B19-16169

CINVESTAV

Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara de Ingeniería Avanzada

Excelencia en Investigación, Educación y Desarrollo Tecnológico

Laboratorio de Ingeniería Eléctrica y Ciencias
de la Computación-Guadalajara

Programación y Simulación del Estándar MELP para Codificación de Voz a 2400 bps

Tesis que presenta
Alejandro Leñero Beracoechea

Para obtener el grado de
Maestro en Ciencias

En la especialidad de
Ingeniería Eléctrica

Guadalajara, Jal., Octubre de 1998

Programación y Simulación del Estándar MELP para Codificación de Voz a 2400 bps

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:

Alejandro Leñero Beracoechea

Ingeniero Electrónico
Instituto Tecnológico y de Estudios Superiores de Occidente
1992-1996

Becario del CONACYT, expediente no. 116195

Director de Tesis
Dr. Arturo Veloz Guerrero

CINVESTAV del IPN Unidad Guadalajara, Octubre de 1998

Agradecimientos

A Dios por darme la oportunidad de realizar mis estudios de maestría.

A mis Padres y a mis Hermanos por todo el apoyo que me brindaron durante mis estudios.

A mi novia Celia Edith por todo su cariño y su apoyo.

A todos mis Profesores del CINVESTAV, especialmente al Dr. Manuel Guzmán y a mi asesor el Dr. Arturo Veloz.

A mis amigos y mis compañeros del CINVESTAV, especialmente al grupo de codificación de voz, Omar, Ramón y Mario.

ÍNDICE

Capítulo I. Introducción

La codificación digital de voz es un tema que se ha venido estudiando por mucho tiempo, sin embargo, es en las últimas décadas donde ha jugado un papel de suma importancia en la comunicación y almacenamiento de la información de la voz humana. Además, la codificación digital de voz ha permitido que sistemas como el telefónico puedan funcionar conjuntamente con otros sistemas de comunicación de datos, como redes de computación.

El tipo de codificación de voz utilizada en telefonía digital convencional (PCM a 64 Kbps ley-A o ley-μ) tiene una complejidad de implementación muy baja, sin embargo, no es considerada una codificación óptima ya que existe una gran redundancia de información. En muchas aplicaciones donde el ancho de banda es restringido este tipo de codificación no es funcional, por ello se han estudiado otras técnicas de codificación de voz que disminuyan la velocidad de transmisión o la cantidad de información a almacenar.

Como se mencionó en el párrafo anterior no todos los codificadores de voz utilizan las mismas técnicas ni la misma velocidad de transmisión, por lo que se puede hacer una clasificación de los codificadores ya sea por la velocidad o por el método o técnica que se utiliza para codificar la voz, donde en general la velocidad está directamente relacionada con la técnica de codificación. De una manera muy general se puede clasificar a un codificador por su velocidad de transmisión como alta, media, baja o muy baja. Por otra parte un codificador de voz también puede clasificarse por la técnica que utiliza para codificar la voz como, codificador de forma de onda, paramétrico o híbrido. Una descripción más profunda de estas clasificaciones se verá en el capítulo II.

El problema con los codificadores digitales es que todos contienen un grado de pérdida de información que va desde el error de cuantificación hasta el modelado inexacto del sistema de producción de la voz. En el caso de los codificadores considerados de baja o muy baja velocidad, las pérdidas son tales que la señal de voz reconstruida es una versión distorsionada de la original y su calidad muchas veces no es la deseada. Un ejemplo clásico de lo mencionado anteriormente es el antiguo estándar federal de los Estados Unidos de Norte América LPC 10E (FS-1015) a 2400 bps [10], en el cual la voz reconstruida está considerada como no natural, áspera, ruidosa, y poco inteligible.

Un punto que vale la pena recalcar es que a medida que la velocidad de transmisión del codificador disminuye, los beneficios en cuanto a la transmisión de información y almacenamiento aumentan, sin embargo, también la complejidad del procesamiento tiende a aumentar y la calidad de la voz a disminuir, por lo que debe buscarse un compromiso entre la complejidad del algoritmo, la velocidad de

transmisión, la calidad de la señal de voz reconstruida y el retardo que se genera por el procesamiento de la señal.

Muchas técnicas para mejorar la calidad de voz reconstruida manteniendo la misma velocidad han sido propuestas por diversos autores (en el capítulo II se mencionan algunas). Para el caso del estándar LPC 10E una variedad de codificadores fueron propuestos para sustituirlo como nuevo estándar federal a 2400 bps de los Estados Unidos de Norte América. Los algoritmos propuestos debían tratar de mejorar considerablemente la calidad de la voz reconstruida. Después de extensas pruebas se eligió un codificador conocido como MELP [1] (Mixed Excitation Linear Predictor), el cual fue un trabajo desarrollado conjuntamente por Texas Instruments y Atlanta Signal Processors.

