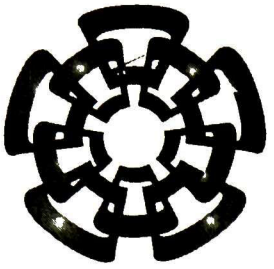

CINVESTAV - IPN

Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara

LABORATORIO DE INGENIERÍA ELÉCTRICA Y CIENCIAS
DE LA COMPUTACIÓN – GUADALAJARA

CODIFICACIÓN DE FUENTE Y CANAL EN EL CELP FS1016

TESIS QUE PRESENTA
OMAR HUMBERTO LONGORIA GÁNDARA

PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS

EN LA ESPECIALIDAD DE
INGENIERÍA ELÉCTRICA

Guadalajara, Jal., Septiembre de 1998

CODIFICACIÓN DE FUENTE Y CANAL EN EL CELP FS1016

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

por:

Omar Humberto Longoria Gándara

Ingeniero en Electrónica y Comunicaciones
Instituto Tecnológico y de Estudios Superiores de Monterrey, 1989–1993

Becario del CONACYT, expediente no. 112936

Directores de Tesis:

M.C. José Ramón Rodríguez Cruz
Dr. Arturo Veloz Guerrero

CINVESTAV del IPN Unidad Guadalajara, Septiembre de 1998.

Agradecimientos.

Un canto de agradecimiento brota desde mi corazón hacia el Señor Jesús, pues al final de este pequeño tramo, que ha sido la maestría, no me queda más que agradecerle el cariño y la comprensión que me tiene.

Agradezco a mis padres el don de la vida y la confianza que me han tenido.

Quiero expresar el más sincero agradecimiento a mi asesor de tesis, el Ing. Ramón Rodríguez por la motivación, el trabajo en equipo, la comprensión y el cariño a la investigación, que ha infundido en mi persona a lo largo de este tiempo.

También agradezco la colaboración y ayuda del Dr. Veloz, pues a través de él he aprendido que no importa que pertenezcamos a un país subdesarrollado, ya que el gusto por el trabajo, la constancia y la responsabilidad son piezas claves para "hacer investigación".

Reitero mi agradecimiento al Dr. Manuel Guzmán, de quien destaco la motivación que ha depositado en nosotros los estudiantes, de superarnos y aprender a aprender.

Envió las gracias a mis compañeros de maestría, Iván y Alejandro, por su apoyo en los momentos difíciles.

Le doy gracias a mi tía Fabiola, por su cariño y apoyo para poder llevar a cabo mis estudios.

Finalmente, deseo que estas palabras de agradecimiento se traduzcan en servicio a mi comunidad desde el lugar en que me encuentre.

"Nunca encontró Gerónimo, como otros ciertos hombres encontraban, placer en el prodigio. Nunca le regocijaron la vastedad del universo, la belleza extrema de algunas criaturas o la infinitud de las combinaciones matemáticas, pues en cada uno de esos asombros hallaba probada la miseria de su inteligencia, que, si había sido hasta entonces ignorante de ideas, seres o materias tan sobresalientes como aquellos que al revelársele admiraba, podría seguir siéndolo perpetuamente de otros, semejantes o diversos, no menos excelentes. Fue temeroso siempre de lo magnífico porque advertía en ello la mezquindad de su discernimiento y la estrechura de su memoria."

La dulce ira.
Luis G. Martín.

"No mucho el saber harta y satisface el ánima, sino el gustar de las cosas internamente"

Ignacio de Loyola.

ÍNDICE

CAPÍTULO I

INTRODUCCIÓN

Introducción a la Tesis.

"Si es cierto que sin pensamientos no podemos pensar y que aprendemos a pensar a través de las palabras, entonces el lenguaje limita y rodea todo el pensamiento humano... ".

Herder.

La historia del hombre está marcada por dos sistemas de información, los códigos genéticos como el ADN, para la transmisión de la información de los caracteres en un árbol genealógico, y el lenguaje, visto como el medio técnico empleado por el hombre para comunicar lo que inventa, lo que conserva y observa; en pocas palabras, los hechos históricos. Así el lenguaje viene a ser imprescindible para constituir la sociedad humana y hacer que el hombre promueva su facultad de comunicación.

La voz o señal de voz, como se referirá en esta tesis, constituye el lenguaje hablado. Cuando se desea procesar y transmitir esta señal en su forma eléctrica a través del espacio, se tienen dos formas de hacerlo: la analógica y la digital. Durante el proyecto de tesis se trabajó con el procesamiento y la transmisión digital, que permiten la implementación de técnicas de codificación, de protección contra el ruido del canal y de seguridad cuando son explotadas.

El concepto de codificación permite manipular las señales de voz digitalizadas de manera que es posible tener señalización y recuperación, seguridad e integración con otro tipo de servicios y sistemas. Es muy importante notar que el hombre y en particular su oído es el que percibe o mide la señal de voz reproducida. Así pues, podemos decir que el hombre es quien mide la calidad de una reproducción de voz.

Las técnicas de codificación de señales de voz avanzan en vías de la implementación de sistemas de comunicación en tiempo real que funcionen cada vez más a menores razones de transmisión de bits, que sean robustos al ruido agregado en la señal por el canal y, simultáneamente, que la señal reproducida conserve una calidad aceptable, objetiva y subjetivamente, pues el hombre es el receptor de la señal de voz, él es quien escucha.

La evolución de los codificadores de voz ha pasado por aquéllos que funcionan a 64 Kbps como los codificadores PCM (Pulse Code Modulation), los de 32 Kbps como ADPCM (Adaptive Diferencial PCM), hasta los vocoders que van desde los 1200 a los 2400 bps; dentro de estos últimos tenemos al conocido LPC-10 (Linear Predictive

Coder). Sin embargo, al analizar calidad, la robustez contra el ruido y la complejidad, encontraremos que entre menor sea la razón de transmisión de bits, la calidad de la señal reproducida disminuye (se escucha más robótica) y la robustez contra los errores inducidos aminora. El precio a pagar es pues la calidad de la señal y la complejidad de los algoritmos; pero por otro lado se tiene, como motivación y aliento, el rápido progreso de la tecnología de los procesadores digitales que hacen posible la implementación de algoritmos altamente complejos trabajando en tiempo real.

Así, en un principio se tenía como un estándar federal en los Estados Unidos, al LPC-10 (Linear Predictive Coding), con el avance de las nuevas tecnologías como el análisis por síntesis (AbS), se optó por seleccionar a los codificadores híbridos; entre ellos están los codificadores CELP (Code-Excited Linear Predictive) y VSELP (Vector Sum Excitation Linear Prediction) que trabajan alrededor de los 4800 y 9600 bps. En la actualidad han surgido nuevos codificadores que persiguen transmitir a una razón de transmisión del orden de los vocoders y a la vez tener una calidad equiparable a los codificadores híbridos; dentro de este tipo de codificadores encontramos al MELP (Mixed Excited Linear Predictive) que funciona a razones de transmisión que comprenden los 1700 a los 2400 bps aunado a que la señal reproducida presenta buena calidad. Durante el trabajo de tesis se ha puesto la atención en el codificador de voz híbrido, CELP FS1016 a 4800 bps. Este codificador se encuentra embebido en el estándar de radio móvil (Federal Standard 1024) a 8000 bps, incluida la protección de canal, propuesto en los Estados Unidos.

Los conceptos de codificación conjunta de canal y fuente (CCCF) serán manejados; adelantando un poco, la CCCF está referida a dos esquemas de trabajo: el primero es tratar el transmisor con el propósito de diseñar un codificador que tome en cuenta, y de manera simultánea, la fuente de información y el modelo de canal utilizado para la transmisión; el segundo esquema es diseñar un decodificador que aproveche la redundancia de información que el codificador fuente no logró eliminar.

La meta de esta tesis ha sido la tarea de implementar algoritmos de codificación conjunta de canal y fuente en el codificador comercial CELP FS1016 a 4800 bps, con el objetivo de explorar las posibilidades de mejoría en la calidad de voz, sin aumentar el ancho de banda y considerando un canal ruidoso. Para lograrlo ha sido necesario el conocimiento e implementación de algunos algoritmos como el LBG y la cuantificación trellis (TQ) - cuando no se toma en cuenta al canal en la codificación fuente- que no se consideran algoritmos de codificación conjunta de canal y fuente y, finalmente, la realización de los algoritmos como el temple simulado (Simulated Annealing, SA), la cuantificación vectorial optimizada para el canal (Channel Optimized Vector Quantization, COVQ) y el algoritmo de Viterbi con salidas suaves (Soft Output Viterbi Algorithm, SOVA), que sí pertenecen a los algoritmos de CCCF.

El contexto en el que se desarrolló esta tesis fue el siguiente: Recientemente en el CINVESTAV Unidad Guadalajara se ha estado promoviendo la elaboración de tesis de maestría que vayan de la mano con las investigaciones y tesis de estudiantes de doctorado, con el propósito, según lo he experimentado, de trabajar en equipo y

ofrecer un acompañamiento más cercano a los estudiantes de maestría. Esta tesis, en lo particular, se llevó a cabo con el propósito de colaborar con un trabajo de investigación de mayor magnitud, propuesto en la tesis de doctorado (aún en progreso) del M.C. José Ramón Rodríguez Cruz; ambas tesis se proponen en un proyecto de Codificación Conjunta aprobado por CONACYT y dirigido por el Dr. Arturo Veloz Guerrero y el Dr. Mauricio Lara.

Quisiera expresar que la experiencia personal adquirida, bajo este esquema de realización de tesis, ha sido de mucho beneficio, pues mantiene la responsabilidad, sostiene la motivación, ayuda al aprendizaje y es posible llegar a resultados objetivos en la investigación emprendida.

A continuación se muestra la organización de la tesis: En el capítulo 2 se describe el funcionamiento del codificador CELP FS1016 a 4800 bps, pues éste es el objeto de prueba; el capítulo 3 comprende una descripción sencilla y la clasificación de los algoritmos de CCCF, así como el trabajo realizado con los algoritmos LBG, SA y COVQ; en el capítulo 4 se expone la cuantificación tipo trellis y, por último, el capítulo 5 explica y describe el algoritmo SOVA. Los resultados y las conclusiones obtenidas se comentan en cada capítulo. El apéndice contiene algunas de las rutinas implementadas, sin embargo los programas del codificador y sus modificaciones se citan haciendo referencia a un compendio de programas que se encuentran en la biblioteca del CINVESTAV Unidad Guadalajara. También se puede encontrar en la biblioteca un escrito sobre la simulación y comparación del funcionamiento del algoritmo de Viterbi y el SOVA. En el capítulo correspondiente se hace referencia explícita a estos documentos.

Los resultados obtenidos durante esta tesis ha motivado a nuestro Equipo de Voz a compartir el trabajo en el Simposium Internacional de Teoría de la Información a celebrarse en el mes de octubre del presente año con el artículo titulado "Joint Source Channel Coding with the FS-1016 CELP Coder" Un segundo artículo titulado "TCVQ of the Secondary Excitation of the FS-1016 CELP Coder", se propondrá para el ICASSP'99 (International Conference in Audio Signals and Signal Processing), esperando tener resultados positivos.

Esperamos que en una segunda tesis se lleve a cabo la implementación del algoritmo de trellis mediante el uso de un procesador de señales.

Creemos que un buen trabajo a futuro es modificar el algoritmo de trellis con la finalidad de tener un algoritmo orientado a la codificación conjunta de canal y fuente. Otro proyecto propuesto a desarrollar consistiría en emplear códigos turbo para la codificación-decodificación de canal tomando en cuenta las estadísticas de la fuente de información.

CAPÍTULO II

EL CODIFICADOR HÍBRIDO CELP (CODE-EXCITED LINEAR PREDICTIVE)

El CELP Federal Standard 1016 (FS1016) @ 4800 bps

1 Introducción

" ¿Cúal es el significado de todo, Mr. Holmes? " " Ah, no tengo datos, no se lo puedo decir, " contestó. *Arthur Conan Doyle*

Existen dos clases básicas de codificadores de voz: el codificador de forma de onda y el vocoder. Los codificadores de forma de onda intentan preservar la forma de onda de la señal que va a ser codificada. Estos son capaces de proporcionar alta calidad de voz a razón de transmisión de bits media (e.g. ADPCM @ 32 kbps), sin embargo no pueden ser usados para codificación de voz a bajas razones de transmisión (< 8 Kbps) debido a que la calidad de la señal reproducida disminuye de forma considerable. Los vocoders, por otro lado, tratan de producir una señal que se escuche o se perciba como la señal de voz original. Estos pueden ser utilizados para trabajar a muy bajas razones de transmisión (< 4800 bps), pero la voz resultante, aunque es altamente inteligible, tiende a sonar muy sintética o robótica. Por lo tanto es viable pensar en una mezcla de ambas técnicas, para tener alta calidad de voz operando a bajas razones de transmisión. Tal mezcla se conoce como codificador híbrido, que intenta preservar las partes perceptualmente importantes de la forma de onda de la voz de entrada. Dentro de los codificadores híbridos podemos encontrar al SELP (Self Excited Linear Predictor), MLPC (Multi-Pulse Linear Predictive Coding), VSELP (Vector Sum Excitation Linear Prediction) y al CELP. El trabajo hecho en esta tesis se remite a tomar como objeto de prueba, para la realización de las simulaciones, al codificador híbrido CELP.

El Codificador híbrido CELP, fue presentado por primera vez en 1984 en la Internacional Communications Conference por B.S. Atal y M.A. Schroeder [1]. Desde entonces, el esquema primitivo ha sido sometido a infinidad de estudios que persiguen un diseño realizable en tiempo real, con las posibilidades que ofrece el hardware existente.

En ese mismo año, el Departamento de Defensa (DoD) de los Estados Unidos lanzó un programa para desarrollar la tercera generación de Unidad Telefónica Segura, o

STU-III (Third generation Secure Telephone Unit), capaz de proveer comunicaciones seguras para voz. Esta investigación tenía como finalidad sustituir al vocoder LPC-10e a 2400 bps. En 1988 el CELP desarrollado por la DoD y AT&T Bell Laboratories fue seleccionado. Las pruebas subjetivas y las pruebas formales tales como DAM (Diagnostic Acceptability Measure) y DRT (Diagnostic Rhyme Test) mostraron el revolucionario desempeño de este codificador operando por debajo de los 16,000 bps; robusto frente al ruido acústico, a errores de canal y comparable a un codificador tipo CVSD (Continously Variable Slope Delta Modulation) a 32,000 bps.

El codificador CELP FS1016 está basado en una versión mejorada de este codificador hecha por Campbell, Welch and Tremain, donde en [2] hacen referencia a sus investigaciones. Este codificador ha sido aprobado para formar parte del estándar de comunicaciones de radio móvil a 8,000 bps, incluida la codificación de canal, (Fed-Std-1024); además de ser propuesto para estándares militares. Es importante notar que en esos años se esperaba que este nuevo sistema de codificación de voz reemplazara a muchos sistemas existentes, basados en 12,000 y 16,000 bps.

En el presente capítulo de esta tesis, se describirá el funcionamiento del codificador CELP FS1016. La organización será de la siguiente manera: Basándonos en el esquema general del codificador se expone la teoría general del análisis por síntesis (AbS), luego se explica el funcionamiento del receptor y el transmisor haciendo hincapié en la partes constitutivas de éste último, por la razón de que contiene en su estructura una réplica del decodificador; debido a lo anterior, el tipo de análisis hecho en el transmisor se denomina AbS.

2 Esquema General

Al igual que todas las técnicas de cuantificación vectorial, la codificación hecha por el CELP es una técnica de tramas que separa la señal de entrada una vez muestreada, en bloques de muestras llamados vectores. Las piezas claves en la codificación CELP - como le llamaremos a la codificación que utiliza el algoritmo CELP- son: la técnica de Análisis por Síntesis (AbS), la Cuantificación Vectorial (VQ) de vectores perceptualmente ponderados y la Predicción Lineal (LP).

El filtro de predicción lineal es de 10º orden y se usa para modelar el espectro de término corto o estructura de formantes de la señal de voz. La redundancia de término largo de la señal de voz, conocida como altura tonal o "pitch", es modelada mediante un libro de códigos adaptable. El residual obtenido como fruto del análisis LP de término corto y la cuantificación vectorial del "pitch", es cuantificado vectorialmente usando un libro de códigos estocástico llamado también contenedor de excitaciones secundarias. Las excitaciones óptimas escaladas (ganancia asociada) de los libros adaptable y estocástico, son seleccionadas por medio de la minimización, en el dominio del tiempo, de una medida de distorsión perceptualmente pesada que mejora subjetivamente la calidad de la voz, explotando las propiedades de enmascaramiento del oído humano.

El costo computacional requerido por el codificador CELP es dominado por la búsqueda de los vectores óptimos dentro de los dos libros de código existentes. La calidad de la voz y la complejidad computacional dependen del tamaño del libro de códigos para realizar las búsquedas, por lo que existe un compromiso entre calidad de voz y complejidad computacional. Podemos manipular un número menor de vectores, o únicamente realizar la búsqueda en un sector del libro de códigos, para cumplir con las limitaciones de nuestro procesador a expensas de que la calidad de voz disminuya.

El FS1016 utiliza una razón de muestreo de 8 kHz y el análisis por trama se realiza cada 30 milisegundos divididos en cuatro subtramas de 7.5 ms. El análisis consiste en tres funciones básicas: 1.) La predicción lineal de término corto, 2.) la búsqueda adaptativa del "pitch" y 3.) la búsqueda del vector estocástico óptimo. La etapa de síntesis consiste también de tres funciones que se realizan de la siguiente manera: 1.) Obtener el vector óptimo a partir del índice seleccionado, 2.) Obtener la contribución hecha por el "pitch". es decir, el vector de muestras con el retardo correspondiente, 3.) Introducir o sumar las dos contribuciones a través del filtro de síntesis o todo polos.

Los parámetros transmitidos son los índices y las ganancias de cada libro de códigos, así como los 10 coeficientes del filtro predictor que se envían en forma de parámetros LSP (Line Spectral Pairs).

En el receptor las operaciones son sencillas en comparación con el emisor: Una cuarta etapa, opcional y aunada a las tres funciones de la etapa de síntesis, es el postfiltrado, que permite mejorar la calidad de la voz de salida.

2.1 Análisis por Síntesis (AbS).

Al filtrar la señal de voz mediante el filtro de predicción de término largo, LTP (Long Term Predictor) y el filtro de predicción de término corto, STP (Short Term Predictor), tenemos lo que llamamos señal residual. Si ésta se transmitiera íntegra al receptor y, en este último, colocáramos los filtros STP y LTP inversos, obtendríamos la señal de voz original. Sin embargo no es posible mandar la señal residual cuando queremos trabajar a bajas razones de transmisión, debido a la gran cantidad de información que sería necesario enviar. Una buena opción para no transmitir la señal residual es utilizar los codificadores paramétricos o un codificador híbrido en el que se tengan libros de códigos con vectores de muestras, representando a la señal residual.

A partir de aquí podemos comparar la señal residual obtenida con algún vector del libro de códigos, teniendo como patrón de comparación alguna medida de distorsión. Los estudios reportados en la literatura [5] señalan que la minimización de esta medida de distorsión no garantiza que la diferencia entre la señal de voz original y la reconstruida en el receptor, sea la mínima. En otras palabras podemos decir que una minimización de algún tipo de error (e.g. MSE (Minimum Squared

Error)), en el espacio de las muestras residuales, no implica una minimización de ese error en el espacio de las muestras de voz.

Como a nosotros nos interesa que la voz transmitida por un interlocutor sea percibida por el oído humano, es necesario entonces, hacer el esfuerzo para que la señal de voz reconstruida sea lo más parecida a la señal de voz que el oído escucharía.

Para tener un mejor control sobre las distorsiones en la señal de voz reconstruida, el residual debe ser cuantificado (representado por uno de los vectores del libro de códigos) de acuerdo con la minimización del error entre la señal de voz original y la voz reconstruida [3]. Tal procedimiento de codificación es llamado Análisis por Síntesis y es utilizado por el codificador FS1016. Este procedimiento tiene la ventaja adicional de que es sencillo para incorporar modelos de percepción humana (lo que el oído escucha), por medio del uso de medidas de distorsión "pesadas" perceptualmente.

2.2 El Receptor del CELP FS1016.

Comenzamos con el receptor debido a que el transmisor tiene en su núcleo la etapa de decodificación de la voz. La **Fig. 1** ilustra a manera de bloques la estructura del receptor del FS1016.

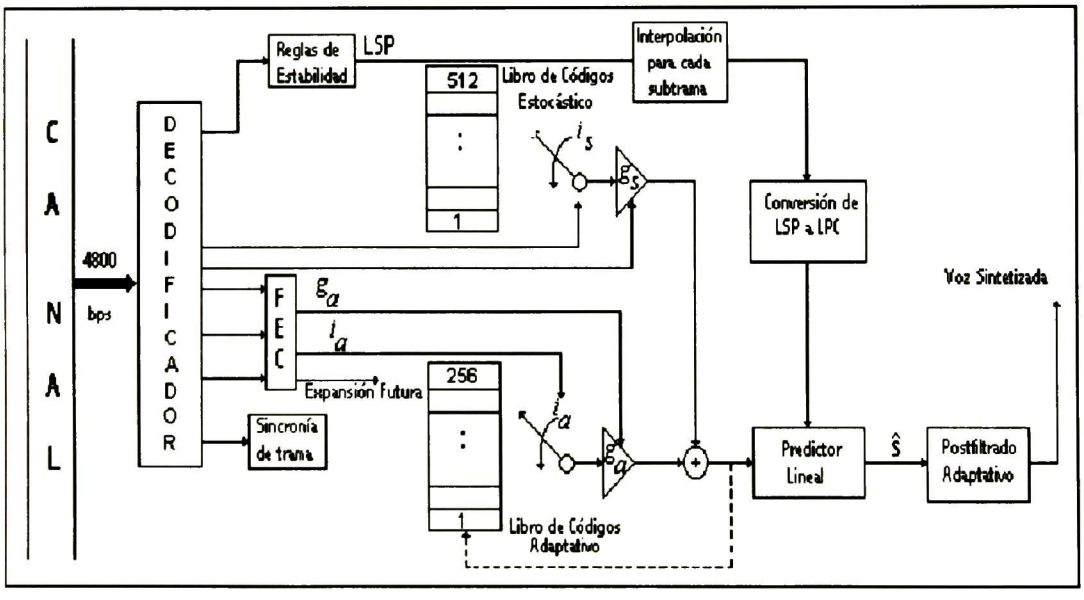

Fig. 1 Esquema del Rx de un Codificador CELP

Luego de que el flujo de bits correspondiente a una trama de 4,800 bits es recogido por el decodificador, éste decodifica los parámetros enviados por el transmisor

incluyendo los bits de protección de canal (para los parámetros del filtro adaptable) como lo especifica la norma del FS1016. El receptor sintetiza la voz por medio de una excitación cuantificada y codificada vectorialmente a través de un filtro de predicción lineal.

Para cada subtrama (60 muestras), la excitación es proporcionada por un libro de códigos estocástico fijo y un libro adaptable. El libro de códigos estocástico contiene lo que conocemos como excitación secundaria o redundancia de término corto; por su parte, el libro de códigos adaptable, contiene la información referente al "pitch" o redundancia de término largo. La entrada al filtro de predicción lineal está formada añadiendo una excitación secundaria, dada por un índice i_s y escalada por una ganancia g_s, a un vector proveniente del libro adaptable, dado por un índice i_a y escalado por un factor g_a. El libro de códigos adaptable es entonces actualizado con esta excitación para poder usarse en la siguiente subtrama. Con esto podemos apreciar que el libro de códigos adaptable es una memoria que guarda la historia de las señales de excitación pasadas; de manera que el índice i_a indica la posición a partir de la cual se obtendrá el mejor bloque de muestras del pasado que serán usadas para la predicción en la siguiente subtrama. La cantidad de muestras retrasadas en el tiempo es lo que conocemos como "pitch"; el cual representa un índice del libro de códigos adaptable. Posteriormente, la señal sintética puede o no ser introducida a una etapa de postfiltrado con el propósito de mejorar, bajo criterios subjetivos, la señal de voz que será escuchada por el oído humano.

2.3 El Transmisor del CELP FS1016.

El transmisor CELP, mostrado en la **Fig. 2** contiene una réplica del sintetizador del receptor, excepto la etapa de postfiltrado.

Como podemos apreciar, la señal de voz estimada s es obtenida como se describe para el receptor. Posteriormente es restada a la señal s que representa la señal de voz original; la diferencia es pasada por un filtro perceptual. Este error perceptualmente ponderado es utilizado para controlar el procedimiento de búsqueda, usando cuantificación vectorial y el método de análisis por síntesis, que minimizará dicho error. El procedimiento de búsqueda consiste en encontrar los índices y las ganancias correspondientes que minimicen el error perceptualmente ponderado.

El filtro de predicción lineal se determina por medio de las técnicas de análisis de predicción lineal en lazo abierto actuando sobre la señal de voz original.

Una vez encontrados los parámetros CELP a transmitir, se codifican en conjunto con un bit de sincronía y un bit para futuras expansiones de acuerdo como lo especifica el FS1016. La **Tabla 1** relaciona el acomodo de los bits codificados en una trama.

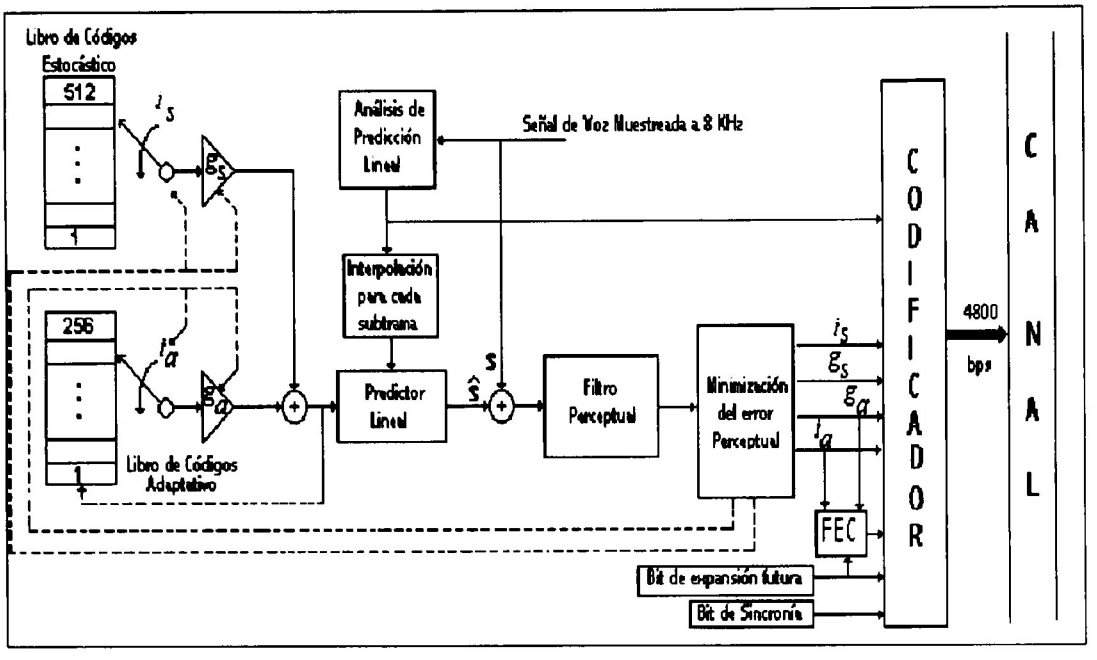

Fig. 2 Esquema del Tx de un Codificador CELP

Tabla 1.

Principales	Predicción Lineal	Libro de Códigos Adaptable	Libro de Códigos Estocástico
Actualización	30 ms	30/4 = 7.5 ms	30/4 = 7.5 ms
Parámetros	10 LSPs	1 ganancia, 1 retardo 256 vectores	1 ganancia, 1 índice 512 vectores
Análisis	Lazo abierto 10º Orden Autocorrelación 30 ms ventana de Hamming, no preénfasis 15 Hz de expansión Interpolados por 4	Lazo cerrado Dimensión 60 MSPE VQ Factor de peso,γ= 0.8 Búsqueda tipo delta Rango: 20 a 147 Retardos no enteros	Lazo cerrado Dimensión 60 MSPE VQ Factor de peso,γ= 0.8 Desplazamiento por –2 77% de esparcimiento Muestras Ternarias
Bits por Trama	34 (3,4,4,4,4,3,3,3,3,3)	Índice: 8+6+8+6 ± Ganancia: 5×4	Índice: 9×4 ± Ganancia: 5×4
Razón	1133.33 bps	1600 bps	1866.67 bps
Misceláneo	Los restantes 200 bps están conformados de la manera siguiente: 1 bit por trama para sincronización, 4 bits por trama para corrección de errores (FEC) y 1 bit por trama para expansiones futuras.		

2.4 El Análisis de Predicción Lineal (LPC).

El propósito del análisis de predicción lineal (Linear Predictive Coding analysis) es obtener los coeficientes del filtro de predicción lineal para la trama de voz correspondiente. La interpretación temporal del predictor es sencilla: con el conjunto de coeficientes queremos predecir el valor de una muestra como una combinación lineal de las anteriores. Veamos:

Si $s'(n)$ es la señal predicha, $s(n)$ la señal original y $\{a_k\}$los coeficientes de predicción lineal con p etapas; entonces $s'(n)=\sum_{k=1}^{p} a_k \cdot s(n-k)$ y

$e(n)=s(n)-s'(n)$ Representando al error de predicción o residual.

