



UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

**CCITT**

COMITÉ CONSULTIVO  
INTERNACIONAL  
TELEGRÁFICO Y TELEFÓNICO

**G.728**

(09/92)

**ASPECTOS GENERALES DE LOS SISTEMAS  
DE TRANSMISIÓN DIGITAL;  
EQUIPOS TERMINALES**

---

**CODIFICACIÓN DE SEÑALES VOCALES A  
16 kbit/s UTILIZANDO PREDICCIÓN LINEAL  
CON EXCITACIÓN POR CÓDIGO  
DE BAJO RETARDO**

**Recomendación G.728**

---



Ginebra, 1992

## PREFACIO

El CCITT (Comité Consultivo Internacional Telegráfico y Telefónico) es un órgano permanente de la Unión Internacional de Telecomunicaciones (UIT). Este órgano estudia los aspectos técnicos, de explotación y tarifarios y publica Recomendaciones sobre los mismos, con miras a la normalización de las telecomunicaciones en el plano mundial.

La Asamblea Plenaria del CCITT, que se celebra cada cuatro años, establece los temas que han de estudiarse y aprueba las Recomendaciones preparadas por sus Comisiones de Estudio. La aprobación de Recomendaciones por los miembros del CCITT entre las Asambleas Plenarias de éste es el objeto del procedimiento establecido en la Resolución N.º 2 del CCITT (Melbourne, 1988).

La Recomendación G.728 ha sido preparada por la Comisión de Estudio XV y fue aprobada por el procedimiento de la Resolución N.º 2 el 1 de septiembre de 1992.

---

## NOTAS DEL CCITT

- 1) En esta Recomendación, la expresión «Administración» se utiliza para designar, en forma abreviada, tanto una Administración de telecomunicaciones como una empresa privada de explotación reconocida de telecomunicaciones.
- 2) En el anexo F figura la lista de abreviaturas utilizadas en la presente Recomendación.

© UIT 1992

Es propiedad. Ninguna parte de esta publicación puede reproducirse o utilizarse, de ninguna forma o por ningún medio, sea éste electrónico o mecánico, de fotocopia o de microfilm, sin previa autorización escrita por parte de la UIT.

## Recomendación G.728

# CODIFICACIÓN DE SEÑALES VOCALES A 16 kbit/s UTILIZANDO PREDICCIÓN LINEAL CON EXCITACIÓN POR CÓDIGO DE BAJO RETARDO

(1992)

## 1 Introducción

Esta Recomendación contiene la descripción de un algoritmo para la codificación de señales vocales a 16 kbit/s utilizando predicción lineal con excitación por código de bajo retardo (LD-CELP, *low-delay code excited linear prediction*). Esta Recomendación está organizada como se describe a continuación.

En el § 2 se da una breve descripción del algoritmo LD-CELP. En los § 3 y 4 se tratan, respectivamente, los principios del codificador LD-CELP y del decodificador LD-CELP. En el § 5 se definen los detalles de cálculo que pertenecen a cada bloque funcional del algoritmo. Los anexos A, B, C y D contienen tablas de constantes utilizadas por el algoritmo LD-CELP. En el anexo E se describen las secuencias de adaptación y utilización de las variables. Finalmente, en el apéndice 1 se da información sobre los procedimientos aplicables a la verificación de la realización del algoritmo.

Está en estudio la futura incorporación de tres apéndices adicionales (que han de publicarse por separado) que tratan los aspectos de red del LD-CELP, la descripción de la realización de punto fijo del LD-CELP y los procedimientos de verificación del LD-CELP de punto fijo.

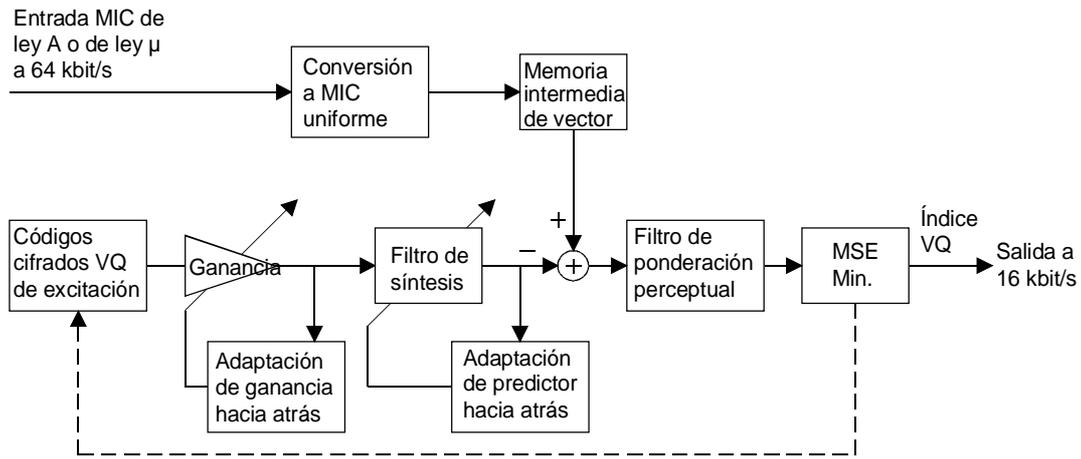
## 2 Breve descripción del LD-CELP

El algoritmo LD-CELP consta de un codificador y un decodificador, descritos respectivamente en los § 2.1 y 2.2, e ilustrados en la figura 1/G.728.

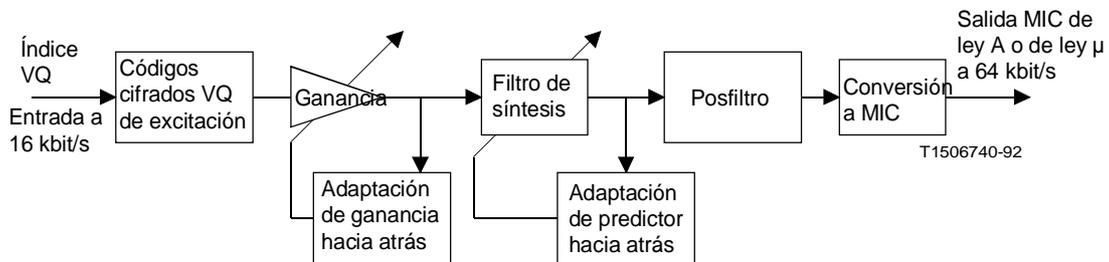
El LD-CELP conserva la esencia de la técnica de predicción lineal con excitación por código (CELP, *code excited linear prediction*), que aplica un método de análisis y síntesis a la búsqueda de código cifrado. Sin embargo, el LD-CELP utiliza adaptación hacia atrás de los predictores y la ganancia para lograr un retardo algorítmico de 0,625 ms. Se transmite únicamente el índice del código cifrado de excitación. Los coeficientes del predictor se actualizan mediante análisis de codificación predictiva lineal (LPC, *linear prediction coding*) de las señales vocales previamente cuantificadas. La ganancia de excitación es actualizada utilizando la información de ganancia incluida en la excitación previamente cuantificada. El tamaño del bloque correspondiente al vector de excitación y la adaptación de ganancia es de sólo cinco muestras. Se actualiza un filtro de ponderación perceptual mediante análisis LPC de las señales vocales no cuantificadas.

### 2.1 Codificador LD-CELP

Después de pasar de una codificación MIC de ley A o  $\mu$  a una codificación MIC uniforme, la señal de entrada es subdividida en bloques de cinco muestras consecutivas. Para cada bloque de entrada, el codificador pasa cada uno de 1024 vectores de código cifrado posibles (almacenados en una tabla de códigos cifrados de excitación) a través de una unidad de escalamiento de ganancia y un filtro de síntesis. De entre los 1024 vectores de señal cuantificada resultantes, el codificador identifica el que minimiza la medida del error cuadrático medio ponderado en frecuencia con relación al vector de la señal de entrada. Se transmite al decodificador índice del código cifrado de diez bits del correspondiente mejor vector de código cifrado (o «vector de código») que da lugar a dicho mejor vector de señal cuantificada. El mejor vector de código pasa entonces a través de la unidad de escalamiento de ganancia y del filtro de síntesis a fin de establecer la memoria de filtro correcta como preparación para la codificación del siguiente vector de señal. Los coeficientes del filtro de síntesis y la ganancia son actualizados periódicamente de manera adaptativa hacia atrás, basándose en la señal previamente cuantificada y en la excitación con escalamiento de ganancia.



a) Codificador LD-CELP



b) Decodificador LD-CELP

FIGURA 1/G.728

**Diagrama de bloques simplificado del codificador LD-CELP**

2.2 *Decodificador LD-CELP*

La operación de decodificación se realiza también bloque a bloque. Después de recibir cada índice de diez bits, el decodificador consulta una tabla para extraer el vector de código correspondiente de la tabla de códigos cifrados de excitación. El vector de código extraído pasa entonces a través de una unidad de escalamiento de ganancia y un filtro de síntesis a fin de producir el vector de señal decodificada actual. Los coeficientes del filtro de síntesis y la ganancia son entonces actualizados de la misma manera que en el codificador. El vector de la señal decodificada pasa entonces a través de un posfiltro adaptativo, a fin de mejorar la calidad perceptual. Los coeficientes del posfiltro se actualizan periódicamente utilizando la información disponible en el decodificador. Las cinco muestras del vector de la señal del posfiltro se convierten entonces en cinco muestras de salida MIC de ley A o  $\mu$ .

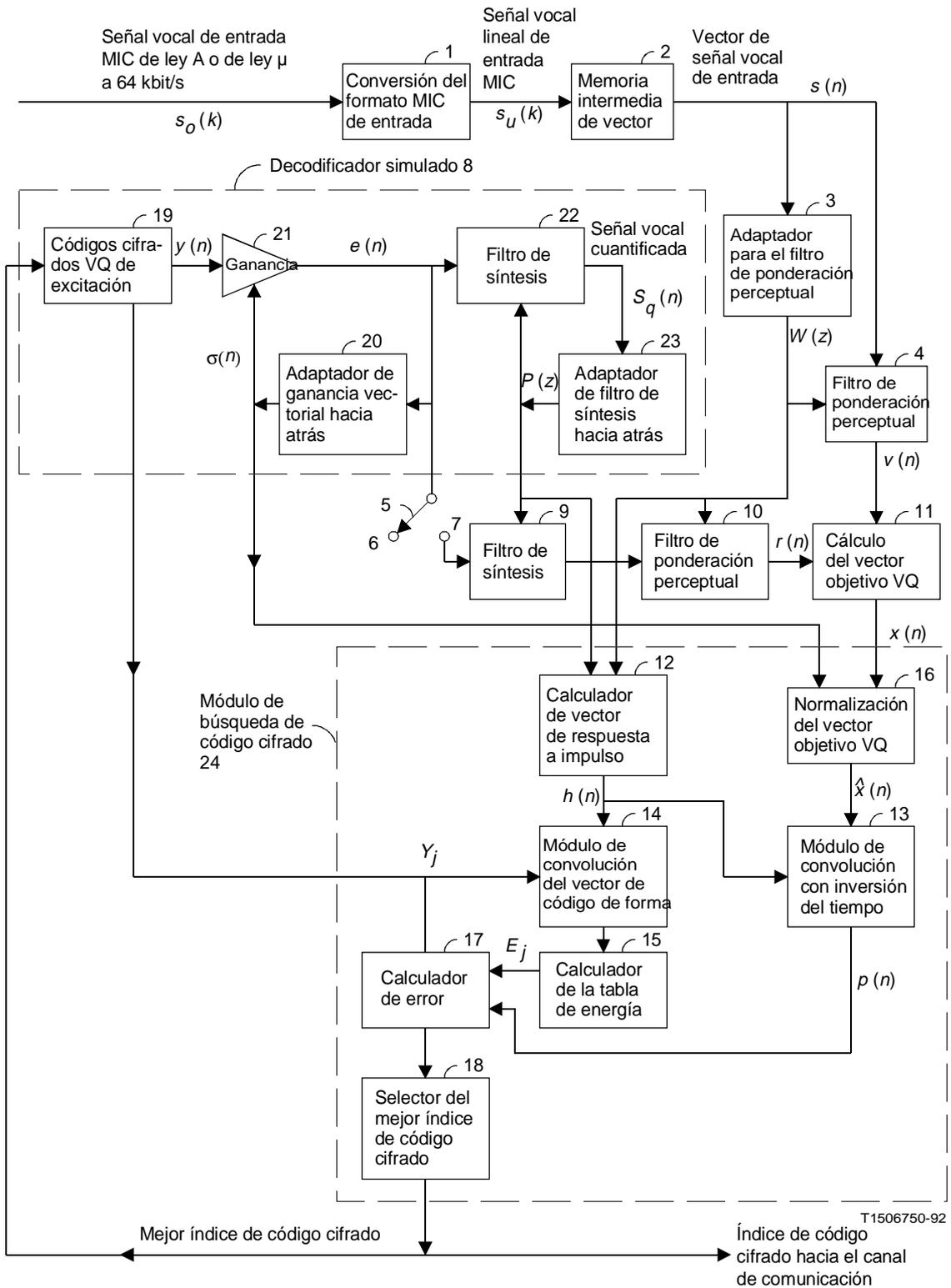


FIGURA 2/G.728

Diagrama de bloques del codificador LD-CELP

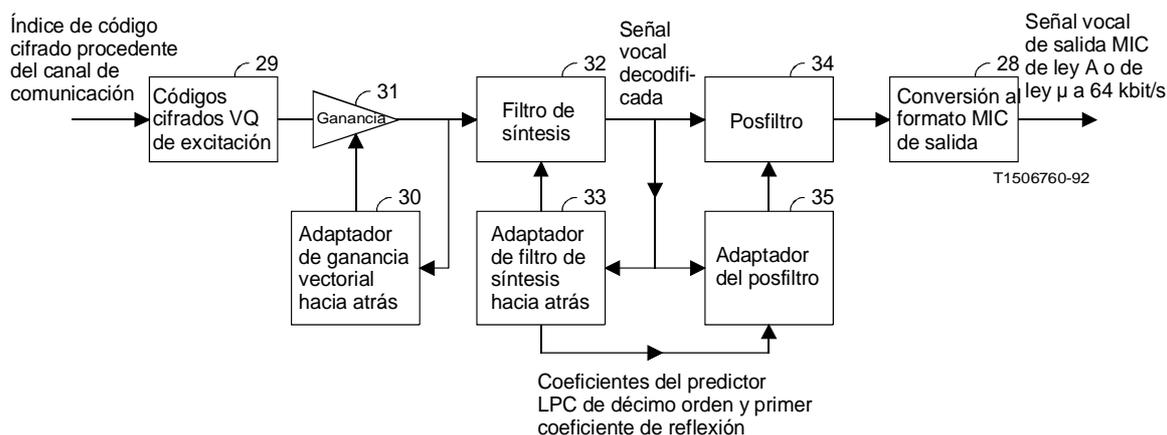


FIGURA 3/G.728

Diagrama de bloques del decodificador LD-CELP

### 3 Principios de funcionamiento del codificador LD-CELP

La figura 2/G.728 es un diagrama de bloques detallado del codificador LD-CELP. El codificador de la figura 2/G.728 es matemáticamente equivalente al codificador mostrado anteriormente en la figura 1/G.728, pero su realización es más eficaz desde el punto de vista de los cálculos.

En la descripción siguiente,

- para cada variable descrita,  $k$  es el índice de muestreo, y las muestras se toman a intervalos de 125  $\mu$ s;
- un grupo de cinco muestras consecutivas de una señal dada recibe el nombre de *vector* de esa señal. Por ejemplo, cinco muestras consecutivas de señal vocal forman un vector de señal vocal, cinco muestras de excitación forman un vector de excitación, etc.;
- $n$  designa el índice de vector, que es diferente del índice de muestra  $k$ ;
- cuatro vectores consecutivos constituyen un *ciclo de adaptación*. En un punto posterior designamos también los ciclos de adaptación como *tramas*. Los dos términos se emplean indistintamente.

El índice de código cifrado de cuantificación de vector (VQ, *vector quantization*) de excitación es la única información transmitida explícitamente del codificador al decodificador. Otros tres tipos de parámetros serán actualizados periódicamente: la ganancia de excitación, los coeficientes del filtro de síntesis y los coeficientes del filtro de ponderación perceptual. Estos parámetros se obtienen de manera adaptativa hacia atrás a partir de las señales que aparecen antes del vector de señal actual. La ganancia de excitación es actualizada una vez por vector, mientras que los coeficientes del filtro de síntesis y los coeficientes del filtro de ponderación perceptual son actualizados una vez cada cuatro vectores (es decir, un periodo de actualización de 20 muestras, o 2,5 ms). Obsérvese que, si bien la secuencia de procesamiento del algoritmo tiene un ciclo de adaptación de cuatro vectores (20 muestras), el tamaño de la memoria tampón básica es de sólo un vector (cinco muestras). El pequeño tamaño de la memoria intermedia permite obtener un retardo en un solo sentido inferior a 2 ms.

A continuación se da una descripción de cada bloque del codificador. Dado que el codificador LD-CELP se emplea ante todo para la codificación de señales vocales, en adelante supondremos, para facilitar la descripción, que la señal de entrada es vocal, si bien en la práctica puede haber también señales no vocales.

### 3.1 *Conversión del formato MIC de entrada*

Este bloque convierte la señal MIC de entrada de ley A o  $\mu$   $s_o(k)$  en una señal MIC uniforme  $s_u(k)$ .

#### 3.1.1 *Niveles internos de la MIC lineal*

Al pasar de la MIC de ley A o  $\mu$  a la MIC lineal, son posibles diferentes representaciones internas, según el dispositivo. Por ejemplo, las tablas normalizadas definen una gama lineal de  $-4015,5$  a  $+4015,5$  para la MIC de ley  $\mu$ . La gama correspondiente para la MIC de ley A es  $-2016$  a  $+2016$ . Ambas tablas incluyen algunos valores de salida que tienen una parte fraccionaria igual a  $0,5$ . Estas partes fraccionarias no pueden representarse en un dispositivo de enteros, a menos que toda la tabla se multiplique por  $2$  para convertir todos los valores en enteros. Esto es realmente lo que suele hacerse en las microplaquetas (chips) de procesamiento de señal digital (DSP, *digital signal processing*) de punto fijo. Por otra parte, las microplaquetas DSP de punto flotante pueden representar los mismos valores que figuran en las tablas. En esta Recomendación se supone que la señal de entrada tiene una gama máxima de  $-4095$  a  $+4095$ , que comprende ambos casos de ley  $\mu$  y ley A. En el caso de ley A, esto implica que cuando la conversión lineal da lugar a una gama de  $-2016$  a  $+2016$ , estos valores deben multiplicarse por  $2$  antes de proseguir con la codificación de la señal. En el caso de una entrada de ley  $\mu$  a un procesador de punto fijo en el que la gama de entrada es transformada en  $-8031$  a  $+8031$ , implica que los valores deben dividirse por  $2$  antes de empezar el proceso de codificación. Otra posibilidad sería tratar estos valores como si estuvieran en formato Q1, lo que significa que hay un bit a la derecha del punto (o coma) decimal. Todos los cálculos en que intervengan los datos deberían entonces tener en cuenta este bit.

En el caso de señales de entrada MIC lineal de  $16$  bits que tengan la gama dinámica total de  $-32\ 768$  a  $+32\ 767$ , deberá considerarse que los valores de entrada están en formato Q3, lo cual significa que los valores de entrada deben dividirse por  $8$ . A la salida del decodificador se efectuaría una multiplicación por  $8$  para reconstituir estas señales.

### 3.2 *Memoria intermedia de vector*

Este bloque almacena cinco muestras vocales consecutivas  $s_u(5n)$ ,  $s_u(5n + 1)$ , ...,  $s_u(5n + 4)$  para formar un vector vocal de cinco dimensiones  $s(n) = [s_u(5n), s_u(5n + 1), \dots, s_u(5n + 4)]$ .

### 3.3 *Adaptador para el filtro de ponderación perceptual*

En la figura 4/G.728 se ilustra el funcionamiento detallado del adaptador del filtro de ponderación perceptual (bloque 3 de la figura 2/G.728). Este adaptador calcula los coeficientes del filtro de ponderación perceptual una vez cada cuatro vectores vocales, basándose en el análisis de predicción lineal (denominado frecuentemente análisis LPC) de las señales vocales no cuantificadas. Las actualizaciones de los coeficientes se producen en el tercer vector vocal de cada ciclo de adaptación de cuatro vectores. Los coeficientes permanecen constantes entre dos actualizaciones sucesivas.