El objetivo de la presente tesis es el estudio de técnicas de codificación de fuente para la señal de voz catalogadas como codificadores de muy baja velocidad, por esto, en el capítulo II se hace especial énfasis en este tipo de codificadores. Además se eligió el codificador MELP a 2400 bps para realizar una simulación en *Matlab* de este algoritmo, con el fin de comprender los principios en los que se basa. La programación del algoritmo está basada en el "draft" del nuevo estándar federal de E.U.A. [12] (MELP a 2400 bps), debido a que la versión definitiva del mismo aun no ha sido terminada. De igual manera se busca verificar el desempeño y robustez del algoritmo bajo condiciones de canal ideal y ruidoso. Este trabajo pretende ser la base para realizar una implementación en hardware del algoritmo MELP con el fin de probarlo en condiciones reales de trabajo.

Los principios en que está basado el nuevo modelo y su implementación son considerados en el capítulo III de esta tesis, mientras que la evaluación del desempeño del algoritmo es considerado en el capítulo IV. Los resultados obtenidos junto con las conclusiones son discutidos en el ultimo capítulo del presente trabajo.

Algunos de los temas relacionados con la implementación y funcionamiento del algoritmo son tratados en los apéndices anexos al final de esta tesis.

Capítulo II. Panorama de Codificadores de Voz

2.1 Introducción a los codificadores de voz

Un codificador digital de voz tiene como objetivo representar la información de la señal de voz, por medio de símbolos binarios (o palabras del código). La información de la señal puede ser representada por medio de muestras o de parámetros de la misma. Las principales aplicaciones de los codificadores de voz caen en dos grandes categorías: La transmisión digital de la señal de voz y el almacenamiento de la señal de voz. Dentro de la primera categoría se encuentran aplicaciones como sistemas de comunicación digital, telefonía, radio móvil, etc. mientras que dentro de la segunda categoría se encuentran aplicaciones como contestadoras digitales y sistemas de respuesta con voz. En ambos campos de aplicación el objetivo principal es obtener la mejor calidad de voz, con el menor gasto posible (baja complejidad) y con la menor velocidad de transmisión. En la primera categoría las condiciones del canal y el retardo son aspectos importantes que deben también ser considerados.

De acuerdo con el capítulo anterior se puede hacer una clasificación de los codificadores de voz ya sea por velocidad o por la técnica utilizada en la codificación. En este capítulo se presenta un esquema que trata de relacionar lo mejor posible a cada una de las técnicas de codificación de voz con su velocidad de transmisión. Además se explican los principios en que están basadas las diferentes técnicas (codificadores de forma de onda, híbridos y paramétricos), sus ventajas y desventajas.

Como se mencionó en el capítulo anterior se hace especial énfasis en los codificadores catalogados en el rango de muy baja velocidad de transmisión, tratando de ubicarlos en un esquema general de codificadores de voz. Es importante hacer notar que los codificadores considerados en este trabajo son codificadores de fuente y no de canal. Otro punto que debe mencionarse es que la mayoría de los codificadores consideran a la señal de voz como una señal que está limitada en frecuencia, por ejemplo, para telefonía está limitada dentro del rango entre 200 Hz y 3400 Hz, por lo que de acuerdo con el teorema de muestreo, la frecuencia mínima de muestreo debe de ser 6800 Hz. En la codificación de voz normalmente se utiliza una frecuencia de muestreo de 8000 Hz, para evitar traslape en frecuencia ("aliasing").

2.2 Esquema de codificadores de voz

La técnica de codificación de voz está estrechamente relacionada con su velocidad de transmisión. En general los codificadores que utilizan la codificación de forma de onda operan a razones de bits considerada como alta o media (16 Kbps o mayor), por otra parte los codificadores híbridos normalmente operan a razones de bits media o baja (mayor que 2.4 Kbps y hasta 16 Kbps) y por último la técnica de

codificación que opera a velocidad de transmisión considerada muy baja (2.4 Kbps o menos) es la codificación paramétrica. Lo anterior se ilustra en la figura 1.

Con lo anterior no se quiere decir que todos los codificadores que utilizan determinada técnica de codificación deban encontrarse forzosamente operando dentro del rango estipulado anteriormente, es decir, los codificadores paramétricos no siempre van a estar dentro del rango considerado como codificadores a muy baja velocidad de transmisión, como es el caso del MELP a 4800 bps [2], o que un híbrido no pueda ser considerado dentro de este mismo rango [3]. Sin embargo, la asociación realizada de la técnica de codificación con su velocidad de transmisión es la más común.

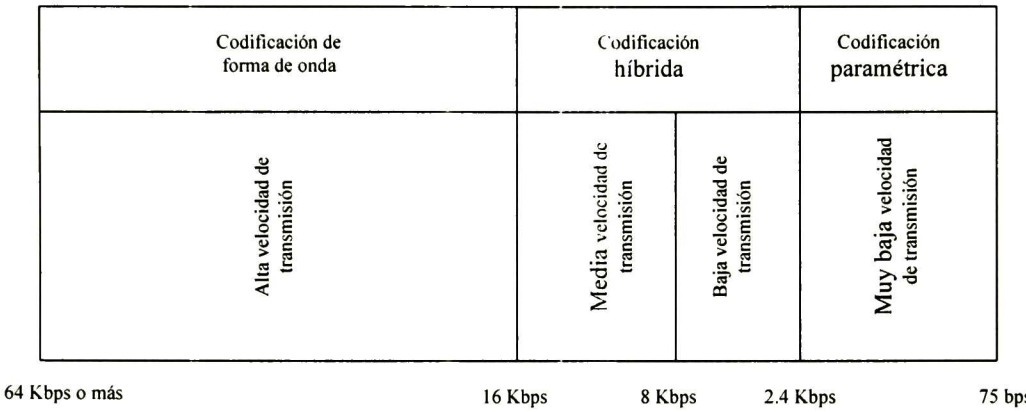

Figura 1. Relación de la técnica de codificación con su velocidad de transmisión.