Al sustituir y obtener la transformada Z para $S(z)$ tenemos:

$$S(z)=E(z)+S(z)\cdot\sum_{k=1}^{p} a_k \cdot z^{-k} \quad \text{y}$$

$$H(z)=\frac{S(z)}{E(z)}=\frac{1}{1-\sum_{k=1}^{p} a_k \cdot z^{-k}}$$

como función de transferencia del filtro de síntesis. De aquí que

$$H^{-1}(z)=A(z)=1-\sum_{k=1}^{p} a_k \cdot z^{-k}$$ Sea el filtro de análisis.

En el dominio de la frecuencia, la magnitud de los coeficientes de predicción lineal, representa la envolvente espectral (formantes) del fragmento de voz del cual se han extraído.

En la **Fig. 2** podemos apreciar que el error residual es el resultado de extraer de la señal de voz original $s(n)$ toda la información espectral o redundancia de término corto por el análisis de predicción lineal, y en el caso ideal su espectro sería plano, como el ruido blanco. Es por eso que en el análisis LPC hablamos de "blanquear" la señal de voz. No obstante, el filtro LPC es sólo un modelo, una estimación espectral de término corto y, por lo tanto, $e(n)$ conserva características tales como la información referente al carácter sonoro/sordo, la forma de onda y la no estacionariedad de la señal de voz.

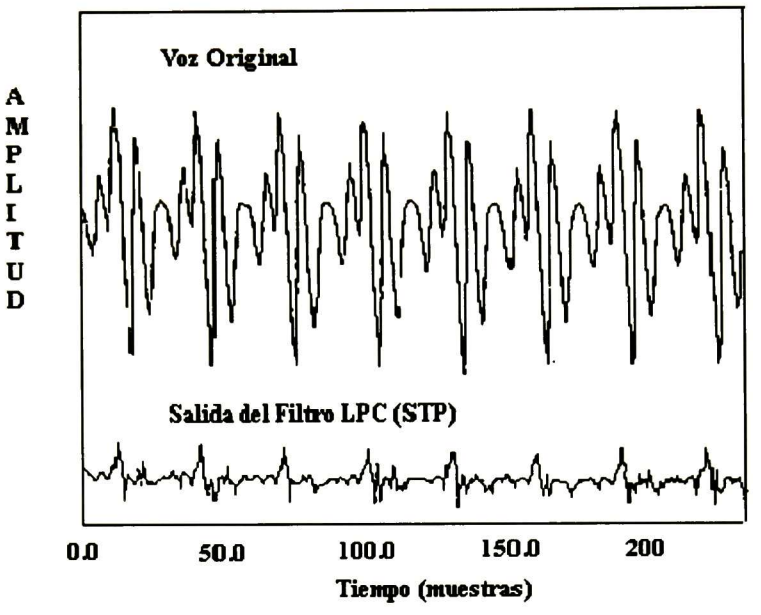

Fig. 3 Formas de Onda de las Señales Original y la Salida del Filtro LPC de Análisis

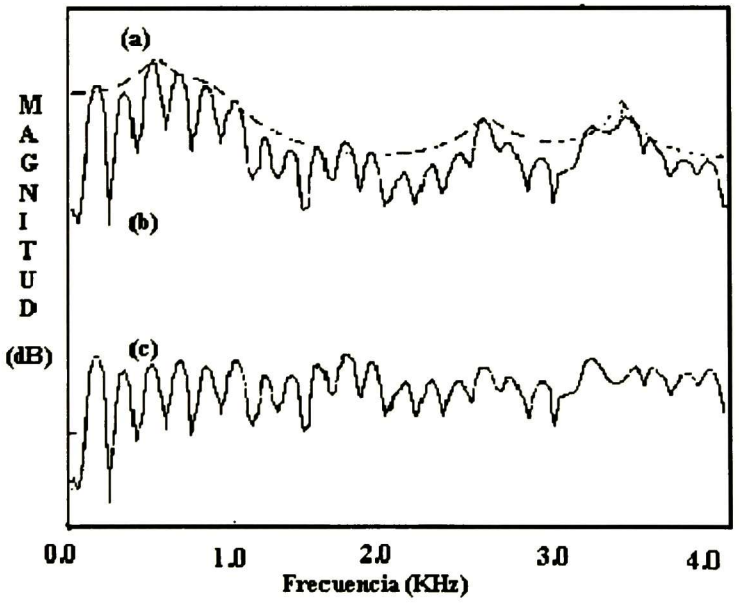

Fig. 4 Espectro de: a) La envolvente de la voz original, b) La voz original, c) La salida del filtro LPC de análisis.

Dos ventajas posteriores al análisis LPC son:

- La señal residual $e(n)$ al tener menor variación espectral que la señal, $s(n)$, refleja con mayor nitidez la periodicidad de la señal fuente y por lo tanto es muy útil para obtener la información referente al "pitch".

- Como se ha extraído redundancia de $s(n)$, la varianza de $e(n)$ es menor y con esto, su codificación, puede realizarse de manera más eficiente.

Una desventaja o problema importante en el análisis LPC es que los polos del filtro de síntesis son muy sensibles a la cuantificación de sus coeficientes debido a la varianza de los mismos. Es por eso que es necesario realizar una transformación de estos coeficientes a un espacio de señales donde su varianza sea menor para obtener una codificación más robusta y garantizar la estabilidad del filtro. Una buena opción es transformar los coeficientes LPC en los parámetros LSP (Line Spectral Pairs).

Una vez que se ha descrito grosso modo el análisis por síntesis, procedamos a las particularidades del FS-1016.

2.4.1 Análisis de Predicción Lineal en el FS1016

El análisis de predicción lineal de término corto, STP (Short Term Predictor), es realizado una vez por cada trama en lazo abierto. Se utiliza el método de la Autocorrelación para obtener los coeficientes del filtro de predicción de 10° orden usando una ventana de Hamming de 30 ms, con una expansión de ancho de banda de 15 Hz, sin preénfasis.

El procesado de la voz, se realiza sobre tramas enventanadas en el tiempo. Esto introduce siempre una distorsión espectral. Al ser la señal de la ventana $w(n)$ finita en tiempo, su espectro presentará una distribución infinita de frecuencias, con un lóbulo principal y una distribución infinita de lóbulos secundarios. La ventana que minimice los efectos indeseables del enventanado será aquella cuyo lóbulo principal sea más estrecho, y cuyos lóbulos secundarios tengan la menor magnitud. La ventana de Hamming es habitualmente usada.

La operación de expansión del ancho de banda reemplaza los coeficientes LP, a_k por $a_k\gamma^k$. Esto desplaza los polos radialmente hacia el origen en el plano z por el factor de ponderación, γ, para $0<\gamma<1$. Estos coeficientes expandidos definen el filtro LP $1/A(z)$; donde γ equivale a 0.994 para una expansión de 15 Hz. Además de mejorar la calidad de voz, esta expansión de 15 Hz es benéfica para la cuantificación LSP ya que disminuye la distorsión introducida por las transformaciones entre coeficientes, además de propiciar métodos de búsqueda rápidos que convierten los coeficientes del filtro predictor directamente en los parámetros LSP cuantificados [4]. La relación entre el parámetro γ, y la expansión Δf está dada por :

$$\Delta f = \frac{f_s}{\pi} \cdot \ln(\gamma)\ \text{[Hz]}$$ donde f_s es la frecuencia de muestreo.

2.4.1.1 El filtro perceptual.

El objetivo del filtro perceptual es transformar el espectro de voz, ya sea de la fuente o sintetizada, en un espectro que es más parecido al que el oído humano capta. Además, el ruido superpuesto se tolera mayormente en las frecuencias de los formantes que en los valles del espectro. La acción realizada por el filtro perceptual se denomina enmascaramiento de ruido, es decir, pretendemos que la potencia del ruido quede por debajo de la potencia de la señal. Con esto el análisis por síntesis estará controlado por un criterio de error que es más afín al criterio de percepción humano. Este filtro perceptual fue propuesto por Atal y está dado por

$$W(z) = \frac{A(z)}{A(z/\gamma)} = \frac{1 - \sum_{k=1}^{p} a_k z^{-i}}{1 - \sum_{k=1}^{p} a_k \gamma^k z^{-k}} \qquad 0 \le \gamma \le 1$$

El efecto del factor γ no altera la frecuencia central de los formantes, pero expande el ancho de banda de los mismos por medio del factor, Δf. La **Fig. 5** ilustra la envolvente espectral en el trazado superior, mientras que las curvas contiguas corresponden al módulo espectral del filtro $W(z)$ para valores de γ =0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 y 0.9 tomando de referencia la ordenada en el origen [8]. Esta expansión es idéntica a la que se describió anteriormente para el enventanado. El factor γ para el filtro perceptual empleado en el FS1016 equivale a 0.8.

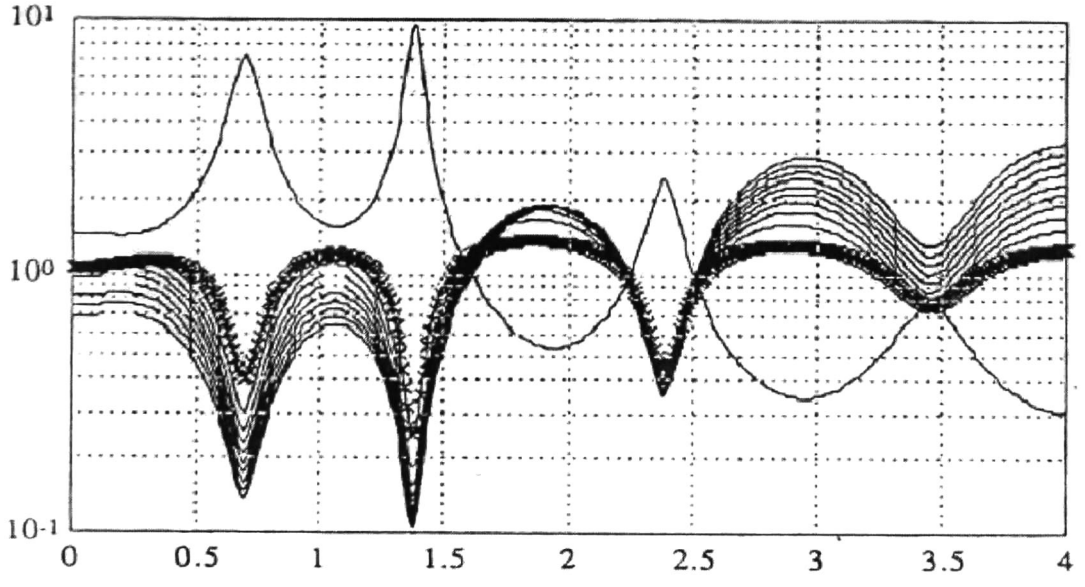

Fig. 5 Gráficas de la envolvente espectral y módulo(s) espectral del filtro $W(z)$.

2.4.1.2 Retardo de procesamiento

El retardo interno está determinado por el análisis de predicción y es de 15ms, equivalente a dos subtramas. Esto es debido a que la ventana de análisis (30ms) está centrada en la terminación de la trama pasada. En concreto, para la trama 1, se transmiten el espectro de la trama 1 junto con los parámetros de excitación ("pitch" y libro de códigos), por omisión (valores codificados cercanos a cero), para las dos primeras subtramas y, los parámetros analizados para las siguientes 2 subtramas. Asimismo, las subsecuentes tramas son transmitidas con los parámetros espectrales 2 subtramas delante de los parámetros de excitación.

Típicamente, el retardo total de la codificación por medio del CELP, incluyendo almacenamiento, es de 105ms [2].

2.4.1.3 Conversión de los parámetros LPC a LSP.

El principal objetivo del procedimiento de cuantificación es codificar los parámetros LPC con el menor número de bits posible sin introducir distorsión espectral adicional. El sistema debe cuantificar los coeficientes del filtro predictor representados por el vector $\boldsymbol{a}$, usando el análisis LSP, ya que los términos a_k son 10 números reales que requieren demasiados bits (10*16=160 bits) para su representación. Por otro lado, los parámetros LSP son más eficientes para ser codificados debido a que fluctúan entre valores enteros de 0 a 8 (o 16), correspondiendo a las entradas de una tabla conveniente de LSP´s. Como no es posible una reconstrucción perfecta, lo que se logra es una "transparencia subjetiva" (aparentemente el espectro no se escucha distorsionado).

Para mostrar la conversión, utilizamos la estructura del filtro LPC de síntesis en términos de los parámetros PARCOR (Partial Correlation). De la estructura de PARCOR [9] tenemos que

$$A_{p-1}(z) = A_p(z) + k_p B_{p-1}(z) \tag{1}$$

$$B_p(z) = z^{-1}\left[B_{p-1}(z) - k_p A_{p-1}(z)\right] \tag{2}$$

donde $A_0(z) = 1$ y $B_0(z) = z^{-1}$. Resolviendo para $p = 1$ en (1) y (2)

$$A_0(z) = A_1(z) + k_1 B_0(z) = 1$$

$$B_1(z) = z^{-1}\left[B_0(z) - k_1 A_0(z)\right] = z^{-1-1}\left[1 - k_1 z\right] = z^{-1-1} A_1\left(z^{-1}\right)$$

Por inducción, obtenemos: $$B_p(z) = z^{-(p+1)} A_p\left(z^{-1}\right) \tag{3}$$

Asumiendo que el filtro PARCOR es estable y el orden es par. Descomponemos $A_p(z)$ en dos funciones de transferencia; una con simetría impar y la otra con simetría par para $k_{p+1} = \pm 1$ respectivamente, entonces

Con $k_{p+1} = 1$ tenemos $A_p(z) - B_p(z) = A_p(z) - z^{-(p+1)} A_p(z^{-1}) = P_{p+1}(z)$ y

Con $k_{p+1} = -1$ tenemos $A_p(z) + B_p(z) = A_p(z) + z^{-(p+1)} A_p(z^{-1}) = Q_{p+1}(z)$

Sumando las dos ecuaciones y despejando para, $A_p(z)$, resulta

$$A_p(z) = \frac{1}{2}\left[P_{p+1}(z) + Q_{p+1}(z)\right] \tag{4}$$

Así, se puede ver que el vector ***a*** (vector de coeficientes del filtro LPC) puede representarse por los valores z_i que son ceros de los polinomios $P_{p+1}(z)$ y $Q_{p+1}(z)$. Recuérdese que $H^{-1}(z) = A(z)$.

Más aún, z_i puede ser representada completamente, porque las magnitudes son iguales a uno, por $\omega_i = \arg(z_i)$; $i = 0,\ldots,9$; para p=10, donde $\arg(\bullet)$ es el argumento de una variable compleja.

Del modelo del filtro sabemos que existen dos raíces ($k_{p+1} = \pm 1$), entonces el orden de $P_{p+1}(z)$ y $Q_{p+1}(z)$ se reduce en uno.

$$P'(z) = \frac{P_{p+1}(z)}{(1-z)} \quad \text{y} \quad Q'(z) = \frac{Q_{p+1}}{(1+z)} \tag{5}$$

Ahora es necesario encontrar 20 ceros. Más aún, todos los z_i son simétricos sobre el eje real y descansan sobre el círculo unitario. Así que, 10 ceros (debajo del eje real) son redundantes y, los otros 10 ceros significantes, corresponden a 10 valores de ω_i para $0 \le i \le 9$. Los parámetros LSPs son las posiciones angulares de las raíces de los polinomios $P'(z)$ y $Q'(z)$ con $0 \le \omega_i \le \pi$. Una propiedad importante es que las raíces de $Q'(z)$ y $P'(z)$ se encuentran alternadas sobre el círculo unitario; lo que satisface: $0 \le \omega_{q,0} \le \omega_{p,0} \le \omega_{q,1} \le \omega_{p,1} \ldots, \le \pi$.

Con esto concluimos que estos 10 valores de ω_i pueden reconstruir todos los ceros de $P_{p+1}(z)$ y $Q_{p+1}(z)$ además de que representan al vector de coeficientes, **a**. Las raíces en el FS1016 se calculan usando el método de bisección o el de Chebyshev.

Habiendo obtenido los valores ω_i, el algoritmo de cuantificación, en el FS-1016, utiliza la tabla $LSPTable[i, j]$, véase [6], que contiene una lista de 8 o 16 valores cuantificados de ω_i, de acuerdo al número de bits asignados para cada uno de los 10 coeficientes (3+4+4+4+4+3+3+3+3+3=34bits). Existen 10 listas, llamadas $lista_i$ para $0 \le i \le 9$. La cuantificación de ω_i se realiza seleccionando j tal que $LSPTable[i, j]$ sea el valor más cercano para ω_i en $lista_i$.

2.5 Cuantificación Vectorial en el CELP FS1016.

Como ya se ha hecho mención, el FS1016 posee dos libros de código que sirven para producir las excitaciones que se introducen al filtro STP para producir la señal de voz sintética que tratará de minimizar el error de predicción cuadrático, MSPE (Minimum Square Prediction Error). Este error es la medida de distorsión considerada para realizar la cuantificación vectorial dentro del codificador que como ya se sabe, se encuentra en el esquema de AbS.

El tipo de cuantificación vectorial utilizada se le conoce como forma-ganancia en paralelo (Parallel Shape-Gain), ya que los libros contienen vectores que tratan de acoplarse a la forma de onda y paralelamente o de manera conjunta se obtiene la ganancia para cada uno de ellos, que es cuantificada de acuerdo a una tabla señalada por la norma. De esta manera al final de la cuantificación tendremos un par de índices que indica la posición dentro del libro de códigos respectivo, además de la ganancia asociada con cada vector.

Los procedimientos de búsqueda del vector óptimo tanto en el libro estocástico como en el adaptable son idénticos, excepto porque actúan sobre una señal objetivo distinta. Para reducir los cálculos, el procedimiento de cuantificación se realiza en dos etapas secuenciales. La primera es realizar una búsqueda en el libro adaptable donde la señal objetivo es la señal de voz pesada perceptualmente mas los errores de codificación introducidos en las tramas anteriores (respuesta de entrada cero) y que afectan la trama presente. La segunda etapa corresponde a la búsqueda en el libro de códigos estocástico cuya señal objetivo es la señal descrita en la primera etapa menos la contribución hecha por la excitación del libro adaptable.

A continuación se describe como se lleva a cabo la búsqueda dentro de cada uno de los libros.

2.5.1 El Libro de Códigos Adaptable.

La búsqueda del "pitch" óptimo se realiza en lazo cerrado tomando vectores de muestras del libro de códigos adaptable con el propósito de minimizar el error cuadrático medio perceptual. A tal libro se le llama adaptable porque a partir de una memoria construida por la suma de dos excitaciones previamente seleccionadas, una proveniente del libro estocástico y otra del adaptable, para un interlocutor en particular, se construye el libro. La **Fig. 1** ilustra la retroalimentación para actualizar

la memoria asociada al libro adaptable a partir de la excitación que se introduce al filtro predictor de síntesis.

Para la codificación se reservan 8 bits, por lo que podemos elegir de un conjunto de 256 vectores. Para cada subtrama impar, la codificación consiste en la elección de un vector relacionado con un retardo (asociado al "pitch") entre 20 y 147 de un conjunto de 128 retardos enteros y 128 fraccionarios. En cambio para las subtramas pares se busca el incremento o disminución en el retardo y se codifica con 6 bits relativo a la subtrama previa. Esto reduce la complejidad y la razón de datos a transmitir sin que se tenga una pérdida considerable en la calidad de la señal de voz. Los parámetros codificados son transmitidos cuatro veces por trama, es decir cada 7.5 ms. La ganancia se codifica con 5 bits empleando una cuantificación escalar no uniforme entre –1 y 2.

Cuando el retardo elegido es mayor o igual que la subtrama entonces se cuenta con un vector de excitación previo que actualiza el libro de códigos adaptable. Pero ¿qué sucede cuando el retardo es menor que la longitud de la subtrama? Cuando esto pasa se observa que no se tiene un vector de muestras completo.

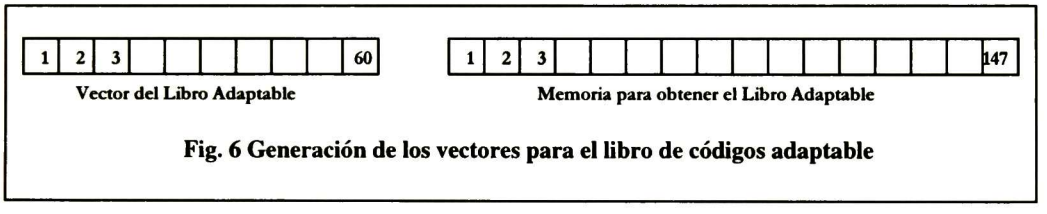

Fig. 6 Generación de los vectores para el libro de códigos adaptable

Con la ayuda de la **Fig. 6** se observa que el problema es llenar un vector de 60 muestras teniendo un "pitch" menor a 60. El procedimiento es ubicarnos en la memoria y a partir del "pitch" extraer las 60 muestras correspondientes (Hace 60, 59, ..., hace 1). Es claro que para un "pitch" < 60 sólo tenemos un número de muestras igual al "pitch" así que es necesario hacer réplicas de estas muestras para formar un vector completo (60 muestras). Como el "pitch" está acotado entre 20 y 147 tendríamos que el FS1016, para un "pitch"=20 forma el vector completo repitiendo 3 veces el pequeño vector de la memoria con 20 muestras. De esta manera el problema queda solucionado.

El procedimiento matemático para construir las palabras de código asociadas con un "pitch" (retardo) entero o fraccionario del libro adaptable, se describe a continuación.

Sea $\boldsymbol{r}$ el libro de códigos adaptable expresado como un arreglo lineal del solapamiento de palabras de código, $\boldsymbol{r} = (r(-147), r(-146), \ldots, r(-1))$, $\boldsymbol{r}'$ un candidato derivado del libro de códigos adaptable, $\boldsymbol{r}' = (r'(0), r'(1), \ldots, r'(59))$ y $\boldsymbol{r}''$ la

concatenación de $\boldsymbol{r}$ y $\boldsymbol{r}'$ Para un retardo entero M, el vector $\boldsymbol{r}'$ se construye a partir del vector $\boldsymbol{r}$, retardado M muestras. Así el candidato para un retardo entero M es:

$$r'_M(i) = r''_M(i) = r''_M(i - M), \text{ donde } i = 0, 1, \ldots, 59; \text{ M} = 20, 21, \ldots, 147 \quad (6)$$

De esta igualdad se aprecia la periodicidad o el "pitch" que se le quiere imprimir a la señal cada retardo ("pitch") M. Recuerde que para retardos $M<60$ es necesario hacer réplicas de las muestras para formar el vector candidato.

Para retardos fraccionarios, las palabras de código se forman por interpolación. La interpolación emplea una ventana de Hamming para calcular los pesos con los que se ponderan las muestras del vector.

$$h(k) = 0.54 + 0.46 \cdot Cos\left(\frac{k}{6N}\right), \text{ donde } k = -6N, 6N+1, \ldots, 6N$$

$$w_f(j) = h(12(j+f)) \cdot \frac{Sen((j+f)\pi)}{(j+f)\pi}, \text{ donde } j = -N/2, -N/2+1, \ldots, N/2-1 \quad (7)$$

$f = 1/4, 1/3, 1/2, 2/3, 3/4$, representa la división intervalo en una fracción de un entero y N, los puntos de interpolación; un valor aceptable es $N = 8$.

Así el retardo fraccionario consiste de una parte entera M mas una parte fraccional f. La siguiente fórmula describe de manera recursiva cómo se calcula la palabra de código r' para un retardo $M+f$.

$$r'_{M+f}(i) = r''_{M+f}(i) = \sum_{j=-N/2}^{N/2-1} w_f(i) \cdot r''_{M+f}(i - M + j) \quad i, M, f, \text{ conservan sus valores.} \quad (8)$$

Finalmente se actualiza el libro de códigos con el vector de excitación seleccionado ***ea***, que indica la suma de las excitaciones estocástica y adaptable escaladas. El desplazamiento se expresa así:

$r(i) = r(i+60)$ para $i = -147, -146, \ldots, -61$ y

$$r(i) = ea(i+60) \text{ para } i = -60, -59, \ldots, -1 \quad (9)$$

2.5.2 El Libro de Códigos Estocástico

La búsqueda de la excitación estocástica óptima se realiza en lazo cerrado tomando vectores de 60 muestras del libro de códigos estocástico con el propósito de minimizar el error cuadrático medio perceptual. La **Fig. 1** ilustra la retroalimentación para seleccionar el vector. La selección se realiza de forma exhaustiva, es decir, se prueba con cada uno de los 512 vectores (también puede ser 256 o 128).

Para la codificación se reservan 9 bits, por lo que podemos elegir de un conjunto de 512 vectores. El procedimiento de elección se describe a continuación.

Se asume que $\mathbf{s}$, $\hat{\mathbf{s}}$ y $\mathbf{e}$ son vectores de dimensión 60 que representan a la señal de voz original, la señal de voz sintética y la señal de error pesada, respectivamente. Por otro sea $\mathbf{v}$ el vector de excitación (forma y ganancia) buscado para la etapa presente y $\mathbf{u}$ el vector de excitación de la etapa anterior (recuerde que se habla de dos etapas secuenciales). Con N como el número de vectores del libro de códigos se utiliza la variable i para realizar el barrido; así $1 \leq i \leq N$. De esta manera cada vector tiene asociada una ganancia optimizada llamada g_i. Basándonos en lo anterior podemos escribir que

$$\mathbf{v}^{(i)} = g_i \mathbf{x}^{(i)} \tag{10}$$

Donde $\mathbf{x}^{(i)}$ representa un vector del libro de códigos estocástico o la forma del vector $\mathbf{v}^{(i)}$

Tómese $\mathbf{H}$ y $\mathbf{W}$ como matrices triangulares cuyas columnas contienen las respuestas al impulso truncadas del filtro de predicción y el filtro perceptual aplicado a la señal de error, respectivamente.

De la **Fig. 1** se puede expresar la voz sintética como la convolución de la respuesta al impulso del filtro predictor con su entrada y su respuesta de entrada cero, $\hat{\mathbf{s}}^{(0)}$.

$$\hat{\mathbf{s}}^{(i)} = \mathbf{H}\left(\mathbf{u} + \mathbf{v}^{(i)}\right) + \hat{\mathbf{s}}^{(0)} \tag{11}$$

Para la señal de error se tiene la siguiente expresión:

$$\begin{aligned} \mathbf{e}^{(i)} &= \mathbf{W}\left(\mathbf{s} - \hat{\mathbf{s}}^{(i)}\right) = \mathbf{W}\left(\mathbf{s} - \left[\mathbf{H}\left(\mathbf{u} + \mathbf{v}^{(i)}\right) + \hat{\mathbf{s}}^{(0)}\right]\right) = \\ &\mathbf{W}\left(\mathbf{s} - \hat{\mathbf{s}}^{(0)}\right) - \mathbf{WH}\left(\mathbf{u} + \mathbf{v}^{(i)}\right) = \mathbf{W}\left(\mathbf{s} - \hat{\mathbf{s}}^{(0)}\right) - \mathbf{WHu} - \mathbf{WHv}^{(i)} \end{aligned} \tag{12}$$

Donde la señal objetivo es entonces:

$$\mathbf{e}^{(0)} = \mathbf{W}\left(\mathbf{s} - \hat{\mathbf{s}}^{(0)}\right) - \mathbf{WHu} \tag{13}$$

Así la ecuación (12) toma la siguiente forma:

$$\mathbf{e}^{(i)} = \mathbf{e}^{(0)} - \mathbf{WHv}^{(i)} \tag{14}$$

Si sustituimos $\mathbf{y}^{(i)} = \mathbf{WHx}^{(i)}$ y usamos $\mathbf{v}^{(i)} = g_i \mathbf{x}^{(i)}$, la ecuación 14 se expresa así:

$$\mathbf{e}^{(i)} = \mathbf{e}^{(0)} - g_i \mathbf{y}^{(i)} \tag{15}$$

Ahora se analiza el error cuadrático para el vector i representado por E_i.

$$E_i = \left\| \mathbf{e}^{(i)} \right\|^2 = \left\langle \mathbf{e}^{(i)}, \mathbf{e}^{(i)} \right\rangle = \mathbf{e}^{(i)T}\mathbf{e}^{(i)}$$
$$= \mathbf{e}^{(0)T}\mathbf{e}^{(0)} - 2g_i\mathbf{y}^{(i)T}\mathbf{e}^{(0)} + {g_i}^2\mathbf{y}^{(i)T}\mathbf{y}^{(i)} \tag{16}$$

Donde T denota la traspuesta. Al analizar la ecuación (16) observamos que el error cuadrático depende del factor de ganancia g_i y del índice i. Así para un valor dado de i, la ganancia óptima puede ser calculada igualando a cero la derivada del error cuadrático con respecto a la ganancia. Esto es:

$$\frac{\partial E_i}{\partial g_i} = -2\mathbf{y}^{(i)T}\mathbf{e}^{(0)} + 2g_i\mathbf{y}^{(i)T}\mathbf{y}^{(i)} = 0 \tag{17}$$

Resolviendo para g_i tenemos que es igual a la correlación cruzada de la señal objetivo (ecuación (13)) y la excitación estocástica filtrada y pesada perceptualmente entre la energía de esta última señal:

$$g_i = \frac{\mathbf{y}^{(i)T}\mathbf{e}^{(0)}}{\mathbf{y}^{(i)T}\mathbf{y}^{(i)}} \tag{18}$$

Como se puede ver, se está realizando una cuantificación de forma (excitación) y ganancia en paralelo.