Véase la figura 4a)/G.728. El cálculo se lleva a cabo como sigue. Se hace pasar primero, el vector de señal vocal de entrada (no cuantificado) a través de un módulo de ventanización híbrida (bloque 36), que coloca una ventana sobre los vectores de señal vocal anteriores y calcula a modo de salida los  $11$  primeros coeficientes de autocorrelación de la señal vocal ventanizada. El módulo de recursión (repetición) de Levinson-Durbin (bloque 37) convierte entonces estos coeficientes de autocorrelación en coeficientes de predicción. Basándose en estos coeficientes de predicción, el calculador de los coeficientes del filtro de ponderación (bloque 38) calcula los coeficientes deseados del filtro de ponderación. Estos tres bloques se tratan más detalladamente a continuación.

Describamos primero los principios de funcionamiento de la ventanización híbrida. Dado que esta técnica se utilizará en tres tipos diferentes de análisis LPC, haremos primero una descripción más general de la técnica y examinaremos después los diferentes casos. Supongamos que el análisis LPC ha de realizarse una vez cada  $L$  muestras de señal, y, para más generalidad, que las muestras de señal correspondientes al ciclo de adaptación LD-CELP actual son  $s_u(m)$ ,  $s_u(m + 1)$ ,  $s_u(m + 2)$ , ...,  $s_u(m + L - 1)$ . Entonces, para el análisis LPC adaptativo hacia atrás, la ventana híbrida se aplica a todas las muestras de señal anteriores cuyo índice de muestra es inferior a  $m$  (como se muestra en la figura 4b)/G.728). Supongamos que hay  $N$  muestras no recursivas en la función ventana híbrida. Entonces, las muestras de señal  $s_u(m - 1)$ ,  $s_u(m - 2)$ , ...,  $s_u(m - N)$  son todas ponderadas por la parte no recursiva de la ventana. Todas las muestras que están a la izquierda de  $s_u(m - N - 1)$  (incluida ésta) son ponderadas por la parte recursiva de la ventana, que tiene los valores  $b$ ,  $b\alpha$ ,  $b\alpha^2$ , ..., donde  $0 < b < 1$  y  $0 < \alpha < 1$ .

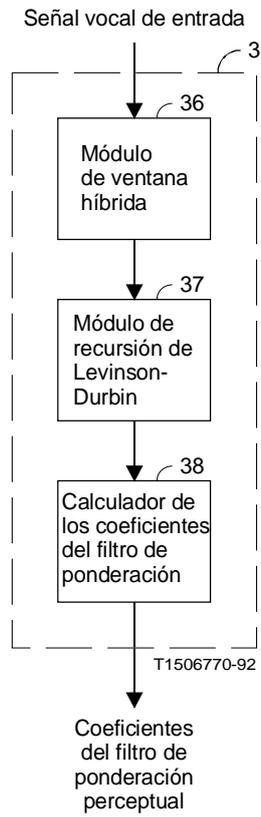


FIGURA 4a)/G.728

**Adaptador del filtro de ponderación perceptual**

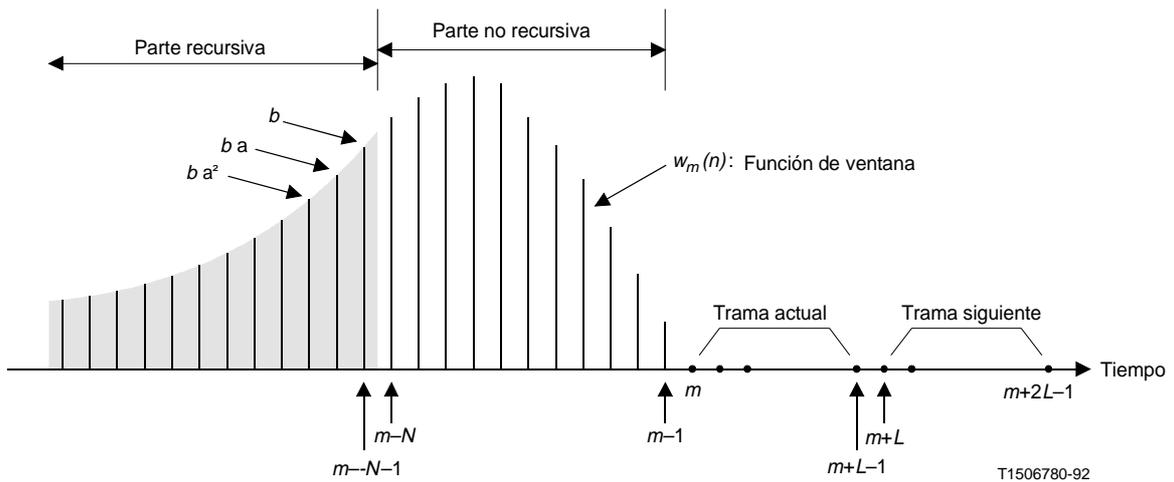


FIGURA 4b)/G.728

**Ilustración de una ventana híbrida**

En el instante  $m$ , la función de ventana híbrida  $w_m(k)$  se define como

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

y la señal ponderada por la ventana es

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

En el anexo A se especifican las muestras de la parte no recursiva  $g_m(k)$  y la sección inicial de la parte recursiva  $f_m(k)$  para diferentes ventanas híbridas. Para un análisis LPC de orden  $M$ , tenemos que calcular  $M+1$  coeficientes de autocorrelación  $R_m(i)$  para  $i = 0, 1, 2, \dots, M$ . El  $i$ -ésimo coeficiente de autocorrelación para el ciclo de adaptación actual puede expresarse como

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

donde

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

En el segundo miembro de la ecuación (3-1c), el primer término  $r_m(i)$  es la «componente recursiva» de  $R_m(i)$ , mientras que el segundo es la componente «no recursiva». La suma finita de la componente no recursiva se calcula para cada ciclo de adaptación. Por otra parte, la componente recursiva se calcula recursivamente. Los párrafos siguientes explican la forma de hacerlo.

Supongamos que hemos calculado y almacenado todos los  $r_m(i)$  para el ciclo de adaptación actual y deseamos pasar al ciclo de adaptación siguiente, que empieza en la muestra  $s_u(m+L)$ . Después de desplazar la ventana híbrida  $L$  muestras a la derecha, la nueva señal con ponderada de ventana para el siguiente ciclo de adaptación es

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

La componente recursiva de  $R_{m+L}(i)$  puede escribirse

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

o

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

Por consiguiente,  $r_{m+L}(i)$  puede calcularse recursivamente a partir de  $r_m(i)$  utilizando la ecuación (3-1g). Este último  $r_{m+L}(i)$  se almacena en la memoria para utilizarlo en el ciclo de adaptación siguiente. El coeficiente de autocorrelación  $R_{m+L}(i)$  se calcula entonces como

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

Hasta ahora hemos descrito de modo general los principios del procedimiento de cálculo de ventana híbrida. Los valores de los parámetros del módulo 36 de ventanización híbrida de la figura 4a)/G.728 son

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

Una vez calculados los 11 coeficientes de autocorrelación  $R(i)$ ,  $i = 0, 1, \dots, 10$  por el procedimiento de ventanización híbrida antes descrito, se aplica un procedimiento de «corrección por ruido blanco». Esto se hace aumentando la energía  $R(0)$  en una pequeña cantidad:

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

Esto tiene por efecto llenar las depresiones espectrales con ruido blanco para reducir la gama dinámica espectral y atenuar el desajuste de la recursión de Levinson-Durbin subsiguiente. El factor de corrección por ruido blanco (WNCF, *white noise correction factor*) de 257/256 corresponde a un nivel de ruido blanco de unos 24 dB por debajo de la potencia de señal vocal media.

Después, utilizando los coeficientes de autocorrelación con corrección por ruido blanco, el módulo 37 de recursión Levinson-Durbin calcula recursivamente los coeficientes de predicción del primer orden al décimo orden. Sean  $a_j^{(i)}$  los  $j$ -ésimos coeficientes del predictor de  $i$ -ésimo orden. Entonces, el procedimiento recursivo puede especificarse como sigue:

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

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

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

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

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

Las ecuaciones (3-2b) a (3-2e) son evaluadas recursivamente para  $i = 1, 2, \dots, 10$ , y la solución final viene dada por

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

Si definimos  $q_0 = 1$ , el «filtro de error de predicción» de décimo orden (a veces denominado «filtro de análisis») tiene la función de transferencia

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

y el correspondiente predictor lineal de décimo orden viene definido por la siguiente función de transferencia

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

El calculador de los coeficientes del filtro de ponderación (bloque 38) calcula los coeficientes del filtro de ponderación perceptual de acuerdo con las ecuaciones siguientes:

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

y

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

El filtro de ponderación perceptual es un filtro de polos y ceros de décimo orden definido por la función de transferencia  $W(z)$  en la ecuación (3-4a). Los valores de  $\gamma_1$  y  $\gamma_2$  son respectivamente 0,9 y 0,6.

Véase la figura 2/G.728. El adaptador del filtro de ponderación perceptual (bloque 3) actualiza periódicamente los coeficientes de  $W(z)$  según las ecuaciones (3-2) a (3-4), y suministra los coeficientes al calculador del vector de respuesta a impulsos (bloque 12) y a los filtros de ponderación perceptual (bloques 4 a 10).

### 3.4 Filtro de ponderación perceptual

En la figura 2/G.728, se hace pasar el vector de señal vocal de entrada actual  $s(n)$  a través del filtro de ponderación perceptual (bloque 4), dando lugar al vector vocal ponderado  $v(n)$ . Obsérvese que, salvo durante la inicialización, la memoria del filtro (es decir las variables de estado internas o los valores mantenidos en las unidades de retardo del filtro) no debe reponerse a cero en ningún momento. Por otra parte, la memoria del filtro de ponderación perceptual (bloque 10) necesitará un tratamiento especial, como se indica más adelante.

#### 3.4.1 Funcionamiento con señales no vocales

Para señales de módem u otras señales no vocales, los resultados de las pruebas del CCITT indican que es conveniente desactivar el filtro de ponderación perceptual. Esto equivale a hacer  $W(z) = 1$ . El modo más fácil de conseguirlo es haciendo iguales a cero  $\gamma_1$  y  $\gamma_2$  en la ecuación (3-4a). Los valores nominales de estas variables en el modo conversación son respectivamente 0,9 y 0,6.

### 3.5 Filtro de síntesis

En la figura 2/G.728 hay dos filtros de síntesis (bloques 9 y 22) con coeficientes idénticos. Ambos filtros son actualizados por el adaptador del filtro de síntesis hacia atrás (bloque 23). Cada filtro de síntesis es un filtro «todos polos» de quincuagésimo orden que consiste en un bucle de realimentación con un predictor LPC de quincuagésimo orden en la rama de realimentación. La función de transferencia del filtro de síntesis es  $F(z) = 1/[1 - P(z)]$ , donde  $P(z)$  es la función de transferencia del predictor LPC de quincuagésimo orden.

Después de haber obtenido el vector de señal vocal ponderada  $v(n)$ , se generará un vector de respuesta a entrada cero  $r(n)$  utilizando el filtro de síntesis (bloque 9) y el filtro de ponderación perceptual (bloque 10). Para hacerlo, empezamos por abrir el conmutador 5, es decir, lo llevamos al nodo 6. Esto implica que la señal que va del nodo 7 al filtro de síntesis 9 será cero. Dejamos entonces que el filtro de síntesis 9 y el filtro de ponderación perceptual 10 «suenen» para cinco muestras (un vector). Esto significa que continuamos la operación de filtrado durante cinco muestras con una señal cero aplicada en el nodo 7. La salida resultante del filtro de ponderación perceptual 10 es el vector de respuesta a entrada cero deseado  $r(n)$ .

Obsérvese que, salvo para el vector que sigue inmediatamente a la inicialización, la memoria de los filtros 9 y 10 es generalmente distinta de cero; por consiguiente, el vector de salida  $r(n)$  también es generalmente distinto de cero, incluso si la entrada del filtro desde el nodo 7 es cero. En efecto, este vector  $r(n)$  es la respuesta de los dos filtros a los vectores de excitación con escalamiento de ganancia anteriores  $e(n-1)$ ,  $e(n-2)$ , . . . Este vector representa en realidad el efecto debido a la memoria del filtro hasta el tiempo  $(n-1)$ .

### 3.6 Cálculo del vector objetivo VQ

Este bloque sustrae el vector respuesta a entrada cero  $r(n)$  del vector de señal vocal ponderada  $v(n)$  para obtener el vector  $x(n)$  objetivo usado en la búsqueda en la tabla de códigos cifrados VQ.

### 3.7 Adaptador del filtro de síntesis hacia atrás

Este adaptador 23 actualiza los coeficientes de los filtros de síntesis 9 y 22. Toma como entrada la señal vocal cuantificada (sintetizada) y produce un conjunto de coeficientes de filtro de síntesis como salida. Su funcionamiento es bastante similar al del adaptador del filtro de ponderación perceptual 3.

En la figura 5/G.728 se ilustra una versión ampliada de este adaptador. El funcionamiento del módulo de ventanización híbrida 49 y del módulo de recursión de Levinson-Durbin 50 es exactamente igual al de sus equivalentes (36 y 37) en la figura 4a)/G.728, con las tres diferencias siguientes:

- La señal de entrada es ahora la señal vocal cuantificada y no la señal vocal de entrada no cuantificada.
- El orden del predictor es 50 y no 10.
- Los parámetros de la ventana híbrida son diferentes:  $N = 35$ ,  $\alpha = \left(\frac{3}{4}\right)^{40} = 0,992833749$ .

Obsérvese que el periodo de actualización sigue siendo  $L = 20$ , y el factor de corrección de ruido blanco sigue siendo  $257/256 = 1,00390625$ .

Sea  $\hat{P}(z)$  la función de transferencia del predictor LPC de quincuagésimo orden; por lo que presenta la forma

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

donde los  $\hat{a}_i$  son los coeficientes del predictor. Para mejorar su resistencia a los errores de canal, estos coeficientes se modifican de manera que los picos del espectro LPC resultante tengan anchuras de banda ligeramente mayores. El módulo de expansión de anchura de banda 51 ejecuta este procedimiento de expansión de anchura de banda como se indica a continuación. Dados los coeficientes  $\hat{a}_i$  del predictor LPC, se calcula un nuevo conjunto de coeficientes  $a_i$  por la fórmula

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

donde  $\lambda$  viene dado por

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

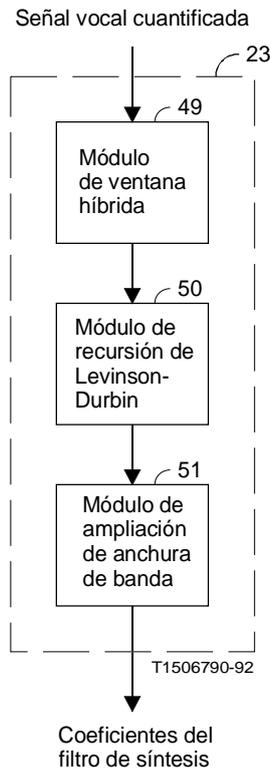


FIGURA 5/G.728

### Adaptador del filtro de síntesis hacia atrás

Esto hace que se desplacen todos los polos de filtro de síntesis radialmente hacia el origen en un factor  $\lambda$ . Dado que los polos se desplazan alejándose del círculo unitario, se ensanchan los picos de la respuesta de frecuencia.

Después de esta expansión de anchura de banda, el predictor LPC modificado tiene la función de transferencia

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

Los coeficientes modificados se aplican entonces a los filtros de síntesis 9 y 22. Estos coeficientes se aplican también al calculador del vector de respuesta a los impulsos 12.

Los filtros de síntesis 9 y 22 tienen la función de transferencia

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

Análogamente al filtro de ponderación perceptual, los filtros de síntesis 9 y 22 también son actualizados una vez cada cuatro vectores, y las actualizaciones se producen también en el tercer vector de señal vocal de cada ciclo de adaptación de cuatro vectores. No obstante, las actualizaciones se basan en la señal vocal cuantificada hasta el último vector del ciclo de adaptación anterior. En otras palabras, se introduce un retardo de dos vectores antes de que se produzca las actualizaciones. Esto se debe a que el módulo de recursión de Levinson-Durbin 50 y el calculador de la tabla de energía 15 (descrito más adelante) exigen muchos cálculos. Como resultado, aunque se disponga de

autocorrelación de la señal vocal cuantificada anteriormente en el primer vector de cada ciclo de cuatro vectores, los cálculos pueden exigir un tiempo superior al de un vector. Por consiguiente, para mantener un tamaño de memoria intermedia básica de un vector (a fin de mantener bajo el retardo de codificación), y para mantener el funcionamiento en tiempo real, se introduce un retardo de dos vectores en las actualizaciones del filtro a fin de facilitar la aplicación en tiempo real.

### 3.8 Adaptador de ganancia vectorial hacia atrás

Este adaptador actualiza la ganancia de excitación  $\sigma(n)$  para cada índice de tiempo de vector  $n$ . La ganancia de excitación  $\sigma(n)$  es un factor de escala empleado para escalar el vector de excitación seleccionado  $y(n)$ . El adaptador 20 toma como entrada el vector de excitación con escalamiento de ganancia  $e(n)$ , y produce como salida una ganancia de excitación  $\sigma(n)$ . Fundamentalmente, intenta «predecir» la ganancia de  $e(n)$ , basándose en las ganancias de  $e(n - 1)$ ,  $e(n - 2)$ , . . . utilizando predicción lineal adaptativa en el dominio de ganancia logarítmica. Este adaptador de ganancia vectorial hacia atrás 20 se muestra más detalladamente en la figura 6/G.728.

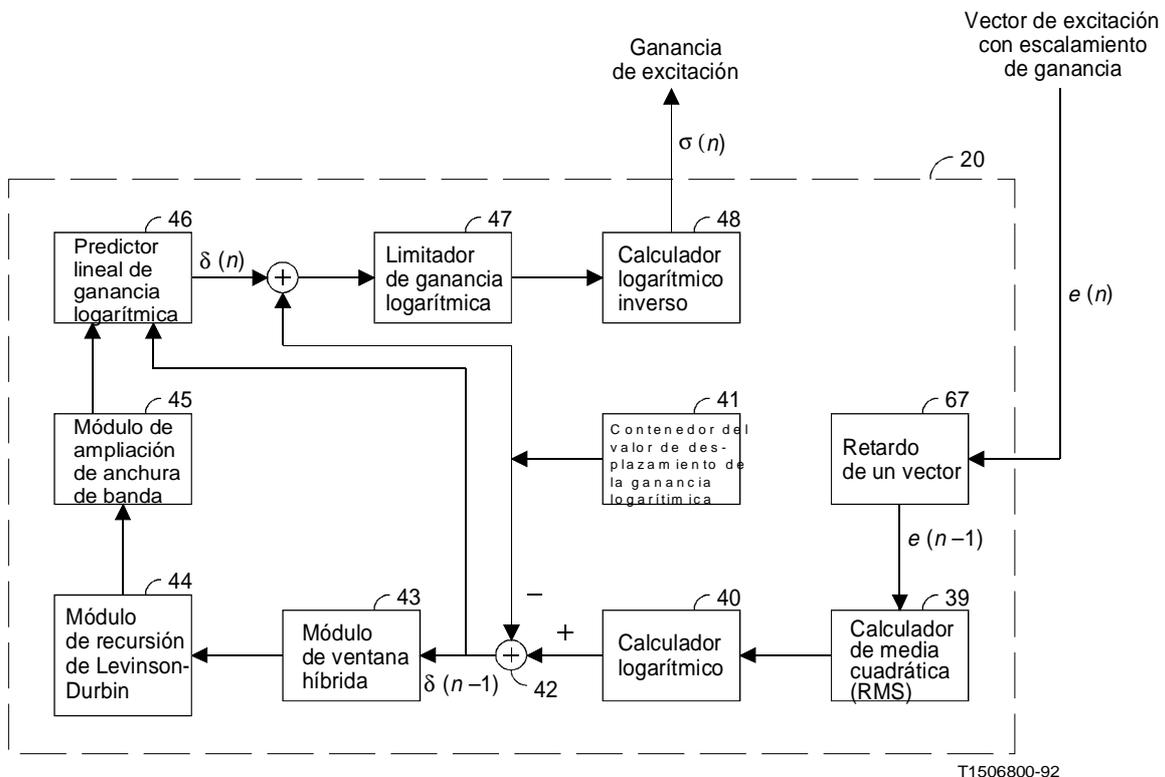


FIGURA 6/G.728

### Adaptador de ganancia vectorial hacia atrás

Véase la figura 6/G.728. El adaptador de ganancia funciona como sigue. La unidad de retardo de un vector (bloque 67) provee el vector de excitación con escalamiento de ganancia anterior  $e(n - 1)$ . El calculador de media cuadrática (RMS, *root-mean-square*) 39 calcula entonces el valor cuadrático medio del vector  $e(n - 1)$ . A continuación, el calculador logarítmico 40 calcula el equivalente en dB del valor cuadrático medio de  $e(n - 1)$ , calculando primero el logaritmo de base 10 y multiplicando luego el resultado por 20.