2.3 Codificación de forma de onda

La codificación de forma de onda es la técnica más sencilla para codificar la voz, y está basada en la codificación de las muestras de la forma de onda ya sea de la señal ó de un residual, como sucede en la codificación DPCM y ADPCM, que se verá más adelante. Las muestras obtenidas son entonces cuantificadas y posteriormente codificadas, estos códigos son transmitidos o almacenados (dependiendo de la aplicación) y son utilizados para reconstruir la forma de onda de la señal.

PCM (*Pulse Code Modulation*) es el más simple de los codificadores de forma de onda. La velocidad de transmisión que se puede obtener es hasta de 64 Kbps, tomando en cuenta que el muestreo de la señal de voz es a 8000 Hz y que las muestras son codificadas en palabras de 8 bits (con 8 bits se obtiene un cuantificador de 256 niveles). Sin embargo, existen codificadores PCM que consideran más niveles de cuantificación, una frecuencia de muestreo mayor o ambos, lo que conlleva a un incremento de la velocidad de transmisión. Un diagrama del codificador PCM se muestra en la figura 2 a).

Otras técnicas de codificación de forma de onda como DPCM y ADPCM buscan reducir la velocidad de transmisión aprovechando la redundancia que existe entre muestras contiguas. En estas técnicas un residual de la señal de voz es codificado, y el código es enviado al decodificador para reconstruir la forma de onda de señal.

El codificador DPCM (*Differential Pulse Code Modulation*) cuyo diagrama es mostrado en la figura 2 b), utiliza un predictor que tiene como función estimar la muestra que se quiere codificar basándose en su pasado. El predictor más sencillo es el de primer orden, sin embargo, se puede utilizar un predictor de orden mayor. El error de predicción (o residual) es el valor que se obtiene cuando a la muestra de la señal original se le resta el valor estimado por el predictor. La ganancia en bits se debe a que la señal a cuantificar (el residual o error de predicción) tiene una varianza más pequeña que la señal de voz, por lo que se necesitan menos niveles de cuantificación para obtener el mismo error de cuantificación, lo que produce una menor velocidad de transmisión. Bajo este esquema también es posible mantener una misma velocidad de transmisión, dando como resultado un error de cuantificación menor y obteniendo por tanto, mejor calidad en la señal reproducida por el decodificador.

En el modelo DPCM tanto el cuantificador como el predictor son fijos, sin embargo, la señal de voz no tiene un comportamiento igual en todo tiempo, por lo que el error de predicción puede llegar a tener valores muy alejados del rango de cuantificación, si es que éste es muy restringido. Una variante del codificador DPCM toma en cuenta la naturaleza no estacionaria de la voz, introduciendo un predictor y un cuantificador adaptable. Esta variante es conocida como ADPCM (*Adaptive Differential Pulse Code Modulation*), figura 2 c), en la cual se logra disminuir la velocidad de transmisión hasta 32 kbps [4] manteniendo una calidad de voz comparable con la voz generada por el codificador PCM a 64 Kbps. Ambas técnicas son actualmente utilizadas en sistemas de telefonía.

Los cuantificadores utilizados en la codificación de forma de onda pueden o no tomar en cuenta las características de la señal que se está muestreando (independientemente del número de niveles de cuantificación). El cuantificador más simple y que no toma en cuenta las características de la señal, es el cuantificador uniforme. Para la señal de voz se han propuesto cuantificadores logarítmicos (ley-A o Ley-μ), que asignan un mayor número de niveles de cuantificación en voltajes pequeños, la razón de esto es que la señal de voz permanece más tiempo en niveles bajos de voltaje. Otros cuantificadores tratan de tomar en cuenta las estadísticas de la señal para asignar las fronteras del cuantificador, un ejemplo de esto es el algoritmo de Max-Lloyd.