Minimizar E_i con respecto al índice i equivale a maximizar el negativo de los términos que dependen de i. Esto corresponde a maximizar el siguiente valor:

$$m_i = g_i\left(2\mathbf{y}^{(i)T}\mathbf{e}^{(0)} - g_i\mathbf{y}^{(i)T}\mathbf{y}^{(i)}\right) \tag{19}$$

Sustituyendo la ecuación (18) en la ecuación (19) el valor m_i se convierte en la correlación cruzada normalizada al cuadrado de la excitación filtrada y pesada perceptualmente, y la señal objetivo. Esto es:

$$m_i = \frac{\left(\mathbf{y}^{(i)T}\mathbf{e}^{(0)}\right)^2}{\mathbf{y}^{(i)T}\mathbf{y}^{(i)}} \tag{20}$$

Obsérvese la diferencia entre m_i y g_i. Con estas ecuaciones el cálculo se facilita.

Recapitulando, podemos decir que el procedimiento de búsqueda sobre el libro de códigos estocástico encuentra la palabra de código representada por el índice i, que maximiza el valor m_i. La palabra de código apunta en la dirección más cercana a la señal objetivo que se encuentra en el espacio de dimensión 60 de las señales de voz pesadas perceptualmente. La magnitud del vector elegido se determina por la ganancia g_i.

La **Fig. 7** ilustra a manera de bloques las operaciones descritas con anterioridad.

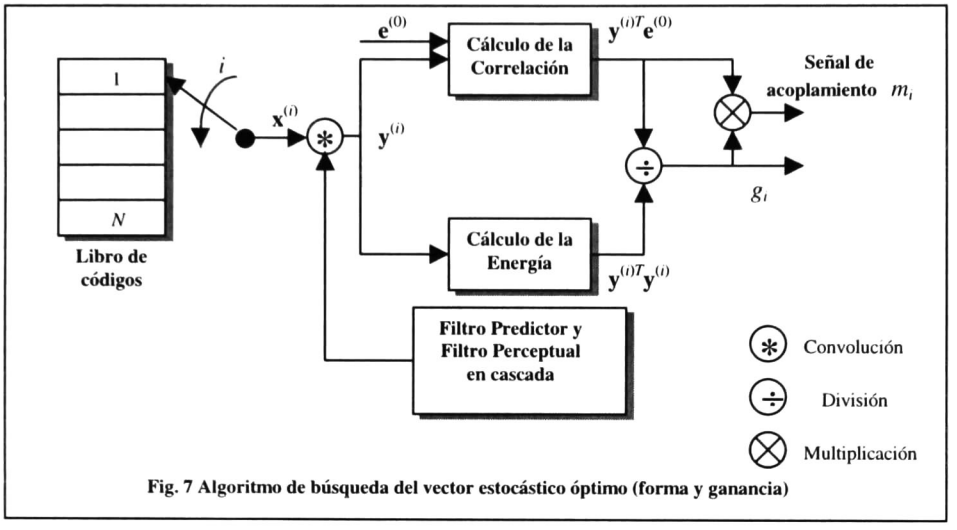

Fig. 7 Algoritmo de búsqueda del vector estocástico óptimo (forma y ganancia)

2.5.3 Evaluación de las Dos Contribuciones.

Las mediciones subjetivas revelan que aunque objetivamente se ha minimizado el error cuadrático de predicción bajo un modelo perceptual, la calidad de la voz puede ser mejorada si se modifica la magnitud (ganancia) de la excitación estocástica. Básicamente se cuenta con un algoritmo adaptable que atenúa la ganancia estocástica cuando el predictor de término largo es eficiente; esta modificación permite que la calidad subjetiva de la voz sea mejorada porque reduce la aspereza y el ruido de cuantificación en los segmentos de voz sonoros. En cambio, cuando la predicción de término largo es ineficiente, la ganancia del vector estocástico se incrementa, proporcionando un acoplamiento subjetivamente favorable entre los segmentos de voz no sonoros de la señal de voz de entrada y la voz sintética [2].

El grado de eficiencia de la predicción de término largo puede ser medido por la similitud, en el sentido de la raíz cuadrada de la correlación cruzada, de la señal objetivo, $\mathbf{e}^{(0)}$, antes y después del cálculo del "pitch". Tomando R como la correlación cruzada normalizada y g_i' como la ganancia modificada para el índice seleccionado, i, tenemos:

$$R = \frac{\left\langle \mathbf{W}\left(\mathbf{s}-\hat{\mathbf{s}}^{(0)}\right), \mathbf{W}\left(\mathbf{s}-\hat{\mathbf{s}}^{(0)}\right)-\mathbf{WHu}\right\rangle}{\left\|\mathbf{W}\left(\mathbf{s}-\hat{\mathbf{s}}^{(0)}\right)\right\|} \tag{21}$$

$$g_i' = \begin{cases} 0.2 g_i, |R| < 0.04 \\ 1.4 g_i (|R|)^{1/2}, |R| > 0.81 \\ g_i (|R|)^{1/2}, COC \end{cases} \quad (22)$$

Posteriormente se realiza la cuantificación y codificación de la ganancia.

Nota.- Los detalles pueden encontrarse en los archivos de programación del CELP FS1016 propuestos en [10].

2.6 Complejidad del Algoritmo.

Según la norma del CELP FS1016 [2], la complejidad estimada para un sistema "full-duplex" es aproximadamente 12.6 millones de instrucciones por segundo, MIPS (Million Instruction Per Second). Estos cálculos comprenden las operaciones de multiplicación, suma, multiplica-acumula y comparación, para el transmisor y el receptor tomados como un sistema "full duplex" Un punto importante que resaltar es que los requerimientos computacionales mayores los domina la búsqueda del vector óptimo dentro del libro de código. Así se tiene que para la búsqueda del vector estocástico óptimo dentro de un libro de 512 vectores, se requieren aproximadamente 8.3 MIPS por trama.

3 Referencias Bibliográficas

[1] B. Atal y M.R. Schroeder. *"Stochastic Coding of Speech Signal at Very Low Bit Rates"*, IEEE Int. Conf. Commun. ICC'84, segunda parte, pp. 1610 -1613.

[2] Campbell, J., V. Welch and T. Tremain, *"The Proposed Federal Standard 1016 4800 bps Voice Coder: CELP"*, Speech Technology, Abril/Mayo 1990.

[3] B.S. Atal, *"Predictive coding of speech at low bit rates,"* IEEE Transactions on Communications., Vol. COM-30, pp. 600 - 614, Abril 1982.

[4] A. Langi, W. Grieder and W. Kinsner, *"Fast CELP Algorithm and Implementation for Speech Compression"*,. Proc. Digital Communications Conference, 1994.

[5] P. Kroon, F. Deprettere, *"A Class of Analysis-by-Synthesis Predictive Coders for High Quality Speech Coding at Rates Between 4.8 and 16 kbits/s"*, IEEE Journal on Selected Areas in Communications, Vol. 6, No.2, Febrero 1988.

[6] J. Campbell et al, *"Archivos de Programación en lenguaje C para el CELP FS1016"*, Tabla de cuantificación de los parámetros LSPs, Biblioteca del CINVESTAV Gdl., 1998.

[7] L. Rabiner, R. Schafer, "Digital Processing of Speech Signal", Prentice Hall.

[8] S. Moreno, *"Introducción al codificador estocástico excitado por código en el dominio temporal"*, Dpto. de Señales, Sistemas y Radiocomunicación, ETSIT.

[9] A. Kondoz, "Digital Speech, Coding for low bit rate communications systems", John Wiley & Sons, 1994.

[10] J. Campbell, *"Archivos de Programación en lenguaje C para el CELP FS1016"*, Biblioteca del CINVESTAV Gdl., 1998.

CAPÍTULO III

CODIFICACIÓN CONJUNTA DE CANAL Y FUENTE (CCCF)

CCCF de la Excitación Estocástica del CELP FS1016 @ 4800 bps

1 Introducción

"La vida es el ajuste continuo del estado interno a las condiciones exteriores".

Spencer.

Los esfuerzos realizados en la codificación de fuente y de canal surgieron a raíz de los estudios de Shannon. Su teorema de separación de la codificación de fuente y de canal [1] enuncia que una comunicación confiable (los errores debidos a la acción del canal tiendan a cero) de una fuente de información sobre un canal ruidoso, puede lograrse empleando un esquema en el que se tengan por separado la codificación de fuente y la codificación de canal. Más aún, estas dos funciones de codificación pueden diseñarse separadamente, sin que para el diseño de una implique el conocimiento de la otra. Sin embargo esto implica utilizar bloques de códigos de longitud grande y como consecuencia un aumento en la complejidad de los algoritmos.

Si se desea trabajar con una comunicación confiable cercana a los límites de Shannon con la condición de tener retardos razonables y bloques de longitud moderada, es necesario considerar un esquema que contemple el diseño de cada una de las funciones de codificación en base a la otra. Dicho de otra forma, se requiere utilizar códigos que de manera conjunta realicen la codificación de fuente y de canal. Es así como se origina la codificación conjunta de canal y de fuente (CCCF) que tiene como propósito encontrar códigos que consideren tanto la cuantificación como una comunicación confiable. Cabe aclarar que no es necesario combinar la codificación de fuente y la de canal, sólo diseñarlas de manera conjunta. Dentro de los diseños de CCCF propuestos, existe una variedad de estructuras y métodos que han sido considerados para este propósito.

Hasta la fecha, no hemos encontrado en la literatura reportes de investigaciones sobre codificación conjunta de canal y fuente, del libro de códigos estocástico que se encuentra en el codificador híbrido de voz CELP FS1016. Es desde aquí de donde parte la motivación de esta investigación.

La plataforma de comunicación considerada para esta investigación es el CELP FS1016 @ 4800 bps, transmitiendo sobre un canal binario simétrico (BSC). Recuérdese del capítulo anterior, que el CELP es una técnica de codificación orientada a tramas que divide la señal de voz en bloques de muestras que son procesadas como una unidad. En los parámetros de la trama de bits que son transmitidos, se incluye a los índices del libro estocástico con sus respectivas ganancias, los índices del libro adaptable con sus ganancias y los parámetros LSP´s (Line Spectral Pairs).

En este capítulo se presenta la utilización de la CCCF aplicada a la excitación estocástica del CELP FS1016 empleando los algoritmos tales como el LBG (Linde, Buzo y Gray) que no pertenece a la CCCF, el COVQ (Channel Optimized Vector Quantization) y el SA (Simulated Annealing) aplicado a la cuantificación vectorial (VQ), que pueden ser vistos como cuantificadores que son modificados para el uso sobre canales ruidosos.

La organización del capítulo será como sigue. La primera sección hablará sucintamente de los antecedentes de la CCCF en general y en esquemas híbridos de voz. Las siguientes secciones describirán los algoritmos LBG, COVQ y SA mostrando la metodología de la simulación. También se describirá el diseño de otros libros de código basados en una mezcla de libros ya conocidos. En la penúltima sección se expondrá una tabla que agrupa los resultados obtenidos y por último, se expresan algunas conclusiones.

2 Antecedentes de la CCCF

2.1 Antecedentes de la CCCF en General.

Una forma de lograr el propósito de la CCCF en el diseño de cuantificadores para operar sobre canales ruidosos, es reemplazar la medida de distorsión con la cual el cuantificador se optimiza, por una medida que también utilice la información asociada a la distorsión ocasionada por un canal ruidoso. Esta simple modificación en la medida de distorsión permite que las estadísticas del canal sean incluidas en el diseño del cuantificador óptimo. Recientemente este método ha sido referido como cuantificación optimizada para el canal, por sus siglas en inglés, COVQ, donde la cuantificación puede ser escalar, vectorial o de trellis.

Esta técnica fue introducida por Kurtenbach y Wintz en 1969 [2]. Se puede destacar de esta investigación las siguientes consideraciones: la fuente puede tener cualquier distribución de probabilidad, la matriz de transición de probabilidades es cualesquiera y se minimiza el error cuadrático medio (MSE) del sistema; el cuantificador es escalar y no se analiza el número óptimo de niveles de cuantificación. Para el año de 1981 Dunham y Gray emplean el teorema de la codificación conjunta de Shannon en un cuantificador de trellis con la distorsión mencionada; pero es hasta 1987 cuando Ayanoglu y Gray proponen una metodología, para el diseño de tales codificadores, basada en el algoritmo de Lloyd.

El algoritmo de Lloyd para cuantificadores vectoriales usando la medida de distorsión modificada fue introducida en 1984 por Kumasawa [3] y con esto nace el primer cuantificador vectorial para canales ruidosos.

Otros esfuerzos en este campo fueron hechos por Farvardin en el año de 1987 y 1990. Farvardin en [4] sugiere un cuantificador vectorial para canales ruidosos. Este cuantificador emplea la técnica de temple simulado por sus siglas en inglés SA (Simulated Annealing), para encontrar el código que optimice el sistema mediante la minimización del error cuadrático medio. La medida de distorsión se modifica para tomar en cuenta el canal ruidoso. Esta técnica es una de las seleccionadas en la presente investigación por lo que más adelante se entrará en detalles.

Es importante notar que los nuevos algoritmos que fueron surgiendo hasta nuestros días toman en cuenta no sólo la minimización del error cuadrático medio, sino el número óptimo de niveles de cuantificación a codificar y el acomodo de los mismos para conformar lo que llamamos el libro de códigos.

El concepto de codificación conjunta de fuente y canal poco a poco se fue extendiendo en los campos de la codificación de imágenes y de señales de voz. En los párrafos siguientes se pondrá particular atención en la CCCF aplicada a esquemas híbridos de voz.

2.2 Antecedentes de la CCCF en Esquemas Híbridos de Voz.

En el año de 1990, Bastian Kleijn [5], operando sobre un CELP, emplea un cuantificador para canales ruidosos con la técnica de temple simulado aplicada a los índices de la excitación estocástica y adaptable, así como a sus respectivas ganancias. Aunque sus resultados indican no lograr una mejora considerable, Kleijn ha demostrado que el uso de cuantificadores para canales ruidosos es un medio práctico y eficiente que mejora la tolerancia de errores de los codificadores diseñados para un rango de errores de canal. Empleando estos procedimientos se logra tener una buena calidad de la señal de voz, sin requerir el uso de mayor redundancia. Algunos puntos que descarta el autor son: la importancia de los parámetros asociados al filtro LTP ya que estos proporcionan una contribución dominante para los sonidos sonoros, de manera que infringir en errores de canal tiene como consecuencia una pérdida del carácter sonoro de la voz; en cambio el efecto en errores asociados a la excitación estocástica es usualmente menor y frecuentemente imperceptible.

Para el año de 1992, Rowe y Secker [6], lanzan una propuesta de codificación de los coeficientes LPC de un MBE-LPC (Multi-Band Excitation) @ 2400 bps, utilizando el esquema de trellis que considera la distorsión del canal propuesto por Ayanoglu y Gray en 1987. Los autores destacan obtener una robustez notable a los errores en el canal sin incrementar la razón de bits.

Popescu et al [7], en el año de 1995, publican su investigación realizada sobre el CELP de Atal, en el que los filtros LTP y STP están en cascada, empleando un cuantificador de trellis que no toma en cuenta las estadísticas del canal e

intercambia aumento en la complejidad por un aumento en la calidad de la señal de voz. Es importante reafirmar que, en su trabajo, Popescu et al no aplican el concepto de CCCF en la medida de distorsión del cuantificador trellis, sin embargo realiza un arreglo de índices, que pertenece a las técnicas de CCCF, para la organización de los vectores del libro de códigos estocástico.

Un trabajo relativamente reciente, 1996, de Alajaji [8] expone la implementación de un decodificador que utiliza una técnica de codificación conjunta para explotar la redundancia residual (redundancia que el codificador fuente no alcanza a eliminar) durante la decodificación de los parámetros LSP's del CELP FS1016. Para esto, diseña el decodificador que emplea un modelo de la fuente de parámetros LSP's, basado en tal redundancia. Alajaji afirma que aproximadamente un tercio de los bits utilizados en la cuantificación de los parámetros LSP's son redundantes, por lo que propone aprovechar la redundancia en la corrección de errores.

3 Tres Métodos de CCCF

Existen una variedad de métodos para aplicar la CCCF a un sistema de comunicación, sin embargo, para este trabajo nos remitiremos a la descripción y estudio de tres métodos básicos descritos por Zahir [8].

3.1 Protección Jerárquica.

La principal motivación para el uso de esta técnica estriba en que una vez que se han estudiado las características de la fuente y se ha realizado una clasificación de los parámetros a trasmitir de acuerdo con los efectos que se incurren al presentarse errores en el canal, como la mencionada por Kleijn [5] para los parámetros del CELP, es posible dedicar mayor protección a la información más sensible, la cual se sospecha de contribuir con grandes errores en el momento de la decodificación. Pues bien, este método es conocido como protección desigual contra errores o protección jerárquica.

Si recordamos, el FS1016 dedica una codificación de canal (Hamming (15,11)) exclusiva a los parámetros asociados al "pitch", debido a que el oído humano es más sensible a los errores ocasionados por un "pitch" erróneo que a errores en cualquier otro parámetro transmitido.

Otro ejemplo que ayuda a ilustrar el uso de esta técnica es aquel en el que de antemano se sabe que los bits más significativos pertenecientes a la representación binaria de un vector de códigos, son más sensibles a los errores de canal que los bits menos significativos. Entonces se procede a proteger únicamente los bits más significativos de la palabra de código.

La **Fig. 1.** muestra un esquema de codificación conjunta de canal y fuente utilizando la técnica de protección jerárquica. Se destaca el bloque de clasificación que puede consistir simplemente en un separador de los bits asociados a un parámetro

específico que se quiere transmitir. Después de la clasificación, los bits pasan por diferentes tipos de codificación de canal.

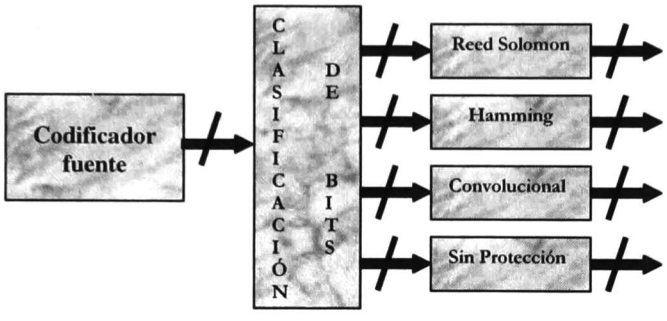

Fig. 1. Diagrama a bloques de CCCF con protección jerárquica

3.2 Asignación de Índices.

Zahir comenta en su publicación que la metodología utilizada para asignar los índices con los que se representa a los vectores de códigos, no afecta la distorsión promedio si se tiene ausencia de ruido en el canal, mientras que en la presencia de ruido en el canal, esta asignación juega un papel determinante en el desempeño de la cuantificación vectorial. Para comprobar lo anterior se realizó el experimento de intercambiar los vectores en el libro de códigos estocástico del FS1016, de manera que los índices tuvieran otra correspondencia. Con la excepción de que el último vector siempre fuera el 512, por cuestiones de la implementación, la SSNR (Segmental Signal to Noise Ratio) fue la misma.

Ahora planteemos el problema cuando transmitimos sobre un canal ruidoso. Al transmitir un índice (palabra binaria) el ruido actuará en algunos de los bits de dicho índice de manera que el vector representado será otro; cuando el decodificador utilice el vector indicado por el índice erróneo, se producirán los errores en la reproducción. Pero, ¿qué sucede si el índice erróneo representa a un vector muy parecido al correcto? Es claro que los efectos ocasionados por el índice erróneo serán menores en la reproducción. Así pues, dice la técnica de asignación de índices, organicemos el libro de códigos de manera que los centroides más parecidos estén "más cerca" para que cuando se tengan índices erróneos, el error conlleve a utilizar un índice que represente al centroide más parecido al original.

Una solución general al problema de asignación de índices (AI) es realizar primeramente el diseño de la cuantificación vectorial y posteriormente permutar los índices de tal manera que el libro de códigos resultante sea más robusto contra el ruido del canal.

La **Fig. 2.** muestra como se lleva a cabo la codificación para canales ruidosos utilizando la técnica de asignación de índices. El vector de datos **x** es codificado por la función **Q** que produce un índice **i**. La función de permutación π cambia la posición dentro del libro de códigos de este índice. Esta información se transmite por el canal binario simétrico donde se contamina con ruido. Al llegar la información al receptor se produce el índice **i** ′, el cual es introducido al decodificador para generar el vector reproducido **y** ′.

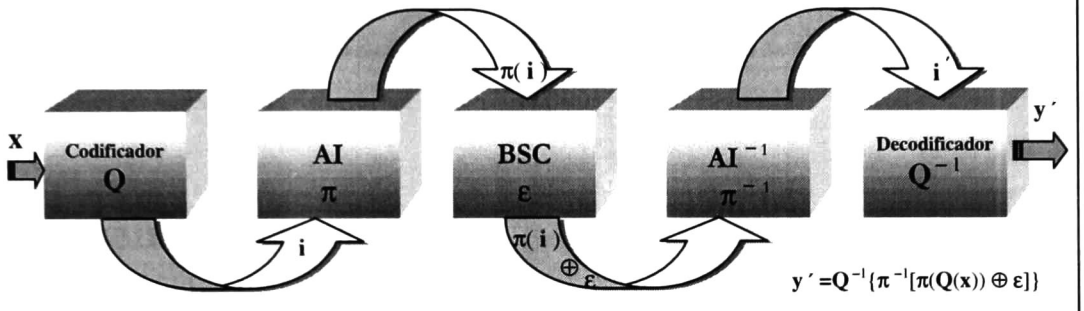

Fig. 2. Diagrama de bloques de un sistema de CCCF con Asignación de Índices

Dos métodos pertenecientes a este tipo de codificación que se utilizan para el diseño de un libro de códigos son el algoritmo de conmutación de índices y el temple simulado. Este último se describirá en esta investigación.

3.3 Cuantificación Vectorial Óptima y Asignación de Índices.

Esta técnica de CCCF tiene como propósito diseñar el cuantificador vectorial que sea óptimo para el uso sobre canales ruidosos empleando una función de asignación de índices incluida en el codificador. En pocas palabras podemos decir que este tipo de codificación intercambia exactitud en la cuantificación por menor sensibilidad al ruido en el canal.

Un ejemplo de este tipo de CCCF es el método llamado Cuantificación Vectorial Optimizada para el Canal, mejor conocido por sus siglas en inglés como COVQ (Channel Optimized Vector Quantization).

La **Fig. 3.** muestra un diagrama de bloques de este tipo de CCCF. La notación está basada en la **Fig. 2.** Nótese que la función de asignación de índices y su función inversa están incluidas en el codificador y decodificador respectivamente.

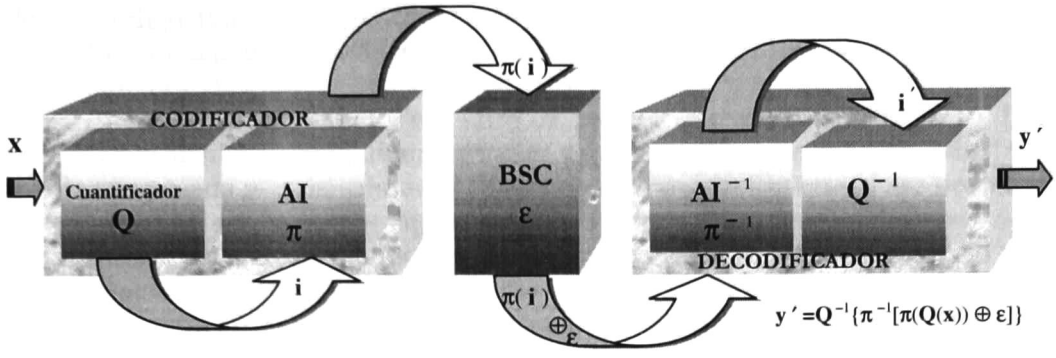

Fig. 3. Sistema de CCCF con VQ óptima y Asignación de Índices

4 Diseño de un Libro de Códigos para el CELP FS1016

Como se ha visto en el capítulo anterior, el CELP posee un libro de códigos estocástico que consideramos como centroides o vectores representativos de lo que conocemos como el vector residual o simplemente residual de la señal de voz. El libro de códigos del FS1016, como ya vimos, tiene características particulares como son la cantidad de ceros, el desfasamiento de muestras entre vectores y el tipo de valores de las muestras (-1, 0, 1). Este libro de códigos ha demostrado ser bastante bueno debido a que disminuye la complejidad del codificador y se almacena en una pequeña memoria para 1082 muestras con valores enteros y a partir de ellas se genera el libro completo.

Como el propósito de este trabajo es aplicar la CCCF para la excitación secundaria, se procedió a diseñar un nuevo libro de códigos que considere las características del canal. Un primer acercamiento, no considerado dentro de la técnica de CCCF, se realizó utilizando el algoritmo LBG, posteriormente se diseñaron otros dos libros con el COVQ y el SA, algoritmos que sí pertenecen a la CCCF; por último, se diseñaron algunos libros utilizando mezclas de libros ya generados.

4.1 Libro de Códigos usando el Algoritmo LBG.

4.1.1 Objetivo.

Usando el algoritmo generalizado de Lloyd o algoritmo LBG y una secuencia de entrenamiento, encontrar un libro de códigos para la excitación del codificador híbrido CELP FS1016. El algoritmo se modifica para encontrar la mínima distorsión en el espacio de señales sintetizadas por el filtro que modela el tracto y no en el espacio de residuales.

4.1.2 Descripción del Algoritmo LBG bajo un esquema de AbS.

Dado un código inicial ***q*** de excitaciones y una secuencia de entrenamiento ***s*** se calculan los coeficientes del filtro predictor para una determinada subtrama de ***s*** usando el método de autocorrelación. Con estos coeficientes se construye un filtro de síntesis ***H(w)***. Se pasa cada vector del libro de códigos inicial a través del filtro **H(w)** y se calcula la distancia euclidiana entre la subtrama (60 muestras de valor real) actual de voz pesada ***p*** y la salida del filtro ***H(w)***. La excitación que al ser procesada por ***H(w)*** resulte ser la mejor aproximación (menor distancia euclidiana) de la subtrama se elige para representar la señal. Cada palabra de código o residual tiene entonces asociadas ciertas subtramas de la secuencia de entrenamiento, mismas que forman una región en el espacio de señales de voz. Una vez que a todas las subtramas de ***s*** se les ha asignado un vector de excitación del libro de códigos se procede a calcular los nuevos centroides a partir de las regiones obtenidas.

El centroide para cada región se calcula de la siguiente manera:

- Se traslada el conjunto de subtramas de la señal de voz pesada, asociadas a un residual en particular, al espacio de los residuales utilizando los filtros de síntesis construidos con los parámetros obtenidos en el análisis de cada segmento de voz.

- El nuevo centroide es igual a la suma de los residuales entre el número de residuales.

El nuevo centroide sustituye ahora al residual que se encuentra en el libro de códigos.

Excepto por el cálculo de los coeficientes de predicción y la voz pesada, las operaciones se realizan de manera iterativa hasta lograr una cierta medida de distorsión global.

El esquema de la **Fig. 4.** ilustra a manera de bloques la estructura y organización del algoritmo LBG bajo un esquema de AbS de manera iterativa.

En [9] se encuentra el programa utilizado para la simulación de este algoritmo.