En la figura 6/G.728, se almacena un valor de desplazamiento de ganancia logarítmica de 32 dB en el contenedor del valor de desplazamiento de la ganancia logarítmica 41. Se supone que este valor es aproximadamente igual al nivel de ganancia de excitación media (en dB) durante la conversación. El sumador 42 sustrae este valor de desplazamiento de ganancia logarítmica de la ganancia logarítmica que sale del calculador logarítmico 40. La

ganancia logarítmica resultante (trás deducirle el desplazamiento,  $\delta(n - 1)$ ), es utilizada entonces por el módulo de ventanización híbrida 43 y el módulo de recursión de Levinson-Durbin 44. Una vez más, los bloques 43 y 44 funcionan exactamente de la misma manera que los bloques 36 y 37 del módulo adaptador de filtro de ponderación perceptual [figura 4a)], salvo que los parámetros de la ventana híbrida son diferentes y que la señal analizada es ahora la ganancia logarítmica con desplazamiento sustraído, y no la señal vocal de entrada. (Obsérvese que sólo se produce un valor de ganancia cada cinco muestras vocales.). Los parámetros de ventana híbrida del bloque 43 son

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

La salida del módulo de recursión de Levinson-Durbin 44 consiste en los coeficientes de un predictor lineal de décimo orden cuya función de transferencia es

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

El módulo de expansión de anchura de banda 45 desplaza entonces las raíces de este polinomio radialmente hacia el plano  $z$  de origen de manera similar al módulo 51 de la figura 5/G.728. El predictor de ganancia de anchura de banda ampliada resultante tiene la función de transferencia

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

donde los coeficientes  $\alpha_i$  se calculan por la fórmula

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

Esta expansión de anchura de banda hace el adaptador de ganancia (bloque 20 de la figura 2/G.728) más resistente a los errores de canal. Estos  $\alpha_i$  se utilizan entonces como coeficientes del predictor lineal de ganancia logarítmica (bloque 46 de la figura 6/G.728).

Este predictor 46 es actualizado una vez cada cuatro vectores vocales, y las actualizaciones se producen en el segundo vector vocal de cada ciclo de adaptación de cuatro vectores. El predictor intenta predecir  $\delta(n)$  basándose en una combinación lineal de  $\delta(n - 1)$ ,  $\delta(n - 2)$ , ...,  $\delta(n - 10)$ . La versión predicha de  $\delta(n)$  se designa por  $\hat{\delta}(n)$  y viene dada por:

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

Una vez que el predictor lineal de ganancia logarítmica 46 ha obtenido  $\hat{\delta}(n)$ , añadimos nuevamente el valor del desplazamiento de ganancia logarítmica de 32 dB almacenado en 41. El limitador de ganancia logarítmica 47 verifica entonces el valor de la ganancia logarítmica y lo recorta si es demasiado grande o demasiado pequeño. Los límites inferior y superior se hacen iguales a 0 dB y 60 dB, respectivamente. La salida del limitador de ganancia se aplica entonces al calculador logarítmico inverso 48, que invierte la operación del calculador logarítmico 40 y convierte el valor en dB de la ganancia al dominio lineal. El limitador de ganancia asegura que la ganancia en el dominio lineal esté comprendida entre 1 y 1000.

### 3.9 Módulo de búsqueda de código cifrado

Los bloques 12 a 18 de la figura 2/G.728 constituyen el módulo de búsqueda de código cifrado 24. Este módulo realiza una búsqueda entre los 1024 vectores de código candidatos de la tabla de códigos cifrados VQ de excitación 19 e identifica el índice del mejor vector de código, que da el correspondiente vector de señal vocal cuantificada más próximo al vector de señal vocal de entrada.

Para reducir la complejidad de la búsqueda de código cifrado, la tabla de 1024 códigos cifrados de diez bits se descompone en dos tablas más pequeñas: una «tabla de códigos de forma» de siete bits que contiene 128 vectores de código independientes, y una «tabla de códigos cifrados de ganancia» de tres bits que contiene ocho valores escalares simétricos con respecto a cero (es decir, un bit para el signo y dos bits para la magnitud). El vector de código de salida final es el producto del mejor vector de código de forma (de la tabla de códigos cifrados de forma de siete bits) y del mejor nivel de ganancia (de la tabla de códigos cifrados de ganancia de tres bits). La tabla de códigos cifrados de forma de siete bits, y la tabla de códigos cifrados de ganancia de tres bits figura en el anexo B.

### 3.9.1 Principio de la búsqueda de código cifrado

En principio, el módulo de búsqueda de código cifrado 24 normaliza cada uno de los 1024 vectores de código candidatos mediante la ganancia de excitación actual  $\sigma(n)$  y transmite los 1024 vectores resultantes uno por uno a través de un filtro en cascada que consta del filtro de síntesis  $F(z)$  y del filtro de ponderación perceptual  $W(z)$ . La memoria del filtro se inicializa a cero cada vez que el módulo aplica un nuevo vector de código al filtro en cascada con la función de transferencia  $H(z) = F(z) W(z)$ .

El filtrado de los vectores de código VQ puede expresarse como la multiplicación de matriz por vector. Sea  $y_j$  el  $j$ -ésimo vector de código en la tabla de códigos cifrados de forma de siete bits, y sea  $g_i$  el  $i$ -ésimo nivel en la tabla de códigos cifrados de ganancia de 3 bits. Sea  $\{h(n)\}$  la secuencia de respuesta a los impulsos del filtro en cascada. Entonces, cuando el vector de código especificado por los índices  $i$  y  $j$  de la tabla de códigos cifrados es aplicado al filtro en cascada  $H(z)$ , la salida del filtro puede expresarse como

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

donde

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

El módulo de búsqueda de código cifrado 24 busca la mejor combinación de índices  $i$  y  $j$  que minimiza la siguiente distorsión de error cuadrático medio (MSE, *mean-squared error*):

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

donde  $\hat{x}(n) = x(n) / \sigma(n)$  es el vector objetivo VQ con normalización de ganancia. Desarrollando los términos cuadrado se obtiene:

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

Dado que el término  $\|\hat{x}(n)\|^2$  y el valor de  $\sigma^2(n)$  son fijos durante la búsqueda de código, minimizar  $D$  equivale a minimizar:

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

donde

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

y

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

Obsérvese que  $E_j$  es en realidad la energía de los  $j$ -ésimos vectores de código de forma filtrados, y no depende de  $\hat{x}(n)$ , el vector objetivo VQ. Obsérvese también que el vector de código de forma  $y_j$  es fijo, y la matriz  $\mathbf{H}$  depende únicamente del filtro de síntesis y del filtro de ponderación, que son fijos durante un periodo de cuatro vectores de señal vocal. En consecuencia,  $E_j$  también es fijo durante un periodo de cuatro vectores de señal vocal. Basándonos en esta observación, al actualizar los dos filtros, podemos calcular y almacenar los 128 posibles términos de energía  $E_j$ ,  $j = 0, 1, 2, \dots, 127$  (correspondientes a los 128 vectores de código de forma) y utilizarlos luego repetidamente para la búsqueda de código cifrado durante los cuatro vectores siguientes de señal vocal. Esto reduce la complejidad de la búsqueda de código cifrado.

Para reducir aún más los cálculos, podemos calcular y almacenar previamente los dos ordenamientos

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

y

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

para  $i = 0, 1, \dots, 7$ . Estos dos ordenamientos son fijos, dado que los  $g_i$  son fijos. Ahora podemos expresar  $\hat{D}$  como:

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

donde  $P_j = p^T(n)y_j$ .

Obsérvese que una vez precalculadas y almacenadas las tablas  $E_j$ ,  $b_i$  y  $c_i$ , el término del producto interno  $P_j = p^T(n)y_j$ , que depende solamente de  $j$ , incluye la mayor parte de los cálculos para determinar  $\hat{D}$ . Así, el procedimiento de búsqueda de código cifrado recorre la tabla de códigos cifrados de forma e identifica el mejor índice de ganancia  $i$  para cada vector de código de forma  $y_j$ .

Existen varias maneras de hallar el mejor índice de ganancia  $i$  para un determinado vector de código de forma  $y_j$ .

- La primera manera, y la más obvia, consiste en evaluar los ocho posibles valores  $\hat{D}$  correspondientes a los ocho posibles valores de  $i$ , y seleccionar luego el índice  $i$  que corresponde al  $\hat{D}$  más pequeño. Sin embargo, esto requiere dos multiplicaciones para cada  $i$ .
- Una segunda manera consiste en calcular primero la ganancia óptima  $\hat{g} = P_j / E_j$ , y cuantificar luego esta ganancia  $\hat{g}$  según uno de los ocho niveles de ganancia  $\{g_0, \dots, g_7\}$  de la tabla de códigos cifrados de ganancia de tres bits. El mejor índice  $i$  es el índice del nivel de ganancia  $g_i$  más próximo a  $\hat{g}$ . No obstante, este método exige una operación de división para cada uno de los 128 vectores de código de forma, y la división suele ser poco eficaz al utilizar procesadores DSP.
- Un tercer método, que es una versión ligeramente modificada del segundo, es particularmente eficaz para aplicaciones DSP. La cuantificación de  $\hat{g}$  puede considerarse como una serie de comparaciones entre  $\hat{g}$  y las «fronteras de célula del cuantificador», que son los puntos medios entre niveles de ganancia adyacentes. Sea  $d_i$  el punto medio entre los niveles de ganancia  $g_i$  y  $g_{i+1}$  que tienen el mismo signo. Entonces, la prueba « $\hat{g} < d_i?$ » es equivalente a la prueba « $P_j < d_i E_j?$ ». De esta manera, utilizando la última prueba, podemos evitar la operación de división y necesitar únicamente una multiplicación para cada índice  $i$ . Este es el método utilizado en la búsqueda de código cifrado. Las fronteras de célula del cuantificador de ganancia  $d_i$  son fijas y pueden calcularse y almacenarse previamente en una tabla. Para los ocho niveles de ganancia, se utilizan en la práctica únicamente los seis valores de frontera  $d_0, d_1, d_2, d_4, d_5$  y  $d_6$ .

Una vez identificados los mejores índices  $i$  y  $j$ , se concatenan para formar la salida del módulo de búsqueda de código cifrado – un solo índice de diez bits que corresponde al «mejor» código cifrado.

### 3.9.2 Funcionamiento del módulo de búsqueda de código cifrado

Habiendo introducido el principio de la búsqueda de código cifrado, se describe a continuación el funcionamiento del módulo de búsqueda de código cifrado 24. Véase la figura 2/G.728. Cada vez que se actualizan el filtro de síntesis 9 y el filtro de ponderación perceptual 10, el calculador de vector de respuesta a los impulsos 12 calcula las cinco primeras muestras de la respuesta a los impulsos del filtro en cascada  $F(z) W(z)$ . Para calcular el vector de respuesta a los impulsos, ponemos primero la memoria del filtro en cascada a cero, y excitamos luego dicho

filtro con una secuencia de entrada  $\{1, 0, 0, 0, 0\}$ . Las cinco muestras de salida correspondientes del filtro son  $h(0), h(1), \dots, h(4)$ , que constituyen el vector de respuesta a los impulsos deseado. Después de haber calculado este vector de respuesta a los impulsos, se mantendrá constante y se utilizará en la búsqueda de código cifrado para los cuatro vectores de señal vocal siguientes, hasta que los filtros 9 y 10 sean actualizados nuevamente.

A continuación, el módulo de convolución de vector de código de forma 14 calcula los 128 vectores  $\mathbf{H}y_j, j = 0, 1, 2, \dots, 127$ . En otras palabras, convolucionada cada vector de código de forma  $y_j, j = 0, 1, 2, \dots, 127$  con la secuencia de respuesta a los impulsos  $h(0), h(1), \dots, h(4)$ , donde la convolución es realizada únicamente para las cinco primeras muestras. La energía de los 128 vectores resultantes es entonces calculada y almacenada por el calculador de la tabla de energía 15 de acuerdo con la ecuación (3-20). La energía de un vector se define como la suma de los cuadrados de las componentes de dicho vector.

Obsérvese que los cálculos de los bloques 12, 14 y 15 se realizan una vez cada cuatro vectores de señal vocal, mientras que los demás bloques del módulo de búsqueda de código cifrado llevan a cabo cálculos para cada vector de señal vocal. Obsérvese asimismo que las actualizaciones de la tabla  $E_j$  están sincronizadas con las actualizaciones de los coeficientes del filtro de síntesis. Es decir, la nueva tabla  $E_j$  se utilizará a partir del tercer vector de señal vocal de cada ciclo de adaptación. (Véase el § 3.7.)

El módulo de normalización de vector objetivo VQ 16 calcula el vector objetivo VQ con normalización de ganancia  $\hat{x}(n) = x(n) / \sigma(n)$ . En las aplicaciones DSP, es más eficaz calcular primero  $1 / \sigma(n)$ , y multiplicar luego cada componente de  $x(n)$  por  $1 / \sigma(n)$ .

Acto seguido, el módulo de convolución con inversión del tiempo 13, calcula el vector  $p(n) = \mathbf{H}^T \hat{x}(n)$ . Esta operación es equivalente a invertir primero el orden de las componentes de  $\hat{x}(n)$ , convolucionar luego el vector resultante con el vector de respuesta a los impulsos, e invertir finalmente el orden de las componentes de la salida una vez más (lo que justifica el nombre de «convolución con inversión del tiempo»).

Una vez precalculadas y almacenadas las tablas  $E_j, b_i$  y  $c_i$ , y una vez calculado el vector  $p(n)$ , el calculador de error 17 y el selector del mejor índice de código cifrado 18 trabajan juntos para ejecutar el algoritmo de búsqueda de código cifrado siguiente, que se caracteriza por su eficiencia.

- a) inicializar  $\hat{D}_{\min}$  a un número mayor que el máximo valor posible de  $\hat{D}$  (o utilizar el mayor número posible del sistema DSP de representación de números);
- b) poner el índice de código cifrado de forma a  $j = 0$ ;
- c) calcular el producto interno  $P_j = p^t(n)y_j$ ;
- d) si  $P_j < 0$ , ir a la etapa h) para buscar entre los valores negativos de ganancia; de no ser así, ir a la etapa e) para buscar entre las ganancias positivas;
- e) si  $P_j < d_0 E_j$ , poner  $i = 0$  e ir a la etapa k); de no ser así, pasar a la etapa f);
- f) si  $P_j < d_1 E_j$ , poner  $i = 1$  e ir a la etapa k); de no ser así, pasar a la etapa g);
- g) si  $P_j < d_2 E_j$ , poner  $i = 2$  e ir a la etapa k); de no ser así, poner  $i = 3$  e ir a la etapa k);
- h) si  $P_j > d_4 E_j$ , poner  $i = 4$  e ir a la etapa k); de no ser así, pasar a la etapa i);
- i) si  $P_j > d_5 E_j$ , poner  $i = 5$  e ir a la etapa k); de no ser así, pasar a la etapa j);
- j) si  $P_j > d_6 E_j$ , poner  $i = 6$ ; de no ser así, poner  $i = 7$ ;
- k) calcular  $\hat{D} = -b_i P_j + c_i E_j$ ;
- l) si  $\hat{D} < \hat{D}_{\min}$ , poner  $\hat{D}_{\min} = \hat{D}, i_{\min} = i$ , y  $j_{\min} = j$ ;
- m) si  $j < 127$ , poner  $j = j + 1$  e ir a la etapa c); de no ser así, pasar a la etapa n);
- n) cuando el algoritmo llega a ese punto, se ha realizado la búsqueda en la totalidad de las 1024 combinaciones posibles de ganancia y forma. Los  $i_{\min}$  y  $j_{\min}$  resultantes son los índices de canal deseados para la ganancia y la forma, respectivamente. El mejor índice de código cifrado de salida (diez bits) es la concatenación de estos dos índices, y el mejor vector de código de excitación correspondiente es  $y(n) = g_{i_{\min}} y_{j_{\min}}$ . El índice de código cifrado de diez bits seleccionado es transmitido al decodificador a través del canal de comunicación.

### 3.10 Decodificador simulado

Si bien hasta este punto el codificador ha identificado y transmitido el mejor índice de código cifrado, deben llevarse a cabo algunas tareas adicionales para preparar la codificación de los siguientes vectores de señal vocal. Primero, se introduce el mejor índice de código cifrado al módulo de códigos cifrados VQ de excitación a fin de extraer el mejor vector de código correspondiente  $y(n) = g_{i_{\min}} y_{j_{\min}}$ . Este mejor vector de código es entonces escalado mediante la ganancia de excitación actual  $\sigma(n)$  en la etapa de ganancia 21. El vector de excitación con escalonamiento de ganancia resultante es  $e(n) = \sigma(n) y(n)$ .

Este vector  $e(n)$  pasa entonces a través del filtro de síntesis 22 para obtener el vector de señal vocal cuantificada actual  $s_q(n)$ . Obsérvese que los bloques 19 a 23 forman el decodificador simulado 8. Por consiguiente, el vector de señal vocal cuantificada  $s_q(n)$  es en la práctica el vector de señal vocal decodificada simulado cuando no hay errores de canal. En la figura 2/G.728 el adaptador del filtro de síntesis hacia atrás 23 necesita este vector de señal vocal cuantificada  $s_q(n)$  para actualizar los coeficientes del filtro de síntesis. De modo similar, el adaptador de ganancia vectorial hacia atrás 20 necesita el vector de excitación con escalonamiento de ganancia  $e(n)$  para actualizar los coeficientes del predictor lineal de ganancia logarítmica.

Una última tarea antes de proceder a codificar el siguiente vector de señal vocal consiste en actualizar la memoria del filtro de síntesis 9 y del filtro de ponderación perceptual 10. Para hacerlo empezamos por guardar la memoria de los filtros 9 y 10 que fue dejada de lado antes de realizar el cálculo de respuesta a entrada cero descrito en el § 3.5. Acto seguido, ponemos la memoria de los filtros 9 y 10 a cero y cerramos el conmutador 5, es decir lo conectamos al nodo 7. Entonces, el vector de excitación con escalonamiento de ganancia  $e(n)$  pasa a través de los dos filtros 9 y 10 con memoria cero. Obsérvese que, dado que  $e(n)$  tiene una longitud de sólo cinco muestras y los filtros tienen una memoria cero, el número de multiplicaciones-adiciones asciende sólo de cero a cuatro para el periodo de cinco muestras. Esto es un ahorro considerable de cálculo, dado que habría 70 multiplicaciones-adiciones por muestra si la memoria de los filtros fuera distinta de cero. A continuación, añadimos nuevamente la memoria del filtro original, guardada, a la memoria del filtro nuevamente establecida después de filtrar  $e(n)$ . Esto, en efecto, añade las respuestas de entrada cero a las respuestas de estado cero de los filtros 9 y 10. Esto genera al conjunto deseado de memoria de filtro que se utilizará para calcular la respuesta a entrada cero durante la codificación del siguiente vector de señal vocal.

Obsérvese que después de la actualización de la memoria de los filtros, los cinco elementos superiores de la memoria del filtro de síntesis 9 son exactamente los mismos que las componentes del vector de señal vocal cuantificada deseado  $s_q(n)$ . Por consiguiente, podemos omitir el filtro de síntesis 22 y obtener  $s_q(n)$  a partir de la memoria actualizada del filtro de síntesis 9. Esto significa un ahorro adicional de 50 multiplicaciones-adiciones por muestra.

El funcionamiento del codificador descrito hasta ahora especifica la manera en que se codifica un solo vector de señal vocal de entrada. La codificación de la totalidad de la forma de onda de señal vocal se obtiene repitiendo la operación anterior para cada vector de señal vocal.