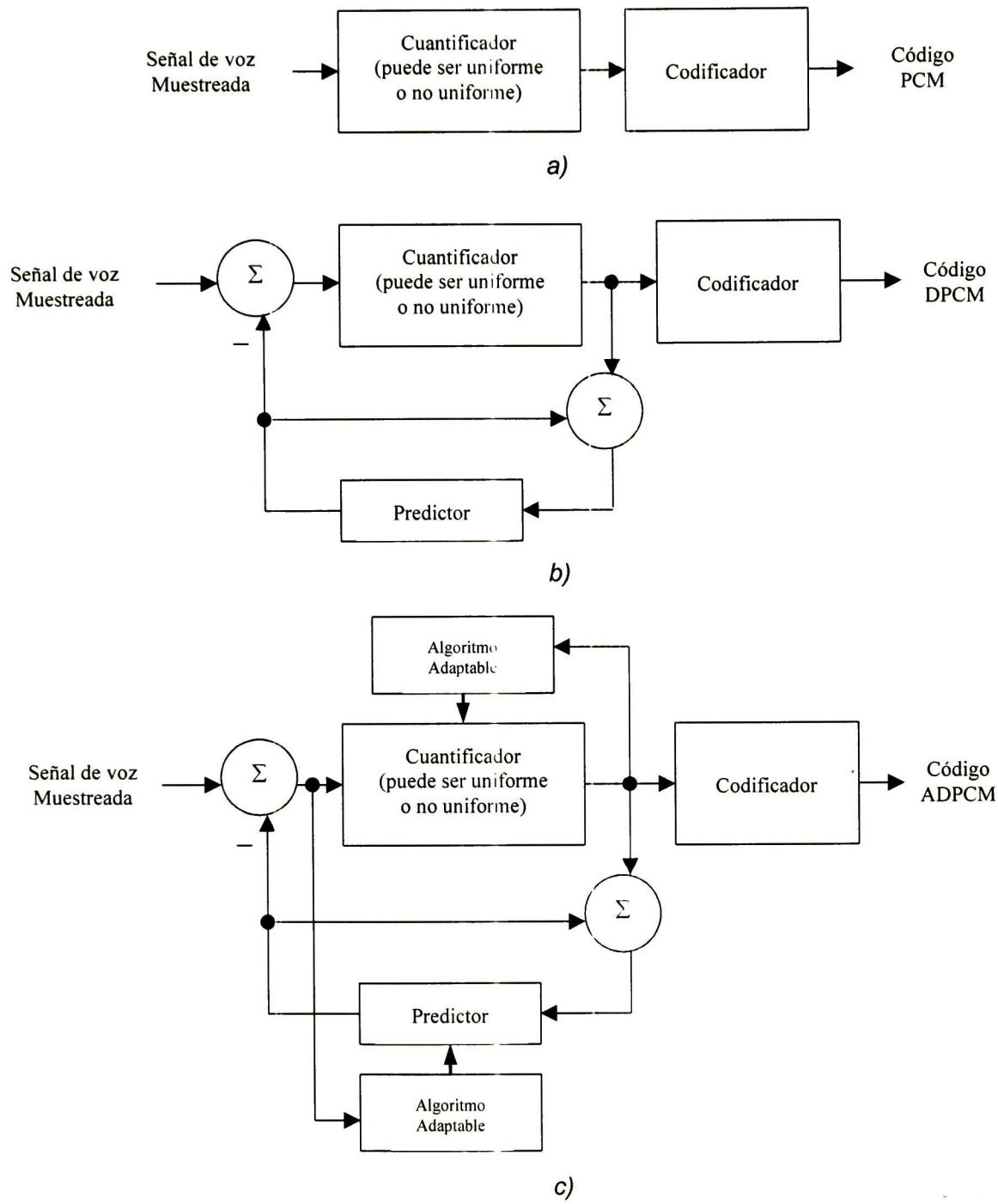

Figura 2. Codificadores de forma de onda a) PCM, b) DPCM, c) ADPCM.

Una de las ventajas de la codificación de forma de onda además del bajo costo de implementación y la alta calidad de la señal reconstruida, es que puede ser utilizada para codificar cualquier tipo de señal, es decir, la codificación no está restringida a la señal de voz, como sucede normalmente en la codificación paramétrica e híbrida.

Aunque los codificadores de forma de onda generan voz de alta calidad y su complejidad no es muy alta, existen aplicaciones como telefonía celular, radio móvil, almacenamiento y otras, en que la compresión de información es un factor importante, ya sea por un ancho de banda restringido o por capacidad de almacenamiento restringida. En este tipo de aplicaciones utilizar un codificador de forma de onda puede llegar a ser prohibitivo, por lo cual se emplean otro tipo de técnicas, que aunque tengan mayor costo de implementación logran disminuir la velocidad de transmisión, estas técnicas son estudiadas a continuación.

2.4 Codificación paramétrica

La voz humana es una señal cuyas estadísticas varían con el tiempo, sin embargo, en períodos de tiempo cortos (entre 10 y 40 mseg) puede considerarse como una señal cuyas estadísticas temporales se conservan (es decir, puede considerarse a la señal de voz como un proceso estacionario). Durante estos períodos de tiempo se puede definir un sistema (lineal o no lineal) que modele el sistema del tracto vocal, el cual al ser excitado por una señal adecuada es capaz de sintetizar voz. De lo anterior pueden distinguirse dos partes fundamentales de la codificación paramétrica, el modelo del sistema del tracto vocal y el modelo para generar la excitación.

En la codificación paramétrica se busca obtener un modelo adecuado tanto para el tracto vocal como para la excitación. Los parámetros de dichos modelos son enviados (o almacenados) y utilizados para generar un estimado de la voz. Al proceso de extracción de parámetros se le conoce como análisis mientras que al proceso de reconstrucción de la señal se le llama síntesis.