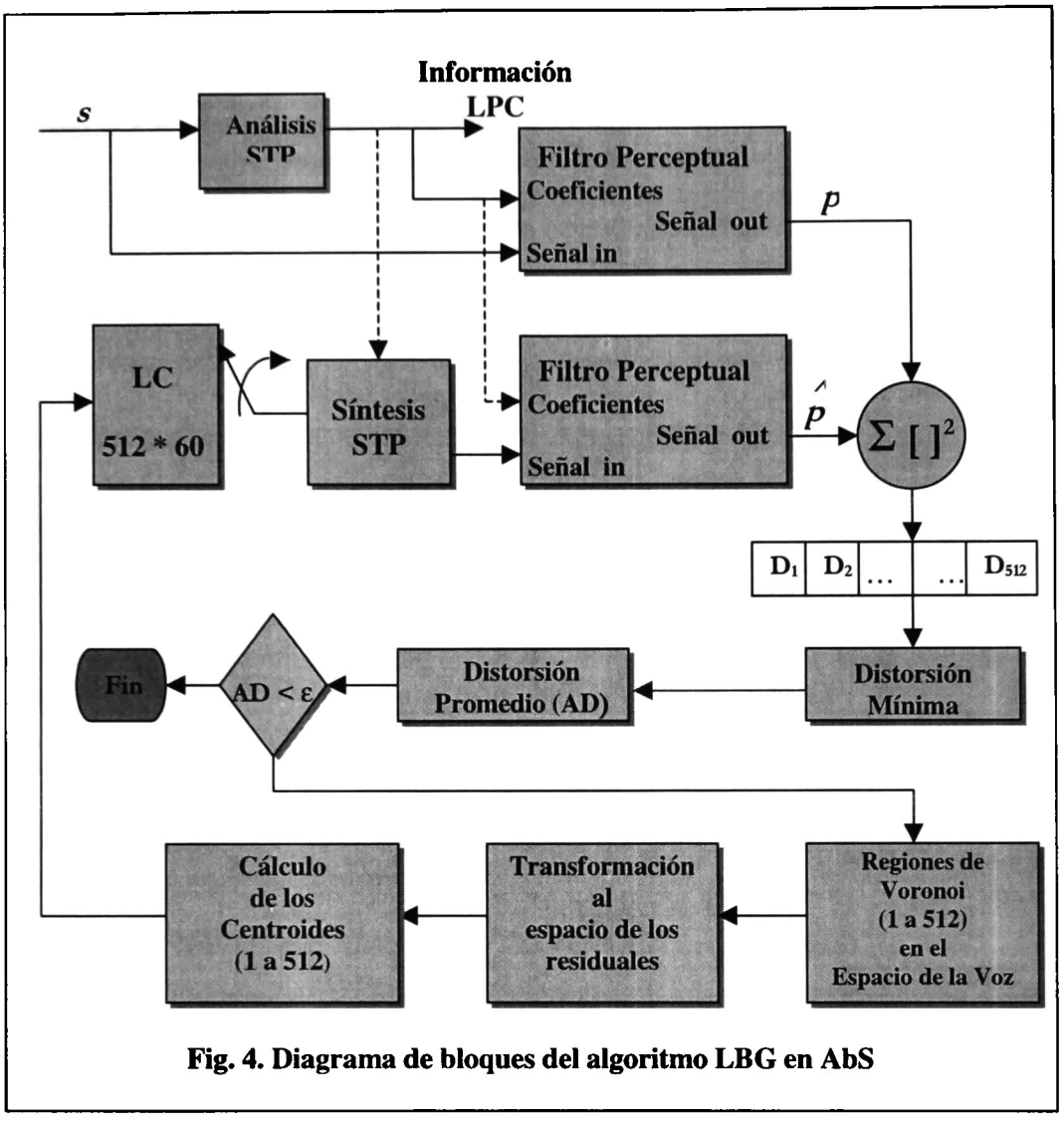

Fig. 4. Diagrama de bloques del algoritmo LBG en AbS

4.1.3 Simulación del Experimento.

Como libro de códigos inicial se utilizó un libro formado por muestras aleatorias Gaussianas con media cero y varianza uno. La secuencia de entrenamiento tiene las siguientes características:

Archivos de entrada :
T_S_voz.spd.
Formato: Crudo.
Idioma: Inglés.
Timbre: Cuatro voces de mujer y cuatro de hombre.

No. de muestras: 240000 muestras (30s) aproximadamente.
Razón de muestreo: 8 Khz.

El programa se realizó en Matlab y posteriormente se compiló en lenguaje C++ con una tasa de error o umbral de terminación de 0.001 El algoritmo obtuvo la convergencia en 39 iteraciones. Posteriormente se utilizó el libro generado para implantarlo en el FS1016 y sustituir el libro de códigos ternario.

En la **Tabla 1** de la sección 5 se muestran los resultados para un archivo de voz codificado por el CELP FS1016 con el libro de códigos generado por el algoritmo LBG en AbS (cb20LBG).

4.2 Libro de Códigos usando el Algoritmo COVQ.

Como se mencionó en la sección 3.3, el algoritmo COVQ permite el diseño de un libro de códigos para tener CCCF mediante la técnicas de cuantificación óptima y asignación de índices simultáneamente. Este algoritmo es muy parecido al LBG sólo que la medida de distorsión se modifica, además de utilizar una asignación de índices que minimice el error bajo condiciones de canal ruidoso. La medida de distorsión involucra entonces al error de cuantificación y al error debido a la perturbación en el canal.

4.2.1 Objetivo.

Realizar una modificación al algoritmo LBG descrito en el punto 4.1 para encontrar un libro de códigos que tenga una distribución de índices que minimice el error bajo condiciones de un canal ruidoso.

4.2.2 Descripción del Algoritmo COVQ bajo un esquema de AbS.

Utilizando un modelo de canal binario simétrico con una cierta probabilidad de error es posible construir una matriz de transición que contenga las probabilidades condicionales de recibir un índice j dado que se transmitió un índice i. Dicha matriz se usa para modificar la medida de distorsión que se emplea para determinar el vector de código más apropiado que represente a la señal de voz pesada. Lo que se elige en realidad es un índice que representa una región cuyo centroide se calcula considerando la matriz de transición. Kumazawa propone la siguiente medida de distorsión:

$$D = \sum_{j=1}^{N} P(j/i) * d(p(n), \hat{p}(j)) \qquad \forall \quad 1 \le i \le N \tag{1}$$

Donde $P(j/i)$ representa la coordenada (i, j) dentro de la matriz de transición de probabilidades P y $d(a, b)$ indica la distancia euclidiana entre los argumentos a y b. Los vectores a comparar son p (señal de voz pesada) y $\hat{p}$ (señal de voz sintetizada) para una subtrama Esta medida de distorsión es calculada para cada elemento del libro de códigos y se elige el vector q que la minimice. De esta forma se construyen

regiones formadas por todos los segmentos de voz de la secuencia de entrenamiento que fueron estimados usando la síntesis del residual correspondiente a la palabra de código.

Los nuevos centroides se calculan utilizando siguiente ecuación:

$$q_{nuevo(l)} = \frac{\sum_{j=1}^{N} P(l/j) * \sum_{i:p_i \in R_j} p_i}{\sum_{j=1}^{N} P(l/j) * |R_j|} \qquad 1 \le l \le 512 \qquad (2)$$

donde $|R_j|$ representa la cantidad de subtramas de la secuencia de entrenamiento que fueron cuantificadas con el residual $q_{(l)}$, R_j representa la región asociada al residual $q_{(l)}$ y la sumatoria $\sum_{i:p_i \in R_j} p_i$ indica la suma de todas las subtramas de voz pesada p_i que pertenezcan a la región R_i.

Empleando estas fórmulas de manera iterativa obtenemos el libro de códigos robusto al canal que se pretende.

4.2.3 Simulación del Experimento.

Como libro de códigos inicial se utilizó un libro formado por muestras aleatorias Gaussianas con media cero y varianza uno. La secuencia de entrenamiento fue la misma que se usó para el experimento con el LBG y la tasa de error fue de 0.001

El programa se realizó en Matlab y posteriormente se compiló en lenguaje C++. El algoritmo obtuvo la convergencia en 25 iteraciones. Posteriormente se utilizó el libro generado para implantarlo en el FS1016 y sustituir el libro de códigos ternario.

En la **Tabla 1** de la sección 5 se muestran los resultados para un archivo de voz codificado por el CELP FS1016 con el libro de códigos generado por el algoritmo COVQ en AbS (cb20COVQ). Se aprecia que la SSNR aumenta aproximadamente en 1 dB.

4.3 Libro de Códigos usando el Algoritmo de Temple Simulado (SA).

Este algoritmo es utilizado para problemas de optimización combinatoria y quizás, parafraseando a Farvardin [4], lo más relevante a favor de la cuantificación vectorial para canales ruidosos, es el trabajo en el cual el SA es usado para diseñar códigos fuente. Como se vio en la **sección 3.2**, el algoritmo de temple simulado (SA) pertenece a la técnica de asignación de índices.

El algoritmo de temple simulado tiene sus fundamentos en la descripción del proceso de templado para la producción de cristales (metales, etc) cuya finalidad es enfriar poco a poco el material de manera que éste adquiera la máxima fuerza de cohesión, que puede traducirse como la mínima energía potencial almacenada por la estructura cristalina. De esta manera, la temperatura del material al inicio del proceso de templado es la de fundición; luego se perturba el estado en que se encuentra y la temperatura aumenta o disminuye. El herrero al observar el color del material decide si lo vuelve a calentar o baja más la temperatura introduciéndolo en el agua. Así el algoritmo de SA observa una función de costo de muchas variables y en base a los resultados decide si aumenta o disminuye la temperatura mediante un factor específico que representa la probabilidad o la certeza de que se va avanzando en el camino correcto o hacia la minimización de la función objetivo.

Los experimentos reportados en la literatura señalan que si en el proceso de templado se disminuye la temperatura de manera muy fina, se logra llegar al mínimo global; sin embargo este proceso es demasiado tardado. El algoritmo de SA trata de llegar al mínimo global de una manera más rápida que el procedimiento anterior, al librarse ("hill climbing") de los mínimo locales cuando la temperatura es bastante alta.

La implementación de dicho algoritmo en codificadores híbridos ha sido reportado por Bastian Kleijn [5]

4.3.1 Objetivo.

Ubicar las palabras de código de un libro estocástico del CELP FS1016 mediante la técnica de SA de modo que al ocurrir un error debido al ruido en el canal, los índices corruptos conduzcan a palabras de código similares a la que constituye el centroide del algoritmo.

4.3.2 Descripción del Experimento.

Con un canal binario simétrico como modelo de canal se utiliza una matriz de transición de probabilidades para calcular una función de costo para medir las consecuencias de recibir un índice j dado que se envió un índice i. La medida de distorsión es la representación analítica de un parámetro objetivo de comparación que permite asociar los índices que menor distorsión produzcan al momento de la decodificación. El algoritmo iterativo perturba los índices del libro de códigos con el propósito de minimizar la función de costo. A cada arreglo o combinación de índices del libro de código se le conoce como estado. En el caso concreto del CELP donde se tiene una asignación de índices, se persigue minimizar una medida de distorsión basada en el promedio de la diferencia entre las distorsiones producidas entre dos asignaciones de índices distintas.

En otras palabras podemos decir que cuando enviamos un índice a través del canal existe cierta probabilidad de que el receptor reciba uno distinto que será decodificado como una palabra diferente. Asumiendo que la palabra decodificada representa una excitación para sintetizar un segmento de voz, la señal sintética

estará a una cierta distancia de la señal objetivo. Esta distancia se multiplica por la probabilidad condicional de recibir el índice j dado que se envió el índice i, y se promedia para todas las posibles transiciones. Este promedio corresponde a la asignación o estado en particular. Se busca entonces, la asignación que minimice dicho promedio.

La primer información a calcular es la señal objetivo o señal de voz pesada a partir de la secuencia de entrenamiento propuesta en la sección 4.1.3.

El segundo cálculo es la probabilidad a priori referida a la elección de los vectores de un libro estocástico (ternario, gaussiano, etc.) la cual se calcula agregando una pequeña rutina en el programa de simulación del CELP FS1016 [11], para calcular la frecuencia de elección de cada vector. De esta manera se obtiene un histograma que representa la frecuencia de elección de cada vector (1 a 512) para una secuencia de entrenamiento dada, que si es lo suficientemente grande, entonces la probabilidad a priori será bastante representativa.

Otra información necesaria es el cálculo de las distancias de Hamming de todas las palabras de código. Es decir, la distancia de Hamming entre cada palabra de código con respecto a las demás.

Al contar con la información anterior se procede como lo indica el siguiente algoritmo:

1. Se tiene una temperatura inicial llamada T, que se manipula en el algoritmo.

2. Se obtiene una medida de distorsión promedio (ecuación 3), para el arreglo de índices inicial o estado inicial $b \equiv (b(\mathbf{c}_1), b(\mathbf{c}_2), \cdots, b(\mathbf{c}_M))$. Esta nomenclatura se lee como el índice b asociado con la palabra de código $\mathbf{c}_i$ M es un entero positivo cuya representación binaria es la palabra binaria asignada a la palabra de código $\mathbf{c}_i$. $P(\mathbf{c}_i)$ es la probabilidad a priori del vector $\mathbf{c}_i$ y $d(\mathbf{c}_i, \mathbf{c}_j)$ es la distancia euclidiana entre $\mathbf{c}_i$ y $\mathbf{c}_j$.

$$D_c(b) = \frac{1}{k} \sum_{i=1}^{M} \sum_{j=1}^{M} P(\mathbf{c}_i) P\left({}^{b(\mathbf{c}_j)}\!/\!_{b(\mathbf{c}_i)} \right) \cdot d(\mathbf{c}_i, \mathbf{c}_j) \qquad (3)$$

3. En forma aleatoria se intercambian dos índices y se calcula una nueva medida de distorsión, para este nuevo estado, llamada $D_c(b')$.

4. Se calcula la diferencia $\Delta D_c = D_c(b') - D_c(b)$

   a.) Si $\Delta D_c < 0$ se acepta el reemplazo de b con b'.

   b.) Si $\Delta D_c \geq 0$ se acepta el reemplazo de b con b' pero con una probabilidad igual a $\exp\{-\Delta D_c / T\}$; donde *T es la temperatura del sistema.*

5. Si en el paso 3 el número de caídas de energía excede un número prescrito o si hay muchas perturbaciones que no resultan en caídas de energía, entonces ir al paso 5, de lo contrario, ir al paso 2.

6. Disminuir la temperatura T. Si T está debajo de una temperatura de congelación preescrita T_f o si aparentemente se ha alcanzado un estado estable, el algoritmo se detiene con los valores de b se detiene con el estado b como estado final, de lo contrario se va al paso 2.

4.3.3 Simulación del Experimento.

Se tomó como libros iniciales al libro de códigos ternario y un libro gaussiano con media cero y varianza uno. Sin perturbaciones en el cambio de posición dentro del libro se dice que el proceso se encuentra en su estado inicial.

La secuencia de entrenamiento fue la misma que para el LBG y el COVQ. Las tasas de error para las simulaciones fueron de 0.001 y 0.01.

En la **Tabla 1** de la sección 5 se muestran los resultados para un archivo de voz codificado por el CELP FS1016 con el libro de códigos generado por el algoritmo SA en AbS (cb20COVQ). Se aprecia que no hay un cambio representativo en la SSNR al aplicar el algoritmo SA al libro ternario y al gaussiano.

4.4 Libros de Código Mixto.

Un problema que se presenta al utilizar libros de código que contienen muestras con valor real es la cantidad de memoria requerida para almacenar tales muestras. Por ejemplo, si el libro de códigos es de 9 bits para referenciar vectores de 60 muestras cada uno, entonces el número de localidades a almacenar será de $2^9 \times 60$, esto corresponde a 30790 localidades. Kondoz [12] describe algunas técnicas como son el entrelazado y el esparcimiento de ceros; el autor presenta también resultados para diferentes libros basados en un código gaussiano.

El entrelazado consiste en tener un conjunto de muestras de las cuales se puede formar el libro de códigos; cada vector del libro de códigos varía de los otros en ciertas muestras únicamente. Esto ayuda a tener menos localidades de almacenamiento y reducir los cálculos durante la búsqueda dentro del libro de códigos. Por otro lado, el esparcimiento de ceros consiste en tener una serie fija de muestras con valor cero después de una muestra distinta de cero. Esto permite que los cálculos tengan menor complejidad, ya que en las implementaciones del CELP se pueden librar algunos cálculos cuando las muestras son cero.

De acuerdo a las motivaciones anteriores se procedió durante la investigación a probar con diferentes arreglos de libros de código conformados por:

- Una población del tipo ternario-gaussiano. La organización de este libro es tener una primera sección que agrupa a la primera mitad (256 vectores) del libro ternario perteneciente al CELP FS1016; la segunda sección agrupa a un conjunto

de muestras (256 × 60) con distribución gaussiana obtenidas en Matlab. Al libro se le llamó cbGT1. Los resultados obtenidos al utilizar este libro se muestran en la **Tabla 1** del punto 5.

- Una población del tipo gaussiano-esparcido. Este libro consiste en una transformación del libro ternario perteneciente al CELP FS1016 mediante la siguiente ecuación:

  *cbGSPAR = codebook * Abs(cbGAUSS)*

  Donde *codebook* representa el libro de códigos ternario, *Abs* es la operación de valor absoluto y *cbGAUSS* es un libro de códigos gaussiano de 512 × 60, obtenido en Matlab.

  En la **Tabla 1** de la sección 5 se muestran los resultados obtenidos con este libro.

5 Comparación de Resultados

La **Tabla 1** agrupa los resultados obtenidos para los diferentes experimentos expuestos en las secciones anteriores. Una explicación sencilla acerca La nomenclatura se expone en la columna de comentarios.

Tabla 1. Resultados para diferentes libros de código

Libros de Código	Segmental SNR (BER=0.0)	Segmental SNR (BER=0.001)	Comentarios :
origin	10.39 dB	9.25 dB	LC del FS-1016 CELP
cbOSA	10.39 dB	9.23 dB	LC con Temple Simulado para (BER=0.01) a partir del LC original
cbGAUSS	10.55 dB	9.51 dB	LC gaussiano
cbGSA	10.53 dB	9.52 dB	LC con Temple Simulado para (BER=0.001) a partir de un LC gaussiano
cb20LBG	11.18 dB	10.01 dB	LC generado con el algoritmo LBG
cb20COVQ	11.38 dB	9.94 dB	LC generado con el algoritmo COVQ
cbGT1	10.58 dB	9.32 dB	LC Ternario-Gaussiano
cbGSPAR	10.61 dB	9.46 dB	LC Sparse-Gaussian

La SNR segmental toma en cuenta únicamente las subtramas acotadas entre −10 y 64 dB

El número de convoluciones por trama aumenta de 1.029573 (Dato proporcionado por la norma del CELP FS1016) al siguiente valor:

C =[(1+2+... +30) + (30)*(30)]*(512*4)/10^6 = (465+900)*(2048)/10^6 = 2.7955 MIPS

Esto se debe a que no se maneja un solapamiento entre vectores como se tiene en el libro ternario del CELP FS1016.

6 Conclusiones

- Los resultados obtenidos muestran que el libro diseñado con el método COVQ es el que tuvo mejores resultados en la relación SSNR en ausencia de ruido. Aproximadamente 1 dB más que con el libro ternario del CELP FS1016. Además el archivo de voz codificado-decodificado presenta buena calidad e inteligibilidad comparándolo con el archivo de voz sin procesar.

- El libro de códigos diseñado con el algoritmo LBG fue con el que se obtuvo los mejores resultados para el archivo de voz presentado en la **Tabla 1** bajo un BER=0.001.

- El costo de tener mayor SSNR implica mayor complejidad en el algoritmo, pues como se muestra en la sección 5 el número de convoluciones es mayor que la utilizada por la norma con el libro ternario.

- A manera de recomendación, este trabajo sugiere utilizar la norma del CELP FS1016 con su libro ternario cuando se quiere menor complejidad y cuando se quiere mayor calidad en la voz es posible utilizar un libro como lo es el diseñado con el método del COVQ.

- Otra ventaja que ofrece el libro de códigos del CELP FS1016 en comparación con los otros libros diseñados es la utilización de una memoria pequeña que almacene sólo 1082 muestras con las que se genera el libro ternario, en cambio para los otros libros se necesita una memoria para 30790 localidades.

7 Referencias Bibliográficas

[1] T. Cover , *"Elements of Information Theory"*, John Wiley & Sons, 1991, pp. 215-219.

[2] A. Kurtenbach y P. Wintz, *"Quantizing for Noisy Channels"*, IEEE Transactions on Communication Technology, Vol. COM-17, No. 2, abril de 1969.

[3] H. Kumasawa, *"A Construction of Vector Quantizers for Noisy Channels "*, Electronics and Engineering in Japan, Vol. 67-B, No. 4, 1984.

[4] Farvardin, *"A Study of Vector Quantization for Noisy Channels"*, IEEE Transactions on Information Theory, Vol. 36, No. 4, julio de 1990.

[5] B. Kleijn, *"Source-Dependt Channel Coding for CELP"*, AT&T Bell Laboratories, Naperville, IL 60566, USA..

[6] D. Rowe, P. Secker, *"A Robust 2400 bit/s MBE-LPC Speech Coder Incorporating Joint Source and Channel Coding"*, Department of Electrical and Computer Engineering, The University of Wollogong, P.O. Box 1144, Wollogong, NSW 2500, Australia, 1992.

[7] A. Popescu, N. Moreau, C. Lamblin,*"CELP Coding Using Trellis-Coded Vector Quantization of the Excitation"*, IEEE Transactions on Speech and Audio Processing, Vol. 3, No. 6 Noviembre 1995.

[8] S. Zahir, P. Duhamel, O. Rioul, *"Combined Source-Channel Coding : Panorama of Methods"*, CNES Workshop on Data Compression : 13-14 Nov. 1996, Toulouse.

[9] Apéndice Programación del Algoritmo LBG en Análisis por Síntesis.

[10] P. Dymarski et al. *"Optimal and sub-optimal algorithms for selecting the excitation in linear predictive coders"*, Proc. ICASSP, pp 485-488, 1990.

[11] O. Longoria, R. Rodríguez, *"Archivos de Programación en lenguaje C para el CELP FS1016 con TCVQ"*, Biblioteca del CINVESTAV Gdl., 1998.

[12] A. Kondoz, *"Digital Speech"*, John Wiley & Sons, 1994.

[13] C. Thomas, Elements of Information Theory, John Wiley & Sons, Inc., 1991.

CAPÍTULO IV

CODIFICACIÓN DE TRELLIS EN EL CELP FS1016

Codificación Vectorial Trellis para la excitación estocástica del codificador CELP FS1016

1 Introducción

"La memoria es la que condiciona nuestros actos presentes en función del impacto de los mensajes del mundo exterior y del residuo de los actos pasados que constituyen nuestro aprendizaje y nuestra experiencia".

Abraham A. Moles.

En la cuantificación vectorial sin memoria, cada símbolo o vector cuantificado es determinado por el vector presente; en cambio la cuantificación vectorial con memoria que utiliza una máquina de estados (FSVQ), un diagrama de trellis (TCVQ) o un método de predicción, se rige por el historial de símbolos. Estos tres tipos de cuantificación con memoria pertenecen a los sistemas recursivos de cuantificación vectorial.

En otras palabras, podemos decir que la cuantificación vectorial sin memoria emplea únicamente las muestras de entrada presentes en el codificador para encontrar el centroide o símbolo que represente los parámetros que se desean transmitir. Y la cuantificación vectorial recursiva, además del símbolo presente, utiliza memorias que almacenan la información referida a los vectores previos de entrada.

El esquema de cuantificación vectorial es bueno principalmente por dos puntos: Primero, cuando utilizamos codificadores cuyo retardo es pequeño, el dato que entra se codifica y se envía. Segundo, todos los centroides de un libro de códigos se usan para encontrar el mejor (de acuerdo con una medida) en un instante dado.

Como dicen Popescu et al:

"Intuitivamente el codificador hace lo mejor para la cuantificación del vector presente sin preocuparse de las consecuencias que acarrea tal decisión" [1]

En contraste, la estrategia al utilizar cuantificación recursiva es evaluar varias y buenas alternativas de codificación para el símbolo presente, y escoger aquella que conduzca a la mínima distorsión en una función de costo para cierto momento en el

futuro. El precio principal a pagar es el tiempo de decisión conocido como "decisión retardada", el segundo es la complejidad adquirida.

Como sabemos del capítulo II, el codificador CELP forma una trama de bits en la que están inmersos cuatro índices correspondientes a la excitación secundaria para una trama de voz de 30 ms. Conformada la trama entonces es transmitida. Con la anterior observación podemos notar que es posible emplear cuantificación recursiva, en particular la cuantificación vectorial tipo trellis TCVQ, para la codificación de las cuatro subtramas que constituyen una trama de voz, cargando con la complejidad requerida pero no con el retardo en el número de vectores codificados, ya que al término de la cuantificación de la cuarta subtrama se toma una decisión y, al igual que en la codificación vectorial, se procede a conformar la trama de bits.

Es desde aquí de donde parte nuestra motivación para implementar una cuantificación vectorial tipo trellis que sustituya la cuantificación vectorial que se emplea para la excitación estocástica del CELP FS1016.

El desarrollo de este capítulo será de la siguiente manera: Una primera sección está destinada a describir la cuantificación vectorial tipo trellis; la segunda expondrá la implementación hecha sobre el CELP FS1016 y finalmente, se mostrarán los resultados obtenidos y las conclusiones generadas durante la investigación.

2 Cuantificación-Codificación Vectorial tipo Trellis (TCVQ)

2.1 Cuantificador Vectorial Recursivo

La cuantificación tipo trellis es un caso especial de los codificadores de cuantificación vectorial recursivos cuya estructura se muestra en la **Fig. 1**

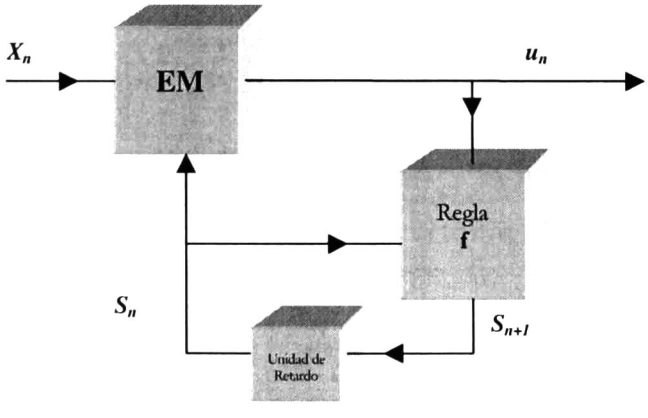

Fig. 1 Cuantificador Recursivo: Codificador

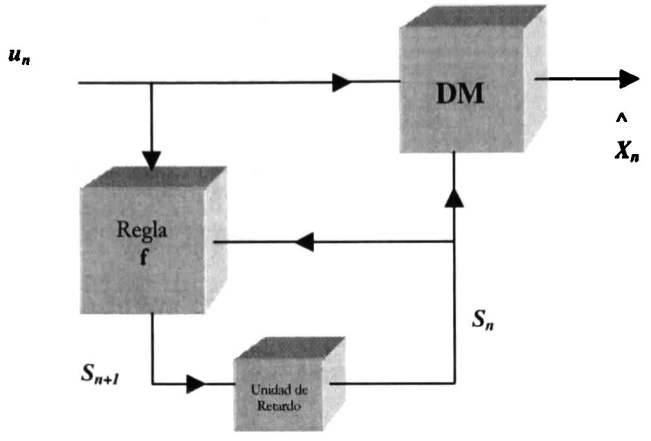

Fig. 2 Cuantificador Recursivo: Decodificador

Las estructuras mostradas funcionan como una máquina de Mealy, es decir, la salida depende de las transiciones entre estados y del estado actual.

2.2 Matemáticas Aplicadas

Dada una secuencia de vectores aleatorios de entrada X_n , n = 0, 1, ..., el codificador produce una secuencia de símbolos u_n n = 0, 1, ..., y una secuencia de estados S_n , n=0, 1, ..., que describen el comportamiento del codificador como respuesta a los vectores de entrada. Se asume que la fuente de vectores X_n proporciona vectores de dimensión k con muestras reales, esto es, $X_n \in \Re^k$. Los símbolos u_n pueden representarse como un índice que toma valores de un conjunto de números enteros desde 1 hasta $N = 2^R$ donde R es la razón de codificación dada en bits por vector de entrada.

Con el fin de asegurar que el decodificador pueda seguir la evolución de estados que marca el codificador a partir de la secuencia de símbolos recibida sin agregar información adicional, se requiere que dicha evolución comience en un estado determinado llamado $S_0 = s_0$, éste debe ser conocido tanto por el codificador como por el decodificador. Explícitamente, se asume que el próximo estado S_{n+1} es determinado a partir del estado actual y el símbolo u_n mediante la transformación hecha por la regla f llamada también función de transición de estados.

$$\text{Así} \quad S_{n+1} = f(u_n, S_n), \text{ con } n = 1, 2, \ldots, \tag{1}$$

Dado un estado S_n, entonces la transformación de X_n a u_n depende del estado actual de acuerdo con

$$u_n = \text{EM}\,(X_n\,,\,S_n) \tag{2}$$

De la **Fig. 2** observamos que el decodificador también utiliza el estado actual para generar el vector estimado a partir del símbolo u_n y la función DM.

$$\hat{X}_n = \text{DM}\,(u_n\,,\,S_n) \tag{3}$$

De manera similar a como trabajan los cuantificadores vectoriales sin memoria se asume que el codificador opera bajo la modalidad del vecino más cercano o de mínima distorsión para una función de costo.

Para comprender el funcionamiento del codificador y el decodificador supongamos que ambos se encuentran en un estado s. Entonces el decodificador tiene una colección, dependiente del estado, con la que puede generar un conjunto de vectores de reproducción o salidas.

$$C_s = \{\text{DM}\,(u\,,\,s)\};\ \text{para todo}\ u \in 1\ \text{hasta}\ N = 2^R \tag{4}$$

Esta colección es llamada partición de un libro de códigos asociada al estado s. De aquí en adelante se utilizará el término estado, para referirse a una partición de un libro de código o viceversa.

Ahora, si elegimos una medida de mínima distorsión asociada a EM con el fin de aproximar al máximo el vector estimado $\hat{X}$ al vector proporcionado por la fuente X, tendremos

$$\text{EM}\,(X\,,\,s) = \min_{u}{}^{-1}\, d(X\,,\,\text{DM}\,(u\,,\,s)) \tag{5}$$

Que se lee como el símbolo u que minimiza la medida de distorsión entre X y $\hat{X}$

Puesto de otro modo, entonces la distorsión es

$$d(X\,,\,\text{DM}\,(\,\text{EM}(X\,,\,s),\,s\,)) = \min_{\hat{X}\in C_s} d(X\,,\hat{X}) \tag{6}$$

Así que podemos afirmar que el cuantificador vectorial recursivo siempre escoge el índice que representa al vector del libro de códigos que produce la mínima distorsión posible en el decodificador para el vector reproducido, sabiendo que tanto el codificador como el decodificador se encuentran en el mismo estado s.

