\_depth, next\_stage, TCQ\_resid

**Sortie:** distortion\_metric, Q\_resid, prev\_node, TCQ\_channel\_indices

**Fonction:** pour chaque état de treillis "next\_stage" (nœud) 0, 1, 2 et 3 (Figure J.3), sélectionner une des deux branches d'arrivée et la marquer comme "nouvel" élément subsistant pour cet état de treillis (nœud).

NOTE – Chaque nœud d'état suivant est associé à un seul élément survivant (une des deux branches d'arrivée). Par conséquent, il est préférable d'associer le "nouvel" élément subsistant à un nœud d'état suivant et de le référencer au moyen du pointeur next\_stage.

La complexité d'exécution de ce bloc (millions d'instructions par seconde, MIPS) est considérable.

```
| Pour chaque état de treillis, quantifier les résidus associés en utilisant  
| le sous-ensemble qui étiquette les deux branches d'arrivée (Figure J.3).
```

```
| Effectuer 8 quantifications. CALL BLOCK #J.11 8 fois.  
TCQ_quantize_resid ( TCQ_resid [0], 0, &ch_index[0], &qerror[0] )  
TCQ_quantize_resid ( TCQ_resid [0], 2, &ch_index[1], &qerror[1] )  
TCQ_quantize_resid ( TCQ_resid [1], 1, &ch_index[2], &qerror[2] )  
TCQ_quantize_resid ( TCQ_resid [1], 3, &ch_index[3], &qerror[3] )  
TCQ_quantize_resid ( TCQ_resid [2], 2, &ch_index[4], &qerror[4] )  
TCQ_quantize_resid ( TCQ_resid [2], 0, &ch_index[5], &qerror[5] )  
TCQ_quantize_resid ( TCQ_resid [3], 3, &ch_index[6], &qerror[6] )  
TCQ_quantize_resid ( TCQ_resid [3], 1, &ch_index[7], &qerror[7] )
```

```
| Calculer les valeurs de distorsion associées à chaque nœud.
```

```
AA0 = (TCQ_resid[0] - qerror[0]) * (TCQ_resid[0] - qerror[0])  
AA0 = AA0 >> 2  
temp_metric[0] = distortion_metric[0] + AA0
```

```
AA0 = (TCQ_resid[0] - qerror[1]) * (TCQ_resid [0] - qerror[1])  
AA0 = AA0 >> 2  
temp_metric[1] = distortion_metric[0] + AA0
```

```
AA0 = (TCQ_resid[1] - qerror[2]) * (TCQ_resid[1] - qerror[2])  
AA0 = AA0 >> 2  
temp_metric[2] = distortion_metric[1] + AA0
```

```
AA0 = (TCQ_resid[1] - qerror[3]) * (TCQ_resid[1] - qerror[3])  
AA0 = AA0 >> 2  
temp_metric[3] = distortion_metric[1] + AA0
```

```
AA0 = (TCQ_resid[2] - qerror[4]) * (TCQ_resid[2] - qerror[4])  
AA0 = AA0 >> 2  
temp_metric[4] = distortion_metric[2] + AA0
```

```
AA0 = (TCQ_resid[2] - qerror[5]) * (TCQ_resid[2] - qerror[5])  
AA0 = AA0 >> 2  
temp_metric[5] = distortion_metric[2] + AA0
```

```
AA0 = (TCQ_resid[3] - qerror[6]) * (TCQ_resid[3] - qerror[6])  
AA0 = AA0 >> 2  
temp_metric[6] = distortion_metric[3] + AA0
```

```
AA0 = (TCQ_resid[3] - qerror[7]) * (TCQ_resid[3] - qerror[7])  
AA0 = AA0 >> 2  
temp_metric[7] = distortion_metric[3] + AA0
```

```
| Sélectionner le nouvel élément subsistant pour chaque nœud.
```

```
| nœud 0
```

```
IF (temp_metric[0] < temp_metric[4])  
    distortion_metric[0] = temp_metric[0]  
    Q_resid[0] = qerror[0]
```

```

    prev_node [next_stage][0] = 0
    TCQ_channel_indices [next_stage] [0] = ch_index[0]
ELSE
    distortion_metric[0] = temp_metric[4]
    Q_resid[0] = qerror[4]
    prev_node [next_stage][0] = 2
    TCQ_channel_indices [next_stage] [0] = ch_index[4]

| nœud 1

IF (temp_metric[1] < temp_metric[5])
    distortion_metric[1] = temp_metric[1]
    Q_resid[1] = qerror[1]
    prev_node [next_stage][1] = 0
    TCQ_channel_indices [next_stage] [1] = ch_index[1] OR 0x10
ELSE
    distortion_metric[1] = temp_metric[5]
    Q_resid[1] = qerror[5]
    prev_node [next_stage][1] = 2
    TCQ_channel_indices [next_stage] [1] = ch_index[5] OR 0x10

| nœud 2

IF (temp_metric[2] < temp_metric[6])
    distortion_metric[2] = temp_metric[2]
    Q_resid[2] = qerror[2]
    prev_node [next_stage][2] = 1
    TCQ_channel_indices [next_stage] [2] = ch_index[2]
ELSE
    distortion_metric[2] = temp_metric[6]
    Q_resid[2] = qerror[6]
    prev_node [next_stage][2] = 3
    TCQ_channel_indices [next_stage] [2] = ch_index[6]

| nœud 3

IF (temp_metric[3] < temp_metric[7])
    distortion_metric[3] = temp_metric[3]
    Q_resid[3] = qerror[3]
    prev_node [next_stage][3] = 1
    TCQ_channel_indices [next_stage] [3] = ch_index[3] OR 0x10
ELSE
    distortion_metric[3] = temp_metric[7]
    Q_resid[3] = qerror[7]
    prev_node [next_stage][3] = 3
    TCQ_channel_indices [next_stage] [3] = ch_index[7] OR 0x10

```

#### J.4.3.7 Bloc J.11 – Résidu de quantification TCQ

**Entrée:** résidu, état

**Sortie:** index, qerror

**Fonction:** quantification d'un seul résidu au moyen d'un seul sous-ensemble.

NOTE – Ce sous-programme est exécuté huit fois par échantillon; chaque quantification est une quantification sur 4 bits et 16 niveaux.

```

index = 0
while ( (Xk[state][index] < resid) && (index < Q_CELLS) ) | Exécution de la ligne
suivante
    index = index+1
*qerror = Yk[state][index] | Produire le résidu quantifié.

```

#### J.4.3.8 Bloc J.10 – Calcul du signal reconstitué

**Entrée:** next\_stage DLQ\_GAIN, DLQ\_nls\_p\_cb\_q\_m\_18, Q\_resid, prev\_node

**Sortie:** TCQ\_reconstructed

**Fonction:** pour chaque nœud 0, 1, 2, 3 (Figure J.3), calculer le niveau reconstitué.

NOTE – Le niveau reconstitué est désigné par next\_stage. Voir Note au J.4.3.6.