A diferencia de la codificación de forma de onda, este tipo de codificación intenta tomar en cuenta las características de la señal para reducir la velocidad de transmisión, sin embargo, la calidad de voz obtenida en el proceso de decodificación tiende a ser muy pobre; lo anterior se debe a un modelado incorrecto del tracto vocal, de la excitación o de ambos. Este tipo de codificadores también trata de tomar en cuenta el mecanismo perceptual del oído humano, en el cual la magnitud de las componentes frecuenciales de la señal tiene una gran importancia, mientras que la fase juega un papel poco relevante, es por ello que la forma de onda de la señal sintetizada por un codificador paramétrico puede llegar a ser muy diferente a la forma de onda de la señal original.

2.4.1 Modelos de la excitación y del tracto vocal

Existen diferentes técnicas para modelar cada una de las partes que conforman a un codificador paramétrico. El modelo más simple para la excitación se basa en una discriminación de la señal de voz en sonora o no sonora. Normalmente una porción de la señal se declara no sonora cuando tanto la energía como su función de autocorrelación (o una medida similar) son muy bajas, esta porción de la señal puede ser modelada entonces por medio de ruido blanco y niveles bajos de ganancia. Por otro lado cuando la porción de la señal que se está analizando contiene un nivel de energía alto y el valor de su función de autocorrelación es también alto (existe periodicidad), se dice que es una porción de voz sonora, la cual puede ser modelada por un tren de pulsos cuya separación entre pulsos es equivalente al *pitch*. El ***pitch*** puede ser obtenido por muchos métodos, y **es el menor tiempo en el que un segmento de la forma de onda de la señal de voz se repite**.

A pesar de la simplicidad del modelo anterior, no es considerado bueno ya que la señal de excitación se aproxima burdamente. Debido a lo anterior se propuso un modelo de excitación que mezcla ruido y pulsos [5], en este modelo se hace un análisis por bandas de la señal de voz, determinándose la sonoridad en cada una de las bandas por medio de la función de autocorrelación (o una medida similar). Cuando existe periodicidad en una banda, la excitación en esa banda es dominada por una señal periódica con período de *pitch* (por ejemplo, un tren de pulsos). Por otro lado, si no existe energía periódica, la excitación en esa banda es dominada por ruido. Ambas excitaciones se suman formando un modelo más preciso de excitación.

Otro modelo de excitación es realizar una suma de señales senoidales a diferente frecuencia. A este tipo de codificación se le conoce como codificación armónica en la cual, existe una señal senoidal con frecuencia fundamental (el inverso del *pitch*) a la cual se le suman señales senoidales cuyas frecuencias son múltiplos de la frecuencia fundamental [6].

La excitación modelada es introducida al sistema que modela el tracto vocal, teniendo como salida un estimado de la voz. Algunos de los modelos del tracto son la determinación de formantes, muestras de la envolvente espectral, coeficientes cepstral, predicción lineal, etc. La primera técnica para modelar el tracto busca encontrar la posición de los formantes y mandar esta información al decodificador. Los formantes son las regiones de mayor energía del espectro de la señal de voz. Pueden determinarse de muchas maneras, una de ellas es modelarlos como funciones de densidad de probabilidad [7] cuya media corresponda a la posición del formante y la varianza al ancho de banda.

Una técnica que ha dado buenos resultados, es el modelo del tracto por medio de un filtro todo polos, esta técnica propone que cada una de las muestras de la señal de voz se puede obtener como una combinación lineal de su pasado, una de las

razones de su eficiencia puede deberse a que los polos del filtro corresponden en gran medida a los formantes de la señal. A esta técnica se le conoce como predicción lineal y muchos codificadores paramétricos e híbridos la utilizan como modelo del tracto. Debido a que el algoritmo simulado en la presente tesis está basado en esta técnica, y a su importancia en los codificadores a bajas razones de bits, esta técnica se explica con detalle en el apéndice A.

2.4.2 Algunos codificadores paramétricos

LPC-10E

Este modelo utiliza el modelo más simple de excitación. El cual como ya se había mencionado, discrimina una porción de voz en sonora o no sonora. Como modelo del tracto utiliza la técnica de predicción lineal, obteniendo parámetros para un filtro de orden 10. Los parámetros que se requiere transmitir son el pitch, coeficientes del filtro, ganancia y una decisión binaria de sonoridad (V/UV). Estos parámetros se obtienen por tramas en el análisis, el intervalo de tiempo de la trama varia entre 10 y 40 mseg. Un diagrama del modelo se presenta en la figura 3.