2.3 Cuantificación Vectorial de Trellis

Como se ha visto hasta el momento, el cuantificador recursivo descrito, logra la mínima distorsión para un vector de entrada X en una unidad de tiempo n. Este procedimiento es óptimo en ese instante, pero ¿qué sucede después de la cuantificación de varios vectores de entrada? Podemos intuir que si acumulamos la

distorsión calculada hasta un tiempo $n+k$ descubriremos que tener la mínima distorsión a cada instante no implica tener la mínima distorsión acumulada al tiempo $n+k$; y esto se debe a que existe una correlación entre los vectores de entrada.

El cuantificador recursivo descrito trata en pocas palabras de que el centroide o índice elegido minimice el error entre el vector X y el vector reproducido $\hat{X}$ en el decodificador. Sin embargo sólo se está pensando en una minimización a corto plazo y por consiguiente no le importa que suceda después.

La cuantificación vectorial en trellis en cambio, está pensada para minimizar el error a largo plazo. Lo que suceda en el futuro influencia la decisión tomada en el presente, es decir haremos una decisión retardada. Para ilustrar lo anterior

Supóngase un conjunto de vectores de entrada $\{X_0 \ldots X_{T-1}\}$. Conociendo tal secuencia de vectores y el estado inicial s_0 se pueden generar los símbolos $\{u_0 \ldots u_{T-1}\}$ y los estados $\{s_1 \ldots s_T\}$ que minimicen la distorsión acumulada

$$D_{Acc} = \sum_{n=0}^{T-1} d\left[X_n, \hat{X}_n\right]$$

El problema ahora es encontrar la combinación de símbolos que minimice la distorsión del conjunto de vectores de entrada. La solución es utilizar algún algoritmo de máxima verosimilitud (ML) como lo es el algoritmo de Viterbi (descrito en la **sección 5.1** del capítulo V).

2.3.1 Representación de la Máquina de Estados en el Tiempo

La estructura de trellis es una manera de representar el comportamiento de una máquina de estados finitos a través del tiempo, por lo que se considera como una herramienta de visualización del comportamiento de un autómata.

Un diagrama de transición de estados usando la máquina de la **Fig.1** nos daría una estructura de retícula o trellis si interpretamos cada estado como una partición de un diccionario o libro de códigos y los representamos con puntos alineados en forma vertical. Repetimos esta estructura para cada unidad de tiempo y unimos con líneas o ramas dos estados localizados en unidades de tiempo contiguas. De esta manera llegamos a la representación gráfica de la **Fig. 3**

Las ramas constituirían la representación gráfica de las palabras de un libro de códigos. La rama une dos puntos contiguos, el punto de origen representa el estado previo de la máquina, es decir, una partición específica del libro, mientras que el punto de arribo indica el nuevo estado de la máquina, la partición en el nuevo instante de tiempo, que no necesariamente es la misma que la del instante anterior.

Se tendría entonces, una serie de caminos distintos que recorren los puntos del trellis. Estos caminos son una representación de todas las posibles secuencias de palabras del libro de códigos tomado como una totalidad. En un cuantificador que

no fuese de estados finitos el pasado no tiene ninguna influencia en la codificación del dato actual por lo que elegir cualquier palabra del libro es igualmente válido en cualquier instante; en contraste, el cuantificador de trellis restringe la búsqueda de la palabra óptima a una partición del libro llamada estado. La elección de dicha partición depende del estado actual de donde salió el último símbolo u y del nuevo dato que se desea codificar.

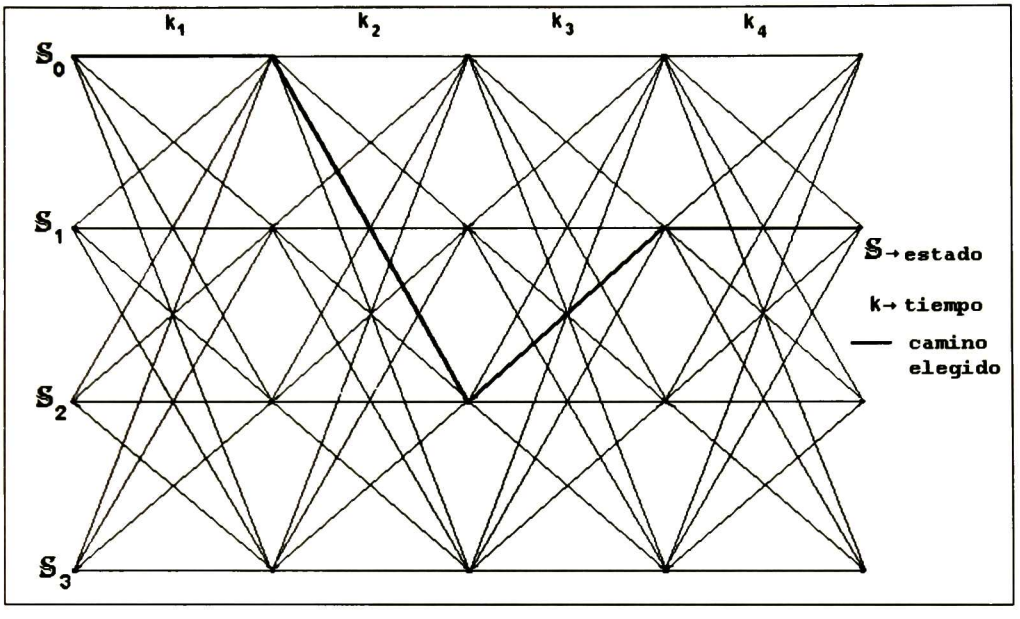

Fig. 3 Diagrama de trellis

2.3.2 Diagrama de Bloques

La **Fig. 4** muestra un cuantificador-codificador vectorial de trellis basándose en la representación de un cuantificador vectorial recursivo que inicia en el estado S_0 y a partir de aquí se construye el diagrama de trellis. Se distingue que el mapeo del codificador (EM) se sustituye por el algoritmo de Viterbi que tiene la tarea de encontrar la secuencia que minimice el error o el camino con menor costo a través del trellis. La regla f al recibir un símbolo u_n y un estado S_n, que indica el libro de códigos de donde se seleccionó el símbolo u_n, genera el próximo estado S_{n+1} que indica el libro de códigos de donde se obtendrán los siguientes símbolos a evaluar por el algoritmo de Viterbi que posee una métrica particular. Después del tiempo marcado (t) en el caso de que se trunque el diagrama de trellis, el algoritmo de Viterbi entrega una secuencia de símbolos u a los cuales se les asocia un índice que corresponde a la codificación. En el caso de que no se trunque, el algoritmo de Viterbi entregará una secuencia de tamaño variable según el tiempo de convergencia.

En la etapa de decodificación se procede de una manera similar al decodificador del cuantificador vectorial recursivo, pues se conoce el estado de inicio S_0 y se cuenta con la misma regla f, de manera que el decodificador sólo realiza un seguimiento de la secuencia recibida de índices, que se traducen a símbolos cuando se cogen de un libro de códigos determinado por la regla.

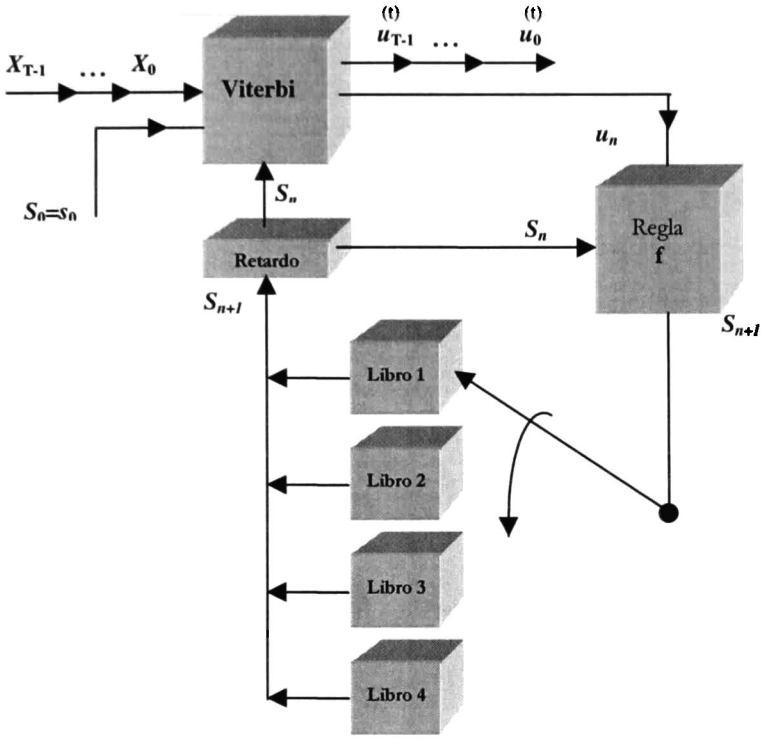

Fig. 4 Cuantificador Vectorial de Trellis: Codificador

2.4 Partición del Libro de Códigos

Se ha dicho que cada estado es una partición de un libro de códigos de manera que podemos considerar a esta partición como un libro de códigos. Así el esquema de la **Fig. 4** tiene 4 libros provenientes de un libro A al que se le aplicó un algoritmo de partición. Pero ¿cómo podemos realizar la partición un libro? En el caso particular en el que el libro de códigos está constituido por vectores de muestras o derivados de la voz como el residual, no es posible utilizar literalmente las reglas de Ungerboeck [3] sobre la partición, debido a que los vectores no se encuentran regularmente espaciados como en las constelaciones de modulación PSK. Por eso recurrimos a algoritmos iterativos publicados en la literatura.

Popescu et al [1] sugieren dos algoritmos distintos para repartir las palabras del libro. Una combinación de dichos algoritmos proporciona la partición final que él utilizó en su implementación de cuantificación de trellis sobre un CELP.

El objetivo principal en la partición del libro es tener vectores que se encuentren lo más lejos posible si pertenecen al mismo grupo o estado. La razón para esto es que se quiere facilitar el discernimiento del vector más apropiado para la codificación una vez que se determinó el próximo estado. La forma de hacerlo, según Popescu et al, es la siguiente:

1. Dado el libro de códigos *A* se eligen 2 vectores en forma aleatoria y se determinan dos subconjuntos (libro *B* y libro *C*), uno con cada vector.

2. Se mide la distancia de los vectores restantes en el libro *A* con el vector del libro *B*, se elige el más alejado (distancia euclidiana) y se agrega al subconjunto. De los vectores restantes se elige el más alejado del vector del libro *C* y se agrega al mismo. Ahora tenemos 2 vectores en cada subconjunto y 4 vectores menos en el libro *A*.

3. Se mide la distancia de los vectores en *A* con cada uno de los vectores de *B*. De los vectores que tengan la mayor distancia respecto a cada elemento del libro *B*, se agrega el de mayor distancia.

4. Se hace lo anterior para el libro *C*.

5. Volvemos al paso 3 hasta consumir los elementos del libro *A*.

El segundo algoritmo propuesto por Popescu et al consiste en intercambiar dos vectores entre subconjuntos siempre que esto aumente la distancia mínima en ambos subconjuntos. Popescu et al utilizan este algoritmo para mejorar el primero después de que se han hecho varios intentos con éste. Dado que al inicio se eligen dos vectores aleatorios, entonces al repetir el primer algoritmo, obtendremos resultados distintos. De modo que se realizan varias corridas y se escoge la mejor partición. En este trabajo solamente se utilizó el primer algoritmo para dividir un libro de códigos en cuatro subconjuntos o estados.

2.5 La Regla de Asignación *f*

La regla *f* es la unidad fundamental del cuantificador vectorial de trellis ya que es la encargada del funcionamiento de la máquina de estados. Una buena analogía es pensar en la matriz generadora de un codificador convolucional. Otra forma de interpretarla sería pensar en un predictor que como su nombre lo indica, tiene la función de predecir el próximo estado que será utilizado para cuantificar el siguiente vector de entrada. O dicho de otro modo, proporciona el libro de códigos de donde se sacará el centroide que minimice la distorsión.

Entonces surge la pregunta ¿cómo encontrar tal regla de asignación? Bei [4] propone varios algoritmos de solución, de los cuales, el método del histograma condicional es el que dio mejores resultados en sus experimentos. Con una secuencia de entrenamiento se realiza un conteo estadístico de transiciones, de las cuales, aquellas con el mayor número de ocurrencia son las que forman la regla. El resultado es una

lista en la que a cada índice del libro de códigos se le asigna un estado siguiente según el histograma obtenido. La estructura de trellis generada tendrá de esta manera que a cada estado arriban transiciones provenientes de todos los estados.

Popescu et al en [1], utilizan una máquina de estados finitos arbitraria para el cambio de estados como regla, y agrupa los vectores que llevan a determinado estado en el mismo libro de códigos procurando tener la máxima distancia entre ellos. El trellis resultante está restringido por el codificador convolucional (regla *f*), a una estructura en la que para cada estado sólo llegan las transiciones provenientes de dos estados.

2.6 Algoritmo de Viterbi en la Cuantificación

Ya que se tiene el libro de códigos particionado y a cada partición se le asigna un estado que la represente, además de contar con la regla de asignación *f*, es posible implementar el algoritmo de Viterbi de manera que desempeñe la función del bloque EM del codificador recursivo de la **Fig. 1**. Este algoritmo deberá comparar los vectores del libro de códigos según el estado-partición en que se encuentren con la señal objetivo, empleando una métrica específica y almacenando para cada unidad de tiempo la información asociada a la rama o transición ganadora por estado.

Al final de la evolución del diagrama de trellis (terminado o truncado) se elige la trayectoria que minimice el error global, es decir, aquel camino cuya distorsión acumulada es mínima.

El algoritmo de Viterbi en su forma óptima (diagrama de trellis sin truncar) considera todas las trayectorias posibles a través del trellis y en cada unidad de tiempo en la que se produce una transición, elimina todas las trayectorias excepto la perteneciente a la rama ganadora que llega a cada estado. Al eliminar segmentos conforme pasa el tiempo, ocurre que en ciertos instantes no hay ramas llegando a todos los estados anteriores. Así, la estructura de enrejado va perdiendo complejidad al mirarla en forma retrospectiva, hasta que descubrimos una unidad de tiempo en el diagrama donde solamente un nodo es alcanzado por una transición (nodo inmortal). En este punto se tiene una trayectoria óptima cuyos parámetros pueden ser transmitidos. La distancia en unidades de tiempo entre un punto de convergencia y el siguiente es variable por lo que la secuencia óptima encontrada varía en la cantidad de vectores codificados. Es necesario que el monitoreo de la estructura de retícula sea constante (cada unidad de tiempo) para poder transmitir información en cuanto se haya cuantificado en forma óptima.

A fin de tener un tiempo fijo para la transmisión de datos, Popescu et al establecen el tiempo máximo de convergencia. En la investigación que aquí se reporta, el trellis se trunca a cuatro unidades de tiempo y se decide una trayectoria. Esto da origen a una cuantificación subóptima, sin embargo el retardo y la complejidad disminuyen. En la descripción de la simulación se expondrán las razones de tal truncamiento.

3 Sustitución del Cuantificador Vectorial del CELP FS1016

3.1 Propósitos

En el capítulo sobre el CELP FS1016 se explica el cuantificador vectorial utilizado, que, en pocas palabras, elige un centroide de un libro de códigos de vectores residuales dentro del contexto de análisis por síntesis.

El primer propósito de hacer una sustitución que utiliza un cuantificador de trellis es obtener una codificación que reduzca la velocidad de transmisión de 4800 bps a 4533.33 bps (8 bits menos por cada trama de 144 bits) con una buena calidad de voz medida de manera objetiva y subjetivamente.

El segundo es utilizar el cuantificador de trellis para obtener una codificación igual a la del CELP FS1016 pero con mayor calidad de voz.

3.2 Descripción de la Sustitución

3.2.1 Cálculos Previos a la Cuantificación

Los cálculos previos se realizan en la sección llamada etapa de análisis localizada en el codificador. La metodología es la siguiente:

- Obtención de las muestras correspondientes a una trama de voz.
- Análisis espectral en lazo abierto para obtener los parámetros LPC por el método de autocorrelación.
- Conversión de los parámetros LPC's a los parámetros LSP's.
- Empaquetamiento de los índices asociados a los parámetros LSP's.

3.2.2 Cuantificación Vectorial

La cuantificación vectorial de Trellis es utilizada para la excitación estocástica. Los programas modificados para el FS1016 se localizan en [7]

Un punto muy importante a señalar es el truncamiento del diagrama de trellis realizado después de 4 subtramas, por la razón de que se quiere cumplir con la interfaz que entrega la información de las 4 subtramas codificadas, a la rutina de empaquetamiento. De esta manera se hace en el CELP FS1016. Así, aunque el algoritmo de Viterbi se ejecuta de forma subóptima, no se tiene un retraso en el número de tramas codificadas; éste es el compromiso adquirido.

Los puntos principales de la metodología son:

1. Inicialización.

2. Elección de un estado inicial que indica el libro de códigos de donde se obtendrán los vectores residuales para la primer subtrama de una trama.

3. Ciclo *k* que maneja los 4 tiempos de propagación del diagrama de trellis o el control para las 4 subtramas.

4. Utilizar los coeficientes LPC para el tiempo *k*.

5. Calcular la respuesta al impulso para el tiempo k.

6. Ciclo *s* que maneja los estados de arribo para la formación del diagrama de trellis.

7. Ciclo *sigma* que maneja los estados de partida que indican los libros de códigos de donde se obtienen los vectores residuales que me llevan al estado de arribo.

8. Encontrar el "pitch" utilizando el método de cuantificación vectorial que emplea el libro de códigos adaptativo señalado por la norma FS1016.

9. Ciclo *u* que maneja el índice de los vectores residuales que están en el libro de códigos indicado por *sigma*.

10. Seleccionar el vector residual proveniente del estado *sigma*, que minimiza el error cuadrático medio arribando al estado *s*.

11. Fin del ciclo *u*.

12. Almacenar el estado ganador $sigma_s$ y el vector seleccionado u_s, si la métrica acumulada al estado *sigma* mas la distorsión del vector residual seleccionado en el paso 10, es mayor que la métrica acumulada al estado *s*.

13. Fin del ciclo *sigma*.

14. Almacenar la información que describe el diagrama de trellis al tiempo *k* para el estado *s*, esto es: los estados ganadores, los vectores ganadores, la información sobre el "pitch" y las memorias de los filtros.

15. Fin del ciclo *s*.

16. Actualización de las memorias de los filtros y las distorsiones acumuladas.

17. Fin del ciclo *k*.

18. Recorrer el diagrama de trellis para el retro-rastreo "trace back".

3.2.3 Procedimiento Posterior a la Cuantificación.

Las rutinas posteriores a la cuantificación comprenden una sección de la etapa de análisis y la etapa de decodificación. La organización y logística de las rutinas se describen a continuación.

- Empaquetar los parámetros asociados a la excitación estocástica y al "pitch" de cada subtrama, en el flujo de bits.
- Proteger los bits referentes al "pitch" utilizando codificación de Hamming.
- Generar el archivo que contiene el flujo de bits.
- Etapa de decodificación que desempaca la información y utilizando la regla *f* se mueve a través del libro indicado por ésta, para obtener el vector residual que se alimenta en la etapa de síntesis.

3.3 Especificaciones Técnicas

El trabajo está hecho utilizando como base el programa en lenguaje C, propuesto por el Departamento de Defensa de los Estados Unidos a cargo de Joe Campbell [5].

La sustitución implica varios puntos:

- Las interacciones hechas entre la rutina de cuantificación y las otras funciones (cálculo de los parámetros LPC, etc.), deben ser transparentes.
- Cumplir con las especificaciones de la norma del CELP FS1016 como son:
  - Respetar el formato de la trama de bits.
  - Utilizar las tablas para cuantificación de parámetros.
  - Etc.
- Cumplir con el esquema del programa.
- Cumplir con interfaces entre rutinas y funciones.
- Las modificaciones realizadas deben tener la finalidad de afectar al mínimo las rutinas y funciones que no correspondan a la cuantificación o que sean utilizadas por ésta.
- Las variables adicionales deben ser propias de la modificación hecha y las variables ya existentes deben de respetarse lo más posible.

- Generar los archivos de salida en el formato especificado.

- Etc. Véase [6] si se desea entrar en detalles.

3.4 Almacenamiento y Complejidad

3.4.1 Complejidad

Empleando la tabla sobre los cálculos de complejidad para diferentes tamaños de libros de código que se encuentra en el archivo cgain.c de [5], partimos para hacer un estimado de la complejidad del CELP con la cuantificación de trellis.

Se consideran cuatro secciones principales que son:

C: Convolución; **R**: Correlación; **E**: Energía; **G**: Ganancia de Cuantificación

El número de MIPS para N=1 y N=512 (vectores del libro de códigos) necesarios para cada una de estas tareas en el CELP FS1016 se muestra en la **tabla 1.**

Tabla 1 MIPS para cuatro diferentes tareas en el CELP FS1016

N	C	R	E	G	Total
1	0.089333	0.008000	0.008000	0.002533	0.107867
512	1.029573	4.096000	1.833810	1.297067	8.256450

Nota.- Recuerde que el CELP FS1016 utiliza un libro de códigos ternario que ayuda a bajar el No. de MIPS requeridos.

Basándonos en los valores para N=1 (excepto para la convolución) y los cálculos hechos para la cuantificación de trellis y considerando que el diagrama de trellis, en la primera subtrama, evalúa un solo libro de códigos, tenemos que:

- Para un conjunto de 4 Libros de 128 Vectores c/u

R=0.008/4*(128+512*3) = 3.3280 MIPS

E=0.008/4*(128+512*3) = 3.3280 MIPS

G=0.002533/4*(128+512*3) = 1.0537 MIPS

C=[(1+2+...+30)+(30)*(30)]*(512*3+128) = (465+900)*(1664) /10^6 = 2.27 MIPS

Total = 3.3280 + 3.3280 +1.0537 + 2.27 = **9.9797** MIPS

- Para un conjunto de 4 libros de 512 Vectores c/u

R=0.008/4*(512+2048*3) = 13.3120 MIPS

E=0.008/4*(512+2048*3) = 13.3120 MIPS

G=0.002533/4*(512+2048*3) = 4.2149 MIPS

C=[(1+2+...+30)+(30)*(30)]*(2048*3+512) /10^6 = (465+900)*(6656) /10^6 = 9.0854 MIPS

Total = 13.3120 + 13.3120 + 4.2149 + 9.0854 = **39.9243** MIPS

3.4.2 Almacenamiento

La principal información que necesita ser guardada para tejer el diagrama de trellis es:

A. Matriz de k=4 tiempos por s=4 estados que almacena los estados ganadores y la información relativa a estos como es:

   - Índice del vector ganador a cada estado.
   - Ganancia del vector ganador a cada estado.
   - Parámetros del "pitch".
   - Correlación de la señal objetivo (e0) antes y despúes de obtener el "pitch"

B. Matriz de k=4 tiempos por s=4 estados que almacena las memorias de los filtros STP y STP pesado.

C. Matriz de s=4 estados y se renueva cada tiempo k, que almacena el "buffer", para cada estado, de donde se obtendrán los parámetros del "pitch".

4 Resultados del Experimento

4.1 Archivos de Voz

El primer archivo de voz utilizado para presentar los resultados consiste de 4 voces de mujer y 4 de hombre en idioma inglés de aproximadamente 30 seg. (T_S_voz.spd)

El segundo archivo de voz consiste de 2 voces de mujer y 2 de hombre en idioma español de aproximadamente 26 seg. (probe.spd)

4.2 Resultados Utilizando LC´s de 512 vectores

Emplear un libro de código con 512 vectores para particionarlo en 4 distintos libros de 128 vectores, conduce a codificar el vector residual seleccionado, para cada subtrama, con 7 bits en vez de los 9 bits empleados por el CELP FS1016. Esto implica

una reducción de la velocidad de transmisión, pasando de 4800 bps a 4533.33 bps aproximadamente.

La **Tabla 2** muestra los resultados obtenidos utilizando diversos libros de código de 512 vectores para la codificación del archivo de voz T_S_voz.spd

Tabla 2. Cuantificación usando diferentes libros de códigos: T_S_voz.spd

Libros de código	SNR Segmental (BER = 0.0)	SNR Segmental (BER = 0.001)	Comentarios
original	10.39 dB	9.25 dB	Libro de código (LC) del FS1016
codq	9.76 dB	8.81 dB	L.C original sin particionar
cbOTQ	9.75 dB	8.67 dB	LC del FS1016 para TQ
cbLBGTQ	9.89 dB	8.68 dB	LC producto LBG para TQ
cbCOVQTQ	10.13 dB	8.88 dB	LC producto COVQ para TQ

La **Tabla 3** muestra los resultados obtenidos utilizando diversos libros de código de 512 vectores para la codificación del archivo de voz probe.spd

Tabla 3. Cuantificación usando diferentes libros de códigos: probe.spd

Libros de código	SNR Segmental (BER = 0.0)	SNR Segmental (BER = 0.001)	Comentarios
original	7.39 dB		Libro de códigos (LC) del FS1016
codq	6.85 dB	6.08 dB	LC original sin particionar
cbOTQ	6.72 dB	6.07 dB	LC del FS1016 para TQ
cbLBGTQ	6.81 dB	6.01 dB	LC producto LBG para TQ
cbCOVQTQ	6.75 dB	5.69 dB	LC producto COVQ para TQ

El segundo experimento realizado consistió en utilizar un libro de código de 2048 vectores y dividirlo en 4 libros de 512.

La **Tabla 4** muestra los resultados obtenidos aplicando el libro de código de 2048 vectores al archivo T_S_voz.spd

Tabla 4. Cuantificación usando diferentes libros de códigos: T_S_voz.spd

Libros de código	SNR Segmental (BER = 0.0)	SNR Segmental (BER = 0.001)	Comentarios
cbO2048	10.24 dB	9.13 dB	LC a partir del original de 1082 muestras
cbMIXTQ	10.41 dB	9.02 dB	LC compuesto de 4 L.C. de 512 vx's

5 Conclusiones

- De los resultados podemos apreciar que la SSNR no baja de manera considerable (aproximadamente 0.5 dB) cuando utilizamos un libro de códigos de 512 vectores; además, la complejidad asociada con la búsqueda del vector estocástico óptimo se mantiene en el rango de los 10 MIPS (recuérdese que en el CELP FS1016, este dato es de 8.3 MIPS, aproximadamente). Más aún, la velocidad de transmisión baja de 4800 a 4533.33 bps, de manera que los bits libres pueden ser empleados para protección de canal.

- La utilización de un libro de códigos de 2048 vectores parece ser que no mejora de manera considerable la calidad de voz al compararse con la presentada por el CELP FS1016. Por otro lado, la complejidad del algoritmo aumenta fuertemente.

- El uso de la regla f para un libro de código de 512 vectores obtenida mediante el histograma condicional con una secuencia de entrenamiento de 3.06 min. y 4.12 min. tuvo resultados muy similares en SSNR para los archivos de voz codificados. Por lo anterior, podemos afirmar que una secuencia de entrenamiento de 3 min. (con voces mezcladas) es suficiente para obtener una regla f que conduzca a buenos resultados en la codificación.

6 Referencias Bibliográficas

[1] A. Popescu, N. Moreau, C. Lamblin, *"CELP Coding Using Trellis-Coded Vector Quantization of the Excitation"*, IEEE Transactions on Speech and Audio Processing, Vol. 3, No. 6, Noviembre 1995.

[2] Gersho, Allen, "Vector Quantization and Signal Compression", Kluwer Academic Series.

[3] Ungerboeck, Gottfried, *"Trellis-Coded Modulation with Redundant Signal Sets"*, IEEE Communications Magazine, Vol. 25, No. 2, Febrero 1987, pp. 5-11.

[4] Bei, Chang-Da, *"Simulation of Vector Trellis Encoding Systems"*, IEEE Transactions on Communications, Vol. COM-34, No. 3, Marzo de 1986, pp. 214-218.

[5] J. Campbell, *"Archivos de Programación en lenguaje C para el CELP FS1016"*, Biblioteca del CINVESTAV Gdl., 1998.

[6] I. Peñuelas, Tesis: *"Desarrollo de un modelo para el estudio de interacción de servicios en el software de un conmutador digital "*, Biblioteca del CINVESTAV Gdl., 1998.

[7] O. Longoria, R. Rodríguez, *"Archivos de Programación en lenguaje C para el CELP FS1016 con TCVQ"*, Biblioteca del CINVESTAV Gdl., 1998.

CAPÍTULO V

EL ALGORITMO DE VITERBI CON SALIDAS SUAVES (SOVA)

El algoritmo SOVA aplicado al FS-1016 @ 4800 bps

1 Introducción

"Si os doy probabilidades, ya no me pidáis más".
Platón.

Dentro de los algoritmos de codificación conjunta de canal y fuente está la categoría en la que el decodificador de canal utiliza información de la fuente para sus decisiones. El algoritmo de Viterbi con salidas suaves (SOVA) entra en esta clasificación. En el presente capítulo se trata de exponer la investigación realizada utilizando el algoritmo SOVA en la decodificación de los parámetros referidos a la segunda excitación del codificador CELP FS1016.