### 3.11 Sincronización y señalización dentro de banda

En la anterior descripción del codificador se supone que el decodificador conoce las fronteras de los índices de código cifrado de diez bits, y que conoce también cuándo es preciso actualizar el filtro de síntesis y el predictor de ganancia logarítmica (recordemos que son actualizados una vez cada cuatro vectores). En la práctica, esta información de sincronización puede ponerse a disposición del decodificador añadiendo bits adicionales de sincronización en la parte superior del tren binario de 16 kbit/s. Sin embargo, en muchas aplicaciones es necesario insertar bits de sincronización o de señalización dentro de banda como parte del tren binario de 16 kbit/s. Esto puede hacerse de la manera siguiente. Supongamos que ha de insertarse un bit de sincronización cada  $N$  vectores de señal vocal; entonces, para cada  $N$ -ésimo vector de señal vocal de entrada, podemos hacer la búsqueda únicamente en la mitad del código cifrado de forma y producir un índice de código cifrado de forma de 6 seis bits. De esta manera, quitamos un bit de cada  $N$ -ésimo índice de código cifrado transmitido e insertamos en su lugar un bit de sincronización o de señalización.

Es importante observar que no podemos quitar arbitrariamente un bit de un índice de código cifrado de forma de siete bits ya seleccionado: el codificador tiene que saber a cuáles vectores de señal vocal se les quitará un bit y así efectuar la búsqueda únicamente en la mitad de la tabla de códigos cifrados para dichos vectores de señal vocal. De no ser así, el decodificador no tendrá los mismos vectores de código de excitación decodificados para esos vectores de señal vocal.

Dado que el algoritmo de codificación tiene un ciclo de adaptación básico de cuatro vectores, es razonable que  $N$  sea un múltiplo de 4 para que el decodificador pueda determinar fácilmente las fronteras de los ciclos de adaptación del codificador. Para un valor razonable de  $N$  (por ejemplo, 16, que corresponde a un periodo de «robo» de bits de 10 ms), la degradación resultante en la calidad de la señal vocal es despreciable. En particular, hemos encontrado que un valor de  $N = 16$  produce poca distorsión adicional. La tasa del «robo» de bits es en este caso de sólo 100 bit/s.

Si se aplica el procedimiento anterior, recomendamos que cuando el bit deseado sea un cero, la búsqueda se realice únicamente en la primera mitad de la tabla de códigos cifrados de forma, es decir, en los vectores cuyos índices estén entre 0 y 63. Cuando el bit deseado sea un uno, la búsqueda se realizará en la segunda mitad de la tabla de códigos cifrados, y el índice resultante estará comprendido entre 64 y 127. El significado de esta elección es que el bit deseado será el bit que figure en el extremo izquierdo de la palabra de código, dado que los 7 bits del vector de código de forma preceden a los tres bits correspondientes al signo y al código cifrado de ganancia. Recomendamos además que el bit de sincronización se quite del último vector en un ciclo de cuatro vectores. Una vez detectado, la siguiente palabra de código recibida puede empezar el nuevo ciclo de vectores de código.

Si bien afirmamos que la sincronización causa muy poca distorsión, observamos que no se han realizado pruebas oficiales en soportes físicos que utilicen esta estrategia de sincronización. Por consiguiente, la cantidad de degradación no ha sido medida.

No obstante, recomendamos específicamente no utilizar el bit de sincronización para la sincronización en sistemas en que el codificador se activa y desactiva repetidamente. Por ejemplo, consideremos un sistema que utilice un detector de actividad de señal vocal para desactivar el codificador en ausencia de señales vocales. Cada vez que sea activado el codificador, el decodificador tendrá que localizar la secuencia de sincronización. A 100 bit/s, esto tomaría probablemente varios centenares de milisegundos. Además, debe darse tiempo para la alineación del estado del decodificador con el del codificador. El resultado combinado sería un fenómeno conocido como recorte a la entrada, en el que se perdería el comienzo de la señal vocal. Si el codificador y el decodificador arrancan en el mismo instante en que comienza la señal vocal, no se perderá ninguna señal. Esto es posible únicamente en sistemas que utilizan sincronización externa y señalización externa para los instantes de arranque.

#### **4 Principios de funcionamiento del decodificador LD-CELP**

La figura 3/G.728 es un diagrama de bloques del decodificador LD-CELP. En las secciones siguientes se da una descripción funcional de cada bloque.

##### *4.1 Tabla de códigos cifrados VQ de excitación*

Este bloque contiene una tabla de códigos cifrados VQ (que incluye las tablas de códigos cifrados de forma y de ganancia) idéntica a la tabla de códigos cifrados 19 del codificador LD-CELP. Emplea el índice recibido del mejor código cifrado a fin de extraer el mejor vector de código  $y(n)$  seleccionado en el codificador LD-CELP.

##### *4.2 Unidad de escalonamiento de ganancia*

Este bloque calcula el vector de excitación escalonado  $e(n)$  multiplicando cada componente de  $y(n)$  por la ganancia  $\sigma(n)$ .

##### *4.3 Filtro de síntesis*

Este filtro tiene la misma función de transferencia que el filtro de síntesis del codificador LD-CELP (suponiendo una transmisión sin errores). Filtra el vector de excitación escalonado  $e(n)$  para producir el vector de señal vocal decodificado  $s_d(n)$ . Obsérvese que, a fin de evitar cualquier acumulación posible de errores de redondeo durante la decodificación, a veces es deseable duplicar exactamente los procedimientos utilizados en el codificador para obtener  $s_q(n)$ . Si éste es el caso, y si el codificador obtiene  $s_q(n)$  a partir de la memoria actualizada del filtro de síntesis 9, el decodificador debe calcular también  $s_d(n)$  como la suma de la respuesta a la entrada cero y la respuesta al estado cero del filtro de síntesis 32, como se hace en el codificador.

##### *4.4 Adaptador de ganancia vectorial hacia atrás*

La función de este bloque se describe en el § 3.8.

4.5 *Adaptador del filtro de síntesis hacia atrás*

La función de este bloque se describe en el § 3.7.

4.6 *Posfiltro*

Este bloque filtra la señal vocal decodificada para mejorar la calidad perceptual, y se amplía en la figura 7/G.728. Véase dicha figura. El posfiltro consta básicamente de tres partes principales: un posfiltro de largo plazo 71, un posfiltro de corto plazo 72 y una unidad de escalonamiento de ganancia de salida 77. La función de los otros cuatro bloques de la figura 7/G.728 consiste en calcular el factor de escala apropiado que se utiliza en la unidad de escalonamiento de ganancia de salida 77.

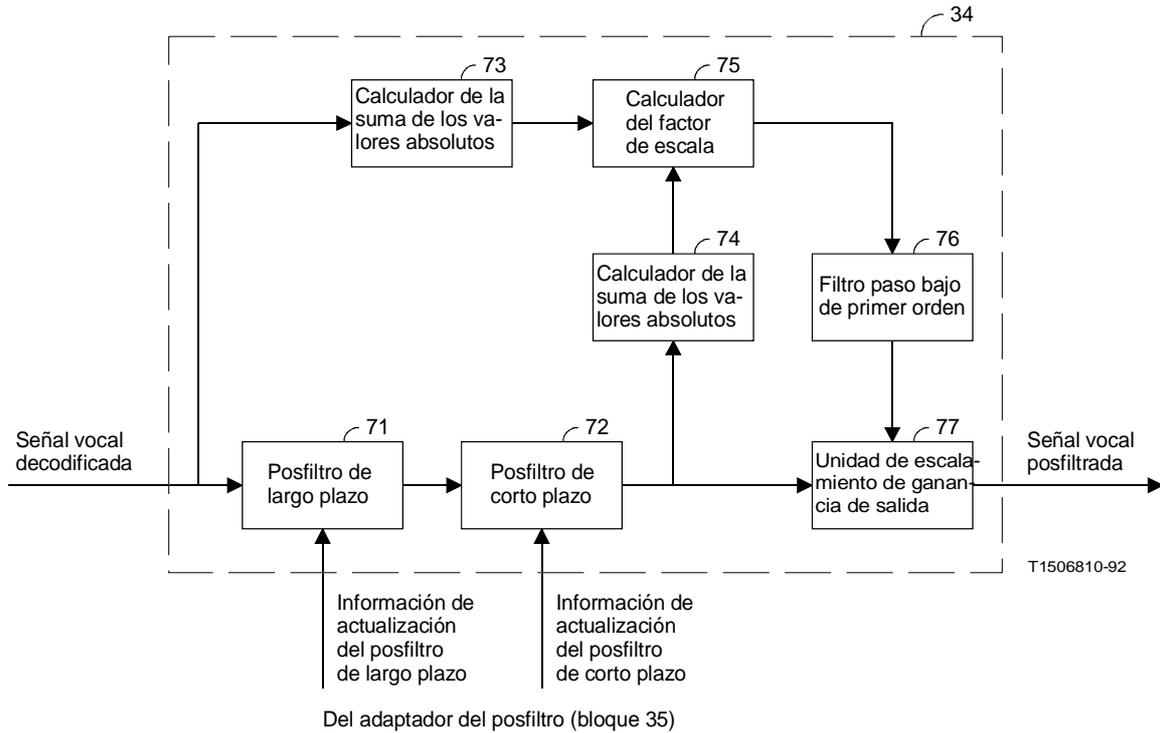


FIGURA 7/G.728

Diagrama de bloques del posfiltro

El *posfiltro de largo plazo* 71, denominado a veces *posfiltro de tono*, es un filtro de peine cuyos picos espectrales están situados en múltiplos de la frecuencia fundamental (o *frecuencia de tono*) de la señal vocal que debe posfiltrarse. El inverso de la frecuencia fundamental se denomina *periodo de tono*. El periodo de tono puede extraerse de la señal vocal decodificada utilizando un detector de tono (o extractor de tono). Sea  $p$  el periodo de tono fundamental (en las muestras) obtenido por un detector de tono; entonces, la función de transferencia del posfiltro de largo plazo puede expresarse como:

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

donde los coeficientes  $g_l$ ,  $b$  y el periodo de tono  $p$  se actualizan una vez cada cuatro vectores de señal vocal (un ciclo de adaptación) y las actualizaciones reales se producen en el tercer vector de señal vocal de cada ciclo de adaptación. Por comodidad, en adelante denominaremos *tramas* a los ciclos de adaptación. El cálculo de  $g_l$ ,  $b$  y  $p$  se describirá más adelante, en el § 4.7.

El posfiltro de corto plazo 72 consta de un filtro de polos y ceros de décimo orden en cascada con un filtro de primer orden todos ceros. El filtro de polos y ceros de décimo orden atenúa los componentes de frecuencia comprendidos entre los picos formantes, mientras que el papel del filtro de primer orden de todos ceros consiste en tratar de compensar la inclinación espectral en la respuesta de frecuencia del filtro de décimo orden de polos y ceros.

Sean  $\tilde{a}_i$ ,  $i = 1, 2, \dots, 10$  los coeficientes del predictor LPC de décimo orden obtenido por análisis LPC hacia atrás de la señal vocal decodificada, y sea  $k_1$  el primer coeficiente de reflexión obtenido mediante el mismo análisis LPC. Entonces, tanto los  $\tilde{a}_i$ , como  $k_1$  pueden obtenerse como derivados del análisis LPC hacia atrás de quincuagésimo orden (bloque 50 de la figura 5/G.728). Todo lo que tenemos que hacer es parar la recursión de Levinson-Durbin de quincuagésimo orden en el décimo orden, copiar  $k_1$  y  $\tilde{a}_1, \tilde{a}_2, \dots, \tilde{a}_{10}$ , y reanudar entonces la recursión de Levinson-Durbin desde el orden 11 hasta el orden 50. La función de transferencia del posfiltro a corto plazo es:

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

donde

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

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

y

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

Los coeficientes  $\bar{a}_i$ ,  $\bar{b}_i$  y  $\mu$  se actualizan también una vez por trama, pero las actualizaciones tienen lugar en el primer vector de cada trama (es decir, tan pronto como se dispone de los  $\tilde{a}_i$ ).

Por lo general, después de que la señal vocal decodificada pasa a través del posfiltro de largo plazo y del posfiltro de corto plazo, la señal vocal filtrada no tendrá el mismo nivel de potencia que la señal vocal decodificada (no filtrada). Para evitar grandes excursiones ocasionales de ganancia, es necesario controlar automáticamente la ganancia para hacer que la señal vocal posfiltrada tenga aproximadamente la misma potencia que la señal vocal no filtrada. Esto es realizado por los bloques 73 a 77.

El calculador de la suma de los valores absolutos (bloque 73) opera vector por vector. Toma el vector de señal vocal decodificada actual  $s_d(n)$  y calcula la suma de los valores absolutos de sus cinco componentes vectoriales. De manera similar, el calculador de la suma de los valores absolutos 74 realiza el mismo tipo de cálculo, pero sobre el vector de salida actual  $s_f(n)$  del posfiltro de corto plazo. El calculador del factor de escala 75 divide entonces el valor de salida del bloque 73 por el valor de salida del bloque 74 a fin de obtener un factor de escala para el vector  $s_f(n)$  actual. Este factor de escala es entonces filtrado por un filtro de paso bajo de primer orden 76 para obtener un factor de escala por separado para cada uno de las cinco componentes de  $s_f(n)$ . El filtro de paso bajo de primer orden 76 tiene la función de transferencia  $0,01/(1 - 0,99 z^{-1})$ . El factor de escala filtrado en paso bajo es utilizado por la unidad de escalamiento de la ganancia de salida 77 para llevar a cabo el escalamiento muestra a muestra de la salida del posfiltro de corto plazo. Obsérvese que, dado que el calculador del factor de escala 75 genera únicamente un factor de escala por vector, tendría un efecto de escalera en la operación de escalamiento muestra a muestra del bloque 77 si no estuviera presente el filtro de paso bajo 76. Efectivamente, el filtro de paso bajo 76 elimina dicho efecto de escalera.

#### 4.6.1 *Funcionamiento no vocal*

Los resultados de las pruebas objetivas del CCITT indican que, para algunas señales no vocales, la calidad de funcionamiento del codificador aumenta cuando se desactiva el posfiltro adaptativo. Como la entrada al posfiltro adaptativo es la salida del filtro de síntesis, esta señal está siempre disponible. En una aplicación real, la señal no filtrada será la salida con el conmutador puesto en posición de desactivación del posfiltro.

4.7 Adaptador del posfiltro

Este bloque calcula y actualiza los coeficientes del posfiltro una vez por trama. Este adaptador del posfiltro se presenta más detalladamente en la figura 8/G.728.

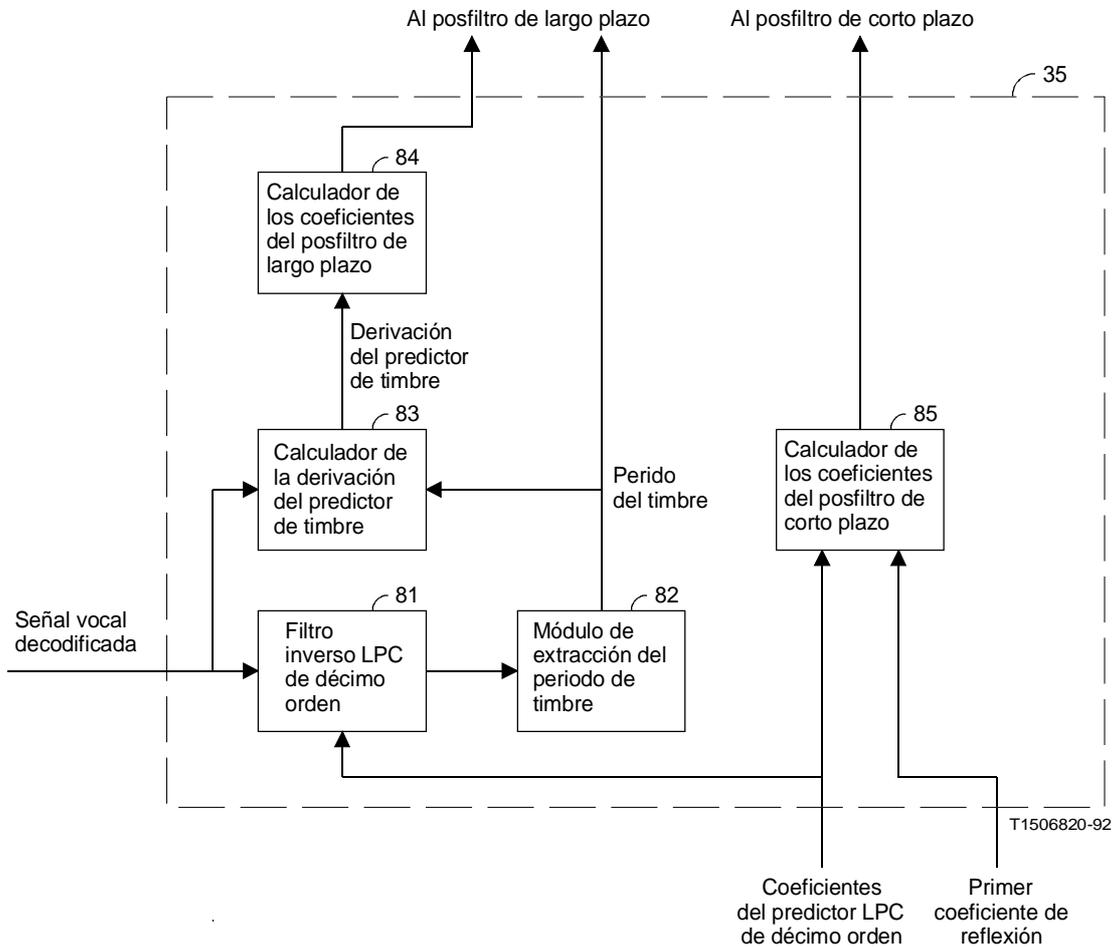


FIGURA 8/G.728

**Diagrama de bloques del adaptador del posfiltro**

Refiérase a la figura 8/G.728. El filtro inverso LPC de décimo orden 81 y el módulo de extracción del periodo de tono 82 trabajan juntos para extraer el periodo de tono de la señal vocal decodificada. De hecho, puede utilizarse cualquier extractor de tono que tenga una calidad razonable (y que no introduzca retardo adicional). Lo que describimos aquí es sólo una manera posible de realizar un extractor de tono.

El filtro inverso LPC de décimo orden 81 tiene la función de transferencia:

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

donde los coeficientes  $\tilde{a}_i$  proceden del módulo de recursión de Levinson-Durbin (bloque 50 de la figura 5/G.728) y son actualizados en el primer vector de cada trama. Este filtro inverso LPC toma como entrada la señal vocal decodificada y genera como salida la secuencia residual de predicción LPC  $\{d(k)\}$ . Empleamos un tamaño de ventana de análisis de tono de 100 muestras y una gama de periodo de tono de 20 a 140 muestras. El módulo de extracción del periodo de

tono 82 mantiene una larga memoria intermedia para conservar las últimas 240 muestras del residuo de predicción LPC. Para comodidad de la indización, las 240 muestras residuales LPC almacenadas en la memoria intermedia se indizan:  $d(-139), d(-138), \dots, d(100)$ .

El módulo de extracción del periodo de tono 82 extrae el periodo de tono una vez por trama, en el tercer vector de cada trama. Por consiguiente, los vectores de salida del filtro inverso LPC deben almacenarse en la memoria intermedia de residuos LPC en un orden especial: el vector residual LPC correspondiente al cuarto vector de la última trama es almacenado como  $d(81), d(82), \dots, d(85)$ , el residuo LPC del primer vector de la trama actual es almacenado como  $d(86), d(87), \dots, d(90)$ , el residuo LPC del segundo vector de la trama actual es almacenado como  $d(91), d(92), \dots, d(95)$ , y el residuo LPC del tercer vector es almacenado como  $d(96), d(97), \dots, d(100)$ . Las muestras  $d(-139), d(-138), \dots, d(80)$  son simplemente las muestras residuales LPC anteriores dispuestas en el orden temporal correcto.