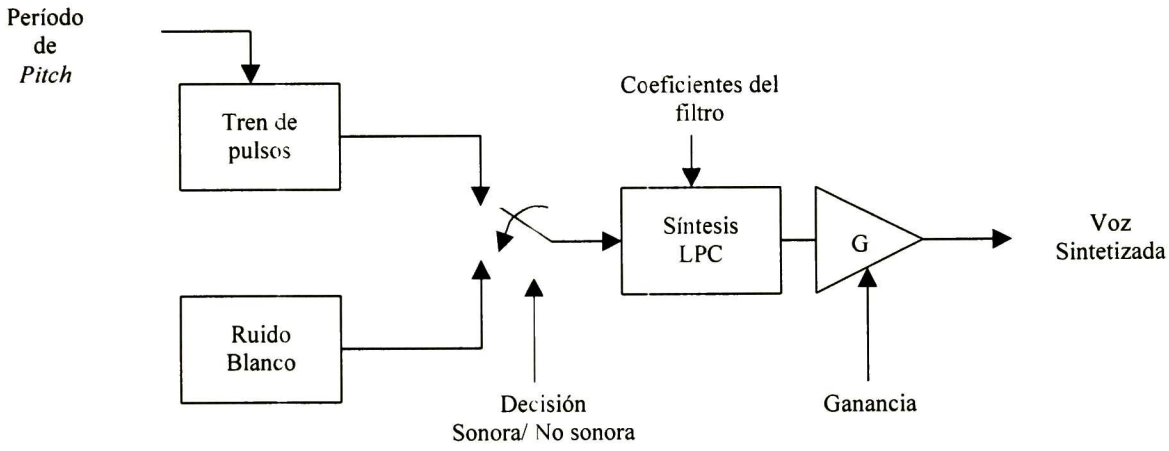

Figura 3. Diagrama a bloques del decodificador LPC-10E.

Cuando en la decisión de sonoridad se indica que la trama es sonora la excitación al filtro es un tren de pulsos con período de *pitch*, de otra manera la excitación al filtro es ruido blanco. Este codificador puede producir voz inteligible a razones de hasta 2400 bps [8],[10], sin embargo, la calidad de la voz sintetizada no es buena, una posible explicación a esto es el pobre modelo de excitación.

Spectral Excitation Coding (SEC)

Este modelo busca generar una excitación en el dominio espectral, lo anterior se logra por medio de un banco de osciladores senoidales. El tracto vocal puede ser

modelado de muchas maneras, Cuperman [6] por ejemplo, utiliza el filtro de predicción lineal como modelo del tracto. Un diagrama del decodificador SEC propuesto por Cuperman se presenta en la figura 4. En el codificador la señal de voz es introducida al filtro LPC de análisis para producir una señal residual, de la cual son extraídos los parámetros. Los parámetros requeridos en el decodificador son el período de *pitch*, las ganancias para cada armónico, un factor de dispersión de fase y los coeficientes del filtro de predicción lineal.

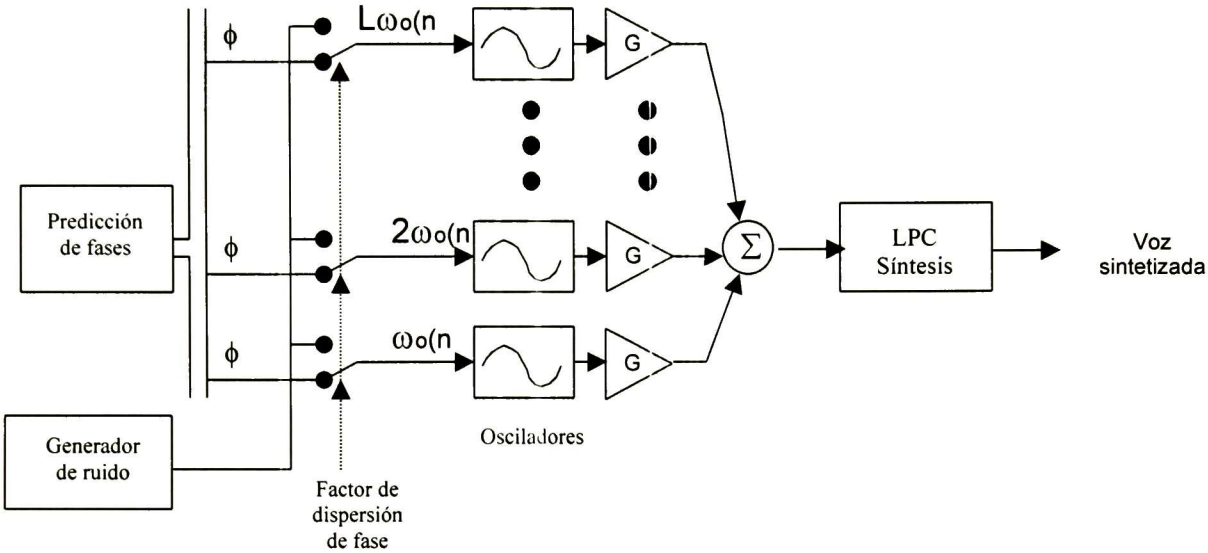

Figura 4. Diagrama a bloques del decodificador SEC.

El factor de dispersión de fase lleva la información de sonoridad. Cuando se quiere reproducir un sonido sonoro de voz, la frecuencia de oscilación de las senoidales corresponde a múltiplos enteros de la frecuencia fundamental (inverso del *pitch*) y las fases se obtienen con ayuda de un predictor de fases. Por otro lado cuando el sonido deseado es no sonoro, las oscilaciones corresponden a múltiplos de un *pitch* preestablecido y las fases se determinan de una manera aleatoria.