Variables internes (définitions en C)

- int src;
- long int AA0;
- short int nls;

```
FOR d = 0, 1, ..., (N_STATES - 1)
| Pour chaque état de treillis (nœud) présenté par la variable d,
| exécuter les blocs suivants jusqu'à END J.10.

src = prev_node [next_stage] [d] | Trouver le nœud précédent.

TCQ_ET [ next_stage ] [ d ] = Q_resid [ d ]

AA0 = Q_resid[d] * DLQ_GAIN

IF (0 <= DLQ_nls_p_cb_q_m_18 )
    AA0 <<= DLQ_nls_p_cb_q_m_18
ELSE
    AA0 >>= abs (DLQ_nls_p_cb_q_m_18)
    AA0 = AA0 + (TCQ_predict_sample_q2 [src] << 16L)
    TCQ_reconstructed_q2 [next_stage] [d] = AA0 >> 16L

END #J.10:
```

#### J.4.3.9 Bloc J.4 – Sélection du meilleur élément subsistant

**Entrée:** distortion\_metric, prev\_node, TCQ\_ET, TCQ\_reconstructed\_q2, TCQ\_channel\_indices

**Sortie:** TCQ\_channel\_symbols, ET, TCQ\_STTMP\_q2

**Fonction:** sélectionner l'élément subsistant/nœud et la séquence de symboles de canaux associée.

NOTE – Ce module est exécuté après l'achèvement de 5 étapes dans le parcours en treillis. Une séquence de 5 symboles de canal (5 bits par échantillon) doit être sélectionnée (élément subsistant/nœud).

Variables internes (définitions en C)

- int src;
- int survivor\_node;
- long int min\_dist;

```
| Décaler les échantillons TCQ_STTMP_q2 Kr vers l'arrière.
FOR i = TCQ_Kr.. (NFRSZ-1)
    TCQ_STTMP_q2 [i - TCQ_Kr] = TCQ_STTMP_q2[i]
| Décaler ET vers l'arrière.
FOR i = TCQ_Kr..(RMS_BUF_LEN-1)
    ET [i - TCQ_Kr] = ET [i]

| Sélectionner le nœud présentant le moins de distorsion de quantification et
| la séquence d'échantillons reconstitués associée.
```

```

min_dist = MAX_NUMBER
FOR d= 0..(N_STATES-1)
    IF (distortion_metric[d] <= min_dist) | Exécuter les 2 lignes suivantes
        min_dist = distortion_metric[d]
        survivor_node = d

| Copier les derniers échantillons Kr de la
| séquence d'éléments subsistants dans TCQ_STTMP_q2.

k = 0 | TCQ_Kr == TCQ_Kd as in Kd/Kr=5/5.
src = survivor_node

FOR i = TCQ_DEPTH..(TCQ_Kd - TCQ_Kr) | Exécution des 2 lignes suivantes
| i est décrémenté.
| Disposer les reproductions en ordre ascendant.

    TCQ_STTMP_q2 [ i + (NFRSZ-(TCQ_DEPTH+1))]=TCQ_reconstructed_q2 [k] [src]
    ET[ i - (TCQ_Kd - TCQ_Kr) + (RMS_BUF_LEN - TCQ_Kr) ] = TCQ_ET [k] [src]
    TCQ_channel_symbols [ i - (TCQ_Kd - TCQ_Kr) ] =TCQ_channel_indices [ k ]
    [ src ]
    src = prev_node [k] [src]

    k = k -1
    IF (k < 0)
        k = TCQ_DEPTH
| Fin de FOR.

| Estimer le gain pour les 5 échantillons suivants.

CALL BLOCK #J.12 | Adaptateur en boucle du gain de TCQ.

```

#### J.4.3.10 Bloc J.12 – Adaptateur en boucle du gain de TCQ

**Entrée:** next\_stage

**Sortie:** DLQ\_GAIN, DLQ\_NLSGAIN

**Fonction:** calcul du gain normalisé s(n) effectué une fois par vecteur.

NOTE – Ce bloc remplace le bloc 20 (Annexe G/G.728) en mode de données.

Variables internes (définitions en C)

- int src;
- int survivor\_node;
- long int min\_dist;

```

| Calcul de la valeur quadratique moyenne du gain logarithmique pour le dernier
| segment de parcours sélectionné.

```

```

CALL BLOCK #J.13 | vbd_log_calc_and_lim97

```

```

| Calcul des paramètres de décision DLQ "dms" et "dml".

```

```

dms = (dms << 5) - dms
dml = (dml << 7) - dml
dms += RMS_Q11
dml += RMS_Q11
dms >>= 5
dml >>= 7
diff = labs(dms-dml)

```

```

IF (diff >= (dml >> 3L))
    | Puissance moyenne variable, signaux non stationnaires
    | comme la parole, ap → 2, et déverrouillage du quantificateur.

```

```

    ap_q11 = (ap_q11 * 15 ) >> 4 | ap = ap * 15.0 /16.0
    ap_q11 += (1 Q11 ) >> 3
ELSE
    IF (DLQ_GAIN >> DLQ_NLSGAIN < 10)
        | Conditions de canal au repos, ap → 2, et déverrouillage
        | du quantificateur.

        ap_q11 = (ap_q11 * 15 ) >> 4
        ap_q11 += (1 Q11 ) >> 3
    ELSE
        | Puissance moyenne constante, signaux stationnaires
        | comme des données en bande vocale ap → 2, et verrouillage
        | du quantificateur.

        ap_q11 = (ap_q11 * 15 ) >> 4

IF (1 Q11 < ap_q11)
    al_q11 = 1 Q11
ELSE
    al_q11 = ap_q11

FOR i = 0..NUPDATE | Sauvegarder GSTATE dans l'Annexe G/G.728.
    GTMP[i] = GSTATE[i]
    | Prédiction du gain.

    | Le bloc 46, p. 39 de l'Annexe G de la Recommandation G.728 est le
    | prédicteur linéaire du gain logarithmique.

CALL BLOCK #G.46 | Prédiction linéaire du gain logarithmique.
CALL BLOCK #G.98 | Limiteur du gain logarithmique 98, partie des blocs 46 et 47,
    | p.39.
CALL BLOCK #J.14 | Pondération du gain logarithmique.

GAIN = after_limiter_98 | GAIN est la moyenne du gain logarithmique.

CALL BLOCK #G.99 | add_log_offset, partie des blocs 46 et 47, p. 39.
CALL BLOCK #G.48 | Exponentiateur.

DLQ_GAIN = GAIN
DLQ_NLSGAIN = NLSGAIN
CALL BLOCK #J.15 | Inverse du GAIN.