Una vez lista la memoria intermedia residual LPC, el módulo de extracción del periodo de tono 82 trabaja de la manera siguiente. Primero, las últimas 20 muestras de la memoria intermedia de residuos LPC [ $d(81)$  a  $d(100)$ ] es filtrada en paso bajo a un kHz por un filtro elíptico de tercer orden (en el anexo D se dan los coeficientes) y luego diezmada 4:1 (es decir, se reduce el número de muestras en un factor 4). Esto produce cinco muestras residuales LPC diezmadas, y filtradas en paso bajo designadas  $\bar{d}(21), \bar{d}(22), \dots, \bar{d}(25)$ , que son almacenadas como las últimas cinco muestras en una memoria intermedia residual LPC diezmada. Además de estas cinco muestras, las otras 55 muestras  $\bar{d}(-34), \bar{d}(-33), \dots, \bar{d}(20)$  de la memoria residual LPC diezmada se obtienen desplazando las tramas anteriores de las muestras residuales LPC diezmadas. La  $i$ -ésima correlación de las muestras residuales LPC diezmadas se calcula por la fórmula

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

para retardos temporales  $i = 5, 6, 7, \dots, 35$  (que corresponden a periodos de tono de 20 a 140 muestras). El retardo temporal  $\tau$  que da el mayor de los 31 valores de correlación calculados es entonces identificado. Dado que este retardo temporal  $\tau$  es el retardo en el dominio residual diezmado 4:1, el retardo temporal correspondiente que da la correlación máxima en el dominio residual no diezmado original debe estar comprendido entre  $4\tau - 3$  y  $4\tau + 3$ . Para obtener la resolución temporal original, empleamos a continuación la memoria intermedia residual LPC no diezmada para calcular la correlación del residuo LPC no diezmado:

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

para siete retardos  $i = 4\tau - 3, 4\tau - 2, \dots, 4\tau + 3$ . Se determina el retardo  $p_0$  que da la mayor correlación, entre los siete retardos temporales.

Es posible que el retardo temporal  $p_0$  hallado de esta manera resulte ser un múltiplo del verdadero periodo de tono fundamental. Lo que necesitamos en el posfiltro de largo plazo es el verdadero periodo de tono fundamental, y no un múltiplo del mismo. Por consiguiente, necesitamos continuar el procesamiento, a fin de hallar el periodo de tono fundamental. Nos valemos de que se hace una estimación del periodo de tono bastante frecuentemente – una vez cada 20 muestras de señal vocal. Como el periodo de tono varía típicamente entre 20 y 140 muestras, la frecuente estimación que hacemos del tono significa que, al comienzo de cada emisión de voz, obtendremos primero el periodo de tono fundamental antes de que puedan aparecer los múltiplos del periodo de tono en el proceso antes descrito de toma de picos para la correlación. A partir de aquí tendremos la oportunidad de enganchar el periodo de tono fundamental comprobando que no haya ningún pico de correlación en la cercanía del periodo de tono de la trama anterior.

Sea  $\hat{p}$  el periodo de tono de la trama anterior. Si el retardo temporal  $p_0$  obtenido antes no está en la cercanía de  $\hat{p}$ , evaluamos también la ecuación (4-8) para  $i = \hat{p} - 6, \hat{p} - 5, \dots, \hat{p} + 5, \hat{p} + 6$ . Se determina, entre estos 13 posibles

retardos temporales, el retardo  $p_1$  que da la mayor correlación. Después hacemos una prueba para ver si este nuevo retardo  $p_1$  debe ser empleado como periodo de tono de salida de la trama actual. Primero, calculamos:

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

que es el peso de la derivación óptima de un predictor de tono de una sola derivación con un retardo de  $p_0$  muestras. El valor de  $\beta_0$  es entonces fijado entre 0 y 1, después calculamos también:

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

que es el peso de la derivación óptima de un predictor de tono de una sola derivación con un retardo de  $p_1$  muestras. También el valor de  $\beta_1$  es entonces fijado entre 0 y 1. Entonces, el periodo de tono de salida  $p$  del bloque 82 viene dado por

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

Después de que el módulo de extracción del periodo de tono 82 extrae el periodo de tono  $p$ , el calculador de derivación de predictor de tono 83 calcula el peso de la derivación óptima de un predictor de tono de una sola derivación para la señal vocal decodificada. El calculador de la derivación del predictor de tono 83 y el posfiltro de largo plazo 71 comparten una larga memoria intermedia que contiene las muestras de señales vocales decodificadas  $s_d(-239)$ ,  $s_d(-238)$ ,  $s_d(-237)$ , ...,  $s_d(4)$ ,  $s_d(5)$  donde  $s_d(1)$  a  $s_d(5)$  corresponden al vector actual de señal vocal decodificada. El posfiltro de largo plazo 71 utiliza esta memoria como unidad de retardo del filtro. Por otra parte, el calculador de la derivación del predictor de tono 83 utiliza esta memoria para calcular:

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

El calculador de los coeficientes del posfiltro de largo plazo 84 toma entonces el periodo de tono  $p$  y la derivación del predictor de tono  $\beta$  y calcula los coeficientes del posfiltro de largo plazo  $b$  y  $g_l$  como sigue:

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

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

Por lo general, mientras más cerca esté  $\beta$  de la unidad, más periódica será la forma de onda de la señal vocal. Como puede verse en las ecuaciones (4-13) y (4-14), si  $\beta < 0,6$ , lo que corresponde aproximadamente a las regiones de señales no vocales o a las regiones de transición,  $b = 0$  y  $g_l = 1$ , y la función de transferencia del posfiltro de largo plazo es  $H_l(z) = 1$ , lo que significa que la operación de filtrado del posfiltro de largo plazo está totalmente desactivada. Por otra parte, si  $0,6 \leq \beta \leq 1$ , el posfiltro de largo plazo está activado, y el grado de filtrado de peine viene determinado por  $\beta$ . Entre más periódica sea la forma de onda de la señal vocal, más filtrado de peine habrá. Finalmente, si  $\beta > 1$ ,  $b$  está limitado a 0,15 a fin de evitar que haya demasiado filtrado de peine. El coeficiente  $g_l$  es un factor de escala del posfiltro de largo plazo, destinado a garantizar que las regiones con señales vocales de las formas de onda de señal vocal no sean amplificadas con relación a las regiones donde no haya señales vocales o a las regiones de transición. (Si  $g_l$  se mantuviera constante, con un valor unitario, las regiones con señales vocales serían amplificadas aproximadamente en un factor  $1 + b$  después del posfiltrado de largo plazo. Esto haría que algunas consonantes, que corresponden a las regiones sin señal vocal o de transición, sonaran sin claridad o demasiado suavemente.)

El calculador de los coeficientes del posfiltro de corto plazo 85 calcula los coeficientes del posfiltro de corto plazo  $\bar{a}_i$ ,  $\bar{b}_i$ , y  $\mu$  en el primer vector de cada trama de acuerdo con las ecuaciones (4-3), (4-4) y (4-5).

#### 4.8 *Conversión al formato MIC de salida*

Este bloque convierte los cinco componentes del vector de señal vocal decodificada en cinco muestras correspondientes MIC de ley A o  $\mu$ , y les da salida secuencialmente a intervalos de 125  $\mu$ s. Obsérvese que si el formato MIC lineal interno ha sido normalizado como se describe en el § 3.1.1, la normalización inversa debe realizarse antes de la conversión a la MIC de ley A o  $\mu$ .

### 5. **Detalles de cálculo**

Esta sección proporciona los detalles de cálculo para cada uno de los elementos del codificador y del decodificador LD-CELP. En los § 5.1 y 5.2 se dan los nombres de los parámetros del codificador y de las variables de procesamiento interno a que se hará referencia en las secciones ulteriores. La especificación detallada de cada bloque de las figuras 2/G.728 a 6/G.728 se da entre el § 5.3 y el final del § 5. Para codificar y decodificar un vector de señal vocal de entrada, los distintos bloques del codificador y del decodificador se ejecutan en un orden que corresponde aproximadamente a la secuencia comprendida entre el § 5.3 y el final del § 5.

#### 5.1 *Descripción de los parámetros básicos del codificador*

En el cuadro 1/G.728 se definen los nombres de los parámetros básicos del codificador. En la primera columna figuran los nombres de los parámetros del codificador que se utilizarán en la descripción detallada ulterior del algoritmo LD-CELP. Si un parámetro ha sido mencionado en los § 3 ó 4, pero ha sido representado mediante un símbolo diferente, dicho símbolo equivalente se presentará en la segunda columna para facilitar la referencia. Cada parámetro del codificador tiene un valor fijo que se determina en la etapa de diseño del codificador. En la tercera columna figuran estos valores fijos de los parámetros, y en la cuarta columna se da una breve descripción de los parámetros del codificador.

#### 5.2 *Descripción de las variables internas*

Las variables de procesamiento interno del LD-CELP se enumeran en el cuadro 2/G.728, cuya disposición es similar a la del cuadro 1/G.728. En la segunda columna figura la gama de los índices de cada ordenamiento. La cuarta columna da los valores iniciales recomendados de las variables. Los valores iniciales de algunos ordenamientos figuran en los anexos A, B o C. Se recomienda (pese a que no es necesario) que las variables internas se pongan a sus valores iniciales cuando el codificador o el decodificador apenas empiezan a funcionar, o cuando se requiere una reinicialización de los estados del codificador (tal como en las aplicaciones los equipos digitales de multiplicación de circuitos (DCME, *digital circuits multiplication equipments*). Estos valores iniciales garantizan que no habrá fallas justo después del inicio o del reinicio.

Obsérvese que algunos ordenamientos de variables pueden compartir los mismos emplazamientos físicos de memoria a fin de ahorrar espacio en memoria, pese a que, para mayor claridad, se les asignan nombres diferentes en los cuadros.

CUADRO 1/G.728

## Parámetros básicos del codificador LD-CELP

Nombre	Símbolo equivalente	Valor	Descripción
AGCFAC		0,99	Factor de control de la velocidad de adaptación de AGC
FAC	$\lambda$	253/256	Factor de ampliación de anchura de banda del filtro de síntesis
FACGP	$\lambda_g$	29/32	Factor de ampliación de anchura de banda del predictor de ganancia logarítmica
DIMINV		0,2	Inverso de la dimensión del vector
IDIM		5	Dimensión del vector (tamaño del bloque de excitación)
GOFF		32	Valor de desplazamiento de la ganancia logarítmica
KPDELTA		6	Desviación autorizada con relación al periodo de tono anterior
KPMIN		20	Mínimo periodo de tono (muestras)
KPMAX		140	Máximo periodo de tono (muestras)
LPC		50	Orden del filtro de síntesis
LPCLG		10	Orden del predictor de ganancia logarítmica
LPCW		10	Orden del filtro de ponderación perceptual
NCWD		128	Tamaño de la tabla de códigos cifrados de forma (número de vectores de código)
NFRSZ		20	Tamaño de trama (tamaño del ciclo de adaptación, en muestras)
NG		8	Tamaño de la tabla de códigos cifrados de ganancia (número de niveles de ganancia)
NONR		35	Número de muestras en la ventana no recursiva para el filtro de síntesis
NONRLG		20	Número de muestras en la ventana no recursiva para el predictor de ganancia logarítmica
NONRW		30	Número de muestras en la ventana no recursiva para el filtro de ponderación
NPWSZ		100	Tamaño de la ventana de análisis de tono (muestras)
NUPDATE		4	Periodo de actualización del predictor (en términos de vectores)
PPFTH		0,6	Umbral de derivación para desactivar el posfiltro de tono
PPFZCF		0,15	Factor de control de ceros del posfiltro de tono
SPFPCF		0,75	Factor de control de polos del posfiltro de corto plazo
SPFZCF		0,65	Factor de control de ceros del posfiltro de corto plazo
TAPTH		0,4	Umbral de derivación para el reemplazo del tono fundamental
TILTF		0,15	Factor de control de la compensación de la inclinación espectral
WNCF		257/256	Factor de corrección por ruido blanco
WPCF	$\gamma_2$	0,6	Factor de control de polos del filtro de ponderación perceptual
WZCF	$\gamma_1$	0,9	Factor de control de ceros del filtro de ponderación perceptual

Como se menciona en las secciones anteriores, la secuencia de procesamiento tiene un ciclo de adaptación básico de cuatro vectores de señal vocal. La variable ICOUNT se utiliza como índice vectorial. En otras palabras,  $ICOUNT = n$  cuando el codificador o el decodificador están procesando el  $n$ -ésimo vector de señal vocal en un ciclo de adaptación.

Debe observarse que, para facilitar la recursión de Levinson-Durbin, el primer elemento de los ordenamientos A, ATMP, AWP, AWZ y GP es siempre uno y no cambia nunca, y, para  $i \geq 2$ , los  $i$ -ésimos elementos son los  $(i-1)$ -ésimos elementos de los símbolos correspondientes del § 3.

En los puntos siguientes, el asterisco (\*) representa la multiplicación aritmética.

Variables de procesamiento interno del algoritmo LD-CELP

Nombre	Gama de los índices del ordenamiento	Símbolo equivalente	Valor inicial	Descripción
A	1 a LPC+1	$-a_{i-1}$	1,0,0,...	Coefficientes del filtro de síntesis
AL	1 a 3		Anexo D	Coefficientes del denominador del filtro de paso bajo de 1 kHz
AP	1 a 11	$-\tilde{a}_{i-1}$	1,0,0,...	Coefficientes del denominador del posfiltro de corto plazo
APF	1 a 11	$-\tilde{a}_{i-1}$	1,0,0,...	Coefficientes del filtro LPC de décimo orden
ATMP	1 a LPC+1	$-a_{i-1}$		Memoria intermedia temporal para los coeficientes del filtro de síntesis
AWP	1 a LPCW+1		1,0,0,...	Coefficientes del denominador del filtro de ponderación perceptual
AWZ	1 a LPCW+1		1,0,0,...	Coefficientes del numerador del filtro de ponderación perceptual
AWZTMP	1 a LPCW+1		1,0,0,...	Memoria intermedia temporal para los coeficientes del filtro de ponderación
AZ	1 a 11	$-\tilde{b}_{i-1}$	1,0,0,...	Coefficientes del numerador del posfiltro de corto plazo
B	1	$b$	0	Coefficientes del posfiltro de largo plazo
BL	1 a 4		Anexo D	Coefficientes del numerador del posfiltro de paso bajo de 1 kHz
DEC	-34 a 25	$\tilde{d}(n)$	0,0,...,0	Residuo de la predicción LPC diezmado 4:1
D	-139 a 100	$d(k)$	0,0,...,0	Residuo de la predicción LPC
ET	1 a IDIM	$e(n)$	0,0,...,0	Vector de excitación con escalamiento de ganancia
FACV	1 a LPC+1	$\lambda^{i-1}$	Anexo C	Vector de ampliación de la anchura de banda del filtro de síntesis
FACGPV	1 a LPCLG+1	$\lambda_g^{i-1}$	Anexo C	Vector de ampliación de la anchura de banda del predictor de ganancia
G2	1 a NG	$b_i$	Anexo B	2 veces los niveles de ganancia en la tabla de códigos cifrados de ganancia
GAIN	1	$\sigma(n)$		Ganancia de excitación
GB	1 a NG-1	$d_i$	Anexo B	Punto medio entre niveles de ganancia adyacentes
GL	1	$g_l$	1	Factor de escala del posfiltro de largo plazo
GP	1 a LPCLG+1	$-\alpha_{i-1}$	1,-1,0,0,...	Coefficientes del predictor lineal de ganancia logarítmica
GPTMP	1 a LPCLG+1	$-\alpha_{i-1}$		Ordenamiento temporal para los coeficientes del predictor lineal de ganancia logarítmica
GQ	1 a NG	$g_i$	Anexo B	Niveles de ganancia en la tabla de códigos cifrados de ganancia
GSQ	1 a NG	$c_i$	Anexo B	Cuadrados de los niveles de ganancia en la tabla de códigos cifrados de ganancia

CUADRO 2/G.728 (cont.)

Nombre	Gama de los índices del ordenamiento	Símbolo equivalente	Valor inicial	Descripción
GSTATE	1 a LPCLG	$\delta(n)$	-32,-32,...,-32	Memoria del predictor lineal de ganancia logarítmica
GTMP	1 a 4		-32,-32,-32,-32	Memoria intermedia temporal de ganancia logarítmica
H	1 a IDIM	$h(n)$	1,0,0,0,0	Vector de respuesta a los impulsos de $F(z)W(z)$
ICHAN	1			Mejor índice de código cifrado que ha de transmitirse
ICOUNT	1			Contador de vector de señal vocal (indizado de 1 a 4)
IG	1	$i$		Mejor índice de código cifrado de ganancia de tres bits
IP	1		IPINIT <sup>b)</sup>	Puntero de dirección del residuo de predicción LPC
IS	1	$j$		Mejor índice de código cifrado de forma de siete bits
KP	1	$p$		Periodo de tono de la trama actual
KP1	1	$\hat{p}$	50	Periodo de tono de la trama anterior
PN	1 a IDIM	$p(n)$		Vector de correlación para la búsqueda de código cifrado
PTAP	1	$\beta$		Derivación del predictor de tono calculada por el bloque 83
R	1 a NR+1 <sup>a)</sup>			Coefficientes de autocorrelación
RC	1 a NR <sup>a)</sup>			Coefficientes de reflexión, también como ordenamiento transitorio
RCTMP	1 a LPC			Memoria intermedia temporal para los coeficientes de reflexión
REXP	1 a LPC+1		0,0,...,0	Parte recursiva de autocorrelación, filtro de síntesis
REXPWG	1 a LPCLG+1		0,0,...,0	Parte recursiva de autocorrelación, predictor de ganancia logarítmica
REXPW	1 a LPCW+1		0,0,...,0	Parte recursiva de autocorrelación, filtro de ponderación
RTMP	1 a LPC+1			Memoria intermedia temporal para los coeficientes de autocorrelación
S	1 a IDIM	$s(n)$	0,0,...,0	Vector de señal vocal de entrada MIC uniforme
SB	1 a 105		0,0,...,0	Memoria intermedia para la señal vocal cuantificada anteriormente
SBLG	1 a 34		0,0,...,0	Memoria intermedia para la ganancia logarítmica anterior
SBW	1 a 60		0,0,...,0	Memoria intermedia para la señal vocal de entrada anterior
SCALE	1			Factor de escala de posfiltro no filtrado

CUADRO 2/G.728 (cont.)

Nombre	Gama de los índices del ordenamiento	Símbolo equivalente	Valor inicial	Descripción
SCALEFIL	1		1	Factor de escala de posfiltro filtrado en paso bajo
SD	1 a IDIM	$s_d(k)$		Memoria intermedia de señal vocal decodificada
SPF	1 a IDIM			Vector de señal vocal posfiltrada
SPFPCFV	1 a 11	$SPFPCF^{i-1}$	Anexo C	Vector de control de polos del posfiltro de corto plazo
SPFZCFV	1 a 11	$SPFZCF^{i-1}$	Anexo C	Vector de control de ceros del posfiltro de corto plazo
SO	1	$s_o(k)$		Muestra de señal vocal de entrada MIC de ley A o $\mu$
SU	1	$s_u(k)$		Muestra de señal vocal de entrada MIC uniforme
ST	-239 a IDIM	$s_d(n)$	0,0,...,0	Vector de señal vocal cuantificada
STATELPC	1 a LPC		0,0,...,0	Memoria de filtro de síntesis
STLPCI	1 a 10		0,0,...,0	Memoria de filtro inverso LPC
STLPF	1 a 3		0, 0, 0	Memoria de filtro de paso bajo de 1 kHz
STMP	1 a 4*IDIM		0,0,...,0	Memoria intermedia para la ventana híbrida del filtro de ponderación perceptual
STPFIR	1 a 10		0,0,...,0	Memoria del posfiltro de corto plazo, sección todos ceros
STPFIIR	10		0,0,...,0	Memoria del posfiltro de corto plazo, sección todos polos
SUMFIL	1			Suma de los valores absolutos de la señal vocal posfiltrada
SUMUNFIL	1			Suma de los valores absolutos de la señal vocal decodificada
SW	1 a IDIM	$v(n)$		Vector de señal vocal ponderada perceptualmente
TARGET	1 a IDIM	$\hat{x}(n), x(n)$		Vector objetivo VQ (con escalamiento de ganancia)
TEMP	1 a IDIM			Ordenamiento transitorio para espacio de trabajo temporal
TILTZ	1	$\mu$	0	Coefficientes de compensación de inclinación del posfiltro de corto plazo
WFIR	1 a LPCW		0,0,...,0	Memoria del filtro de ponderación 4, parte todos ceros
WIIR	1 a LPCW		0,0,...,0	Memoria del filtro de ponderación 4, parte todos polos
WNR	1 a 105	$w_m(k)$	Anexo A	Función de ventana para el filtro de síntesis
WNRLG	1 a 34	$w_m(k)$	Anexo A	Función de ventana para el predictor de ganancia logarítmica
WNRW	1 a 60	$w_m(k)$	Anexo A	Función de ventana para el filtro de ponderación

CUADRO 2/G.728 (cont.)