Una vez obtenidas estas oscilaciones, son multiplicadas por las ganancias correspondientes a cada armónico para después sumarse generando una señal de excitación, esta señal se introduce al filtro LPC de síntesis para obtener un estimado de la señal de voz.

Mixed Excitation Linear Predictor (MELP)

Este codificador está basado en los codificadores LPC-10E y MBE (ver codificación híbrida), ya que toma el modelo de excitación del MBE (con una variante) y el modelo del tracto del LPC-10E. En relación con el codificador LPC-10E el MELP agrega cinco características con el fin de producir voz de mayor calidad. Una de estas

características es el modelo de excitación, las otras cuatro son la bandera de aperiodicidad, el realce espectral adaptable, el filtro de dispersión de pulso y las magnitudes de Fourier de los primeros 10 armónicos del residual de predicción. Una explicación más detallada de las partes de este codificador junto con un diagrama del mismo es presentada en el siguiente capítulo.

Vocoder de Formantes

Este modelo se basa en encontrar la posición de los formantes en el espectro de potencia de la señal de voz. Los formantes pueden ser modelados como funciones de densidad de probabilidad [7], o como niveles de energía en diferentes bandas de paso. Al ubicar los formantes se puede obtener un sistema que modela el tracto, el cual al ser excitado con una señal adecuada es capaz de producir voz inteligible.

2.5 Codificación híbrida

A pesar de que la codificación paramétrica de la señal de voz tiene una velocidad de transmisión que se considera baja o muy baja, la calidad de la señal reconstruida muchas veces deja mucho que desear. Por otro lado, en la codificación de forma de onda, a pesar de que la calidad de la señal de voz reconstruida es alta, se le asocia una velocidad de transmisión considerada alta. Un método de codificación que intenta tomar los beneficios de ambas técnicas es la codificación híbrida, logrando producir voz de muy buena calidad a razones de bits entre 16 kbps y 4.8 kbps.

En la codificación híbrida, como su nombre lo indica, se hace una mezcla entre la codificación paramétrica y la de forma de onda, es decir, el transmisor se encarga de enviar un conjunto de parámetros y un código de la forma de onda de la señal de voz o de su espectro. La codificación híbrida puede considerarse en el dominio de la frecuencia o del tiempo. El modelo híbrido que sobresale en el dominio de la frecuencia es el *MultiBand Excitation Vocoder* (*MBE*) del cual se presenta una breve explicación más adelante.

De acuerdo con Kondoz [9], el modelo de predicción lineal ha sido predominante en la codificación híbrida de dominio temporal; algunos de los codificadores híbridos más populares de este tipo son el APC, RELP, MPLPC, RPELPC, VSELP y CELP. Los cuatro últimos utilizan una técnica conocida como análisis por síntesis (AbS). En el análisis por síntesis el codificador contiene al decodificador. La síntesis en el codificador utiliza como excitación un grupo de estimados del residual de predicción, eligiendo aquel que minimice el error (típicamente el error cuadrático medio) entre la señal original y la sintetizada. El código del índice del residual elegido es enviado junto con los parámetros necesarios para generar un estimado de la voz en el receptor.

El filtro LPC remueve la correlación entre muestras adyacentes por ello es conocido como un predictor de término corto, sin embargo, el residual que se obtiene al

filtrar la señal de voz con este filtro contiene correlación de término largo asociada al *pitch*. Para eliminar la correlación aun presente en el residual de predicción de término corto se introduce, en la técnica de análisis por síntesis, un filtro predictor de término largo. Al poner en cascada los filtros de términos corto y largo se pretende generar un residual que se asemeje al ruido blanco, es decir, se pretende extraer toda la correlación existente en la señal de voz. Una explicación más profunda de los filtros de predicción de término corto y término largo se presenta en el apéndice A.

2.5.1 Algunos Codificadores híbridos

MultiBand Excitation Vocoder (MBE)

La base de estos sistemas es la excitación que se genera como una combinación de ruido y una señal periódica, el modelo del tracto puede variar. La mayoría de los trabajos que utilizan el modelo de excitación por multibandas están basados en el trabajo realizado por Griffin y Lim [5]. Un diagrama del decodificador de este modelo se presenta en la figura 5.