```

#### J.4.3.11 Bloc J.13 – vbd\_log\_calc\_and\_lim97

**Entrée:** ET

**Sortie:** GSTATE[0], RMS\_Q11

**Fonction:** calculer la valeur quadratique moyenne du gain logarithmique.

NOTE – Ce bloc remplace les blocs 67, 39 et 40 (retard de 1 vecteur, calculateur de valeur quadratique moyenne et calculateur logarithmique) dans la version en virgule flottante, et les blocs 93, 94, 96 et 97 de l'Annexe G/G.728, dans la version à virgule fixe.

Variables internes (définitions en C)

- long int AA0;
- long int AA1;

```

AA0 = 0
FOR i=(RMS_BUF_LEN - 1), (RMS_BUF_LEN - 2),..., 1
    AA0 += ET [i] * ET [i]
AA0 += ET[0] * ET[0]

```

```

| Division par RMS_BUF_LEN et normalisation de AA0 de Q22 à Q11.
AA0 = (AA0 + 1 Q13) >> 14

RMS_Q11 = AA0 | opération sur 32 bits.

AA0 = calc_10log10(AA0) | Le bloc J.16 est mis en œuvre en tant que fonction.

AA0 = AA0 + (after_limiter_98 << 2) | opération sur 32 bits.

AA0 = AA0 >> 2 | Normaliser au format Q9.

| bloc 97, limiter GSTATE à -32dB.

IF (AA0 < -16384)
    AA0 = -16384

| Si immédiatement après une transition, utiliser GSTATE du mode de transmission
| de la parole.
IF (speech_to_vbd_transition == 1)
    speech_to_vbd_transition = 0
ELSE
    GSTATE[0]=AA0

```

#### J.4.3.12 Bloc J.25 – Calculateur de moyenne du gain

**Entrée:** GSTATE[0], UNSTEADY

**Sortie:** G\_DIFF, G\_CNT

**Fonction:** calcul d'une valeur quasi-moyenne du gain.

Variables internes (définitions en C)

- long int G\_AVE

```

GDIFF=GSTATE[0]-G_AVE;
IF UNSTEADY=1 | exécuter les 3 lignes suivantes
    G_AVE=GSTATE[0]
    G_CNT=0
    UNSTEADY=0
ELSE | exécuter les 3 lignes suivantes
    IF GDIFF<G_TRS | exécuter les 2 lignes suivantes:
        G_AVE=((G_AVE<<G_CONST-G_AVE+GSTATE[0])>>G_CONST
        G_CNT++

```

#### J.4.3.13 Bloc J.26 – Classificateur de signal

**Entrées:** ATMP

**Sorties:** GC\_SC\_FLAG

**Fonction:** détection de signal à faible largeur de bande

Variables internes (définitions en C)

- int GC\_ATMP\_SUM
- int GC\_ATMP1

```

GC_ATMP_SUM=0
GC_ATMP1=ATMP[1]
FOR I=2,3,...LPC+1, |exécuter la ligne suivante
    GC_ATMP_SUM=GC_ATMP_SUM+ABS(ATMP[I])

IF ((GC_ATMP_SUM*ATMP_CONST)>>3)<ABS(GC_ATMP1)
    GC_SC_FLAG=1
ELSE
    GC_SC_FLAG=0

```

#### J.4.3.14 Bloc J.27 – Détection d'impulsion

**Entrées:** GDIFF, GC\_LEN, GC\_SC\_FLAG

**Sorties:** GC\_ID\_FLAG

**Fonction:** recherche d'une augmentation soudaine du gain après une période prédéfinie de gain constant.

```
IF GDIFF > G_TRS          | exécuter les 4 lignes suivantes
  IF G_CNT>GC_LEN
    GC_ID_FLAG=1
  ELSE
    GC_ID_FLAG=0
```

#### J.4.3.15 Bloc J.28 – Décision de compensation du gain

**Entrées:** GC\_ID\_FLAG, GC\_SC\_FLAG

**Sorties:** GC\_FLAG, GC\_CNT, UNSTEADY, GC-NLS\_LIMIT, GC\_COMPENSATION

**Fonction:** logique décisionnelle pour le bloc de compensation du gain.

```
GC_LEN=0
GC-NLS_LIMIT=16383
GC_COMPENSATION=0
IF GC_SC_FLAG=1          | exécuter les 3 lignes suivantes
  GC_LEN=GC_CNT_INIT
  GC-NLS_LIMIT=GC-NLS_LIMIT_INIT
  GC_COMPENSATION=GC_COMPENSATION_INIT
IF GC_ID_FLAG=1          | exécuter les 3 lignes suivantes
  UNSTEADY=1
  GC_FLAG=1              | Fanion de compensation du gain
  GC_CNT=GC_LEN
```

#### J.4.3.16 Bloc J.29 – Compensation du gain

**Entrées:** GC\_FLAG, DLQ\_NLSGAIN, GG\_CNT, GC\_COMPENSATION, GC-NLS\_LIMIT

**Sorties:** GC\_FLAG

**Fonction:** décrémenter DLQ\_NLSGAIN d'une valeur fixe pendant une période prédéfinie.

```
IF GC_FLAG=1            | exécuter les 7 lignes suivantes
  IF DLQ_NLSGAIN>GC-NLS_LIMIT | exécuter les 6 lignes suivantes
    GC_CNT=GC_CNT-1
    DLQ_NLSGAIN=DLQ_NLSGAIN-GC_COMPENSATION
  IF DLQ_NLSGAIN< GC-NLS_LIMIT
    DLQ_NLSGAIN= GC-NLS_LIMIT
  IF GC_CNT=0
    GC_FLAG=0
```

#### J.4.3.17 Bloc J.16 – Calculateur logarithmique

**Entrée:** AA0. Format Q11. (Définition en C: long int AA0).

**Sortie:** Valeur logarithmique

**Fonction:** approximation de la fonction logarithmique.

NOTE – L'approximation est valide pour  $1 \leq x < 2$ .