Para esto es necesario contar con un modelo de la fuente de información que será el flujo de bits arrojado por el CELP. Estos bits se introducen a un codificador de canal del tipo convolucional; luego de una codificación de línea (NRZ), los símbolos son transmitidos por el canal, que al actuar sobre ellos, degradará la información que transportan. En la sección del receptor se coloca un decodificador de canal que usa el algoritmo de Viterbi y otro que emplea el algoritmo SOVA, con el fin de comparar los desempeños de decodificación realizados por ambos algoritmos.

Los antecedentes que dieron origen al algoritmo SOVA que se mencionen, servirán de contexto para que el lector descubra cuáles fueron las principales motivaciones que tuvieron los primeros investigadores.

Las matemáticas y términos afines al algoritmo SOVA que se describan, servirán como herramienta para conocer el funcionamiento del algoritmo. Un punto importante que resaltar son las diferencias existentes entre el algoritmo de Viterbi y el SOVA, diferencias que hacen que éste último tenga un mejor desempeño tomando como medidas de evaluación la relación señal a ruido (SNR) y la tasa de error en bits (BER).

2 Antecedentes.

Un resultado básico de la teoría de información de Shannon es que la codificación de fuente y de canal pueden ser tratadas separadamente. Claro que esto es verdad cuando se cumple el siguiente teorema:

> *Mientras la entropía de la fuente sea menor a la capacidad de canal, existe un esquema separable de codificación de fuente y de canal que permite transmisiones sobre un canal con arbitrariamente baja probabilidad de error [17].*

Como fruto de este resultado básico, existía un común acuerdo entre los ingenieros que sugerían que la gente que realizaba codificación de fuente, debía manejar un flujo de bits que fueran estadísticamente independientes e igualmente importantes. De manera que la tarea de los diseñadores de codificación de canal y los ingenieros encargados de la modulación era, proteger estos bits de los errores de transmisión y entregarlos libres de errores -o casi libres- al decodificador, el cual desempeña la tarea de reconstruir la señal origen. Como se puede apreciar, existía una clara interfaz entre codificación y decodificación de fuente y de canal.

Es frecuente notar –hasta finales de los 80's- que la gente involucrada con la codificación de fuente y de canal no hablaba mucho sobre cuestiones como pueden ser la sensibilidad de los bits provenientes de la fuente, la protección contra errores aplicada de manera desigual y la influencia de la protección contra errores. Es claro que cuando uno no espera errores, lo mencionado pasa a ser irrelevante.

Sin embargo, cuando consideramos un canal de transmisión que toma en cuenta los factores como pueden ser el ruido extenso, la interferencia, las multitrayectorias, el desvanecimiento y el "shadowing" es bastante costoso en términos de ancho de banda, retardo y complejidad, querer realizar una codificación de canal perfecta. Por ejemplo, en radio móvil, el canal es extremadamente no estacionario y algunas veces con codificación de canal razonable (varias y distintas etapas de codificación), la tasa de error presentada al decodificador de voz, está todavía en el rango de 10^{-2}.

Por otro lado, por razones de retardo, complejidad y debido a fuentes altamente no estacionarias, muchos esquemas de codificación de fuente no alcanzan a eliminar toda redundancia de información, ocasionando con esto, que a la salida del codificador se tengan bits con distinto significado y diferente sensibilidad a errores.

Popescu et al [9] encontró que dentro de los parámetros enviados por un CELP, los referentes al "pitch" son los más sensibles a errores. También la norma del GSM realiza una clasificación de los bits, aplicable a su codificador [3]. Es posible notar, como se ilustrará más adelante para el FS-1016, que algunos bits están correlacionados, cuando se realiza el análisis empleando tramas de bits; es decir, existe una correlación al menos en la referencia de tiempo.

Bajo estas circunstancias, dice Hagenauer en [3]- la codificación de canal y de fuente, así como la decodificación, no pueden ser tratadas separadamente - y concluye - En efecto, Shannon mencionó ya esta posibilidad en 1948:

> *"No obstante, cualquier redundancia en la fuente será usualmente de ayuda, si ésta es utilizada en el punto de recepción. En particular, si la fuente ya tiene redundancia y ningún intento es hecho para eliminarla y acoplarse al canal, esta redundancia ayudará a combatir el ruido. [7]*

La **Fig. 1** muestra un diagrama a bloques de un sistema de comunicación empleando la redundancia o información a priori que la fuente no pudo eliminar. Como se explicará más adelante, dicha información es difícil de estimar en el receptor debido a que no se cuenta con un modelo exacto de la fuente.

3 Álgebra Logarítmica para una Variable Aleatoria Binaria

Sea u una variable aleatoria binaria perteneciente al campo de Galois módulo 2, **GF(2),** con los elementos {+1, -1} y el +1 como el elemento nulo bajo la operación de adición ⊕. La razón de verosimilitud logarítmica de la variable aleatoria binaria u, $L(u)$, está definida como:

$$L(u) = \log \frac{P(u=+1)}{P(u=-1)} \qquad (1)$$

donde log es el logaritmo natural.

A partir de aquí nos referiremos a $L(u)$ como el "valor suave" de una variable aleatoria binaria. Para cada bit tenemos una cantidad cuyo signo será positivo o negativo, según el bit (decisión dura) de que se trate, y la magnitud $|L(u)|$ indica la certeza o confiabilidad que tenemos acerca de la identidad del dígito binario.

Si la variable aleatoria binaria u está condicionada a una variable aleatoria distinta, y, entonces tenemos una razón de verosimilitud logarítmica condicionada, $L(u \mid y)$, con

$$L(u \mid y) = \log \frac{P(u=+1 \mid y)}{P(u=-1 \mid y)} = \log \frac{P(u=+1)}{P(u=-1)} + \log \frac{P(y \mid u=+1)}{P(y \mid u=-1)} \qquad (2)$$

$$L(u \mid y) = L(u) + L(y \mid u) \qquad (3)$$

Utilizando la relación $P(u,y) = P(y)\ P(u \mid y)$ podemos afirmar que la probabilidad de verosimilitud logarítmica conjunta $L(u,y)$ es igual a la probabilidad de verosimilitud logarítmica condicional $L(u \mid y)$.

Empleando las relaciones

$$P(u_1 \oplus u_2 = +1) = P(u_1 = +1)\,P(u_2 = +1) + (1 - P(u_1 = +1))\,(1 - P(u_2 = +1)) \qquad (4)$$

$$\text{con} \qquad P(u = +1) = \frac{e^{L(u)}}{1 + e^{L(u)}} \qquad (5)$$

De la referencia [4] concluimos que para dos variables aleatorias estadísticamente independientes, u_1 y u_2,

$$L(u_1 \oplus u_2) = \log \frac{1 + e^{L(u_1)} e^{L(u_2)}}{e^{L(u_1)} e^{L(u_2)}}$$

$$\approx sign(L(u_1)) \cdot sign(L(u_2)) \cdot min(|L(u_1)|, |L(u_2)|) \qquad (6)$$

De la ecuación (6) se aprecia que el resultado de la adición de dos valores suaves será igual a la menor magnitud de los valores suaves manipulados, y el signo corresponderá a la operación binaria de **or exclusivo** entre las variables aleatorias binarias u_1 y u_2.

A continuación se define una álgebra especial para los valores suaves, donde $\boxplus$ denota la operación de adición.

$$L(u_1) \boxplus L(u_2) \triangleq L(u_1 \oplus u_2) \qquad (7)$$

$$\text{Cumpliendo con} \qquad L(u) \boxplus \infty = L(u) \quad \text{y} \quad L(u) \boxplus 0 = 0 \qquad (8)$$

Un valor suave igual a 0 indica que la variable aleatoria binaria es equiprobable y un valor suave igual a infinito significa que la variable binaria es determinística.

De (8) se concluye que si uno de los operandos es infinito, el resultado será igual al valor suave distinto de infinito. Esto es, la confiabilidad que tengo es aquella que me proporciona el valor suave con menor magnitud. Por otro lado, si uno de los operandos es cero, el resultado indicará que la confiabilidad de la decisión será nula o, en otras palabras, la probabilidad, $P(u_1 \oplus u_2 = +1) = 0.5$ y $P(u_1 \oplus u_2 = -1) = 0.5$; por lo tanto, la variable aleatoria binaria resultante de la adición es equiprobable.

4 Valores Suaves del Canal

Para entender el término salida suave proveniente del canal, es conveniente observar la **Fig. 1.**

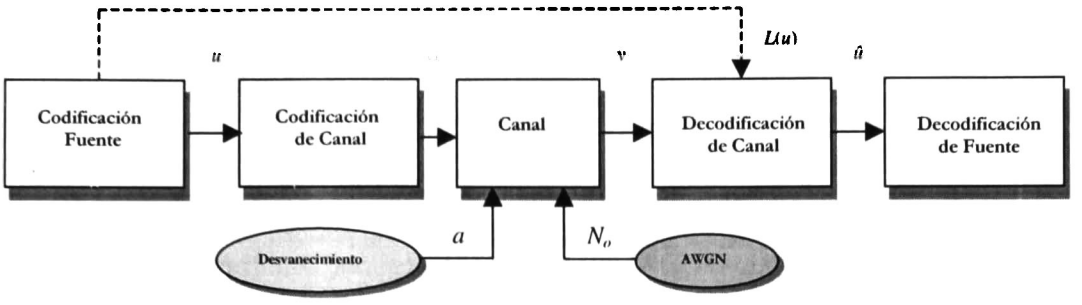

Fig. 1 Diagrama a bloques de una codificación de canal y fuente

Como podemos ver en la sección del transmisor, contamos con un codificador de fuente que tratará de eliminar la redundancia proveniente de la fuente, generando la variable aleatoria binaria, u, con su correspondiente valor suave $L(u)$. Posteriormente utilizamos un codificador de canal que agregará redundancia al codificar los valores binarios, para producir la variable binaria, x, con valores suaves $L(x)$.

La extensión con líneas punteadas desde el codificador de fuente hasta el decodificador de canal es la información a priori de la fuente $L(u)$, que puede ser transmitida, lo que no es usual, o estimada en el receptor con el propósito de que sea utilizada por el decodificador de canal que, en lo particular tiene una implementación del SOVA.

Considerando un canal binario simétrico (BSC) o un canal gaussiano con desvanecimiento transmitimos los valores de x. Luego de la transmisión sobre el canal, podemos calcular la razón de verosimilitud de x condicionada a la salida del filtro acoplado y; esto es:

$$L(x \mid y) = \log \frac{P(x=+1 \mid y)}{P(x=-1 \mid y)} = \log \frac{p(y \mid x=+1)}{p(y \mid x=-1)} \cdot \frac{P(x=+1)}{P(x=-1)} \tag{9}$$

$$L(x \mid y) = \log \frac{\exp(-\frac{E_s}{N_0}(y-a)^2)}{\exp(-\frac{E_s}{N_0}(y+a)^2)} + \log \frac{P(x=+1)}{P(x=-1)}$$

$$L(x \mid y) = L_c y + L(x) \tag{10}$$

$L_c = 4(E_s / No)a$, llamado valor de confiabilidad del canal, donde a, para un canal con desvanecimiento, representa la amplitud de desvanecimiento y para un canal gaussiano $a = 1$. En un BSC indica la razón de verosimilitud logarítmica de la probabilidad de error P_0, es decir $L_c = \log((1-P_0) / P_0)$.

$L_c y$ es la cantidad arrojada por canal y puede interpretarse como la diferencia entre los valores suaves a posteriori y los valores suaves a priori de la variable x.

$$L_c y = L(x|\, y) - L(x)$$

A este valor se le conoce como valor suave del canal que es utilizado junto con los valores suaves de la fuente, por el algoritmo SOVA y así producir una estimación de u con su correspondiente ponderación de confiabilidad asociada a cada bit hecho en la decisión dura.

La **Fig. 2** muestra un diagrama a bloque del decodificador de canal con el algoritmo SOVA y sus elementos de entrada.

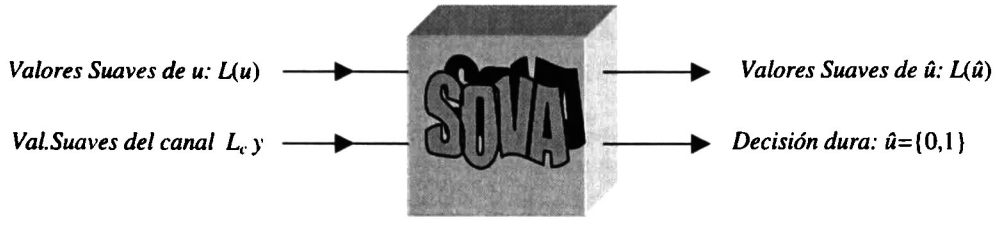

Fig. 2 Entradas y Salidas del Algoritmo SOVA.

5 Funcionamiento del Algoritmo SOVA

Ya que se han visto los valores suaves de la fuente y los provenientes del canal, es momento de describir cómo el algoritmo SOVA emplea tales cantidades para realizar la decisión dura y generar los valores suaves asociados a tal decisión. Para esto, nos basaremos en el algoritmo de Viterbi y el algoritmo HUK propuesto por Hagenauer en [3] que estima los valores suaves de la fuente desde la perspectiva del receptor.

5.1 El Algoritmo de Viterbi

El algoritmo de Viterbi fue propuesto en 1967 [10] como un método de decodificación para códigos convolucionales. A lo largo de los años se ha convertido en una herramienta estándar en otras áreas como pueden ser la demodulación y la equalización. Así también, ha aumentado el número de aplicaciones que utilizan dos etapas con el algoritmo de Viterbi de manera concatenada. Un ejemplo que describe lo anterior es pensar en un sistema de comunicación que emplee una primera etapa para demodulación y una segunda que desempeñe el trabajo de decodificación con corrección de errores (FEC).

El algoritmo de Viterbi asumiendo un proceso estocástico markoviano, esencialmente desempeña una decodificación de máxima verosimilitud es decir, las probabilidades a priori son equiprobables, además de que reduce la carga computacional por la ventaja de utilizar la estructura de un diagrama de trellis.

Aplicando esto a nuestra notación y tomando en consideración la variable aleatoria binaria x, entonces : $P(x = +1) = P(x = -1)$

Una explicación de mayor profundidad sobre el criterio de máxima verosimilitud se encuentra en [11] y la demostración de que, en verdad, el algoritmo de Viterbi es de máxima verosimilitud se encuentra en [12].

En pocas palabras podemos decir que el algoritmo de Viterbi rastrea el estado de un proceso estocástico usando un método recursivo que es óptimo en cierto sentido. Para realizar el rastreo del estado se utiliza un diagrama de trellis que permite observar los cambios de estado a lo largo del tiempo para un autómata como lo es un codificador convolucional. Afirmamos que es óptimo en cierto sentido, por la razón de que utiliza una función de costo o medida de similaridad entre la señal recibida, en el tiempo t_i y todas las trayectorias del trellis que entran a cada estado en el tiempo t_i ; además se considera que la probabilidad a priori es equiprobable lo cual no necesariamente es cierto.

El algoritmo de Viterbi remueve todas aquellas trayectorias que no son posibles candidatas para la elección de la trayectoria de máxima verosimilitud. Cuando dos o más trayectorias entran a un mismo estado se escoge la que tenga la mejor métrica; esta trayectoria es llamada trayectoria superviviente. La selección de la trayectorias supervivientes es calculada para todos los estados. El decodificador continúa de está manera, desplazándose en la profundidad de la estructura de trellis, haciendo decisiones que eliminan las trayectorias de mínima verosimilitud. Es evidente que el rechazo de las trayectorias inverosímiles al principio reduce la complejidad en la decodificación.

Cuando el algoritmo realiza la tarea de eliminación de las trayectorias no sobrevivientes conocida como "trace back" (rastreo retrospectivo), revisa si en un tiempo atrás, t_{i-j}, sólo ha quedado una trayectoria, es decir una única transición a un sólo estado; si esto se cumple, en ese momento realiza la decisión dura. Cabe aclarar que la eliminación de trayectorias puede dejar una sola trayectoria que se mantiene a lo largo de varias etapas contiguas del trellis y, por lo tanto, la decisión dura será sobre una secuencia.

Un ejemplo ilustrado sobre la decodificación convolucional por medio del algoritmo de Viterbi se encuentra en [13]

5.2 Principales Diferencias entre el Algoritmo de Viterbi y el SOVA

Como se mencionó, la tendencia en las comunicaciones desde la perspectiva del receptor es utilizar el algoritmo de Viterbi de manera concatenada: modulación-decodificación. Sin embargo esta implementación tiene la desventaja de que la primera etapa no aporta información sobre los valores suaves de la fuente, limitando así la etapa de decodificación.

Es aquí donde el algoritmo SOVA proporciona una solución a esta limitante, utilizando los valores suaves de la fuente para generar la decisión dura aunada a una ponderación ad hoc a cada decisión hecha.

Como el codificador fuente no logra extraer toda la redundancia de la señal de entrada y los símbolos de salida están correlacionados en alguna forma, el algoritmo SOVA utiliza una métrica en la que se incluye información acerca de dicha correlación. Esta información son los valores suaves de la fuente.

La forma en que estos valores suaves de entrada se procesan dentro del decodificador para producir un valor de confiabilidad en la decisión tomada es la siguiente: La métrica que utiliza el algoritmo de Viterbi es la distancia euclidiana, misma que se sustituye por:

$$M_k^{(m)} = M_{k-1}^{(m)} + \sum_{n=1}^{N} x_{k,n}^{(m)} L_{c k,n} y_{k,n} + u_k^{(m)} L(u_k) \qquad (11)$$

Con la ayuda de las **Fig. 1 y Fig. 3** se describirá la ecuación (11), donde "m" representa una posible trayectoria. "N" es el número de bits en la palabra codificada (por el código convolucional (2,1)), "y" es la salida del canal y "L_c" es el valor de confiabilidad del canal. El tiempo se representa con la variable "k", que para una ventana de tiempo –como en la **Fig. 3** puede tomar los valores enteros $[j\text{-}\sigma, j]$. La variable "u" indica el bit correspondiente a la transición, y "x" son los bits codificados (símbolos con redundancia).

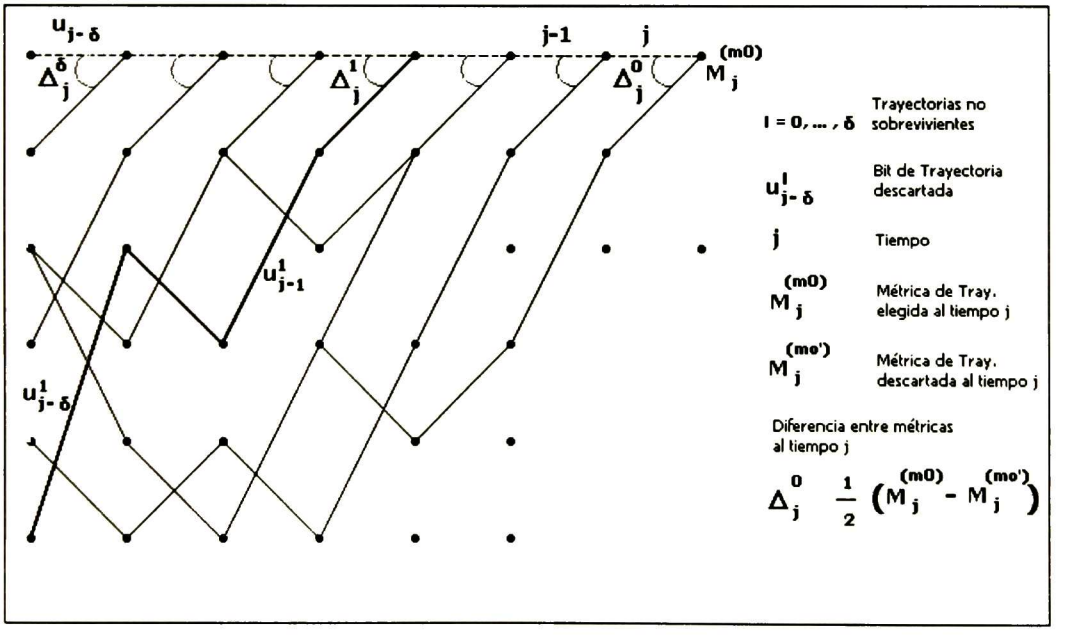

Fig. 3 Estructura de Trellis con diferencias de métricas para el algoritmo SOVA

La métrica puede ser leída así: **La distancia acumulada al tiempo k de la trayectoria m es igual a la distancia acumulada al tiempo k-1 de la misma trayectoria, mas la correlación que hay entre la palabra a la salida del canal y la palabra que corresponde a la trayectoria m mas el valor del dígito binario** (1,–1), **producido por la transición del estado anterior al estado actual, multiplicada por su valor de verosimilitud logarítmica.**

Es importante recordar que, como en el algoritmo de Viterbi, por cada estado sólo habrá una trayectoria superviviente correspondiente al valor de la métrica más grande que llegue a determinado estado.

Además del cambio en la métrica, el SOVA, en el momento del "trace back" evalúa una cantidad de certidumbre para cada trazo realizado por la trayectoria ganadora que se puede describir de la siguiente manera: Como a cada estado llegan únicamente dos trayectorias y de ellas se escoge la que tenga la mayor métrica acumulada, el SOVA almacena la información referente a la diferencia entre las métricas convergentes a cada estado. Una vez que el SOVA ha decidido una trayectoria al tiempo "j" realiza un análisis de las diferencias entre las métricas acumuladas en cada punto de la trayectoria superviviente hasta un tiempo "j-σ" Si el bit "$u^l_{j-\sigma}$" de la trayectoria descartada es igual al bit "$u_{j-\sigma}$', ciertamente no se habrá incurrido en un error si se selecciona la trayectoria descartada. Así pues se dice que la confiabilidad de la decisión para este bit es "∞" ; en cambio, si los bits difieren, el valor de confiabilidad de un error en la decisión del bit es igual a la diferencia entre métricas" Δ^l_j ". Es lógico pensar que la magnitud del valor de confiabilidad para cada bit corresponde a la diferencia con menor magnitud, es decir tuve la competencia más cercana. Entonces la cantidad de certidumbre queda expresada por

$$L(u_{j-\delta}) \approx u_{j-\delta} \cdot \min_{l=0\ldots\delta} \Delta^l_j \tag{12}$$

Con el algoritmo SOVA tenemos, por lo tanto, las mismas decisiones duras que el algoritmo de Viterbi acompañadas de una medida de confiabilidad que se obtiene tomando el mínimo de las diferencias de métricas a lo largo de la trayectoria ganadora.

5.3 El Algoritmo HUK

La existencia y la forma de la redundancia es lo único que el receptor conoce acerca de la fuente, de tal forma que requiere estimar las probabilidades de ocurrencia de cada bit en un instante dado. En la búsqueda de las probabilidades necesarias para obtener la información a priori que se menciona, se procura modelar la fuente matemáticamente.

Debido a que la codificación fuente por medio del FS1016 utiliza un formato de tramas (véase apéndice sobre especificaciones del FS1016) en el que se tienen Q bits por trama en el tiempo k, existe cierta correlación con respecto al tiempo k entre un conjunto de bits con el mismo índice q. Así, en el CELP FS1016, tenemos que para la

segunda excitación se asignan 9 bits para cada uno de los cuatro índices que conforman una trama; por consiguiente, q varía de 1 a 9 considerando un solo índice o de 1 a 36 tomando los cuatro índices.

Un simple modelo de correlación es el modelo de Markov de primer orden donde $P_{k,q}$ denota la probabilidad de que el *qth* bit del conjunto cambie desde el tiempo k-1 al tiempo k. El cambio puede ser calculado sumando en el **GF(2)** el bit previo $u_{k-1,q}$ y el bit $u_{k,q}$, generando el bit de cambio $c_{k,q}$ el cual tiene el valor de verosimilitud logarítmica

$$L(c_{k,q}) = \log \frac{P(c_{k,q} = +1)}{P(c_{k,q} = -1)} = \log \frac{1 - P_{k,q}}{P_{k,q}} \tag{13}$$

Si nosotros conocemos este valor L del bit de cambio, podemos obtener el valor L del nuevo bit $u_{k,q}$ con

$$L(u_{k,q}) = L(u_{k-1,q}) \boxplus L(c_{k,q}) \approx sign(L(u_{k-1,q})) \cdot sign(L(c_{k,q})) \cdot min(|L(u_{k-1,q})|, |L(c_{k,q})|) \tag{14}$$

el cual se utiliza como valor suave de la fuente que entra al algoritmo SOVA.

Ya que la voz es un proceso altamente no estacionario nos vemos obligados a estimar el valor L de los bits de cambio continuamente. Para esto hemos seleccionado el algoritmo HUK presentado por Hagenauer en [3].

El algoritmo HUK adquiere su nombre de una compañía alemana de seguros para automóviles y consiste en una valoración de los bits de cambio de acuerdo al diagrama de estados o niveles de la **Fig. 4.**

Los valores L de los bits de cambio son cuantificados teniendo un rango dinámico de $-L_{máx}$ a $L_{máx}$ Los valores L se mueven a través de una máquina de estados que funciona con el siguiente criterio:

- Los valores L comienzan con un valor de cero.

- Si el bit de cambio c = +1 y L es positivo, éste se incrementa en un nivel de cuantificación que, como máximo, toma el valor de $L_{máx}$; pero si L es negativo su valor se decrementa en un nivel de cuantificación que, como mínimo, toma el valor de $-L_{máx}$.

- Por otro lado si el bit de cambio c = -1 y L es positivo, éste se deprecia o decrementa en un factor D; pero si L es negativo, su valor se incrementa en un factor D.

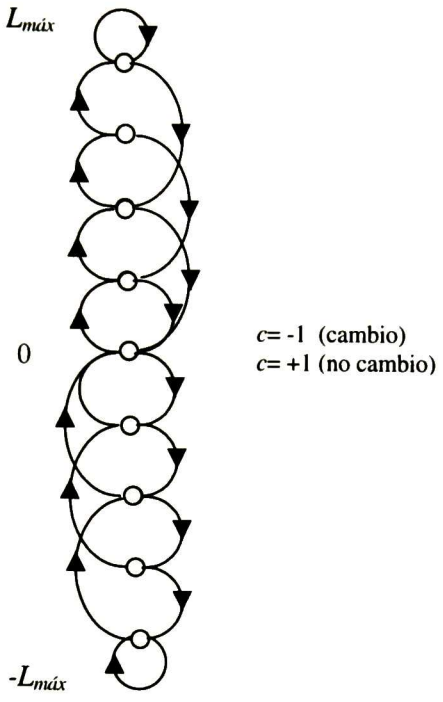

Fig. 4. Diagrama de estados que describe al algoritmo HUK

6 Experimentos Realizados

6.1 Redundancia presente en el Flujo de Bits del CELP FS1016

Para observar la redundancia que no pudo ser extraída por el codificador de fuente se procedió a la realización del siguiente experimento:

1. Procesar el archivo de voz "catedral.spd" de aproximadamente 10 seg. con el CELP FS1016.
   - El valor de la razón señal a ruido segmental (SSNR) fue de 8.34 dB con BER=0%.

2. Pasar el archivo "ofile.chan" a matlab y convertirlo a bits.
   - El archivo "ofile.chan" se encuentra en formato ASCII hexadecimal.
   - Para la conversión es necesario utilizar el algoritmo del archivo "chan2.m", el cual convierte el archivo fuente a un archivo de bits los cuales se manipulan como valores numéricos de 0 y 1.
   - El archivo de salida es "chanbin.bin".

3. Usando el algoritmo "estcelp.m" en matlab, obtenemos las estadísticas de la fuente de bits y extraemos los bits que corresponden a los índices de la segunda excitación.
   - Autocorrelación del flujo de bits arrojados por el CELP → **Fig. 5**.
   - Autocorrelación de los bits correspondientes a los índices de la segunda excitación → **Fig. 6**.
   - Histograma de los índices de la segunda excitación → **Fig. 7**.
   - Autocorrelación de los índices de la segunda excitación agrupados en ternas de 9 bits → **Fig. 8**.
   - Archivo correspondiente a los bits de la segunda excitación llamado "stream2X.bin".
   - Archivo correspondiente a los índices de la segunda excitación llamado "indices.bin".