Nombre	Gama de los índices del ordenamiento	Símbolo equivalente	Valor inicial	Descripción
WPCFV	1 a LPCW+1	$\gamma^{i-1,2}$	Anexo C	Vector de control de polos del filtro de ponderación perceptual
WS	1 a 105			Ordenamiento «espacio de trabajo» para variables intermedias
WZCFV	1 a LPCW+1	$\gamma^{i-1,1}$	Anexo C	Vector de control de ceros del filtro de ponderación perceptual
Y	1 a IDIM*NCWD	$y_j$	Anexo B	Ordenamiento de códigos cifrados de forma
Y2	1 a NCWD	$E_j$	Energía de $y_j$	Energía del vector de código de forma convolucionado
YN	1 a IDIM	$y(n)$		Vector de excitación cuantificada
ZIRWFIR	1 a LPCW		0,0,...,0	Memoria del filtro de ponderación 10, parte todos ceros
ZIRWIIR	1 a LPCW		0,0,...,0	Memoria del filtro de ponderación 10, parte todos polos

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

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

Nota – El asterisco (\*) representa la multiplicación aritmética.

### 5.3 Conversión del formato MIC de entrada (bloque 1)

Entrada: SO

Salida: SU

Función: Convertir una muestra de entrada de ley A o  $\mu$  de 16 bits lineal en una muestra MIC uniforme.

Dado que el funcionamiento de este bloque está completamente definido en la Recomendación G.721 o G.711, no lo repetiremos aquí. No obstante, puede necesitarse cierto grado de normalización para conformarse a la especificación que se da en esta descripción de una gama de entrada de  $-4095$  a  $+4095$  (véase el § 3.1.1).

### 5.4 Memoria intermedia de vector (bloque 2)

Entrada: SU

Salida: S

Función: Poner en memoria intermedia cinco muestras consecutivas de señal vocal MIC uniforme para formar un solo vector de señal vocal de cinco dimensiones.

5.5 *Adaptador para el filtro de ponderación perceptual (bloque 3, figura 4a)/G.728)*

A continuación se describen detalladamente los tres bloques de la figura 4a)/G.728 (36, 37 y 38).

MÓDULO DE VENTANIZACIÓN HÍBRIDA (bloque 36)

Entrada: STMP

Salida: R

Función: Aplicar la ventana híbrida a la señal vocal de entrada y calcular los coeficientes de autocorrelación.

A continuación se describe el funcionamiento de este módulo utilizando un estilo de tipo Fortran, en el que los límites de los bucles se indican mediante sangrados y los comentarios se indican a la derecha del signo «|». El algoritmo siguiente ha de utilizarse una vez en cada ciclo de adaptación (20 muestras). El ordenamiento STMP contiene cuatro vectores consecutivos de señal vocal de entrada, hasta el segundo vector de señal vocal del ciclo de adaptación actual. Es decir, STMP(1) a STMP(5) es el tercer vector de señal vocal de entrada del ciclo de adaptación anterior (cero inicialmente), STMP(6) a STMP(10) es el cuarto vector de señal vocal de entrada del ciclo de adaptación anterior (cero inicialmente), STMP(11) a STMP(15) es el primer vector de señal vocal de entrada del ciclo de adaptación actual, y STMP (16) a STMP(20) es el segundo vector de señal vocal de entrada del ciclo de adaptación actual.

N1 = LPCW + NFRSZ	Calcular algunas constantes
N2 = LPCW + NONRW	(pueden calcularse y almacenarse
N3 = LPCW + NFRSZ + NONRW	previamente en memoria)
For N = 1,2,...,N2, do the next line	
SBW(N) = SBW(N + NFRSZ)	Cambiar la memoria intermedia
	de la señal anterior
For N = 1,2,...,NFRSZ, do the next line	
SBW(N2 + N) = STMP(N)	Introducir la nueva señal
	SBW(N3) es la muestra más reciente
K = 1	
For N = N3,N3 - 1,...,3,2,1, do the next two lines	
WS(N) = SBW(N) * WNRW(K)	Multiplicar la función de ventana
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	Actualizar la componente recursiva
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)	Añadir la componente no recursiva
R(1) = R(1) * WNCF	Corrección por ruido blanco

MÓDULO DE RECURSIÓN DE LEVINSON-DURBIN (bloque 37)

Entrada: R (salida del bloque 36)

Salida: AWZTMP

Función: Convertir los coeficientes de autocorrelación en coeficientes de predictor lineal.

Este bloque se ejecuta una vez por ciclo de adaptación de cuatro vectores. Se hace para ICOUNT = 3, después de que ha terminado el procesamiento del bloque 36. Como la recursión de Levinson-Durbin es bien conocida, se da el algoritmo sin explicaciones.

```

If R(LPCW + 1) = 0, go to LABEL          | Saltar si cero
If R(1) ≤ 0, go to LABEL                 | Saltar si señal cero
RC(1) = R(2)/R(1)                        |
AWZTMP(1) = 1                            |
AWZTMP(2) = RC(1)                        | Predictor de primer orden
ALPHA = R(1) + R(2) * RC(1)              |
If ALPHA ≤ 0, go to LABEL                | Abortar si hay desajuste

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                    | Coeficientes de reflexión
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)
    AWZTMP(IP) = AT                      | Coeficientes del predictor

AWZTMP(MINC + 1) = RC(MINC)              |
ALPHA = ALPHA + RC(MINC) * SUM            | Energía residual de predicción
If Alpha ≤ 0, go to LABEL                 | Abortar si hay desajuste

Repeat the above for the next MINC

Exit this program                         | El programa termina normalmente
                                           | si la ejecución llega hasta aquí

```

LABEL: Si el programa llega aquí, se ha producido desajuste; entonces, saltar el bloque 38, no actualizar los coeficientes del filtro de ponderación (es decir, utilizar los coeficientes del filtro de ponderación del ciclo de adaptación anterior).

CALCULADOR DE LOS COEFICIENTES DEL FILTRO  
DE PONDERACIÓN (bloque 38)

Entrada: AWZTMP

Salidas: AWZ, AWP

Función: Calcular los coeficientes del filtro de ponderación perceptual a partir de los coeficientes del predictor lineal para la señal vocal de entrada.

Este bloque es ejecutado una vez por ciclo de adaptación. Se hace para ICOUNT = 3, después de que ha terminado el procedimiento del bloque 37.

For I = 2,3,...,LPCW + 1, do the next line  
AWP(I) = WPCFV(I) \* AWZTMP(I) | Coeficientes del denominador

For I = 2,3,...,LPCW + 1, do the next line  
AWZ(I) = WZCFV(I) \* AWZTMP(I) | Coeficientes del numerador

---

5.6 *Adaptador del filtro de síntesis hacia atrás (bloque 23, figura 5/G.728)*

A continuación se especifican los tres bloques de la figura 5/G.728 (49, 50 y 51).

MÓDULO DE VENTANIZACIÓN HÍBRIDA (bloque 49)

Entrada: STTMP

Salida: RTMP

Función: Aplicar la ventana híbrida a la señal vocal cuantificada y calcular los coeficientes de autocorrelación.

El funcionamiento de este bloque es esencialmente igual al del bloque 36, salvo en lo concerniente a algunas sustituciones de parámetros y variables, y al instante de muestreo en que se obtienen los coeficientes de autocorrelación. Como se describe en el § 3, los coeficientes de autocorrelación se calculan sobre la base de los vectores de señal vocal cuantificada hasta el último vector del ciclo de adaptación de cuatro vectores anterior. En otras palabras, los coeficientes de autocorrelación empleados en el ciclo de adaptación actual se basan en la información contenida en la señal vocal cuantificada hasta la última muestra (vigésima) del ciclo de adaptación anterior. (De hecho, esta es la manera en que definimos el ciclo de adaptación.) El ordenamiento STTMP contiene los cuatro vectores de señal vocal cuantificada del ciclo de adaptación anterior.

<p>N1 = LPC + NFRSZ                  N2 = LPC + NONR                  N3 = LPC + NFRSZ + NONR</p>	<p>  Calcular algunas constantes (pueden ser                    calculadas y almacenadas previamente                    en memoria</p>
<p>For N = 1,2,...,N2, do the next line                  SB(N) = SB(N + NFRSZ)</p>	<p>  Cambiar la memoria intermedia                    de la señal anterior</p>
<p>For N = 1,2,...,NFRSZ, do the next line                  SB(N2 + N) = STTMP(N)</p>	<p>  Introducir la nueva señal                    SB(N3) es la muestra más reciente</p>
<p>K = 1                  For N = N3,N3 - 1,...,3,2,1, do the next two lines                  WS(N) = SB(N) * WNR(K)                  K = K + 1</p>	<p>  Multiplicar la función de ventana</p>
<p>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</p>	<p>  Actualizar la componente recursiva</p>
<p>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)</p>	<p>  Añadir la componente no recursiva</p>
<p>RTMP(1) = RTMP(1) * WNCF</p>	<p>  Corrección por ruido blanco</p>

---

MÓDULO DE RECURSIÓN DE LEVINSON-DURBIN (bloque 50)

Entrada: RTMP

Salida: ATMP

Función: Convertir los coeficientes de autocorrelación en coeficientes del filtro de síntesis.

El funcionamiento de este bloque es exactamente igual al del bloque 37, salvo en lo que atañe a algunas sustituciones de parámetros y variables. Sin embargo, debe tenerse especial cuidado en su realización. Como se describe en el § 3, si bien el ordenamiento de autocorrelación RTMP está disponible en el primer vector de cada ciclo de adaptación, las actualizaciones reales de los coeficientes del filtro de síntesis no tendrán lugar hasta el tercer vector. Este retardo intencional de las actualizaciones permite al soporte físico, que funciona en tiempo real, ejecutar este módulo durante los tres primeros vectores de cada ciclo de adaptación. En la ejecución de este módulo, durante los dos primeros vectores de cada ciclo, se sigue utilizando el antiguo conjunto de coeficientes del filtro de síntesis (ordenamiento «A») obtenido en el ciclo anterior. Esta es la razón por la cual es necesario mantener un ordenamiento separado ATMP, a fin de evitar la sobreescritura en el antiguo ordenamiento «A». De manera similar, se utiliza RTMP, RCTMP, ALPHATMP, etc., para no causar interferencia a otros módulos de recursión de Levinson-Durbin (bloques 37 y 44).

If RTMP(LPC + 1) = 0, go to LABEL	Saltar si cero
If RTMP(1) ≤ 0, go to LABEL	Saltar si señal cero
RCTMP(1) = -RTMP(2)/RTMP(1)	
ATMP(1) = 1	
ATMP(2) = RCTMP(1)	Predictor de primer orden
ALPHATMP = RTMP(1) + RTMP(2) * RCTMP(1)	
If ALPHATMP ≤ 0, go to LABEL	Abortar si hay desajuste
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	Coeficientes de reflexión
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	Actualizar los coeficientes del predictor
ATMP(MINC + 1) = RCTMP(MINC)	
ALPHATMP = ALPHATMP + RCTMP(MINC) * SUM	Energía residual de predicción
If ALPHATMP ≤ 0, go to LABEL	Abortar si hay desajuste
Repeat the above for the next MINC	
Exit this program	Si la ejecución llega hasta este punto, la recursión ha sido completada normalmente
LABEL:	Si el programa llega a este punto, se ha producido desajuste; entonces, saltar el bloque 51, no actualizar los coeficientes del filtro de síntesis (es decir, utilizar los coeficientes del filtro de síntesis del ciclo de adaptación anterior).

## MÓDULO DE AMPLIACIÓN DE ANCHURA DE BANDA (bloque 51)

Entrada: ATMP

Salida: A

Función: Escalar los coeficientes del filtro de síntesis para ampliar las anchuras de banda de los picos espectrales.

Este bloque es ejecutado una vez por ciclo de adaptación. Se hace después de que ha terminado el procesamiento del bloque 50 y antes de la ejecución de los bloques 9 y 10, para ICOUNT = 3. Cuando se termina la ejecución de este módulo e ICOUNT = 3, copiamos el ordenamiento ATMP en el ordenamiento «A» a fin de actualizar los coeficientes del filtro.

```
For I = 1,3,...,LPC + 1, do the next line      |
  ATMP(I) = FACV(I) * ATMP(I)                | Escalar coeficientes
Wait until ICOUNT = 3, then                  |
For I = 2,3,...,LPC + 1, do the next line    |
  A(I) = ATMP(I)                             | Actualizar los coeficientes en
                                              | el tercer vector de cada ciclo
```

---

### 5.7 Adaptador de ganancia vectorial hacia atrás (bloque 20, figura 6/G.728)

A continuación se especifican los bloques de la figura 6/G.728. Por motivos de eficiencia, algunos bloques se describen juntos, como un solo bloque (se muestran separados en la figura 6/G.728 únicamente para explicar el concepto). Todos los bloques de la figura 6/G.728 se ejecutan una vez por vector de señal vocal, excepto los bloques 43, 44 y 45, que se ejecutan sólo cuando ICOUNT = 2.

### RETARDO DE UN VECTOR, CALCULADOR RMS Y CALCULADOR LOGARÍTMICO (bloques 67, 39 y 40)

Entrada: ET

Salida: ETRMS

Función: Calcular el nivel en dB del valor cuadrático medio (RMS) del vector de excitación con escalamiento de ganancia anterior.

Cuando esos tres bloques son ejecutados (lo que se hace antes de la búsqueda de código cifrado VQ), el ordenamiento ET contiene el vector de excitación con escalamiento de ganancia determinado para el vector de señal vocal anterior. Por consiguiente, la unidad de retardo de un vector (bloque 67) es ejecutada automáticamente. (Aparece en la figura 6/G.728 únicamente para mayor claridad.) Dado que el calculador logarítmico sigue inmediatamente al calculador RMS, la operación de raíz cuadrada en el calculador RMS puede realizarse como una operación de división por dos a la salida del calculador logarítmico. Por consiguiente, la salida del calculador logarítmico (el valor en dB) es  $10 * \log_{10}$  (energía de ET/IDIM). Para evitar el desbordamiento del valor logarítmico cuando  $ET = 0$  (después de la iniciación o reiniciación del sistema), el argumento de la operación logarítmica se recorta a uno si es demasiado pequeño. Asimismo, observamos que ETRMS se conserva usualmente en un acumulador, dado que es un valor temporal que es procesado inmediatamente en el bloque 42.

```
ETRMS = ET(1) * ET(1)                        |
For K = 2,3,...,IDIM, do the next line        |
  ETRMS = ETRMS + ET(K) * ET(K)             | Calcular la energía de ET
ETRMS = ETRMS * DIMINV                       | Dividir por IDIM
If ETRMS < 1, set ETRMS = 1                  | Recortar para evitar el
ETRMS = 10 * log10 (ETRMS)                   | desbordamiento logarítmico
ETRMS = 10 * log10 (ETRMS)                   | Calcular el valor en dB
```

---

SUSTRACTOR DE DESPLAZAMIENTO DE GANANCIA LOGARÍTMICA  
(bloque 42)

Entradas: ETRMS, GOFF  
Salida: GSTATE(1)  
Función: Sustraer de la salida del bloque 40 (nivel de ganancia en dB) el valor del desplazamiento de la ganancia logarítmica conservado en el bloque 41.

GSTATE(1) = ETRMS – GOFF

---

MÓDULO DE VENTANIZACIÓN HÍBRIDA (bloque 43)

Entrada: GTMP  
Salida: R  
Función: Aplicar la ventana híbrida a la secuencia de ganancia logarítmica con sustracción del desplazamiento y calcular los coeficientes de autocorrelación.

El funcionamiento de este bloque es muy similar al del bloque 36, salvo en lo concerniente a algunas sustituciones de parámetros y variables, y al instante de muestreo en que se obtienen los coeficientes de autocorrelación.

Una diferencia importante entre el bloque 36 y este bloque es que se introducen únicamente cuatro (en vez de 20) muestras de ganancia en este bloque cada vez que es ejecutado.

Los coeficientes del predictor de ganancia logarítmica son actualizados en el segundo vector de cada ciclo de adaptación. El ordenamiento GTMP contiene cuatro valores de ganancia logarítmica a los que se ha sustraído el desplazamiento, empezando por la ganancia logarítmica del segundo vector del ciclo de adaptación anterior (GTMP(1)), hasta la ganancia logarítmica del primer vector del ciclo de adaptación actual (GTMP(4)), que es el valor más reciente.

N1 = LPCLG + NUPDATE | Calcular algunas constantes (pueden calcularse  
N2 = LPCLG + NONRLG | y almacenarse previamente en memoria)  
N3 = LPCLG + NUPDATE + NONRLG

For N = 1,2,...,N2, do the next line | Cambiar la antigua memoria intermedia de señal  
SBLG(N) = SBLG(N + NUPDATE)  
For N = 1,2,...,NUPDATE, do the next line | Introducir la nueva señal  
SBLG(N2 + N) = GTMP(N) | SBLG(N3) es la muestra más reciente

K = 1  
For N = N3, N3 - 1, ..., 3, 2, 1, do the next two lines | Multiplicar la función de ventana  
WS(N) = SBLG(N) \* WNRLG(K)  
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)  
REXPLG(I) = (3/4) \* REXPLG(I) + TMP | Actualizar la componente recursiva

For I = 1,2,...,LPCLG + 1, do the next three lines  
R(I) = REXPLG(I)  
For N = N1 + 1, N1 + 2, ..., N3, do the next line  
R(I) = R(I) + WS(N) \* WS(N + 1 - I) | Añadir la componente no recursiva

R(1) = R(1) \* WNCF | Corrección por ruido blanco

---

### MÓDULO DE RECURSIÓN DE LEVINSON-DURBIN (bloque 44)

Entrada: R (salida del bloque 43)

Salida: GPTMP

Función: Convertir los coeficientes de autocorrelación en los coeficientes del predictor de ganancia logarítmica.

El funcionamiento de este bloque es exactamente igual al del bloque 37, excepto en lo concerniente a las sustituciones de los parámetros y las variables indicados a continuación: sustituir LPCW por LPCLG, y AWZ por GP. Este bloque es ejecutado únicamente cuando ICOUNT = 2, después de la ejecución del bloque 43. Obsérvese que, como primera etapa, se comprueba el valor de  $R(LPCLG + 1)$ . Si es cero, saltamos los bloques 44 y 45 sin actualizar los coeficientes del predictor de ganancia logarítmica. (Es decir, seguimos utilizando los antiguos coeficientes del predictor de ganancia logarítmica determinados en el ciclo de adaptación anterior.) Este procedimiento especial tiene por objeto evitar el pequeño mal funcionamiento que podría producirse en ausencia de dicho procedimiento inmediatamente después de la iniciación o reiniciación del sistema. En el caso en que la matriz esté desajustada, saltamos también el bloque 45 y empleamos los antiguos valores.

---

### MÓDULO DE AMPLIACIÓN DE ANCHURA DE BANDA (bloque 45)

Entrada: GPTMP

Salida: GP

Función: Escalar los coeficientes del predictor de ganancia logarítmica para ampliar las anchuras logarítmicas de los picos espectrales.

Este bloque es ejecutado sólo cuando ICOUNT = 2, después de la ejecución del bloque 44.

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

---

### PREDICTOR LINEAL DE GANANCIA LOGARÍTMICA (bloque 46)

Entradas: GP, GSTATE

Salida: GAIN

Función: Predecir el valor actual de la ganancia logarítmica con sustracción del desplazamiento.

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)

---

### SUMADOR DE DESPLAZAMIENTO DE GANANCIA LOGARÍTMICA (entre los bloques 46 y 47)

Entradas: GAIN, GOFF

Salida: GAIN

Función: Sumar de nuevo el valor del desplazamiento de la ganancia logarítmica a la salida del predictor de ganancia logarítmica.

GAIN = GAIN + GOFF

## LIMITADOR DE GANANCIA LOGARÍTMICA (bloque 47)

Entrada: GAIN

Salida: GAIN

Función: Limitar la gama de la ganancia logarítmica predicha.

If GAIN < 0, set GAIN = 0  
If GAIN > 60, set GAIN = 60

| Corresponde a la ganancia lineal uno  
| Corresponde a la ganancia lineal 1000

---

## CALCULADOR LOGARÍTMICO INVERSO (bloque 48)

Entrada: GAIN

Salida: GAIN

Función: Convertir la ganancia logarítmica predicha (en dB) nuevamente al dominio lineal.

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

---

### 5.8 Filtro de ponderación perceptual

#### FILTRO DE PONDERACIÓN PERCEPTUAL (bloque 4)

Entradas: S, AWZ, AWP

Salida: SW

Función: Filtrar el vector de señal vocal de entrada para obtener la ponderación perceptual.