Variables internes (définitions en C)

- long int T;
- short int exp;

```

IF (AA0 < 1)
AA0 = 0 | Nombre d'entrée illégal, écrêtage à 0dB.
ELSE
| Normaliser le nombre d'entrée à la plage 1Q14..2Q14
exp = 3 | Q3 est égal à la différence Q14 - Q11
WHILE ( AA0 < 1 Q14 ) | Exécuter les deux lignes suivantes
AA0 = (AA0 << 1)
exp -= 1

WHILE (2 Q14 <= AA0) | Exécuter les deux lignes suivantes
AA0 = (AA0 >> 1L)
exp += 1

T = AA0 - 1 Q14
T <= 1 | Transposer T de Q14 à Q15
AA0 = log_pol[LOG_POL_ORDER-1]

FOR i = (LOG_POL_ORDER-1), (LOG_POL_ORDER-2).., 1
AA0 = ((T * AA0) * 2 + (log_pol[i-1] << 16) + 1 Q15) >> 16

AA0 = ( (AA0 * T) * 2 + 1 Q15) >> 16
AA0 = AA0 >> 4 | Convertir AA0 à Q11.

AA0 = AA0 * 5 | Diviser AA0 par 2 et multiplier le résultat par 10 pour obtenir
des décibels.

| Ajouter la mantisse et calculer la valeur logarithmique en base 10 des
| décibels.

AA0 = AA0 + (log_2 * exp) >> 2

return(AA0)

```

#### J.4.3.18 Bloc J.14 – Pondération du gain logarithmique

**Entrée:** after\_limiter\_98, GAIN\_state, al\_q11

**Sortie:** after\_limiter\_98

**Fonction:** calculer la moyenne du gain logarithmique et produire la vitesse d'adaptation variable.

Variables internes (définitions en C)

- long int AA0, AA1 | Accumulateurs 32 bits;
- short int locked, unlocked;

```

AA0 = (long)GAIN_state
AA1 = AA0 << 6L | 63/64 iir.
AA1 = AA1 - AA0

```

```

| Ajouter AA1 (format Q9) à after_limiter_98 et arrondir le résultat à Q9 sur
| 16 bits.

```

```

AA0 = after_limiter_98
AA1 = AA1 + AA0

```

```

AA0 = AA1 >> 6L | 63/64 iir.

```

```

locked = AA0
unlocked = after_limiter_98

```

```
AA0 = locked * ( 1 Q11 - al_q11)
AA0 = AA0 + unlocked * al_q11
AA0 >= 11
```

```
after_limiter_98 = AA0
GAIN_state = AA0
```

#### J.4.3.19 Bloc J.15 – Inverse du gain

**Entrée:** DLQ\_GAIN, DLQ\_NLSGAIN

**Sortie:** DLQ\_inv\_GAIN, DLQ\_nls\_p\_cb\_q\_m\_18, DLQ\_inv\_nls\_m

**Fonction:** inversion du gain.

Variables internes (définitions en C)

- long int NUM | Constante 16384: 1 en format Q14
- long int NUMNLS | Constante 14

```
DLQ_nls_p_cb_q_m_18 = 18 - CODEBOOK_Q - DLQ_NLSGAIN |
| Initialisation du numérateur pour l'inversion.
```

```
NUM = 16384
NUMNLS = 14
```

```
divide ( NUM, NUMNLS, DLQ_GAIN, DLQ_NLSGAIN, &DLQ_inv_GAIN, &inv_nls) | Cette
division est une fonction de l'Annexe G/G.728.
```

```
DLQ_inv_nls_m = - 2 - inv_nls + CODEBOOK_Q + 1
```

#### J.4.3.20 Bloc J.5 – Module d'adaptation

**Fonction:** exécution du cycle d'adaptation.

NOTE – Le même cycle d'adaptation est utilisé pour les données et pour la parole. Seule la conversion de quelques variables au format d'adaptation de l'Annexe G/G.728 est requise avant chaque phase d'adaptation.

Variables internes (définitions en C)

- long int nls;
- short int min\_nls;

```
ICOUNT = ICOUNT + 1
```

```
IF (ICOUNT > NUPDATE)
    ICOUNT = 1
```

```
IF (ICOUNT == 4)
```

```
    FOR k = 0, 1, ..., (NUPDATE - 1)
```

```
        | Préparation de STTMP et NLSSTTMP pour le sous-programme
        | d'autocorrélation, -HW_s.
```

```
        VSCALE( &TCQ_STTMP_q2[k * IDIM], IDIM,
                IDIM, 12, &STTMP[k * IDIM], &NLSSTTMP[k])
        NLSSTTMP[k] = NLSSTTMP[k] + 2 | Correction Q2.
```

```
    CALL BLOCK #G.49 | HW_s
```

```
IF (ICOUNT == 2)
```

```
    IF (ILLCOND == 0)
```

```
        durbin (RTMP, ATMP, 10) | Bloc G.50
```

```
        IF (DurbinFaultFlag == 0)
```

```
            CALL BLOCK #J.26 | Classificateur de signal
```

```
            CALL BLOCK #J.51 | bandexpand51_vbd()
```

```

ELSE
    DurbinFaultFlag = 0
    FOR I=1, 2,..LPC
        ATMP[i]=A[i]

IF (ICOUNT == 1)
    CALL BLOCK #G.43 | HW_gain
    IF ( ILLCOND == 0 )
        durbin(R, GPTMP, LPCLG) | Bloc G.44
    IF (DurbinFaultFlag == 0)
        CALL BLOCK #G.45 | bandexpand45
    ELSE
        DurbinFaultFlag = 0

IF (ICOUNT == 3)
    FOR i = 1, 2.., PREDICTOR_ORDER
        A[i] = ATMP[i]

```

#### J.4.3.21 Bloc J.51 – Module d'élargissement de largeur de bande

**Entrée:** voir bloc G.51

**Sortie:** voir bloc G.51

**Fonction:** voir bloc G.51.

NOTE – Ce bloc est identique au bloc G.51, sauf que la table FACV\_vbd remplace FACV, qui comporte 10 éléments non nuls.

#### J.4.3.22 Bloc J.17 – Module d'initialisation de recherche suivante

**Entrée:** survivor\_node

**Sortie:** distortion\_metric

**Fonction:** établissement des paramètres pour une nouvelle recherche.

```

FOR i = 0, 1, ..,(N_STATES-1)
    distortion_metric[i] = (MAX_NUMBER >> 2)

| Mesures prises pour que le parcours sélectionné passe par
| le nœud subsistant.
distortion_metric[survivor_node] = 0L

block_depth = 0
next_stage = 1

```

### J.4.4 Description détaillée du décodeur

La présente section fournit une description détaillée du décodeur.

#### J.4.4.1 Bloc J.20 – Module décodeur

**Entrée:** channel\_symbol

**Sortie:** reconstructed\_sample\_q2

**Fonction:** effectuer les opérations de décodage par vecteur.

```

For I = 1, 2, ..., IDIM | Exécution de la ligne suivante
    CALL BLOCK #J.21 | Module de transition de décodeur.

CALL BLOCK #J.12 | Adaptateur en boucle du gain de TCQ
CALL BLOCK #J.5 | Module d'adaptation.

```

#### J.4.4.2 Bloc J.21 – Module de transition de décodeur

**Entrée:** channel\_symbol

**Sortie:** reconstructed\_sample\_q2

**Fonction:** effectuer les opérations de décodage par échantillon.

NOTE – La prédiction est effectuée à partir de la valeur prédite du codeur pour le nœud 0.