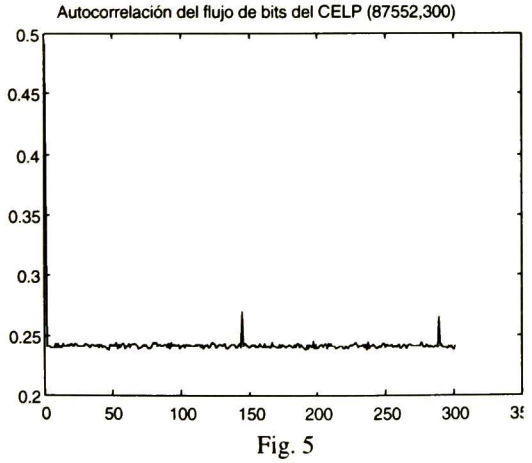

Fig. 5

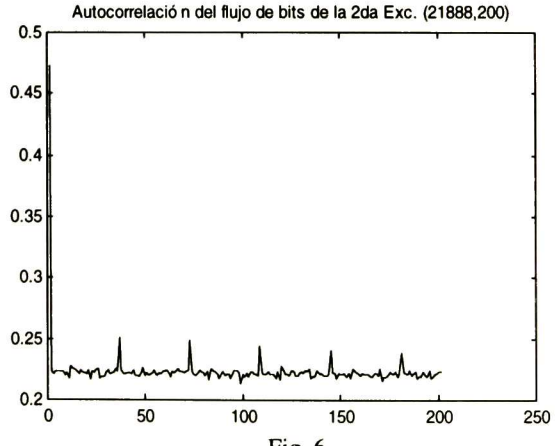

Fig. 6

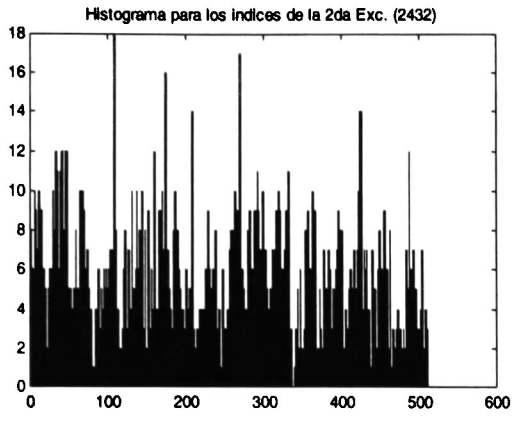

Fig. 7

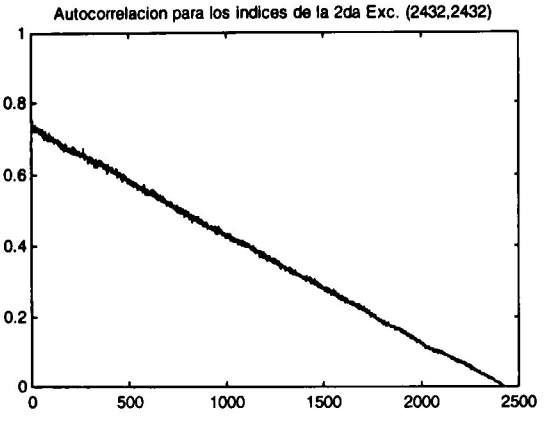

Fig. 8

6.1.1 Observaciones

El resultado del análisis estadístico de la fuente de bits se presenta en las 4 figuras anteriores:

1. En la gráfica de la **Fig. 1** podemos observar un pico periódico en la autocorrelación de la secuencia de bits a la salida del CELP cada 144 bits. Esto coincide con la longitud de la trama.

2. En la gráfica de la **Fig. 2** se muestra solamente la autocorrelación de los bits correspondientes a la segunda excitación en la que se observa el mismo patrón cada 36 bits, que equivalen a los 4 índices en una trama.

3. En la gráfica de la **Fig. 3** se muestran los niveles de ocurrencia de los índices de la codificación de la secuencia de voz correspondiente al archivo "catedral.spd"

4. En la gráfica de la **Fig. 4** se describe la autocorrelación de los índices de la segunda excitación tomados como símbolos de 9 bits.

A partir de esta información y utilizando el algoritmo HUK se han hecho los cálculos para la medida de confiabilidad requerida por el algoritmo SOVA.

6.2 Simulación del Algoritmo SOVA

Para la simulación se eligió como codificador convolucional el que usa la matriz generadora propuesta por la norma del GSM (Global System Mobile) para su codificador híbrido de voz. La razón de código del codificador convolucional es de 1/2.

$$G_0=1+D^3+D^4\,;\; G_1=1+D+D^3+D^4.$$

$$g=\begin{bmatrix}10011\\11011\end{bmatrix}$$

La tarea de codificación de canal trabaja sobre el archivo de voz correspondiente a los bits de la excitación estocástica "stream2X.bin", mediante la metodología siguiente:

1. Se toman tramas de 36 bits para cada ciclo de codificación-decodificación del archivo "stream2X.bin"
2. Se realiza la codificación convolucional por medio de la rutina "encode_block.m"
3. Se emplea la codificación de línea NRZ y se generan los símbolos "x"
4. Se contaminan los símbolos con ruido gaussiano (de acuerdo a una relación señal a ruido Eb/No entre 0, 2, 4, 6, 8, 10 dB seleccionada) y se producen los símbolos degradados "y"
5. Se pesa la salida del canal "y" con la confiabilidad del canal "L_c" y se originan los valores suaves del canal "$L_c \cdot y$"
6. Introducimos los valores suaves del canal "$L_c\, y$" junto con los valores suaves estimados de la fuente $L(\hat{u})$ - que se han estimado con el algoritmo HUK - al decodificador SOVA.
7. Realizamos la decodificación con el algoritmo SOVA produciendo las decisiones duras con su correspondiente valor de confiabilidad.
8. Si no son los últimos 36 bits decodificados, entonces regresamos al paso 1 para agarrar otros 36 bits.
9. Generamos el archivo decodificado "stream2Xd(nivel de Eb/No).bin" y el archivo de valores suaves arrojados por el SOVA "sftval(nivel de Eb/No).bin".
10. Injertamos el archivo "stream2xd(nivel de Eb/No).bin" con su complemento que se encuentra en el archivo "chanbin.bin"
11. Convertimos el archivo entrelazado "chandec.chan" a formato ASCII hexadecimal.

12. Introducimos el archivo "chandec.chan" al CELP para realizar la decodificación.

13. Hacemos comparaciones con el algoritmo de Viterbi para 0, 2, 4, 6, 8, 10 dB de relación Eb/No.

14. Generamos las curvas comparativas.

Una simulación más detallada se encuentra en [14]. Los programas se encuentran en [15]. Los archivos para procesar en [16].

6.3 Resultados Obtenidos

La gráfica de la **Fig. 9** ilustra los resultados obtenidos con el algoritmo SOVA y el algoritmo de Viterbi.

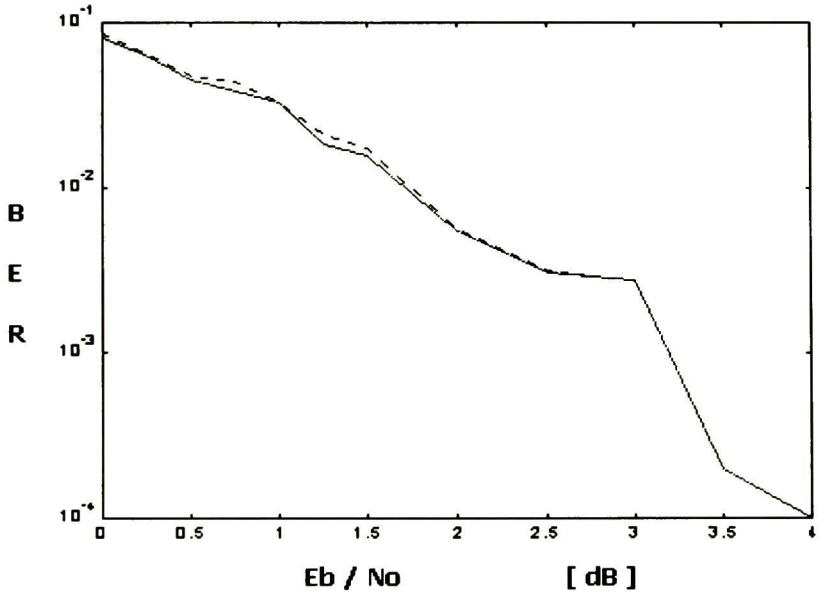

Fig. 9 SNR vs BER con Decodificación por Viterbi [-] y SOVA[.]

7 Conclusiones

- Como en la mayoría de los algoritmos de codificación conjunta, se observa que las mejoras significativas se obtienen cuando el ruido del canal es mayor.

- Los resultados realizados demostraron que si se introducen al SOVA, valores suaves referentes a la fuente con magnitud igual a cero, el SOVA tiene un comportamiento igual al algoritmo de Viterbi.

- Basándonos en la proposición anterior podemos afirmar que los valores $L(u)$ son importantes en la etapa de decodificación porque contribuyen a la métrica utilizada por el SOVA y entre más precisos sean en el momento de la estimación en el receptor, el desempeño del SOVA será mejor. En los experimentos puede verse que el algoritmo HUK no es tan bueno como se esperaba.

- Quizás, los rasgos más importantes que distinguen al SOVA del algoritmo de Viterbi son la utilización de información a priori de la fuente y los valores suaves $L(\hat{u})$ arrojados.

- Ya que el algoritmo SOVA proporciona valores suaves asociados a cada bit de decisión, es posible una decodificación iterativa mediante una forma de retroalimentación para códigos convolucionales. Ejemplos actuales de este tipo de implementaciones son los llamados códigos turbo.

- Otra forma de aprovechar los valores suaves, referentes a la fuente, calculados por el SOVA sería utilizar etapas concatenadas de decodificación con el SOVA, de manera que dichos valores se conviertan en los valores suaves de entrada $L(u)$, para la etapa de decodificación más externa.

8 Referencias Bibliográficas

[1] D. Forney, *"The Viterbi Algorithm"*, Proceedings of the IEEE vol.61 No.3 Marzo 1973.

[2] J. Hagenauer, Peter Hoecher, *"A Viterbi Algorithm with Soft-Decision Outputs and its Applications"*, In Proc. GLOBECOM '89 , pp. 1680-1686, Junio 1989

[3] J. Hagenauer, *"Source-Controlled Channel Decoding"*, IEEE Transactions on Communications, Vol. 43, No.9, Septiembre 1995.

[4] J. Hagenauer, E. Offer, L. Papke, *"Iterative Decoding of Binary Block and Convolutional Codes"*, IEEE Transactions of Information Theory, Vol. 42, No. 2, Marzo 1996.

[5] P. Eck, R. Matzner, *"A low Complexity Soft-Output Decoder for a Broad Class of Trellis Coded Modulation Schemes"*, Institute for Communications Engineering Federal Armed Forces University Munich D-85577 Neubiberg, Germany.

[6] J. Hagenauer, *"Soft is Better than Hard"*, pp. 155-165, "Communications and Cryptography", R. Blahut, D. Costello, U. Maurer, T. Mittelholzer, Kluwer Academic Plublishers 1994.

[7] C. E. Shannon, *"Collected Papers"*, N. J. A. Sloane and A. Wyner. Eds. New York: IEEE Press, 1993, p. 40.

[8] R. Blahut, D. Costello, U. Maurer, T. Mittelholzer, "Communications and Cryptography, Two Sides of One Tapestry", Kluwer Academic Publishers, 1994.

[9] A. Popescu, N. Moreau, C. Lamblin, *"CELP Coding Using Trellis-Coded Vector Quantization of the Excitation"*, IEEE Transactions on Speech and Audio Processing, Vol. 3, No. 6, Noviembre 1995.

[10] A. J. Viterbi, *"Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm"*, IEEE Transactions of Information Theory, Vol. IT-13, pp. 260-269, Abril 1967.

[11] Bernard Sklar, "Digital Communications Fundamentals and Applications", Apéndice B, Prentice Hall, 1988.

[12] Omura, J. K., *"On the Viterbi Decoding Algorithm"* (correspondence), IEEE Transactions of Information Theory, vol. IT-15, pp. 177-179, Enero 1969.

[13] Bernard Sklar, "Digital Communications Fundamentals and Applications", pp. 333 y ss, Prentice Hall 1988.

[14] O. Longoria, *"Simulación del algoritmo de Viterbi y el SOVA"*, Biblioteca del CINVESTAV Gdl.

[15] Apéndice II, *"Programas en Matlab para la Simulación del algoritmo SOVA"*

[16] Archivos para la Simulación del algoritmo SOVA, Disco anexo.

APÉNDICES

APÉNDICE I

COMPENDIO DE PROGRAMAS EN MATLAB PARA LOS ALGORITMOS LBG, COVQ Y SA EN ANÁLISIS POR SÍNTESIS

El Algoritmo LBG en Análisis por Síntesis (AbSLBG)

```
%***************************** AbSLBG *************************
%* Esta rutina encuentra un libro de códigos apropiado para una
%* aplicación de análisis por síntesis.
%* Parámetros importantes:
%* x          Secuencia de entrenamiento (archivo de voz, crudo)*
%* crudo significa muestras enteras desde -2^15 a 2^15
%* x=read_('path\filename.spd');
%* N        Número de palabras del libro de códigos (centroides)
%* q        Matriz que contiene el libro de códigos (512*60)
%* dist_inner Diferencia entre las distorsiones de
%* la iteración anterior y la iteración actual.              *
%* tol        Tolerancia permitida para la convergencia      *
%*************************************************************

%***************** Inicio de la Rutina ************************
x=read_('d:\users\longoria\voz\voces\T_S_voz.spd');
%x=read_('c:\voz\pruebas\T_S_voz.spd');
x=x';
N=512;                          %512 centroides
tol=0.0001;
format long;                    %Usar precisión de 15 dígitos
lx=length(x);                   %Long. de la sec. de entrenamiento
lt=240;                         %Trama de 4*60 muestras
lst=60;                         %longitud de la subrama
Ni=floor(lx/lst);               %Número de subtramas completas
nst=ceil(lx/lst);               %Número total de subtramas
p=zeros(nst,lst);               %Inicialización de la matriz de
                                %señales objetivo (perceptual)
colpc=zeros(1,nst*10);          %Coeficientes LPC
stH=zeros(1,10);                %Estado del filtro de síntesis
stW=zeros(1,10);                %Estado del filtro perceptual
ez=nst*60-lx;                   %Número de ceros a agregar
%********************** Código inicial ***************************
%q=randn(N,lst)*sqrt(acorr_(x,0));
```



```
q=randn(N,lst);
%Guardar la variable q      Activar esta rutina si se quiere grabar el LC
%                           inicial >
% writel_('c:\voz\pruebas\qGLBG.bin',q');
% fid=fopen('c:\voz\pruebas\cbGLBG30.h' 'a+');
% fprintf(fid,'%1.10f,',qnorm');
% fclose(fid);

x=[x zeros(1,ez)];
%************ OBTENCIÓN DE LAS SEÑALES OBJETIVO ********************
for it=1:nst
   lf=(it-1)*lst+1;             % Cota inf. y sup. para tomar las muestras
                                % a partir del vector x
   uf=it*lst;
   lp=(it-1)*10+1;              % Cota inf. y sup. para los coeficientes
                                % del filtro LPC
   up=it*10;
   st=x(lf:uf);                 % Subtrama correspondiente
   %******* CALCULO DE LOS COEFICIENTES DEL FILTRO LPC **************
   R=acorr_(st,10);             % Cálculo de 10 valores para la
                                % autocorrelación
   a=levinson(R,10);            % Coeficientes LPC
   colpc(lp:up)=a(2:11);        % Llenado del vector de coeficientes LPC
   [p(it,:),o]=wsf(st,a(2:11),stW);   %Filtro perceptual
end
clear x;
%******************** INICIO DE LAS ITERACIONES EN ABS ***************
iter=0;
dant=Inf;
dist_inner=Inf;
while (dist_inner   tol)
   iter=iter+1
   %************* FORMACIÓN DE LAS REGIONES ************************
   D=0;
   C=zeros(N,nst);              % Matriz para saber qué subtrama pertenece
                                % que centroide
   for i=1:nst
      lp=(i-1)*10+1;            % Cota inf. y sup. para los coeficientes del
                                % filtro LPC
      up=i*10;
      measure=Inf;
      index=1;
      for j=1:N
         [qf,o]=syfilt(q(j,:),colpc(lp:up),stH);   % Síntesis de la señal
                                                   % de voz
         temp=dist(p(i,:),qf');
         if (temp < measure)
            measure=temp;
            index=j;
         end
      end
      C(index,i)=1;             % Marcar la matriz C
      save_dist(iter,i)=D;      % Guardar la distorsión
      D=D + measure^2;          % Actualizar la distorsion
   end
   dist_inner=abs( dant- (D/nst) )n % Diferencia entre distorsiones
   dant = D/nst;
```

```
   %*********** CALCULO DE LOS NUEVOS CENTROIDES **********************
   for j=1:N
      sums=zeros(1,lst);
      for i=1:nst
         lp=(i-1)*10+1;    % Cota inf. y sup. para los coeficientes del
                           % filtro LPC
            up=i*10;
         if C(j,i)         % Obtención del residual a partir de la señal
                           % perceptual
            sums=sums + filter([1 colpc(lp:up)],1,wsfi(
                 p(i,:),colpc(lp:up) ) );
         end
      end
      summation=sum(C(j,:));
      if (summation > 0)
         cent=sums / summation;
         q(j,:) = cent;    % Asignación de los nuevos centroides
      end
   end
   %q=normaliza(q);
end

% GUARDAR LA VARIABLE Q
% Para leer el LC en matlab
writel_('d:\users\longoria\voz\AbSLBG\qLBG.bin',q');
% Para insertarlo en el programa en C del CELP FS1016
fid=fopen('d:\users\longoria\voz\AbSLBG\cbLBG.h' 'w');
fprintf(fid,'%1.5f,',q');
fclose(fid);
```


El Algoritmo COVQ en Análisis por Síntesis (AbSCOVQ)

```
%***************************** AbSCOVQ **************************
%* Esta rutina encuentra un libro de códigos apropiado para una
%* aplicación de análisis por síntesis.
%* Parámetros importantes:
%* x          Secuencia de entrenamiento (archivo de voz, crudo)*
%* crudo significa muestras enteras desde -2^15 a 2^15
%* x=read_('path\filename.spd');
%* N       Número de palabras del libro de códigos (centroides)
%* q       Matriz que contiene el libro de códigos (512*60)
%* dist_inner Diferencia entre las distorsiones de
%* la iteración anterior y la iteración actual.
%* tol        Tolerancia permitida para la convergencia
%* Se utiliza un modelo de canal binario simétrico.
%* epsi= Probabilidad de error en el canal.
%*****************************************************************

%***************** Inicio de la Rutina ************************
x=read_('d:\users\longoria\voz\voces\T_S_voz.spd');
%x=read_('c:\voz\pruebas\test.bin');
x=x';
% ***************************************************************
%* DEFINES
N=512;              %Número de palabras de código
tol=0.0001;         %Umbral para diferencias entre distorsiones
lt=240;             %Longitud de la trama CELP
lst=60;             %Longitud de la subtrama CELP
la=240;             %Longitud de la trama de análisis CELP
%epsi=0.001;        %Probabilidad de error del canal
epsi=0.001;
%***************************************************************
%* VARIABLES
format long;            % Usar precisión de 15 dígitos
dist_inner=Inf;         % Diferencia entre distorsión actual y anterior
lx=length(x);           % Longitud de la secuencia de entrenamiento
Ni=floor(lx/lst);       % Número de subtramas completas
nst=ceil(lx/lst);       % Número total de subtramas
p=zeros(nst,lst);       % Inicialización de la matriz de señales objetivo
                        % (voz pasada por el filtro perceptual)
colpc=zeros(1,nst*10);  % Coeficientes LPC
stH=zeros(1,10);        % Estado del filtro de síntesis
stW=zeros(1,10);        % Estado del filtro perceptual
ez=nst*60-lx;           % Número de ceros a agregar
x=[x zeros(1,ez)];      % Agregar los ceros al vector x
%***************************************************************
%* CODIGO INICIAL HECHO CON VARIABLES GAUSSIANAS
%q=randn(N,lst)*sqrt(acorr_(x,0));  % Varianza del archivo de voz T_S_voz
q=randn(N,lst);                    % Varianza unitaria
% Guardar el código inicial en un archivo, para leerlo en matlab.
```

```
%fid=fopen('cbGCOVQ.bin','a+');
%fprintf(fid,'%1.10f,',q');
%fclose(fid);
%writel_('qGCOVQ.bin',q');
%************************************************************
%* CALCULO DE LA MATRIZ DE TRANSICION DE PROBABILIDADES DEL CANAL.
Pt=zeros(N,N);
for i=0:N-1
   for j=0:N-1
      Pt(i+1,j+1)=(1-epsi)^(9-biterr(i,j))*epsi^biterr(i,j);
   end
end
% sum(Pt,1) Debe ser un vector de 512 muestras con valores igual a
% Porque P(0/0)+P(1/0)=1  **** Canal Binario Simetrico ****
%************************************************************
%* OBTENER LA SEÑALES OBJETIVO (VOZ PESADA CON EL FILTRO PERCEPTUAL)
for it=1:nst
   lf=(it-1)*lst+1;          % Cota inf. y sup. para tomar las muestras a
                             % partir del vector x
   uf=it*lst;
   lp=(it-1)*10+1;           % Cota inf. y sup. para los coeficientes del
                             % filtro LPC
   up=it*10;
   st=x(lf:uf);              % Subtrama correspondiente
   %* CALCULO DE LOS COEFICIENTES DEL FILTRO LPC
   R=acorr_(st,10);          % 10 valores para la autocorrelación
   a=levinson(R,10);         % Coeficientes LPC
   colpc(lp:up)=a(2:11);     % Llenado del vector de coeficientes LPC
   [p(it,:),o]=wsf(st,a(2:11),stW); % Señal Objetivo
end
% BORRAR LAS VARIABLES QUE NO SON INDISPENSABLES
clear x;

%************* INICIO DE LAS ITERACIONES EN ABS **********************
iter=0;
Dant=Inf;
g_dist=[];   % Arreglo de diferencias entre distorsión actual y anterior
while dist_inner>tol
   iter=iter +1
   % GRUPO DE MUESTRAS CON EL NIVEL DE CUANTIFICACIÓN MÁS CERCANO
   D = 0;
   C   zeros(N,nst);
   for i=1:nst
      up=i*10;
      lp=(i-1)*10+1;
      for j=1:N
         [qf,o]=syfilt(q(j,:),colpc(lp:up),stH);
         dxy(j)= dist( p(i,:), qf' );
      end
      [measure,code]= min(Pt * dxy');
      C(code,i) = 1;     % Marcar la matriz C
      % Actualizar la medida de distorsión
      D = D + measure;
   end
   dist_inner = abs(Dant-D/nst)       % Diferencia entre distorsiones
   g_dist(iter)=dist_inner;           % Guardar la distorsión
   Dant=D/nst;
```



```
%******* DETERMINAR LOS NUEVOS NIVELES DE CUANTIFICACION **********
   for l=1:N                  % Para cada región
      num=zeros(1,lst);
      den=0;
      for j=1:N               % Con los 512 vectores
         sums=zeros(1,lst);
         for i=1:nst          % Cuales vectores de la secuencia de
                              % entrenamiento
            lp=(i-1)*10+1;    % Cota inf. y sup. para los coeficientes
                              % del filtro LPC
            up=i*10;
            if C(j,i)         % Obtención del residual a partir de la
                              % señal perceptual
              sums=sums + filter([1 colpc(lp:up)],1,wsfi(
                   p(i,:),colpc(lp:up) ) );
            end
         end
         summation   sum( C(j,:) );
         if summation>0
            num=num+Pt(j,l)*sums;
            den=den+Pt(j,l)*summation;
         end
      end
      q(l,:)=num/den;
   end
end                       %Fin del While

% GUARDAR LA VARIABLE Q EN UN ARCHIVO
% Para leer el LC en matlab
writel_('d:\users\longoria\voz\AbSCOVQ\qCOVQ.bin',q');
% Para insertarlo en el programa en C del CELP FS1016
fid=fopen('cbCOVQ.bin','w');
fprintf(fid,'%1.5f,',q');
fclose(fid);

% GRAFICA DE LA DISTORSION
plot(g_dist);
```

El Algoritmo SA en Análisis por Síntesis (AbSSA)

```
%***************************** AbSSA *************************
%* Esta rutina encuentra un libro de códigos apropiado para una
%* aplicación de análisis por síntesis.                      *
%* Parámetros importantes:                                   *
%* x          Secuencia de entrenamiento (archivo de voz, crudo)*
%* crudo significa muestras enteras desde -2^15 a 2^15
%* x=read_('path\filename.spd');
%* N        Número de palabras del libro de códigos (centroides)
%* q        Matriz que contiene el libro de códigos (512*60)
%* dist_inner Diferencia entre las distorsiones de
%* la iteración anterior y la iteración actual.
%* tol        Tolerancia permitida para la convergencia
%* Se utiliza un modelo de canal binario simétrico.
%* epsi= Probabilidad de error en el canal.
%*************************************************************

%***************** Inicio de la Rutina ***********************
% x=read_('d:\users\longoria\matprog\T_S_voz.spd');
x=read_('c:\voz\pruebas\T_S_voz.spd');
x=x';
% *************************************************************
% DEFINES
format long;              % Usar precisión de 15 dígitos
N=512;                    % Numero de palabras de codigo
tol=0.00025;              % Umbral para diferencias entre distorsiones
lt=240;                   % Longitud de la trama CELP
lst=60;                   % Longitud de la subtrama CELP
la=240;                   % Longitud de la trama de análisis CELP
epsi=0.001;               % Probabilidad de error del canal
alfa=0.97;
% *************************************************************
% VARIABLES
T0=10;                    % Temperatura inicial
cde=0;                    % Contadores de aumento o disminunción de energía
cue=0;
cpi=0;                    % Contador de permutaciones infructuosas
iter=0;                   % Contador de iteraciones
change=0;
lx=length(x);             % Longitud de la secuencia de entrenamiento
Ni=floor(lx/lst);         % Número de subtramas completas
nst=ceil(lx/lst);         % Número total de subtramas
%p=zeros(nst,lst);        % Inicialización de la matriz de señales objetivo
colpc=zeros(1,nst*10);    % Coeficientes LPC
stH=zeros(1,10);          % Estado del filtro de síntesis
stW=zeros(1,10);          % Estado del filtro perceptual
ez=nst*60-lx;             % Número de ceros a agregar
x=[x zeros(1,ez)];        % Agregar los ceros al vector

% CODIGO INICIAL hecho con variables Gaussianas (índices arbitrarios)
%q=randn(N,lst)*sqrt(acorr_(x,0)); %Varianza igual a la del vector de voz
```



```
 q=randn(N,lst);
% ********** Rutina para usar la q que da origen al codebook.h ******
%qorigin=readl_('e:\voz\pruebas\qorigin.bin');
%y=qorigin';
%N=512;
%dl=1;ul=60;
%for i = 1:N
%   q(i,:)=y(dl:ul);
%  dl=dl+60; ul=ul+60;
%end
size(q)
%clear qorigin;
%clear y;
% *************************************************************
% GUARDAR EL LC
% < Activar esta rutina si se quiere grabar la q origen>
% writel_('e:\voz\Pruebas Marzo\codebooks\qsGSA.bin',q');
% Como ejem. esta la q gaussiana fuente(s=fuente) SA
  fid=fopen('e:\voz\Pruebas Marzo\cbsGSA.h' 'w');
  fprintf(fid,'%1.5f,',q');
  fclose(fid);

% CALCULO DE LA MATRIZ DE TRANSICION DE PROBABILIDADES DEL CANAL
Pt=zeros(N,N);
for i=0:N-1
   for j=0:N-1
      Pt(i+1,j+1)=(1-epsi)^(9-biterr(i,j))*epsi^biterr(i,j);
   end
end
% sum(Pt,1) Debe ser un vector de 512 muestras con valores igual a 1
% Porque P(0/0)+P(1/0)=1  **** Canal Binario Simetrico ****
%*************************************************************
% OBTENER LAS SEÑALES OBJETIVO
for it=1:nst
   lf=(it-1)*lst+1; % Cota inf.para tomar
                    % las muestras a partir del vector x
   uf=it*lst;       % Cota sup.
   lp=(it-1)*10+1;  % Cota inf. y sup. para los coeficientes
                    % del filtro LPC
   up=it*10;
   st=x(lf:uf);     % Subtrama correspondiente

   %********** CALCULO DE LOS COEFICIENTES DEL FILTRO LPC ************
   R=acorr_(st,10); % Calculo de 10 valores para la autocorrelación
   a=levinson(R,10);% Coeficientes LPC
   colpc(lp:up)=a(2:11);  % Llenar el vector de coeficientes LPC
   [p(it,:),o]=wsf(st,a(2:11),stW); % Matriz de Señales Objetivo
end
clear x;
% ******* INICIO DE LAS ITERACIONES en AbS********
   % GRUPO DE MUESTRAS CON EL NIVEL DE CUANTIFICACION MAS CERCANO
   D   0;
   C   zeros(N,nst);
   for i=1:nst
      up=i*10;
      lp=(i-1)*10+1;
      for j=1:N
```

```
        [qf,o]=syfilt(q(j,:),colpc(lp:up),stH);
        dxy(j)= dist( p(i,:), qf' );        % Distancia euclidiana
      end
      [measure,code]= min(Pt   dxy');

      % mark the C matrix
      C(code,i)   1;

      % update the distortion
      D = D + measure;
   end
   %*************** PROBABILIDADES APRIORI *************************
   for i=1:N
      Pa(i)=sum( C(i,:) )/nst;
   end
   clear C;
   %****** MATRIZ DE DISTANCIAS EUCLIDIANAS ENTRE CENTROIDES ****
   for i= 1:N
      for j= 1:N
         dij(i,j)=dist( q(i,:),q(j,:)' );
      end
   end

   Dc=sum( Pa    (Pt   * dij) )/lst;  % Medida de distorsión promedio
while ( (T0 > tol) ,  (cpi < 50000) )
   iter=iter +1;
   Dc_ant=Dc;
   ia=round(rand(1,2)*512)+1;        % Obtención de los índices aleatorios
   if ( max(ia)== 513 )
      ia=ia-1;
   end
   temp=q(ia(1),:);                  % Swap de los índices aleatorios
   q(ia(1),:)= q(ia(2),:);           % Para el LC,
   q(ia(2),:)= temp;
   temp=Pa(ia(1));                   % la probabilidad apriori,
   Pa(ia(1))= Pa(ia(2));
   Pa(ia(2))= temp;
   temp=dij(ia(1),:);                % y la matriz de distancias.
   dij(ia(1),:)= dij(ia(2),:);
   dij(ia(2),:)= temp;
   temp=dij(:,ia(1));
   dij(:,ia(1))= dij(:,ia(2));
   dij(:,ia(2))= temp;

   Dc=sum( Pa  (Pt   * dij) )/lst;  % Actualización de la Med.de disc.
   delta_Dc= Dc - Dc_ant;
   if delta_Dc < 0
      cde= cde + 1;
      change=change+1;
      cpi=0;
   elseif (exp(-delta_Dc/T0)    .5)  % Dejar los índices originales
      temp=q(ia(1),:);               % Swap de los índices aleatorios
      q(ia(1),:)= q(ia(2),:);        % Para el LC,
      q(ia(2),:)= temp;
      temp=Pa(ia(1));                % la probabilidad apriori,
      Pa(ia(1))= Pa(ia(2));
      Pa(ia(2))= temp;
```



```
      temp=dij(:,ia(1));             % y la matriz de distancias.
      dij(:,ia(1))= dij(:,ia(2));
      dij(:,ia(2))= temp;
      temp=dij(ia(1),:);
      dij(ia(1),:)= dij(ia(2),:);
      dij(ia(2),:)= temp;
      cpi=cpi+1;
   else
      cue=cue+1;
      change=change+1;
   end

   if ( (cde==5)| (change==200) )
      T0=T0*alfa
      g_dist(iter)=T0;
      cde=0;change=0;
   end

 end
% GUARDAR LA VARIABLE Q
% Para leer el LC en matlab
writel_('e:\voz\Pruebas Marzo\codebooks\qGSA.bin',q');
% Para insertarlo en el programa en C del CELP FS1016
fid=fopen('e.\voz\Pruebas Marzo\codebooks\cbGSA.h','w');
fprintf(fid,'%1.5f,',q'); % Cuando q proviene de un libro gausiano
%fprintf(fid,'%1.0f,',q');
fclose(fid);

% GRAFICA DE LA DISTORSION
plot(g_dist);
```