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)

|  
| Parte «todos ceros» del filtro

SW(K) = SW(K) + WFIR(1) \* AWZ(2)

WFIR(1) = S(K)

|  
| Tratar la última de manera diferente

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

SW(K) = SW(K) - WIIR(J) \* AWP(J + 1)

WIIR(J) = WIIR(J - 1)

|  
| Parte «todos polos» del filtro

SW(K) = SW(K) - WIIR(1) \* AWP(2)

WIIR(1) = SW(K)

|  
| Tratar la última de manera diferente

Repeat the above for the next K

---

## 5.9 Cálculo del vector de respuesta a entrada cero

En el § 3.5 se explica cómo se calcula un «vector de respuesta a entrada cero»  $r(n)$  en los bloques 9 y 10. A continuación se especifica el funcionamiento de estos dos bloques durante esta fase. Su funcionamiento durante la «fase de actualización de la memoria» se describirá más adelante.

### FILTRO DE SÍNTESIS (bloque 9) DURANTE EL CÁLCULO DE LA RESPUESTA A ENTRADA CERO

Entradas: A, STATELPC

Salida: TEMP

Función: Calcular el vector de respuesta a entrada cero del filtro de síntesis.

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)

| Multiplicar-adicionar

STATELPC(J) = STATELPC(J - 1)

| Cambio de memoria

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

|

STATELPC(1) = TEMP(K)

| Tratar la última de manera diferente

Repeat the above for the next K

---

### FILTRO DE PONDERACIÓN PERCEPTUAL (bloque 10) DURANTE EL CÁLCULO DE LA RESPUESTA A ENTRADA CERO

Entradas: AWZ, AWP, ZIRWFIR, ZIRWIIR, TEMP calculados antes

Salida: ZIR

Función: Calcular el vector de respuesta a entrada cero del filtro de ponderación perceptual.

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)

|

ZIRWFIR(J) = ZIRWFIR(J - 1)

| Parte «todos ceros» del filtro

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

|

ZIRWFIR(1) = TMP

| Tratar la última de manera diferente

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

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

|

ZIRWIIR(J) = ZIRWIIR(J - 1)

| Parte «todos polos» del filtro

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

|

ZIRWIIR(1) = ZIR(K)

| Tratar la última de manera diferente

Repeat the above for the next K

---

5.10 *Cálculo del vector objetivo VQ*

CÁLCULO DEL VECTOR OBJETIVO VQ (bloque 11)

Entradas: SW, ZIR

Salida: TARGET

Función: Sustraer el vector de respuesta a entrada cero del vector de la señal vocal ponderada.

Nota –  $ZIR(K) = ZIRWIIR(IDIM + 1 - K)$  procede del bloque 10 anterior. No requiere una ubicación separada en memoria.

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

---

5.11 *Módulo de búsqueda de código cifrado (bloque 24)*

A continuación se especifican los siete bloques que constituyen el módulo de búsqueda de código cifrado (bloque 24). Una vez más, algunos bloques se describen como un solo bloque de comodidad y eficiencia de realización. Los bloques 12, 14 y 15 son ejecutados una vez por cada ciclo de adaptación cuando ICOUNT = 3, mientras que los demás bloques son ejecutados una vez por cada vector de señal vocal.

CALCULADOR DE VECTOR DE RESPUESTA A LOS IMPULSOS (bloque 12)

Entradas: A, AWZ, AWP

Salida: H

Función: Calcular el vector de respuesta a los impulsos del filtro de síntesis y del filtro de ponderación perceptual colocados en cascada.

Este bloque es ejecutado cuando ICOUNT = 3 y después de que se completa la ejecución de los bloques 23 y 3 (es decir, cuando están listos los nuevos conjuntos de coeficientes A, AWZ, AWP).

TEMP(1) = 1 | TEMP = memoria del filtro de síntesis  
RC(1) = 1 | RC = W(z) memoria de la parte «todos polos»  
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) | Filtrado  
A1 = A1 + AWZ(I) \* TEMP(I) |  
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 | Obtener h(n) invirtiendo el  
For K = 1,2,...,IDIM, do the next line | orden de la memoria de la  
H(K) = RC(ITMP – K) | sección «todos polos» de W(z)

---

MÓDULO DE CONVOLUCIÓN DE LOS VECTORES DE CÓDIGO DE FORMA  
Y CALCULADOR DE LA TABLA DE ENERGÍA (bloques 14 y 15)

Entradas: H, Y

Salida: Y2

Función: Convolver cada vector de código de forma con la respuesta a los impulsos obtenida en el bloque 12, luego calcular y almacenar la energía del vector resultante.

Este bloque es ejecutado también cuando ICOUNT = 3 después de completada la ejecución del bloque 12.

```
For J = 1,2,...,NCWD, do the following: | Un vector de código por bucle
  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) | Convolución
  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) | Calcular la energía
Repeat the above for the next J
```

---

NORMALIZACIÓN DEL VECTOR OBJETIVO VQ (bloque 16)

Entradas: TARGET, GAIN

Salida: TARGET

Función: Normalizar el vector objetivo VQ utilizando la ganancia de excitación predicha.

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

---

MÓDULO DE CONVOLUCIÓN CON INVERSIÓN DEL TIEMPO (bloque 13)

Entradas: H, TARGET (salida del bloque 16)

Salida: PN

Función: Realizar la convolución con inversión del tiempo del vector de respuesta a los impulsos y del vector objetivo VQ normalizado (para obtener el vector  $p(n)$ ).

*Nota* – El vector PN puede conservarse en una memoria temporal.

```
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

---

CALCULADOR DE ERROR Y SELECTOR DEL MEJOR ÍNDICE  
DE CÓDIGO CIFRADO (bloques 17 y 18)

Entradas: PN, Y, Y2, GB, G2, GSQ

Salidas: IG, IS, ICHAN

Función: Buscar en la tabla de códigos cifrados de ganancia y en la tabla de códigos cifrados de forma para determinar el mejor índice de código cifrado de ganancia y el mejor índice de código cifrado de forma, y combinarlos para obtener el mejor índice de código cifrado de diez bits.

*Nota* – Por lo general, la variable COR utilizada más adelante se conserva en un acumulador, en vez de estar almacenada en memoria. Las variables IDXG y J pueden conservarse en registros temporales, mientras que IG e IS pueden conservarse en memoria.

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

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

|  
| Calcular el producto interno  $P_j$

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

| Mejor ganancia positiva hallada

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

| Mejor ganancia negativa hallada

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

| Calcular la distorsión  $\hat{D}$

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

$DISTM = D$

$IG = IDXG$

$IS = J$

| Guardar la distorsión más baja y los mejores  
| índices de código cifrado hallados hasta ahora  
|

Repeat the above indented section for the next K

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

| Concatenar los índices de códigos  
| cifrados de forma y ganancia

Transmit ICHAN through the communication channel.

Para la transmisión en serie del tren de bits, se transmitirá en primer lugar el bit más significativo de ICHAN. Si ICHAN está representado por la palabra de diez bits  $b_9b_8b_7b_6b_5b_4b_3b_2b_1b_0$ , el orden de transmisión de los bits será  $b_9$ , a continuación  $b_8$ , luego  $b_7$  y al final  $b_0$  ( $b_9$  es el bit más significativo).

### 5.12 Decodificador simulado (bloque 8)

Los bloques 20 y 23 han sido descritos anteriormente. A continuación se especifican los bloques 19, 21 y 22.

#### CÓDIGOS CIFRADOS VQ DE EXCITACIÓN (bloque 19)

Entradas: IG, IS

Salida: YN

Función: Examinar la tabla para extraer el mejor vector de código de forma y la mejor ganancia, y multiplicarlos luego para obtener el vector de excitación cuantificada.

$NN = (IS-1) * IDIM$

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

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

---

#### UNIDAD DE ESCALAMIENTO DE GANANCIA (bloque 21)

Entradas: GAIN, YN

Salida: ET

Función: Multiplicar el vector de excitación cuantificada por la ganancia de excitación.

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

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

---

#### FILTRO DE SÍNTESIS (bloque 22)

Entradas: ET, A

Salida: ST

Función: Filtrar el vector de excitación con escalamiento de ganancia para obtener el vector de señal vocal cuantificada.

Como se explica en el § 3, este bloque puede omitirse, y el vector de señal vocal cuantificada puede obtenerse como un subproducto del procedimiento de actualización de la memoria que se describe a continuación. No obstante, si se desea realizar este bloque de todas maneras, debe utilizarse un conjunto separado de memoria de filtro (y no STATELPC) para este filtro de síntesis «todos polos».

---

5.13 *Actualización de la memoria del filtro de los bloques 9 y 10*

La siguiente descripción de los procedimientos de actualización de la memoria del filtro de los bloques 9 y 10 supone que el vector de señal vocal cuantificada ST se obtiene como subproducto de las actualizaciones de la memoria. Para protegerse contra la posible sobrecarga de los niveles de señal, se incorpora un limitador de magnitud, de manera que la memoria del filtro recorta en MAX y MIN, donde MAX y MIN son, respectivamente, los niveles de saturación positiva y negativa de la MIC de ley A o  $\mu$ , según la que se utilice.

ACTUALIZACIÓN DE LA MEMORIA DEL FILTRO (bloques 9 y 10)

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

Salidas: ST, STATELPC, ZIRWFIR, ZIRWIIR

Función: Actualizar la memoria del filtro de los bloques 9 y 10, y obtener también el vector de señal vocal cuantificada.

```

ZIRWFIR(1) = ET(1) | ZIRWFIR es ahora un ordenamiento transitorio
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) | Calcular las respuestas de estado
    A2 = A2 - AWP(I) * TEMP(I) | cero en varias etapas del filtro
                                | en cascada
  ZIRWFIR(1) = A0 |
  TEMP(1) = A0 + A1 + A2 |

Repeat the above indented section for the next K | Actualizar ahora la memoria del filtro sumando
                                                    | las respuestas de estado cero a las
                                                    | respuestas de entrada cero

For K = 1,2,...,IDIM, do the next four lines
  STATELPC(K) = STATELPC(K) + ZIRWFIR(K)
  If STATELPC(K) > MAX, set STATELPC(K) = MAX | Limitar la gama
  If STATELPC(K) < MIN, set STATELPC(K) = MIN |
  ZIRWIIR(K) = ZIRWIIR(K) + TEMP(K)

For I = 1,2,...,LPCW, do the next line | Asignar a ZIRWFIR el valor
  ZIRWFIR(I) = STATELPC(I) | correcto

I = IDIM + 1
For K = 1,2,...,IDIM, do the next line | Obtener la señal vocal cuantificada invirtiendo el
  ST(K) = STATELPC(I - K) | orden de la memoria del filtro de síntesis

```

5.14 *Decodificador (figura 3/G.728)*

A continuación se describen los bloques del decodificador (figura 3/G.728). Exceptuando el bloque de conversión de formato MIC de salida, todos los bloques son exactamente iguales a los bloques del decodificador simulado (bloque 8) de la figura 2/G.728.

El decodificador emplea únicamente un subconjunto de las variables que figuran en el cuadro 2/G.728. Si han de realizarse un decodificador y un codificador en una sola microplaqueta DSP, deben darse nombres diferentes a las variables del decodificador para evitar la sobreescritura en las variables utilizadas en el bloque decodificador simulado del codificador. Por ejemplo, para nombrar las variables del decodificador, podemos añadir un prefijo «d» a los nombres de las variables correspondientes del cuadro 2/G.728. Si ha de realizarse un decodificador en una unidad autónoma, independiente de un codificador, no es necesario cambiar los nombres de las variables.

En la descripción siguiente se supone un decodificador autónomo. Una vez más, los bloques son ejecutados en el mismo orden en que se describen a continuación.

#### ADAPTADOR DE FILTRO DE SÍNTESIS HACIA ATRÁS EN EL DECODIFICADOR (bloque 33)

Entrada: ST

Salida: A

Función: Generar los coeficientes del filtro de síntesis periódicamente a partir de la señal vocal decodificada previamente.

El funcionamiento de este bloque es exactamente igual al del bloque 23 del codificador.

---

#### ADAPTADOR DE GANANCIA VECTORIAL HACIA ATRÁS EN EL DECODIFICADOR (bloque 30)

Entrada: ET

Salida: GAIN

Función: Generar la ganancia de excitación a partir de los vectores de excitación con escalamiento de ganancia anteriores.

El funcionamiento de este bloque es exactamente igual al del bloque 20 del codificador.

---

#### CÓDIGOS CIFRADOS VQ DE EXCITACIÓN DEL DECODIFICADOR (bloque 29)

Entrada: ICHAN

Salida: YN

Función: Decodificar el mejor índice de código cifrado recibido (índice de canal) para obtener el vector de excitación.

Este bloque extrae primero el índice IG de la tabla de códigos cifrados de ganancia de tres bits y el índice IS de la tabla de códigos cifrados de forma de siete bits del índice de canal de diez bits recibido. Después, el resto de la operación es exactamente igual a la del bloque 19 del codificador.

ITMP = integer part of (ICHAN / NG)

| Decodificar (IS - 1)

IG = ICHAN - ITMP \* NG + 1

| Decodificar IG

NN = ITMP \* IDIM

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

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

---

UNIDAD DE ESCALAMIENTO DE GANANCIA DEL DECODIFICADOR  
(bloque 31)

Entradas: GAIN, YN

Salida: ET

Función: Multiplicar el vector de excitación por la ganancia de excitación.

El funcionamiento de este bloque es exactamente igual al del bloque 21 del codificador.

---

FILTRO DE SÍNTESIS DEL DECODIFICADOR (bloque 32)

Entradas: ET, A, STATELPC

Salida: ST

Función: Filtrar el vector de excitación con escalamiento de ganancia para obtener el vector de señal vocal decodificada.

Este bloque puede realizarse como un filtro «todos polos». Sin embargo, como se menciona en el § 4.3, si el codificador obtiene la señal vocal cuantificada como subproducto de la actualización de la memoria del filtro (para ahorrar cálculos), y si la acumulación potencial de errores de redondeo constituye un problema, este bloque debe calcular la señal vocal decodificada exactamente de la misma manera que lo hace el bloque decodificador simulado del codificador. Es decir, el vector de señal vocal decodificada debe calcularse como la suma del vector de respuesta de entrada cero y del vector de respuesta de estado cero del filtro de síntesis. Esto puede hacerse mediante el procedimiento siguiente:

```
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)      | Respuesta de entrada cero
    STATELPC(J) = STATELPC(J - 1)
  TEMP(K) = TEMP(K) - STATELPC(1) * A(2)           |
  STATELPC(1) = TEMP(K)                            | Tratar la última de manera diferente

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)                        | Calcular la respuesta de estado cero
  TEMP(1) = A0

Repeat the above for the next K

| Actualizar la memoria del filtro sumando
| las respuestas de estado cero
| a las respuestas de entrada cero

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      | Limitar la gama
  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)                          | Obtener la señal vocal cuantificada invirtiendo
| el orden de la memoria del filtro de síntesis
```

---

**FILTRO INVERSO LPC DE DÉCIMO ORDEN**  
(bloque 81)

Este bloque es ejecutado una vez por vector, y el vector de salida es escrito secuencialmente en las últimas 20 muestras de la memoria intermedia del residuo de predicción LPC [es decir, D(81) a D(100)]. Utilizamos un puntero IP para la dirección de las muestras del ordenamiento D(K) en el que ha de escribirse. Este puntero IP es inicializado con el valor NPWSZ – NFRSZ + IDIM antes de que este bloque empiece a procesar el primer vector de señal vocal decodificada del primer ciclo de adaptación (trama), y a partir de este punto IP es actualizado de la manera descrita más adelante. Los coeficientes APF(I) del predictor LPC de décimo orden se obtienen en la mitad de la recursión de Levinson-Durbin por el bloque 50, como se describe en el § 4.6. Se supone que antes de que se inicie la ejecución de este bloque, el filtro de síntesis del decodificador (bloque 32 de la figura 3/G.728) ya ha escrito el vector de señal vocal decodificada actual en ST(1) a ST(IDIM)

Entradas: ST, APF

Salida: D

Función: Calcular el residuo de la predicción LPC para el vector de señal vocal decodificada actual.

```

If IP = NPWSZ, then set IP = NPWSZ – NFRSZ           | Comprobar y actualizar IP

  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)   | Filtrado FIR
      STLPCI(J) = STLPCI(J – 1)                   | Cambio de la memoria
    D(ITMP) = D(ITMP) + STLPCI(1) * APF(2)         | Tratar el último
    STLPCI(1) = ST(K)                              | Introducir

IP = IP + IDIM                                       | Actualizar IP

```

**MÓDULO DE EXTRACCIÓN DEL PERIODO DE TONO**  
(bloque 82)

Este bloque es ejecutado en el tercer vector de cada trama, después de la generación del tercer vector de señal vocal decodificada.

Entrada: D

Salida: KP

Función: Extraer el periodo de tono del residuo de la predicción LPC

```

If ICOUNT ≠ 3, skip the execution of this block,
otherwise, do the following:                          | Filtrado de paso bajo y reducción
                                                       | del número de muestras 4:1

  For K = NPWSZ – NFRSZ + 1,...,NPWSZ, do the next seven lines | Filtro 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           | Hacer el filtrado FIR sólo si se necesita
      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)                                   | Cambiar la memoria del filtro de paso bajo
    STLPF(1) = TMP

M1 = KPMIN/4                                           | Iniciar la toma de valores de correlación en el
M2 = KPMAX/4                                           | dominio residual LPC diezmando
CORMAX = most negative number of the machine

```

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 = correlación en el dominio diezmado
If TMP > CORMAX, do the next two lines	Encontrar la correlación máxima
CORMAX = TMP	y el retardo correspondiente
KMAX = J	
For N = -M2 + 1, -M2 + 2, ..., (NPWSZ - NFRSZ)/4, do the next line	Cambiar la memoria intermedia del
DEC(N) = DEC(N + IDIM)	residuo LPC diezmado
M1 = 4 * KMAX - 3	Iniciar la toma de valores de correlación en el
M2 = 4 * KMAX + 3	dominio no diezmado
If M1 < KPMIN, set M1 = KPMIN	Comprobar si M1 está fuera de la gama
If M2 > KPMAX, set M2 = KPMAX	Comprobar si M2 está fuera de la gama
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)	Correlación en el dominio no diezmado
If TMP > CORMAX, do the next two lines	Encontrar la correlación máxima y
CORMAX = TMP	el retardo correspondiente
KP = J	
M1 = KP1 - KPDELTA	Determinar la gama de búsqueda alrededor
M2 = KP1 + KPDELTA	del periodo de tono de la trama anterior
If KP < M2 + 1, go to LABEL	Si es verdad, KP no puede ser un tono
	múltiplo de la frecuencia fundamental de tono
	Verificar si M1 está fuera de la gama
If M1 < KPMIN, set M1 = KPMIN	
CMAX = most negative number of the machine	
For J = M1, M1 + 1, ..., M2, do the next six lines	
TMP = 0	Correlación en el dominio no diezmado
For K = 1, 2, ..., NPWSZ, do the next line	
TMP = TMP + D(K) * D(K - J)	Encontrar la correlación máxima y el retardo
If TMP > CMAX, do the next two lines	correspondiente
CMAX = TMP	
KPTMP = J	
SUM = 0	
TMP = 0	Iniciar el cálculo de los pesos de las derivaciones
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	
If TAP > 1, set TAP = 1	Fijar TAP entre 0 y 1
If TAP < 0, set TAP = 0	
If TAP1 > 1, set TAP1 = 1	Fijar TAP1 entre 0 y 1
If TAP1 < 0, set TAP1 = 0	
	Reemplazar KP por el tono fundamental
	si TAP1 es suficientemente grande
If TAP1 > TAPTH * TAP, then set KP = KPTMP	
LABEL: KP1 = KP	Actualizar el periodo de tono de la trama anterior
For K = KPMAX + 1, -KPMAX + 2, ..., NPWSZ - NFRSZ, do the next line	
D(K) = D(K + NFRSZ)	Cambiar la memoria intermedia del residuo LPC

CALCULADOR DE DERIVACIÓN DE PREDICTOR DE TONO  
(bloque 83)

Este bloque se ejecuta también en el tercer vector de cada trama, inmediatamente después de la ejecución del bloque 82. Comparte la memoria intermedia de señal vocal decodificada (ordenamiento ST(K)) con el posfiltro de largo plazo 71, que se ocupa del cambio del ordenamiento de manera que ST(1) a ST(IDIM) constituyan el vector actual de señal vocal decodificada, y ST(-KPMAX - NPWSZ + 1) a ST(0) sean los vectores anteriores de señal vocal decodificada.

Entradas: ST, KP

Salida: PTAP

Función: Calcular el peso de la derivación óptima del predictor de tono de una sola derivación de la señal vocal decodificada.

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

CALCULADOR DE LOS COEFICIENTES DEL POSFILTRO DE LARGO PLAZO  
(bloque 84)

Este bloque se ejecuta también en el tercer vector de cada trama, inmediatamente después de la ejecución del bloque 83.

Entrada: PTAP

Salidas: B, GL

Función: Calcular el coeficiente  $b$  y el factor de escala  $g_l$  del posfiltro de largo plazo

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

| Fijar PTAP en 1

| Desactivar el posfiltro de tono

| si PTAP es más pequeño que el umbral

B = PPFZCF \* PTAP

GL = 1 / (1 + B)

CALCULADOR DE LOS COEFICIENTES DEL POSFILTRO DE CORTO PLAZO  
(bloque 85)

Este bloque también es ejecutado una vez por trama, pero es ejecutado en el primer vector de cada trama.

Entradas: APF, RCTMP(1)

Salidas: AP, AZ, TILTZ

Función: Calcular los coeficientes del posfiltro de corto plazo.

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)

|

| Escalar los coeficientes del denominador

| Escalar los coeficientes del numerador

| «Inclinar» los coeficientes del filtro de

| compensación

### POSFILTRO DE LARGO PLAZO (bloque 71)

Este bloque se ejecuta una vez por vector.

Entradas: ST, B, GL, KP

Salida: TEMP

Función: Realizar la operación de filtrado del posfiltro de largo plazo.