Variables internes (définitions en C)

- short int next\_state\_label;
- short int level\_selector;
- short int subset;
- short int next\_state;
- short int codebook\_level;
- long int AA0;

TCQ\_predict\_sample\_q2[0] = 0 | Utiliser l'espace du nœud 0 du codeur.

```
FOR i = 0, 1, ..., (PREDICTOR_ORDER-1)
    TCQ_predict_sample_q2[0] -= TCQ_STTMP_q2 [(NFRSZ - 1) - i] * A[i+1]
```

```
FOR i = 0, 1, ..., (NFRSZ-1)
    TCQ_STTMP_q2[i] = TCQ_STTMP_q2[i + 1]
```

```
TCQ_predict_sample_q2[0] =
    rnd_int ( TCQ_predict_sample_q2[0] << 2)
```

| Diviser l'index en deux ensembles; établir l'étiquette de l'état suivant  
| et le sélecteur de niveau.

```
next_state_label = (channel_symbol >> BITS_PER_LEVEL)
level_selector = (channel_symbol & LEVEL_MASK)
next_state = TCQ_next_state [TCQ_decoder_state] [next_state_label]
subset = TCQ_trans_from_src_to_dst [TCQ_decoder_state] [next_state]
TCQ_decoder_state = next_state
```

codebook\_level = Yk[subset][level\_selector] | Obtenir le niveau du répertoire codé.

```
FOR i = 1, 2, ..., (RMS_BUF_LEN - 1)
    ET[i-1] = ET[i]
```

```
ET[RMS_BUF_LEN - 1] = codebook_level
```

```
AA0 = codebook_level * DLQ_GAIN
```

```
IF (0 <= DLQ_nls_p_cb_q_m_18)
    AA0 <<= DLQ_nls_p_cb_q_m_18
ELSE
    AA0 >>= abs (DLQ_nls_p_cb_q_m_18)
```

```
AA0 += (TCQ_predict_sample_q2[0] << 16L)
TCQ_STTMP_q2 [(NFRSZ-1)] = AA0 >> 16L
reconstructed_sample_q2 = AA0 >> 16L
```

#### J.4.5 Description détaillée des modules commutateurs de mode

Cette section donne une description détaillée des modules commutateurs de mode.

#### J.4.5.1 Bloc J.18 – Module de transition parole/données

**Entrée:** NLSSTATE, STATELPC, GAIN, NLSGAIN, IAQ\_for\_VBD, ATMP\_for\_VBD

**Sortie:** TCQ\_STTMP\_q2, DLQ\_GAIN, DLQ\_NLSGAIN, IAQ\_for\_VBD, ATMP\_for\_VBD

**Fonction:** réaliser la transition du mode parole au mode données.

```
| Transposer 20 éléments de STATELPC et NLSSTATE en format SBFL  
| au format Q2 de TCQ_STTMP_q2.  
| L'ordonnancement opposé de STATELPC, NLSSTATE et  
| TCQ_STTMP_q2 est à noter.
```

```
I = 0  
FOR J = 0, 1, 2, 3  
  FOR L = 0, 1, ..., 4  
    k = NLSSTATE [9] - 2  
    IF (k<0)  
      k = -k  
      TCQ_STTMP_q2[NFRSZ - 1 - I] = STATELPC [I] << k  
    ELSE  
      TCQ_STTMP_q2[NFRSZ - 1 - I] = STATELPC [I] >> k  
    I = I + 1
```

```
| VARIABLES DE GAIN.
```

```
speech_to_vbd_transition = 1
```

```
GAIN_state = after_limiter_98
```

```
dms = 0  
dml = 0  
ap_q11 = 0
```

```
DLQ_GAIN = GAIN  
DLQ_NLSGAIN = NLSGAIN
```

```
CALL BLOCK #J.15 | Inverse du GAIN()
```

```
| Utiliser les 10 paramètres de LPC sauvegardés précédemment (pendant  
| le dernier cycle d'adaptation du mode de parole).
```

```
FOR I = 1, 2, ..., PREDICTOR_ORDER  
  ATMP[i] = ATMP_for_VBD[i]
```

```
IAQ = IAQ_for_VBD  
CALL BLOCK #J.51 | bandexpand51_vbd
```

```
FOR I = 1, 2, ..., PREDICTOR_ORDER  
  A[i] = ATMP[i]
```

```
GC_FLAG=0;      | Compensation du gain  
G_CNT=0;        | Compensation du gain
```

#### J.4.5.2 Bloc J.19 – Module de transition données/parole

**Entrée:** voir la description ci-dessous

**Sortie:** voir la description ci-dessous

**Fonction:** réaliser la transition du mode données au mode parole.

## Variables internes (définitions en C)

- long int temp[NFRSZ]

| Réinitialiser les variables internes de pondération perceptive  
| (seulement dans le codeur).

```
AWP[0]=16384
FOR I=1, 2, ...,LPCW
    AWP[i]=0
AWZ[0]=16384
FOR I = 1, 2, ...,LPCW
    AWZ[i]=0
FOR I = 0, 1, 2, ..(NFRSZ-1)
    STMP[i]=0
FOR I = 0, 1,...,(N3weight-1)
    SBW[i]=0

FOR I = 0, 1, ...,LPCW
    REXPW[i]=0
NLSREXPW= 31
```

| Réinitialisation des variables internes du postfiltre  
| (seulement dans le décodeur).

```
ILLCONDP = 1
AP[0] = 16384
FOR I =1, 2, ...,10
    AP[i]=0
AZ[0] = 16384
FOR I =1, 2,...,10
    AZ[i]=0
FOR I = 0,1,...,59
    DEC[i]=0
FOR I = 0, 1,...,239
    D[i]=0
FOR I = 0, 1,...,9
    STLPCI[i]=0
FOR I = 0, 1, ...,9
    STPFIR[i]=STPFIIR[i]=0
FOR I =0, 1, 2, 3
    LPFFIR[i]=0
    LPFIIR[i]=0
FOR I = 0, 1, ..., 244
    SST[i]=0
SCALEFIL= 16384
B=0
GL=16384
GLB = 0
TILTZ=0
APF[0] = 16384
FOR I = 1, 2, ...,10
    APF[i]=0
PF_delay = 100
KP=KP1=50

FOR I =11, 12, ..., LPC
    A[i]=0
    ATMP[i]=0

FOR I = 0, 1, ..., (NFRSZ-1)
    temp[i] = TCQ_STTMP_q2[(NFRSZ-1) - i]
```

```

FOR I = 0, 1, 2, 3
  VSCALE( &temp[i * 5], 5, 5, 12,&STATELPC[i * 5], &NLSSTATE[9-i] )
  NLSSTATE[9-i] += 2 | Correction de Q2.

FOR I = NFRSZ, NFRSZ+1, ..., (LPC - 1)
  STATELPC [i] = 0
FOR I = 0, 1, ..., 5
  NLSSTATE [i] = 16

| Mise à jour de plusieurs blocs internes LD-CELP.
| Ces blocs sont habituellement mis à jour pendant la troisième phase
| d'adaptation
| ( ICOUNT == 3), mais ils sont nécessaires à partir de la phase 1.

CALL BLOCK #G.12 | Calcul du vecteur de réponse impulsionnelle.
CALL BLOCK #G.14 | Convolution du vecteur code de forme et
                  | calcul de l'énergie.

| VARIABLES DE GAIN.

CALL BLOCK #J.13 | vbd_log_calc_and_lim97()
vbd_to_speech_transition = 1