APÉNDICE II

PROGRAMAS EN MATLAB PARA LA SIMULACIÓN DEL ALGORITMO SOVA

Nombre del programa: sovithuk.m

```
% Programa para simular la codificacion-decodificacion (Viterbi y SOVA)
% Usando un codificador convolucional
% Entregando la matriz generadora (g)de la máquina de estados finitos
% msg es el mensaje original
% code es el mensaje codificado
% r es el mensaje codificado y degradado por el canal
% g es la matriz generadora

% DEFINES
con = 1;      % set infinity to an arbitrarily large valu
a=1;          % parámetro del canal
rate = 1/2;   % razón de código del codificador convolucional
lcode = 36;
c=-1;         % escoger el estado con mayor métrica al final del trellis
Lmax=2;
D=0.1;
% ************************************************************************
% ENTRADAS
%L_total = input(' Please enter the frame size (= info + tail, default: 50)   ');
%if isempty(L_total)
%   L_total = 50;   % infomation bits plus tail bits
%end
% Bits correspondientes al flujo arrojado por el CELP
source= read_('d:\users\longoria\voz\Pruebas Mayo\stream2X.bin');
source=(source(1:288))';
LX=length(source);
% ************************************************************************
g = input(' Please enter code generator: ( GSM: g=[1 0 0 1 1;1 1 0 1 1] )     ')
if isempty(g)
   g=[1 0 0 1 1;1 1 0 1 1];
end
%g=[1 1 1;1 0 1];
%g=[1 0 0;1 0 1];
%g=[1 0 0 1 1;1 1 0 1 1]; % g usada por la norma GSM para la clase B de bits de importancia
%g=[1 0 0 0 0;1 1 0 1 1];
disp(g);
[n,K] = size(g);
m = K - 1;
max_state = 2^m;
EbN0db = input(' Please enter Eb/N0 in dB : default [2.0]    ');
if isempty(EbN0db)
   EbN0db = [2 0],
end
% ************************************************************************
% PARÁMETROS CORRESPONDIENTES A LA RELACIÓN SEÑAL A RUIDO
EbN0 = EbN0db;  % Eb/N0 in dB
en = 10^(EbN0/10);
sigma = 1/sqrt(2*rate*en);  % standard deviation of AWGN noise
fprintf('\n Eb / N0 = %6.2f  dB  \n', EbN0db)
Lc = 4*a*en*rate;       % reliability value of the channel
%Lc=1;
% ************************************************************************
counter=1; % Variable auxiliar para el alagoritmo SOVA
for i=0:lcode:LX-lcode
   msg=source(i+1:i+lcode);
      %msg=(sign( randn(1, L_info) ) + 1 )/2; % Flujo aleatorio de bits
   L_total=length(msg) + m;  % Para la aplicación hecha al flujo de bits del CELP
   L_info = L_total - m;
      % CODIFICACIÓN
      en_output =encode_block(g,msg);% Tail bits are automatically % appended to force the encoder back to
the all-zeros state.
      for i=1:length(en_output)
      if (en_output(i) == 0) en_output(i) = -1;
      end
      end
% ************************************************************************
      % CORRUPCIÓN DE LOS VALORES EN EL CANAL
      % corrupted channel output
   r = en_output + sigma * randn (size(en_output));
      % weight channel output by channel reliability, to be used in decoder
   r_Lc = r*Lc;
% ************************************************************************

% DECODIFICACIÓN UTILIZANDO EL ALGORITMO DE VITERBI
% A esta decodificación se le pasa el valor de r, es decir, el valor codificado
% mas el ruido gaussiano.
msg_dec=viterbi2(g,r); % msg_dec no incluye los bits de cola
if counter == 1
   msg_vit=msg_dec;
else
   msg_vit=[msg_vit msg_dec];
end

% ************************************************************************
% USANDO EL SOVA PARA UNA DECODIFICACIÓN POR TRAMAS DE 36 BITS
% EMPLEANDO EL ALGORITMO HUK
% A esta decodificación se le pasa el valor de r multiplicado por Lc
% Para las dos primeras tramas de análisis introducimos L_in=0
if counter == 1
      L_in=zeros(1,L_total); % Probabilidad Apriori para c/bit de msg
      L1=sovaB(L_in,g,r_Lc,c);
   x1 = (sign( L1 ) + 1 )/2;
   x_hat=x1(1:L_info);
      L_sova=L1(1:L_info);
s=[ 0.9000   0.9000   1.0000   0.6000   0.1000   0.1000   1.0000...
    0.3000   0.7000   0.1000   0.5000   1.0000   1.0000   0.6000...
```

```
  0.8000   1.0000   0.1000   0.4000   0.8000   0.3000   0.4000
  0.7000   1.0000   0.5000   0.8000   0.3000   0.7000   1.0000
  0.7000   1.0000   0.4000   1.0000   1.0000   0.6000   0.1000
  0.4000 0 0 0 0];
    s=(1-s);
  L_in=sign(L1).*sign(s).*min(abs(L1),abs(s));
  L_in(37:40)=-con*ones(1,4);
else
    % L_in=zeros(1,L_total);
    L2=sovaB(L_in,g,r_Lc,e);
    x2 =(sign( L2 ) + 1 )/2;
    for i=1:length(x1)
      if x1(i)==x2(i)
          s(i)=min(Lmax,s(i)+D);
      else
      s(i)=max(-Lmax,s(i)-s(i)*D);
        end
  end
  L_in=sign(L2) *sign(s) *min(abs(L2),abs(s));% Ponderar los valores suaves de entrada
  L_in(37:40)=-con*ones(1,4);
    % L_in=zeros(1,36);
    L_sova=[L_sova L2(1:L_info)];
  x_hat=[x_hat x2(1:L_info)];
  x1=x2;x1(37:40)=zeros(1,4);  % Guardar los valores anteriores
end
  counter=counter+1
end % Fin del for principal
% CALCULO DEL NUMERO DE ERRORES
err_vit=find(msg_vit ~= source );
ber_vit = (length(err_vit)/LX)*100;
msg_sova=x_hat(1:LX);
err_sova=find(msg_sova ~= source );
ber_sova = (length(err_sova)/LX)*100;
% ************************************************************************

% DESPLEGAR INFORMACIÓN
fprintf('\n Porcentaje de BER usando Viterbi Clásico: %3.2f\n',ber_vit);
fprintf('\n Porcentaje de BER usando SOVA          %3.2f\n',ber_sova);
%fprintf('\n Bit(s) erróneo(s)  SOVA \n') err
% ************************************************************************
% Archivos a guardar
%write_('d:\users\longoria\voz\Pruebas Mayo\str,m2Xd2.bin',msg_sova );
%write_('d:\users\longoria\voz\Pruebas Mayo\strV,d2.bin',L_sova );
% ************************************************************************
```


Nombre del programa: sovaB.m

```
function  [L_sova]= sovaB(L_in, g, r, e)
%% Copyright 1997 Yufei Wu  M.C. Valenti y Omar Longoria
% for academic use only
% soft decision Viterbi decoding
% takes an encoded (and possibly corrupted) codeword with values between
% [-1 and 1] and returns the soft output
% output, transition, inv_state: description of trellis.
% g: code generator;
% r: Lc *r; where Lc is the channel reliability value, r is the received bits.
% L_in: a priori information of each bit, size(L) = L_total;
% if end>0, the trellis is perfectly terminated, trace back from all-zero state
% if end<0, the trace back from the state with the largest metric

% DEFINES
INIT_STATE = 1;
% ********************************
[n,K] = size(g);
m = K - 1;
max_state = 2^m;
rec_size = length(r);
L_total = rec_size/n;

% info bit = 0, transmission = -1; info bit = 1, transmission = +1.
u(1) = -1; u(2) = 1;

% set infinity to an arbitrarily large value
inf = 10^5;

% initialization
trellis = -inf*ones(max_state,L_total);
path = zeros(max_state,L_total);
path_bit = 10*ones(max_state,L_total);
new_path = path;
delta = inf*ones(max_state,L_total);
L_sova = zeros(1,L_total);
% ************************************************************************
% Seccion tomada del viterbi de M.C. Valenti  (Modificación 28/05/98)
% initialize output and transition matrices

for state=1:max_state
  state_vector = bin_state( state-1, m );
  [out_0, state_0] = encode_bit(g, 0, state_vector);
  [out_1, state_1] = encode_bit(g, 1, state_vector);
  output(state,:) = [out_0 out_1];
  transition(state,:) = [(int_state(state_0)+1) (int_state(state_1)+1)];
end
%*************************************************************************
% Sección tomada del Trellis de Yufei Wu
% find out which two states can come to present state
inv_state = zeros(max_state,2);
for state=1:max_state
 for bit=0:1
   if inv_state(transition(state,bit+1),bit+1)==0
     inv_state(transition(state,bit+1),bit+1)=state;
   else
```

```
      inv_state(transition(state,bit+1),2-bit)=state;
    end
  end
end
% ********************************************************************
% determine trellis and path matrices at time 1.
y_segment = r(1,1:n);
for i=0:1
  hypothesis = 2*output( INIT_STATE, n*i+1:n*(i+1) )-1; % change 0 to -1 and 1 to 1
  next_state = transition(INIT_STATE,i+1);
  path_metric =.5*L_in(1)*u(i+1)+.5*y_segment*hypothesis';
  trellis(next_state,1) = path_metric;
  path(next_state,1) = INIT_STATE;
  path_bit(next_state,1) = i;
  delta(next_state,1)=inf;
  end
% now determine trellis and path matrices for times 2 through L
counter = n +1;
for time=2:L_total
  y_segment = r( 1,counter:counter+n-1);
  counter = counter + n;
  for state=1:max_state
    for i=0:1
      hypothesis = 2*output( state, n*i+1:n*(i+1) )-1;
      next_state = transition( state, i+1 );

      % update path metric
      path_metric = trellis(state,time-1) + .5*L_in(time)*u(i+1)+.5*y_segment*hypothesis';

      if path_metric > trellis( next_state, time )
        new_delta(next_state,1:time)=[delta(state,1:time-1) ...
              abs(path_metric - trellis(next_state,time))];
        trellis( next_state, time ) = path_metric;
        path(next_state, time) = state;
        path_bit(next_state, time) = i;
      else
        new_delta(next_state,time)=abs(path_metric - trellis(next_state,time));
      end
    end
  end
  delta = new_delta;
end

% most likely path: the survived state and infomation bits
if e > 0   %perfectly terminated, trace back from all zero state
  survival(L_total) = 1;
else  % otherwise trace back from the state with the largest metric
  survival(L_total) = min(find (trellis(:,L_total)==max(trellis(:,L_total))));
end
x_hat(L_total) = path_bit(survival(L_total),L_total);
for time = L_total-1:-1:1
  survival(time) = path(survival(time+1),time+1);
  x_hat(time) = path_bit(survival(time),time);
end
% find out nonsurving paths which would have led to a different decision than
% the survived path for the info bit at the time 'window_start'
```



```
for window_start = 1 L_total     %current time 'window_start'
% largest distance between the nonsurving path trace back point and the current state
  delay = min([30,L_total-window_start]);
  compete_bit = 20*ones(delay+1,delay+1);
% competing path start point / time index 'time'
  for time = min(L_total,window_start+delay):-1:window_start
    relative= time-window_start+1;   %relative time index
    if time>1
      previous = survival(time-1);
    else
      previous = 1;              % starting from all-zero state
    end
    discard(relative) = inv_state(survival(time),...
          3-find(inv_state(survival(time),:)==previous));

    compete_sur = discard(relative);
    compete_bit(relative,relative) =  .
      find(transition(discard(relative),:)==survival(time))-1.
    if time == 1
      compete_bit(1,1) = 10.
    end
    if relative>1
      compete_bit(relative, relative-1) = path_bit(compete_sur, time-1).
    end
% competing path point trace back to window start point
    for time1 = time-2:-1:window_start
      compete_sur = path(compete_sur, time1+1);
      compete_bit(relative, time1-window_start+1) = path_bit(compete_sur, time1);
    end
  end
% find out the minimum metric difference among all the chosen nonsurving path
  min_delta = 10000;
  for k = window_start:min([L_total window_start+delay])
    if compete_bit(k-window_start+1, 1) ~= x_hat(window_start)
      min_delta = min( [min_delta delta(survival(L_total),k)]);
    end
  end
  L_sova(window_start) = (2*x_hat(window_start)-1)*min_delta.
%fprintf('\n Time= %d \n',window_start);
end
```

Nombre del programa: viterbi2.m

```
function x_hat = viterbi2( g, r )
% Usage: x_hat = viterbi( g, r )
% Soft decision Viterbi decoding algorithm
% Takes an entire convolutionally encoded (and possibly corrupted)
% codeword with values between [-1 and 1] and returns an estimate
% of the information bits (not including tail bits).
% Original author: M.C. Valenti
% For academic use only
[n,K] = size(g);
m = K - 1;
max_state = 2^m;
[temp, rec_size] = size(r);
L_total = rec_size/n;
L_info = L_total - m;
% set infinity to an arbitrarily large value
inf = 10^5;
% initialize trellis and path matrices
trellis = inf*ones(max_state,L_total);
path = zeros(max_state,L_total);
new_path = path;
% initialize output and transition matrices
for state=1:max_state
  state_vector = bin_state( state-1, m );
  [out_0, state_0] = encode_bit(g, 0, state_vector);
  [out_1, state_1] = encode_bit(g, 1, state_vector);
  output(state,:) = [out_0 out_1];
  transition(state,:) = [(int_state(state_0)+1) (int_state(state_1)+1)];
end
% determine trellis and path matrices at time 1
y_segment = r(1,1:n);
for i=0:1
  hypothesis = 2*output( 1, n*i+1:n*(i+1) )-1;
  next_state = transition(1,i+1);
  dist_euclidian = dist( hypothesis, y_segment' );
  path_metric = dist_euclidian^2;
  trellis(next_state,1) = path_metric;
  path(next_state,1) = i;
end
% now determine trellis and path matrices for times 2 through L
counter = n +1;
for time=2:L_total
  y_segment = r( 1,counter:counter+n-1 );
  counter = counter + n;
  for state=1 max_state
    for i=0:1
      hypothesis = 2*output( state, n*i+1:n*(i+1) )-1;
      next_state = transition( state, i+1 );
      % compute squared Euclidian distance
      square_dist = 0;
      for j = 1:n
        % y_semgment(j) == 0 indicates an erasure
        if y_segment(j)
          square_dist = square_dist + (hypothesis(j)-y_segment(j))^2;
        end
      end
      % update path metric
      path_metric = square_dist + trellis(state,time-1);
      if path_metric < trellis( next_state, time )
        trellis( next_state, time ) = path_metric;
        new_path( next_state, 1:time ) = [path(state,1:time-1) i];
      end
    end
  end
  path = new_path;
end
x_hat = path(1,1:L_info);
```


CINVESTAV, GDL Apéndice I I

Nombre del Programa: chan2.m

```
%Rutina para obtener las estadísticas del flujo
%de bits arrojado por el CELP FS 1016
%El archivo a leer se encuentra en formato hexadecimal (ofile.chan)
fid=fopen('d:\longoria\voz\Pruebas Mayo\ofile.chan');
S = fscanf(fid,'%s');
l=length(S);
Sbin='2'; %Para el encabezado
for i=1:l
  if ( S(i) == '0' )
    Sbin=[Sbin '0000'];
  end
  if ( S(i) == '1' )
    Sbin=[Sbin '0001'];
  end
  if ( S(i) == '2' )
    Sbin=[Sbin '0010'];
  end
  if ( S(i) == '3' )
    Sbin=[Sbin '0011'];
  end
  if ( S(i) == '4' )
    Sbin=[Sbin '0100'];
  end
  if ( S(i) == '5' )
    Sbin=[Sbin '0101'];
  end
  if ( S(i) == '6' )
    Sbin=[Sbin '0110'];
  end
  if ( S(i) == '7' )
    Sbin=[Sbin '0111'];
  end
  if ( S(i) == '8' )
    Sbin=[Sbin '1000'];
  end
  if ( S(i) == '9' )
    Sbin=[Sbin '1001'];
  end

  if ( S(i) == 'A' )
    Sbin=[Sbin '1010'];
  end
  if ( S(i) == 'B' )
    Sbin=[Sbin '1011'];
  end
     if ( S(i) == 'C' )
    Sbin=[Sbin '1100'];
  end
     if ( S(i) == 'D' )
    Sbin=[Sbin '1101'];
  end
     if ( S(i) == 'E' )
    Sbin=[Sbin '1110'];
  end
```



```
     if ( S(i) == 'F' )
       Sbin=[Sbin '1111'];
     end

  end

  for i=2:length(Sbin)
     Snum(i-1)=str2num(Sbin(i));
  end
  write_('d:\longoria\voz\Pruebas Mayo\chanbin.bin',Snum)
```

Nombre del Programa: estcelp.m

```
% Programa que obtiene las estadísticas de los diferentes parámetros
% del CELP FS-1016
% Se tienen 144 bits por trama de los cuales sólo se analizan los bits asociados al LC estocástico
% lsp 1  1-3  lsp 6      20-22
% lsp 2  4-7  lsp 7      23-25
% lsp 3  8-11lsp 8      26-28
% lsp 4  12-15 lsp 9    29-31
% lsp 5  16-19 lsp 10   32-34
% cbindex 48-56  73-81          100-108      125-133
% cbgain  57-61  82-86          109-113      134-138
% DEFINES
TRAMA = 144;
LCBINDEX = 9;
% ****************************************************************
x = read_('d:\longoria\voz\Pruebas Mayo\chanbin.bin');% Flujo de bits del CELP
LX=length(x);
% ESTADISTICAS PARA LOS INDICES DE LA SEGUNDA EXCITACION (cbindex)
%Cargar los bits correspondientes a los indices
j=1;
for i = 1:TRAMA:LX   % Hasta i<LX porque recorre hasta la ultima trama (LX-TRAMA)
  cbindex(j:j+LCBINDEX-1) = x(i+47:i+47 + LCBINDEX-1);
  j=j+LCBINDEX;
  cbindex(j:j+LCBINDEX-1) = x(i+72:i+72 + LCBINDEX-1);
  j=j+LCBINDEX;
  cbindex(j:j+LCBINDEX-1) = x(i+99:i+99 + LCBINDEX-1);
  j=j+LCBINDEX;
  cbindex(j:j+LCBINDEX-1) = x(i+124:i+124 + LCBINDEX-1);
  j=j+LCBINDEX;
end
% Convertir a string
for i=1:length(cbindex)
  cbindex_str(i)=num2str(cbindex(i));
end

write_('d:\longoria\voz\Pruebas Mayo\stream2X.bin',cbindex);
% "0" equivale a 48
% "1" equivale a 49
% Agrupar c/9 bits y convertirlos a un número decimal
j=1;
for i=1:LCBINDEX:length(cbindex_str)
  cbi(j) = bin2dec(cbindex_str(i:i+LCBINDEX-1));
  j=j+1;
end

% Histograma
cbi_his=zeros(1,512);
for j=1:length(cbi)
    cbi_his(cbi(j)+1)= cbi_his(cbi(j)+1)+1;
end
plot(cbi_his);
```


Nombre del archivo: enlaza.m

```
% Programa que inserta los bits decodificados por el algoritmo
% del SOVA (archivo stream2Xd(Eb/No en dB).bin )en el archivo binario
% "chanbin.bin" y genera el archivo "chandec(Eb/No en dB).bin".
% Posteriormente convierte el archivo "chandec(Eb/No en dB).bin"
% a formato ASCII hexadecimal y forma el archivo "chandec(Eb/No en dB).chan'

% CELP FS-1016
% 144 bits por trama
% lsp 1  1-3  lsp 6      20-22
% lsp 2  4-7  lsp 7      23-25
% lsp 3  8-11lsp 8      26-28
% lsp 4  12-15 lsp 9    29-31
% lsp 5  16-19 lsp 10 32-34
% cbindex 48-56  73-81          100-108      125-133
% cbgain  57-61  82-86          109-113      134-138

% DEFINES
TRAMA = 144;
LCBINDEX = 9;
% ****************************************************************************
% Obtener los bits del canal del CELP
x = read_('d:\longoria\voz\Pruebas Mayo\chanbin.bin');% Flujo de bits del CELP
xOrigin=x;
LX=length(x);
% ****************************************************************************
% Obtener los bits de la segunda excitación
y = read_('d:\longoria\voz\Pruebas Mayo\stream2Xd4.bin');% Flujo de bits del CELP
%Cargar los bits correspondientes a los indices
j=1;
for i = 1:TRAMA:LX   % Hasta i<LX porque recorre hasta la ultima trama (LX-TRAMA)
  x(i+47:i+47 + LCBINDEX-1)= y(j:j+LCBINDEX-1);
  j=j+LCBINDEX;
  x(i+72:i+72 + LCBINDEX-1)= y(j:j+LCBINDEX-1);
  j=j+LCBINDEX;
  x(i+99:i+99 + LCBINDEX-1)= y(j:j+LCBINDEX-1);
  j=j+LCBINDEX;
  x(i+124:i+124 + LCBINDEX-1)= y(j:j+LCBINDEX-1);
  j=j+LCBINDEX;
end
err=find(xOrigin ~= x);
  write_('d:\longoria\voz\Pruebas Mayo\chandec4 bin',x);
% Rutina que convierte el archivo "chandec(Eb/No en dB).bin"
% en el archivo "chandec(Eb/No en dB).chan" (Formato ASCII hexadecimal)
x=x';
xHex='7'; %Para el encabezado
for i=1:4:LX
  if ( x(i:i+3) == [0 0 0 0] )
    xHex=[xHex '0'];
  end
  if ( x(i:i+3) == [0 0 0 1] )
    xHex=[xHex '1'];
  end
  if ( x(i:i+3) == [0 0 1 0] )
    xHex=[xHex '2'];
  end
```

```
    if ( x(i:i+3) == [0 0 1 1] )
      xHex=[xHex '3'];
    end
    if ( x(i:i+3) == [0 1 0 0] )
      xHex=[xHex '4'];
    end
    if ( x(i:i+3) == [0 1 0 1] )
      xHex=[xHex '5'];
    end
    if ( x(i:i+3) == [0 1 1 0] )
      xHex=[xHex '6'];
    end
    if ( x(i:i+3) == [0 1 1 1] )
      xHex=[xHex '7'];
    end
    if ( x(i:i+3) == [1 0 0 0] )
      xHex=[xHex '8'];
    end
    if ( x(i:i+3) == [1 0 0 1] )
      xHex=[xHex '9'];
    end
    if ( x(i:i+3) == [1 0 1 0] )
      xHex=[xHex 'A'];
    end
    if ( x(i:i+3) == [1 0 1 1] )
      xHex=[xHex 'B'];
    end
    if ( x(i:i+3) == [1 1 0 0] )
      xHex=[xHex 'C'];
    end
    if ( x(i:i+3) == [1 1 0 1] )
      xHex=[xHex 'D'];
    end
    if ( x(i:i+3) == [1 1 1 0] )
      xHex=[xHex 'E'];
    end
    if ( x(i:i+3) == [1 1 1 1] )
      xHex=[xHex 'F'];
    end
  end
  xHex=xHex(2:l);
  l=length(xHex);
  fid=fopen('d:\longoria\voz\Pruebas Mayo\chandec4.chan','w');
  fprintf(fid,'%s',xHex');
  fclose(fid);
```


Nombre del programa: papriori.m

```
% Programa para capturar la probabilidad apriori
% de los vectores del libro de códigos
% Esta probabilidad se obtuene del programa celp c
% que al ejecutarse con un archivo de voz (nomarch.spd)
% captura la frecuencia de uso de cada uno de los 512
% vectores del libro de código.
% Dicha información se recupera en un archivo editable
% luego se carga en una variable de matlab posteriormete
% se graba esta variable en un archivo (nomarch.bin)
% Este archivo será utilizado por el algoritmo de SA
% function papriori()
pa =[1 4 3 5 4 4 4 3 6 2 7 3 4 4 3 6 7 17 7 1 17 7 3 ...
   7 10 7 12 7 7 10 8 5 3 5 8 6 6 6 11 10 2 5 9 9 5 4 8 ...
   3 5 10 2 4 5 5 3 2 3 5 2 2 9 4 5 8 6 4 6 4 13 3 4 6 ..
   0 9 6 9 5 10 16 13 8 4 17 10 2 6 5 4 10 3 14 4 6 11 3 4 . .
   4 6 8 4 4 4 2 10 2 5 6 4 9 2 1 3 5 7 4 5 7 7 6 5 7 4 8 ...
   4 10 4 4 2 7 8 2 10 3 3 15 14 4 6 3 3 7 6 11 2 2 9 2 4 ...
   5 8 7 6 3 8 10 4 5 9 6 15 3 2 2 5 4 4 6 7 9 5 1 9 4 2 5 ...
   12 6 5 4 7 6 7 7 1 8 6 3 1 10 5 5 4 3 10 5 3 12 3 2 7 12 ...
   2 6 3 5 4 2 5 4 5 4 2 11 5 7 4 1 8 5 9 5 6 9 6 3 5 7 4 ...
   8 12 7 4 1 8 7 4 1 5 7 11 2 4 4 8 0 7 6 6 2 5 7 5 4 2 ...
   10 5 9 6 4 3 7 7 10 9 4 3 3 8 8 5 5 1 3 4 8 6 4 3 7 13 ..
   12 6 5 2 8 2 8 2 4 9 6 12 4 8 4 4 11 7 6 7 11 6 5 5 4 .
   2 2 4 7 4 3 5 7 4 10 3 9 4 8 7 3 8 6 8 13 6 8 5 6 3 9 ...
   7 1 9 6 8 4 4 4 5 7 3 10 2 9 3 4 10 6 6 5 4 8 7 8 4 9 ...
   2 7 9 11 4 0 3 2 6 11 5 2 13 13 5 0 0 4 6 11 5 7 5 9 ...
   12 10 10 4 7 6 5 10 9 10 0 3 4 6 8 2 4 8 7 3 1 9 5 5 ...
   8 3 4 9 9 4 9 10 3 5 3 4 9 6 6 7 5 2 5 9 4 4 1 7 8 4 ...
   7 4 9 15 18 8 14 6 7 8 8 10 9 5 4 5 2 7 2 3 8 4 6 12 ...
   4 9 4 1 7 9 4 3 6 3 16 5 5 9 8 2 2 7 8 2 2 7 8 4 6 8 ...
   6 6 5 4 3 3 11 5 12 1 3 1 8 2 3 4 5 7 5 1 2 2 3 15 ...
   10 5 5 3 6 4 12];
write_('d:\users\longoria\voz\codebooks\papriori.bin',pa);
```

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN
UNIDAD GUADALAJARA

El Jurado designado por el Laboratorio de Ingeniería Eléctrica y Ciencias de la Computación del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó la tesis: **Codificación de Fuente y Canal en el CELP FS1016** el día 9 de Octubre de 1998.

El jurado:

p.a. Manuel Guzmán

A. Ramírez

Dr. Manuel Guzmán Rentería
Investigador CINVESTAV 3A
Jefe del LIECC DEL
CINVESTAV DEL IPN
Guadalajara

Dr. Antonio Ramírez Treviño
Investigador CINVESTAV 2A
CINVESTAV DEL IPN
Guadalajara

p.a Arturo Vel.

Dr. Óscar Yañez Suárez
Profesor Titular C tiempo completo
Facultad: División de Ciencias Básicas
e Ingeniería.
Universidad Autónoma Metropolitana