```
For K = 1,2,...,IDIM, do the next line
  TEMP(K) = GL * (ST(K) + B * ST(K - KP))          | Posfiltrado de largo plazo
For K = -NPWSZ - KPMAX + 1,...,-2,-1,0, do the next line
  ST(K) = ST(K + IDIM)                            | Cambiar la memoria intermedia de
                                                    | señal vocal decodificada
```

---

### POSFILTRO DE CORTO PLAZO (bloque 72)

Este bloque es ejecutado una vez por vector, inmediatamente después de la ejecución del bloque 71.

Entradas: AP, AZ, TILTZ, STPFIR, STPFIIR, TEMP (salida del bloque 71)

Salida: TEMP

Función: Realizar la operación de filtrado del posfiltro de corto plazo.

```
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)                     | Parte «todos ceros» del filtro
  TEMP(K) = TEMP(K) + STPFIR(1) * AZ(2)           | Ultimo multiplicador
  STPFIR(1) = TMP
  For J = 10,9,...,3,2, do the next two lines
    TEMP(K) = TEMP(K) - STPFIIR(J) * AP(J + 1)    |
    STPFIIR(J) = STPFIIR(J - 1)                  | Parte «todos polos» del filtro.
  TEMP(K) = TEMP(K) - STPFIIR(1) * AP(2)          | Ultimo multiplicador
  STPFIIR(1) = TEMP(K)
  TEMP(K) = TEMP(K) + STPFIIR(2) * TILTZ          | Filtro de compensación de
                                                    | la «inclinación» espectral
```

---

### CALCULADOR DE LA SUMA DE LOS VALORES ABSOLUTOS (bloque 73)

Este bloque se ejecuta una vez por vector después de la ejecución del bloque 32.

Entrada: ST

Salida: SUMUNFIL

Función: Calcular la suma de los valores absolutos de las componentes del vector de señal vocal decodificada.

SUMUNFIL = 0

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

---

CALCULADOR DE LA SUMA DE LOS VALORES ABSOLUTOS  
(bloque 74)

Este bloque es ejecutado una vez por vector después de la ejecución del bloque 72.

Entrada: TEMP (salida del bloque 72)

Salida: SUMFIL

Función: Calcular la suma de los valores absolutos de los componentes del vector de salida del posfiltro de corto plazo.

SUMFIL = 0

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

SUMFIL = SUMFIL + absolute value of TEMP(K)

---

CALCULADOR DEL FACTOR DE ESCALA (bloque 75)

Este bloque es ejecutado una vez por vector después de la ejecución de los bloques 73 y 74.

Entradas: SUMUNFIL, SUMFIL

Salida: SCALE

Función: Calcular el factor de escala global del posfiltro

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

---

FILTRO DE PASO BAJO DE PRIMER ORDEN (bloque 76) y UNIDAD DE  
ESCALAMIENTO DE LA GANANCIA DE SALIDA (bloque 77)

Estos dos bloques son ejecutados una vez por vector después de la ejecución de los bloques 72 y 75. Es más conveniente describir los dos bloques al mismo tiempo.

Entradas: SCALE, TEMP (Salida del bloque 72)

Salida: SPF

Función: Realizar el filtrado de paso bajo del factor de escala una vez por vector y utilizar el factor de escala filtrado para escalar el vector de salida del posfiltro de corto plazo.

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

SCALEFIL = AGCFAC \* SCALEFIL +

(1 - AGCFAC) \* SCALE

SPF(K) = SCALEFIL \* TEMP(K)

| Filtrado de paso bajo

| Escalar salida

---

## CONVERSIÓN AL FORMATO MIC DE SALIDA (bloque 28)

Entrada: SPF

Salida: SD

Función: Convertir los cinco componentes del vector de señal vocal decodificada en las 5 muestras correspondientes MIC de ley A o  $\mu$  y sacarlas secuencialmente con intervalos de 125  $\mu$ s.

Las reglas de conversión de la MIC uniforme a la MIC de ley A o  $\mu$  se especifican en la Recomendación G.711.

---

ANEXO A  
(a la Recomendación G.728)

**Funciones de ventana híbrida para varios análisis LPC en LD-CELP**

En el codificador LD-CELP utilizamos tres análisis LPC separados para actualizar los coeficientes de tres filtros: el filtro de síntesis, el predictor de ganancia logarítmica y el filtro de ponderación perceptual. Cada uno de estos tres análisis LPC tiene su propia ventana híbrida. Para cada ventana híbrida, damos la lista de los valores de las muestras de función de ventana que se utilizan en el procedimiento de cálculo de ventanización híbrida. Estas funciones de ventana se diseñaron primero con aritmética de punto flotante y luego se cuantificaron de acuerdo con los números que pueden ser representados de manera exacta mediante representaciones de 16 bits con 15 bits de parte fraccionaria. Para cada ventana, damos primero una tabla que contiene el equivalente en punto flotante de los números de 16 bits y luego una tabla con las correspondientes representaciones enteras de 16 bits.

A.1 *Ventana híbrida para el filtro de síntesis*

La tabla siguiente contiene las 105 primeras muestras de la función de ventana para el filtro de síntesis. Las 35 primeras muestras constituyen la porción no recursiva, y las demás constituyen la porción recursiva. La tabla debe leerse de izquierda a derecha empezando por la primera línea, luego de izquierda a derecha en la segunda línea, y así sucesivamente (de la misma manera que la exploración por barrido de líneas).

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 tabla siguiente contiene la representación entera de 16 bits correspondiente. Al dividir cada elemento de la tabla por  $2^{15} = 32\,768$ , se obtiene la tabla anterior.

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 *Venta híbrida para el predictor de ganancia logarítmica*

La tabla siguiente contiene las 34 primeras muestras de la función de ventana para el predictor de ganancia logarítmica. Las 20 primeras muestras constituyen la parte no recursiva, y las demás constituyen la parte recursiva. La tabla debe leerse de la misma manera que las dos tablas anteriores.

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 tabla siguiente contiene la representación entera de 16 bits correspondiente. Al dividir los elementos de la tabla por  $2^{15} = 32\ 768$ , se obtiene la tabla anterior.

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 *Ventana híbrida para el filtro de ponderación perceptual*

La tabla siguiente contiene las 60 primeras muestras de la función de ventana para el filtro de ponderación perceptual. Las 30 primeras muestras constituyen la parte no recursiva y las demás constituyen la parte recursiva. La tabla debe leerse de la misma manera que las cuatro tablas anteriores.

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 tabla siguiente contiene la representación entera de 16 bits correspondiente. Al dividir los elementos de la tabla por  $2^{15} = 32\ 768$ , se obtiene la tabla anterior.

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

ANEXO B

(a la Recomendación G.728)

**Tablas de códigos cifrados de forma y de ganancia de excitación**

En este anexo se da primero la tabla de códigos cifrados de forma VQ de excitación de siete bits. Cada línea de la tabla especifica uno de los 128 vectores de código de forma. La primera columna es el índice de canal asociado con cada vector de código de forma (obtenido mediante un algoritmo de asignación de índice de código Gray). Las columnas dos a seis representan las componentes uno a cinco de los 128 vectores de código de forma representados en punto fijo de 16 bits. Para obtener el valor en punto flotante a partir del valor entero, hay que dividir el valor entero por 2048. Esto es equivalente a multiplicar por  $2^{-11}$  o a desplazar el punto binario 11 bits a la izquierda.

Índice de canal	Componentes de los vectores de código				
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

>>

&lt;&lt;

Índice de canal	Componentes de los vectores de código				
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

&gt;&gt;

&lt;&lt;

Índice de canal	Componentes de los vectores de código				
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

A continuación damos los valores de los códigos cifrados de ganancia. Esta tabla no sólo incluye los valores de GQ, sino también los valores de GB, G2 y GSQ. Tanto GQ como GB pueden representarse exactamente en aritmética de 16 bits utilizando el formato Q13. La representación en punto fijo de G2 es la misma que la de GQ, salvo que el formato es ahora Q12. Bastará una representación de GSQ aproximada al entero más cercano en punto fijo de formato Q12.

**Valores de los ordenamientos relativos a los códigos cifrados de ganancia**

Índice del ordenamiento	1	2	3	4	5	6	7	8
GQ <sup>b)</sup>	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) Puede ser cualquier valor arbitrario (no utilizado).

b) Obsérvese que  $GQ(1) = 33/64$ , y  $GQ(i) = (7/4) GQ(i - 1)$  para  $i = 2, 3, 4$ .

#### ANEXO C

(a la Recomendación G.728)

#### Valores utilizados para la ampliación de la anchura de banda

La tabla siguiente da los valores enteros para los vectores de control de polos, control de ceros y ampliación de anchura de banda mencionados en el cuadro 2/G.728. Para obtener el valor en punto flotante, hay que dividir el valor entero por 16 384. Los valores de esta tabla representan los valores de punto flotante en el formato Q.14, que es el formato utilizado más comúnmente para representar números inferiores a dos en aritmética de punto fijo de 16 bits.

<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					
41	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					

## ANEXO D

(a la Recomendación G.728)

### Coefficientes del filtro elíptico de paso bajo de 1 kHz utilizado en el módulo de extracción del periodo de tono (bloque 82)

El filtro de paso bajo de 1 kHz empleado en el módulo de extracción y codificación del retardo del tono (bloque 82) es un filtro de polos y ceros de tercer orden cuya función de transferencia es

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

donde los coeficientes  $a_i$  y  $b_i$  vienen dados en la tabla siguiente:

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

## ANEXO E

(a la Recomendación G.728)

### Organización en el tiempo de la secuencia de cálculos

Todos los cálculos realizados en el codificador y el decodificador pueden dividirse en dos clases. En la primera se incluyen los cálculos que tienen lugar una vez por vector. En los § 3 a 5.14 se indica de qué cálculos se trata. Generalmente, son los cálculos en que intervienen la cuantificación real de la señal de excitación y la síntesis de la señal de salida, o que llevan a ellas. Haciendo referencia específica a los números de los bloques de la figura 2/G.728, esta clase incluye los bloques 1, 2, 4, 9, 10, 11, 13, 16, 17, 18, 21 y 22. En la figura 3/G.728, esta clase incluye los bloques 28, 29, 31, 32 y 34. En la figura 6/G.728, esta clase incluye los bloques 39, 40, 41, 42, 46, 47, 48 y 67. (Obsérvese que la figura 6/G.728 se aplica tanto al bloque 20 de la figura 2/G.728 como al bloque 30 de la figura 3/G.728. Los bloques 43, 44 y 45 de la figura 6/G.728 no forman parte de esta clase. Así, los bloques 20 y 30 forman parte de ambas clases.)

En la otra clase se incluyen los cálculos que se hacen sólo una vez cada cuatro vectores. Una vez más, refiriéndose a las figuras 2/G.728 a 8/G.728, esta clase incluye los bloques 3, 12, 14, 15, 23, 33, 35, 36, 37, 38, 43, 44, 45, 49, 50, 51, 81, 82, 83, 84 y 85. Todos los cálculos de esta segunda clase están asociados con la actualización de uno o más filtros adaptativos o predictores del codificador. En el codificador hay tres de estas estructuras adaptativas, a saber, el filtro de síntesis LPC de quincuagésimo orden, el predictor de ganancia vectorial y el filtro de ponderación perceptual. En el decodificador hay cuatro de estas estructuras, a saber, el filtro de síntesis, el predictor de ganancia y los filtros adaptativos de largo plazo y de corto plazo. En las descripciones de los § 3 a 5.14 se indican los instantes y las señales de entrada de cada una de estas cinco estructuras adaptativas. Si bien es redundante, en este anexo se dan explícitamente todas estas informaciones de temporización en un solo sitio, para comodidad del lector. En el cuadro E-1/G.728 se resumen las cinco estructuras adaptativas, sus señales de entrada, sus tiempos de cálculo y el instante en que se utilizan por primera vez los valores actualizados. A efectos de referencia cruzada, la cuarta columna del cuadro E-1/G.728 incluye los números de bloque empleados en las figuras y en los § 3 a 5.

CUADRO E-1/G.728

Temporización de las actualizaciones del adaptador			
Adaptador	Señal(es) de entrada	Primera utilización de los parámetros actualizados	Bloques de referencia
Adaptador del filtro de síntesis hacia atrás	Señal vocal de salida del filtro de síntesis (ST) hasta el vector 4	Codificación/decodificación del vector 3	23, 33 (49,50,51)
Adaptador de ganancia vectorial hacia atrás	Ganancias logarítmicas hasta el vector 1	Codificación/decodificación del vector 2	20, 30 (43,44,45)
Adaptador para el filtro de ponderación perceptual y la búsqueda rápida de código cifrado	Señal vocal de entrada (S) hasta el vector 2	Codificación del vector 3	3 (36,37,38) 12, 14, 15
Adaptador para el posfiltro adaptativo de largo plazo	Señal vocal de salida del filtro de síntesis (ST) hasta el vector 3	Sintetización del vector 3 posfiltrado	35 (81 a 84)
Adaptador para el posfiltro adaptativo de corto plazo	Señal vocal de salida del filtro de síntesis (ST) hasta el vector 4	Sintetización del vector 1 posfiltrado	35 (85)

Con mucho, la mayor cantidad de cálculo concierne a la actualización del filtro de síntesis de quincuagésimo orden. La señal de entrada necesaria es la señal vocal de salida del filtro de síntesis (ST). Tan pronto como se ha decodificado el cuarto vector del ciclo anterior, pueden calcularse los coeficientes de autocorrelación mediante el método de ventana híbrida (bloque 49). Al terminar lo anterior, puede empezar la recursión de Durbin para obtener los coeficientes de predicción (bloque 50). En la práctica, nos pareció necesario ampliar este cálculo a más de un ciclo de un vector. Empezamos los cálculos de ventana híbrida antes de que el vector 1 haya sido recibido en su totalidad. Antes de que pueda completarse la recursión de Durbin, debemos interrumpirla para codificar el vector 1. Esta recursión no se completa hasta el vector 2. Finalmente, se aplica la ampliación de anchura de banda (bloque 51) a los coeficientes del predictor. Los resultados de este cálculo no se utilizan hasta que no se haya codificado o decodificado el vector 3, ya que, en el codificador, necesitamos combinar estos valores actualizados con los valores actualizados del filtro de ponderación perceptual y las energías de los vectores de código. Estos valores actualizados no están disponibles antes del vector 3.

La adaptación de ganancia procede de dos maneras. El predictor adaptativo es actualizado una vez cada cuatro vectores. No obstante, el predictor adaptativo produce un nuevo valor de ganancia una vez por vector. En esta sección describimos la temporización de la actualización del predictor. Para calcular esto, es necesario llevar a cabo primero el método de la ventana híbrida en las ganancias logarítmicas anteriores (bloque 43), luego la recursión de Durbin (bloque 44), y la ampliación de anchura de banda (bloque 45). Todo esto puede completarse durante el vector 2 empleando las ganancias logarítmicas disponibles hasta el vector 1. Si el resultado de la recursión de Durbin indica que no hay ninguna singularidad, el nuevo predictor de ganancia es utilizado inmediatamente en la codificación del vector 2.

La actualización del filtro de ponderación perceptual es calculada durante el vector 3. La primera parte de esta actualización consiste en llevar a cabo el análisis LPC de la señal vocal de entrada hasta el vector 2. Podemos empezar este cálculo inmediatamente después de que el vector 2 ha sido codificado, sin esperar que haya sido recibido totalmente el vector 3. Esto consiste en llevar a cabo el método de la ventana híbrida (bloque 36), la recursión de Durbin (bloque 37) y los cálculos de los coeficientes del filtro de ponderación (bloque 38). Acto seguido, tenemos que combinar el filtro de ponderación perceptual con el filtro de síntesis actualizado para realizar los cálculos del vector de respuesta a los impulsos del bloque 12. Debemos también convolucionar cada vector de código de forma con esta respuesta a los impulsos, a fin de hallar las energías de los vectores de código (bloques 14 y 15). Tan pronto como están completos estos cálculos, podemos utilizar todos los valores actualizados para codificar el vector 3.

*Nota* – Debido a la gran cantidad de cálculos que intervienen para determinar las energías de los vectores de código, no pudimos completar la actualización del filtro de ponderación perceptual como parte del cálculo durante el tiempo del vector 2, incluso desplazando a otro sitio la actualización del predictor de ganancia. Esta es la razón por la cual fue diferida hasta el vector 3.

El posfiltro adaptativo de largo plazo es actualizado con base en un algoritmo rápido de extracción de timbre que emplea como entrada la señal vocal de salida del filtro de síntesis (ST). Dado que el posfiltro se utiliza únicamente en el decodificador, el tiempo señalado para realizar este cálculo se basó en las otras cargas de cálculo del decodificador. El decodificador no tiene que actualizar el filtro de ponderación perceptual y las energías de los vectores de código, de suerte que se dispone del intervalo de tiempo del vector 3. La palabra de código del vector 3 es decodificada, y queda disponible la señal vocal de salida del filtro de síntesis, junto con todos los vectores anteriores de salida del filtro de síntesis. Éstos se introducen al adaptador, que produce entonces el nuevo periodo de timbre (bloques 81 y 82) y los coeficientes del posfiltro de largo plazo (bloques 83 y 84). Estos nuevos valores se utilizan inmediatamente para calcular la salida posfiltrada para el vector 3.

El posfiltro adaptativo de corto plazo es actualizado como subproducto de la actualización del filtro de síntesis. La recursión de Durbin es parada en el décimo orden, y los coeficientes de predicción son guardados para la actualización del posfiltro. Como el cálculo de Durbin comienza generalmente durante el vector 1, la actualización del posfiltro adaptativo de corto plazo es completada a tiempo para el posfiltrado del vector de salida 1.

#### ANEXO F

(a la Recomendación G.728)

##### **Lista por orden alfabético de las abreviaturas contenidas en esta Recomendación**

CELP	Predicción lineal con excitación por código ( <i>code excited linear prediction</i> )
DCME	Equipo digital de multiplicación de circuitos ( <i>digital circuit multiplication equipment</i> )
DSP	Procesamiento de señal digital ( <i>digital signal processing</i> )
LD-CELP	Predicción lineal con excitación por código de bajo retardo ( <i>low-delay code excited linear prediction</i> )
LPC	Codificación predictiva lineal ( <i>linear prediction coding</i> )
MIC	Modulación por impulsos codificados
MSE	Error cuadrático medio ( <i>mean-squared error</i> )
RMS	Calculador de media cuadrática ( <i>root-mean-square</i> )
VQ	Cuantificación de vector ( <i>vector quantization</i> )
WNCF	Factor de corrección por ruido blanco ( <i>white noise correction factor</i> )

#### APÉNDICE I

(a la Recomendación G.728)

##### **Verificación de la realización**

Se ha preparado un conjunto de instrumentos para facilitar la comprobación de que las diferentes realizaciones se ajustan al algoritmo definido en esta Recomendación. Estos instrumentos de comprobación están disponibles en un conjunto de disquetes distribuidos por la UIT.