```

#### J.4.5.3 Bloc J.22 – Modifications de l'adaptateur en boucle du gain vectoriel de l'Annexe G/G.728

**Fonction:** les lignes suivantes remplacent les blocs G.93, G.94, G.96 et G.97.

NOTE – Pendant la transition données/parole, les valeurs de retard de l'indice des répertoires de gain et de forme ne sont pas disponibles. Par conséquent, GSTATE[0] est calculé pendant la transition.

```

IF ( vbd_to_speech_transition == 1)
  vbd_to_speech_transition = 0
ELSE
  | Fonctionnement par défaut de l'Annexe G/G.728.
  | Exécuter les blocs G.93, G.94, G.96 et G.97.

```

#### J.4.5.4 Bloc J.23 – Modifications de l'application du postfiltre de l'Annexe G/G.728

**Fonction:** contourner le postfiltre pendant les 500 premiers échantillons (62,5 ms) après la transition et utiliser le signal reconstitué comme signal de sortie du décodeur pour cette période. Au bout de 62,5 ms, lorsque le postfiltre est prêt à être utilisé, sa sortie devient la sortie du décodeur.

```

IF (PF_delay == 0)
  FOR I =0, 1, ..., (IDIM-1)
    ST[i]=SPF[i]<<1
ELSE
  PF_delay--
  FOR I = 0, 1, ..., (IDIM-1)
    ST[i]=ST[i]<<(16-NLSST+3)
    ST[i]=rnd_int(ST[i])

```

#### J.4.5.5 Bloc J.24 – Modifications devant être apportées à l'Annexe G/G.728 pour la sauvegarde des paramètres LPC provisoires

**Fonction:** sauvegarder les 10 paramètres LPC provisoires. Ces paramètres sont requis pour la transition parole/données.

NOTE – Les 10 paramètres LPC provisoires doivent être sauvegardés pendant l'exécution du bloc G.23. L'exécution du bloc de Durbin doit être interrompue (tel que décrit dans l'Annexe G/G.728 pour les coefficients du postfiltre, APF). Les coefficients doivent être sauvegardés et l'exécution du bloc de Durbin doit être reprise.

```

IF (DurbinFaultFlag == 0)
  FOR I =1, 2,..., 10
    ATMP_for_VBD[I]=ATMP[I];
  IAQ_for_VBD = IAQ;

```

#### J.4.6 Tables de transition en treillis

La présente section donne les tables de transition en treillis. Ces tables décrivent une forme de réalisation du codeur à convolution libre à réaction minimale.

##### J.4.6.1 TCQ\_prev\_state

Cette table donne l'état de treillis (nœud) précédent (source), par branche de treillis, pour chaque état de treillis (nœud), comme le montrent les Figures J.2 et J.3.

Définition en C: int TCQ\_prev\_state[N\_STATES][N\_BRANCHES]

TCQ_prev_state		
Etat de treillis (nœud)	prev_state	
	b[0]	b[1]
s[0]	0	2
s[1]	2	0
s[2]	1	3
s[3]	3	1

##### J.4.6.2 TCQ\_next\_state

Cette table donne l'état de treillis (nœud) suivant (cible) par branche de treillis, pour chaque état de treillis (nœud), comme le montrent les Figures J.2 et J.3.

Définition en C: int TCQ\_next\_state[N\_STATES][N\_BRANCHES];

TCQ_next_state		
Etat de treillis (nœud)	next_state	
	b[0]	b[1]
s[0]	0	1
s[1]	2	3
s[2]	0	1
s[3]	2	3

##### J.4.6.3 TCQ\_trans\_from\_src\_to\_dst

Cette table donne le sous-ensemble du répertoire codé associé à une transition, comme le montrent les Figures J.2 et J.3.

Définition en C: int TCQ\_trans\_from\_src\_to\_dst [N\_STATES][N\_STATES];

TCQ_trans_from_src_to_dst				
Etat de treillis (nœud)	étiquette de transition			
	s[0]	s[1]	s[2]	s[3]
s[0]	0	2	X	X
s[1]	X	X	1	3
s[2]	2	0	X	X
s[3]	X	X	3	1

#### J.4.6.4 TCQ\_index\_from\_src\_to\_dst

Cette table donne le bit d'indice de canal qui identifie la transition, comme le montrent les Figures J.2 et J.3, au codeur.

Définition en C: int TCQ\_index\_from\_src\_to\_dst [N\_STATES][N\_STATES];

TCQ_index_from_src_to_dst				
Etat de treillis (nœud)	bit d'indice de canal			
	s[0]	s[1]	s[2]	s[3]
s[0]	0	0x10	X	X
s[1]	X	X	0	0x10
s[2]	0	0x10	X	X
s[3]	X	X	0	0x10

#### J.4.6.5 Limites du quantificateur Xk

Cette table donne les limites des intervalles du super-répertoire en format Q11. La première colonne comporte les indices de cellule et les colonnes suivantes donnent les niveaux de chacun des sous-ensembles.

Définition en C: int Xk [N\_STATES][Q\_CELLS];

Xk				
Index	Limites du répertoire codé			
	s[0]	s[1]	s[2]	s[3]
0	-9 547	-8 509	-7 779	-7 191
1	-6 690	-6 246	-5 845	-5 477
2	-5 134	-4 812	-4 507	-4 217
3	-3 939	-3 672	-3 414	-3 164
4	-2 921	-2 684	-2 453	-2 226
5	-2 003	-1 783	-1 567	-1 353
6	-1 141	-931	-723	-515
7	-309	-103	103	309
8	515	723	931	1 141
9	1 353	1 567	1 783	2 003

<b>Xk</b>				
<b>Index</b>	<b>Limites du répertoire codé</b>			
10	2 226	2 453	2 684	2 921
11	3 164	3 414	3 672	3 939
12	4 217	4 507	4 812	5 134
13	5 477	5 845	6 246	6 690
14	7 191	7 779	8 509	9 547
15	32 767	32 767	32 767	32 767

#### J.4.6.6 Niveaux du quantificateur Yk

Cette table donne les niveaux de quantification du super-répertoire en format Q11. La première colonne comporte les indices de cellule et les colonnes suivantes donnent les niveaux de chacun des sous-ensembles.

Définition en C: `int Yk [N_STATES][Q_CELLS];`

<b>Yk</b>				
<b>Indice</b>	<b>Niveau du répertoire codé</b>			
	s[0]	s[1]	s[2]	s[3]
0	-11 502	-9 955	-8 962	-8 209
1	-7 592	-7 062	-6 595	-6 174
2	-5 788	-5 430	-5 095	-4 780
3	-4 480	-4 194	-3 919	-3 655
4	-3 399	-3 151	-2 910	-2 674
5	-2 444	-2 218	-1 996	-1 777
6	-1 562	-1 348	-1 138	-928
7	-721	-514	-308	-103
8	103	308	514	721
9	928	1 138	1 348	1 562
10	1 777	1 996	2 218	2 444
11	2 674	2 910	3 151	3 399
12	3 655	3 919	4 194	4 480
13	4 780	5 095	5 430	5 788
14	6 174	6 595	7 062	7 592
15	8 209	8 962	9 955	11 502

#### J.4.7 Coefficients du polynôme de calcul logarithmique

Cette table comporte les coefficients du polynôme utilisé dans le calculateur logarithmique.

Définition en C: `int log_poly[LOG_POL_ORDER];`

<b>log_poly</b>		
<b>Indice</b>	<b>Présentation en virgule flottante</b>	<b>Présentation en virgule fixe Q15</b>
0	0,8678284	28 437
1	-0,4255677	-13 945
2	0,2481384	8 131
3	-0,1155701	-3 787
4	0,0272522	893

#### **J.4.8 Coefficients d'élargissement de la largeur de bande pour le mode de données**

Cette table donne les coefficients d'élargissement de la largeur de bande pour le mode de données.

NOTE – Il n'y a que 10 éléments non nuls.

Définition en C: int FACV\_vbd[LPC];

**Tableau J.1/G.728 – Coefficients d'élargissement de la largeur de bande**

<b>FACV_vbd</b>	
<b>Indice</b>	<b>Coefficient</b>
1	15 360
2	14 400
3	13 500
4	12 656
5	11 865
6	11 124
7	10 428
8	9 777
9	9 166
10	8 593
11-50	0

#### **J.4.9 Blocs internes**

Le Tableau J.2 contient une liste des blocs internes.

**Tableau J.2/G.728 – Blocs internes**

<b>Id. de bloc</b>	<b>Nom de bloc</b>
Bloc J.1	Sélection de mode et de commutateur de mode
Bloc J.2	Mode de recherche du mode de données dans la bande vocale
Bloc J.3	Recherche en treillis par module vectoriel
Bloc J.4	Sélection du meilleur élément subsistant
Bloc J.5	Module d'adaptation
Bloc J.6	Transition TCQ
Bloc J.7	Sélection de l'état du prédicteur
Bloc J.8	Calcul des résidus
Bloc J.9	Trouver un "nouvel" élément subsistant
Bloc J.10	Calcul du signal reconstitué
Bloc J.11	Résidu de quantification TCQ
Bloc J.12	Adaptateur en boucle du gain de TCQ
Bloc J.13	vbd_log_calc_and_lim97
Bloc J.14	Pondération du gain logarithmique
Bloc J.15	Inverse du gain
Bloc J.16	Calculer $10\log_{10}$ (Calculateur logarithmique)
Bloc J.17	Module d'initialisation de recherche suivante
Bloc J.18	Module de transition parole/données
Bloc J.19	Module de transition données/parole
Bloc J.20	Module décodeur
Bloc J.21	Module de transition de décodeur
Bloc J.22	Modifications de l'adaptateur en boucle du gain vectoriel de l'Annexe G/G.728
Bloc J.23	Modifications de l'application du postfiltre de l'Annexe G/G.728
Bloc J.24	Modifications devant être apportées à l'Annexe G/G.728 pour la sauvegarde des paramètres LPC provisoires
Bloc J.25	Calculateur de moyenne du gain pour le module de compensation du gain
Bloc J.26	Classificateur de signal pour le module de compensation du gain
Bloc J.27	Détection d'impulsion pour le module de compensation du gain
Bloc J.28	Bloc de décision pour le module de compensation du gain
Bloc J.29	Compensation du gain pour le module de compensation du gain
Bloc J.51	Module d'élargissement de largeur de bande

**J.4.10 Variables et constantes de traitement interne**

Les Tableaux J.3 et J.4 comportent les listes et les descriptions des variables et constantes internes.

**Tableau J.3/G.728 – Variables de traitement interne**

Nom	Plage d'indices de table	Format en virgule fixe	Description
after_limiter_98			Entrée Q9 du bloc additionneur G.99
al_q11	1	Q11	Facteur de verrouillage de quantificateur
ap_q11	1	Q11	Facteur de verrouillage de quantificateur
ATMP_for_VBD	1..11	Q13/Q14/Q15	Tampon temporaire pour paramètres LPC
block_depth	1	Q0	Points à l'étape de treillis courante (temps n)
ch_index	0..7	Q0	Mémoire temporaire de l'indice sélectionné
distortion_metric	[0..(N_STATES-1)]	Q20 32 bits	Valeurs de distorsion accumulées, par état de treillis (nœud)
DLQ_GAIN	1	SFL	Gain d'excitation linéaire, équivalent de GAIN dans l'Annexe G
DLQ_inv_GAIN	1	SBFL	GAIN inversé pour données du mode VBD
DLQ_inv_nls_m	1	SBFL	NLS pour l'inverse du gain du mode VBD
DLQ_nls_p_cb_q_m_18	1	Q0	NLS pour gain linéaire (décalage de correction de format + Q), équivalent de GAIN dans l'Annexe G
dml	1	Q11	Energie à long terme des résidus quantifiés
dms	1	Q11	Energie à court terme des résidus quantifiés
ET	[0..(RMS_BUF_LEN-1)]		Mémoire des résidus quantifiés
FACV_vbd	---	---	Voir FACV dans l'Annexe G
GAIN_state	1	Q9	Mémoire de filtre de pondération
IAQ_for_VBD	1	Q0	Fanion de précision de récurrence de Durbin ATMP
next_stage	1	Q0	Points à l'étape suivante dans le treillis (temps n+1)
prev_node	[0..(BLOCK_LEN-1)]*[0..(N_STATES-1)]	Q0	Mémoire de treillis d'états précédents (nœuds)
Q_resid	[0..(N_STATES-1)]	Q11	Mémoire temporaire des résidus quantifiés
qerror	1	Q11	Mémoire temporaire des résidus quantifiés
RMS_Q11	1	Q11	Valeur quadratique moyenne de ET
sample	1	Q2	Echantillon d'entrée, un seul élément de S
speech_to_vbd_transition	1	Q0	Fanion de transition parole à données dans la bande vocale

**Tableau J.3/G.728 – Variables de traitement interne (*fin*)**

Nom	Plage d'indices de table	Format en virgule fixe	Description
survivor_node	1	Q0	Indice de nœud subsistant
TCQ_channel_indices	[0..(BLOCK_LEN-1)] * [0..(N_STATES-1)]	Q0	Mémoire de treillis d'indice de canal
TCQ_channel_symbols	[0..(IDIM-1)]	Q0	Mémoire de signal comprimé
TCQ_common_part_pred_q2	1	Q2	Partie commune de valeur prédite
TCQ_ET	[0..(BLOCK_LEN-1)] [0..(N_STATES-1)]		Mémoire de treillis des résidus
TCQ_predict_sample_q2	[0..(N_STATES-1)]	Q2	Mémoire de valeur prédite pour chaque état de treillis (nœud)
TCQ_predictor_state_q2	[0..(N_STATES-1)] [0..(PREDICTOR_ORDER-1)]	Q2	Mémoire de treillis d'états de prédicteur
TCQ_reconstructed_q2	[0..(BLOCK_LEN-1)] [0..(N_STATES-1)]	Q2	Mémoire de treillis de niveaux reconstitués
TCQ_reconstructed_q2	[0..(BLOCK_LEN-1)]*[0..(N_STATES-1)]	Q2	Mémoire de treillis d'échantillons reconstitués
TCQ_resid	[0..(N_STATES-1)]	Q11	Mémoire de résidus de nœuds (état de treillis)
TCQ_STTMP_q2	[0..NFRSZ]	Q2	Tampon pour fenêtre hybride de filtre de synthèse
temp_metric	[0..7]	Q20 32 bits	Mémoire temporaire de valeurs de distorsion
vbd_to_speech_transition	1	Q0	Fanion de transition VBD à parole
Xk	[0..(N_STATES-1)] [0..(Q_CELLS-1)]	Q11	Limites d'intervalles de quantification
Yk	[0..(N_STATES-1)] [0..(Q_CELLS-1)]	Q11	Niveaux de quantification
GDIFF	1	Q9 32 bits	Différence entre le gain courant et sa moyenne
G_AVE	1	Q9 32 bits	Moyenne du gain
GC_ATMP_SUM	1		Somme des paramètres absolus de LPC du bloc G.50
GC_ATMP1	1		Second paramètre de LPC du bloc G.50
GC_NLS_LIMIT	1	Q0	Seuil NLS de gain utilisé dans le bloc de compensation du gain
GC_COMPENSATION	1	Q0	Facteur de compensation

**Tableau J.4/G.728 – Constantes de traitement interne**

Nom	Valeur	Symbole	Description
LOG_POL_ORDER	5		Ordre du polynôme d'approximation logarithmique
BITS_PER_LEVEL	4		Nombre de bits, identifiant les niveaux de quantification
LEVEL_MASK	0x0F		Masque de bits de niveau
PREDICTOR_ORDER	10		Ordre du prédicteur en mode de données
CODEBOOK_Q	1 Q11		Présentation Q du répertoire codé
Log_2	24 660		$10 \cdot \log_{10}(2)$ en Q13
RMS_BUF_LEN	8		Longueur du calcul de la valeur quadratique moyenne
BLOCK_LEN	5	$K_d$	Longueur de bloc de treillis
MAX_NUMBER	0x7fffffff		Nombre positif maximal
N_BRANCHES	2		Nombre de branches en provenance ou à destination de chaque état de treillis
N_STATES	4		Nombre d'états de treillis
Q_CELLS	16		Nombre de niveaux de quantification dans chaque sous-ensemble
TCQ_Kd	5	$K_d$	Longueur de retard de treillis
TCQ_Kr	5	$K_r$	Rôle de libération de treillis
TCQ_DEPTH	BLOCK_LEN-1		Longueur de bloc de treillis -1
G_CONST	5		Utilisé pour le calcul de $G_{ave}$ dans le module de compensation du gain
ATMP_CONST	3		Seuil de détection de signal à faible largeur de bande dans le classificateur de signal
G_TRS	1 800		Seuil de compensation du gain $G_{diff}$
GC-NLS_LIMIT_INIT	7		Limiteur de compensation du gain
GC_COMPENSATION_INIT	3		Valeur soustraite du NLS du gain lorsqu'une compensation de gain a lieu
GC_CNT_INIT	11		Période pendant laquelle la compensation de gain est active

## J.4.11 Valeurs initiales

**Tableau J.5/G.728 – Valeurs initiales**

NOM	Valeur initiale
ATMP_for_VBD	16 384, 0, ..., 0
IAQ_for_VBD	14
vbd_to_speech_transition	0
block_depth	0
next_stage	1
TCQ_decoder_state	0
ET	0..0
GAIN	512
NLSGAIN	0
DLQ_GAIN	16 384
DLQ_NLSGAIN	14
DLQ_nls_p_cb_q_m_18	-7
DLQ_inv_GAIN	16 384
DLQ_inv_nls_m	-4
GAIN_state_fx	0L
after_limiter_98	-16 384
TCQ_STTMP_q2	0..0
TCQ_reconstructed_q2	0, ..., 0
distortion_metric	0, (MAX_NUMBER>>2), ..., (MAX_NUMBER>>2)
dms	0
dml	0
RMS_Q11	0
ap_q11	0
al_q11	0
GAVE	0
UNSTEADY	1
G_CNT	0
GC_SC_FLAG	0
GC_ID_FLAG	0
GC_CNT	0
GC_FLAG	0

## **J.5 Bibliographie**

- (1) GERSHO (A.) et GRAY (R.M.), Vector quantization and signal compression.
- (2) MARCELLIN (M.W.) et FISCHER (T.R.), Trellis Coded Quantization of Memoryless and Gauss-Markov Sources, IEEE transactions on communications, Vol. 38, N° 1, janvier 1990.
- (3) TRUSHKIN (A.V.), On the design of an optimal Quantizer, IEEE transactions on information theory, Vol. 39, N° 4, juillet 1993.
- (4) JAYANT (N.S.) et NOLL (P.), Digital coding of waveforms.
- (5) PETR (D.W.), 32 Kbps ADPCM-DLQ coding for network applications, IEEE 1982.
- (6) MOC/Israël, Test results for non-voiced performance assessment of LD-CELP algorithm operating at 40 kbit/s mainly for DCME applications; CE 16, COM-D.278, Santiago, Chili, 18-27 mai 1999.



## SERIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série B	Moyens d'expression: définitions, symboles, classification
Série C	Statistiques générales des télécommunications
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
<b>Série G</b>	<b>Systèmes et supports de transmission, systèmes et réseaux numériques</b>
Série H	Systèmes audiovisuels et multimédias
Série I	Réseau numérique à intégration de services
Série J	Transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	RGT et maintenance des réseaux: systèmes de transmission, de télégraphie, de télécopie, circuits téléphoniques et circuits loués internationaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	Réseaux pour données et communication entre systèmes ouverts
Série Y	Infrastructure mondiale de l'information et protocole Internet
Série Z	Langages et aspects informatiques généraux des systèmes de télécommunication