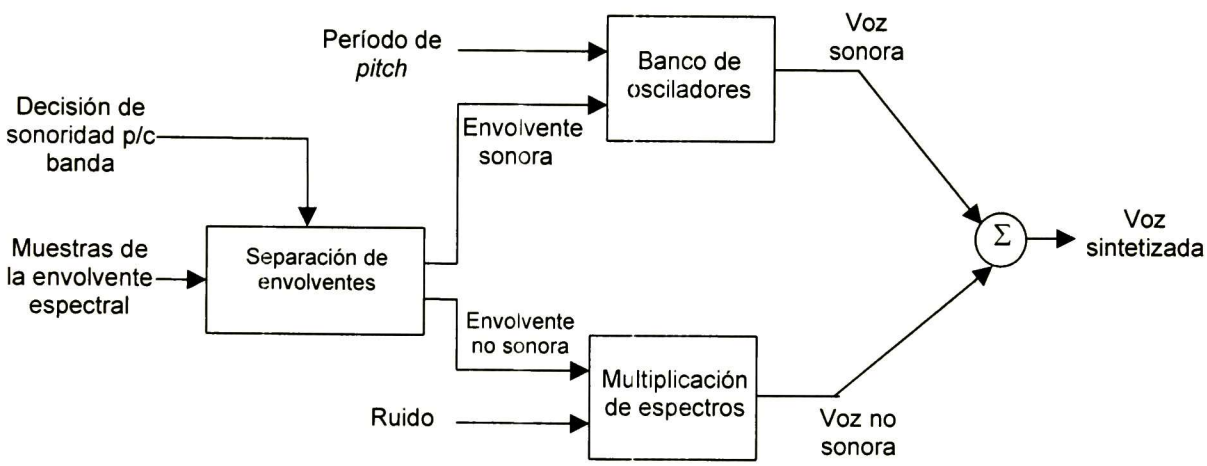

Figura 5. Diagrama a bloques del decodificador MBE.

En el modelo que Griffin propone, el modelo del tracto se envía como muestras del espectro de potencia de la señal de voz, esta envolvente al ser multiplicada por una excitación con espectro plano genera un estimado del espectro de la voz original. La excitación se genera basándose en la información de sonoridad contenida en cada una de las bandas definidas en el análisis, es decir, cuando una banda es declarada como sonora el modelo del espectro de excitación en esa banda es producto de una excitación periódica, de otra manera el espectro de excitación es producto de una excitación ruidosa, al sumar los espectros de excitación de todas las bandas se obtiene

la excitación completa. La decisión de sonoridad es binaria, por lo que una banda de frecuencia únicamente puede ser declarada sonora o no sonora.

Los parámetros que son enviados en este modelo son el *pitch*, las muestras de la envolvente espectral y la información de sonoridad para cada una de las bandas de paso. Se considera a este modelo como un codificador híbrido en el dominio de la frecuencia ya que envía las muestras del espectro de la señal de voz [9].

Residual Excited Linear Predictor (RELP)

En este modelo la señal de voz es filtrada por un predictor de término corto, y se apoya en el hecho de que una porción del residual generado es muy parecida a otra porción del mismo residual, por lo que codificando una porción del residual y enviándola al decodificador, es posible reconstruir un estimado completo del residual.

Antes de enviar la porción del residual, ésta es introducida a un filtro pasa bajas. La salida de este filtro es entonces codificada con alguna técnica de codificación de forma de onda y enviada al decodificador. En el decodificador la porción del residual es reconstruida copiando la porción del residual de banda base a frecuencias superiores. Además de la porción codificada del residual es necesario enviar los parámetros del filtro de predicción lineal. De esta manera se obtiene un modelo de excitación y un modelo del tracto vocal.

Multi-Pulse Linear Predictor Coding (MPLPC)

El primer modelo que utiliza el método de análisis por síntesis es el *Multi-Pulse Linear Predictor Coding*. En este modelo además del predictor de término corto se introduce un predictor de término largo. El residual es modelado como una serie de pulsos (típicamente entre 4 y 6) con diferente amplitud y espaciados de una manera no uniforme.

La posición de los pulsos y las ganancias para cada pulso se determinan por medio de un proceso de análisis por síntesis de manera que, el sintetizador que está contenido en el codificador es excitado por un conjunto de señales constituidas por una serie de pulsos. La señal de pulsos que minimice el error entre la señal original y la sintetizada es elegida como la señal de excitación, y es enviada al decodificador. La posición de los pulsos es determinada de una manera secuencial, es decir, se encuentra la posición óptima para el primer pulso, dejando éste fijo; se busca la posición óptima para el siguiente pulso y así sucesivamente.

Una versión más sencilla de este codificador utiliza pulsos uniformemente espaciados de diferente magnitud. Este codificador es conocido como *Regular Pulse Excited Linear Predictor Coding* (*RPELPC*). Ambos codificadores operan bien a razones de bits alrededor de 9.6 kbps.

Code-Excited Linear Predictor (CELP)

Éste es uno de los codificadores híbridos más populares, un diagrama del modelo se presenta en la figura 6. En este codificador se tiene un libro de excitaciones estocásticas, de las cuales una de ellas es elegida y su índice es enviado al decodificador. La elección al igual que en los otros modelos de análisis por síntesis está basada en la minimización del error entre la señal de voz y la señal sintetizada.

En la síntesis del codificador cada una de las señales del libro estocástico pasa a través de un filtro de síntesis de término largo en cascada con uno de término corto. La señal de excitación a transmitir se elige como la excitación que minimice el error entre la señal de voz y la sintetizada.

Con el fin de dar más importancia al mecanismo del oído humano la señal de referencia (señal de voz) y su estimado pueden ser filtradas por un filtro perceptual [9], cuyo fin es el enmascaramiento del ruido en aquellas regiones de baja energía. El filtro perceptual se aplica únicamente en el codificador, por lo que en el decodificador la señal sintetizada se obtiene tomando directamente la salida del filtro STP Este modelo es capaz de reproducir voz de alta calidad a razones de bits de hasta 4.8 kbps.

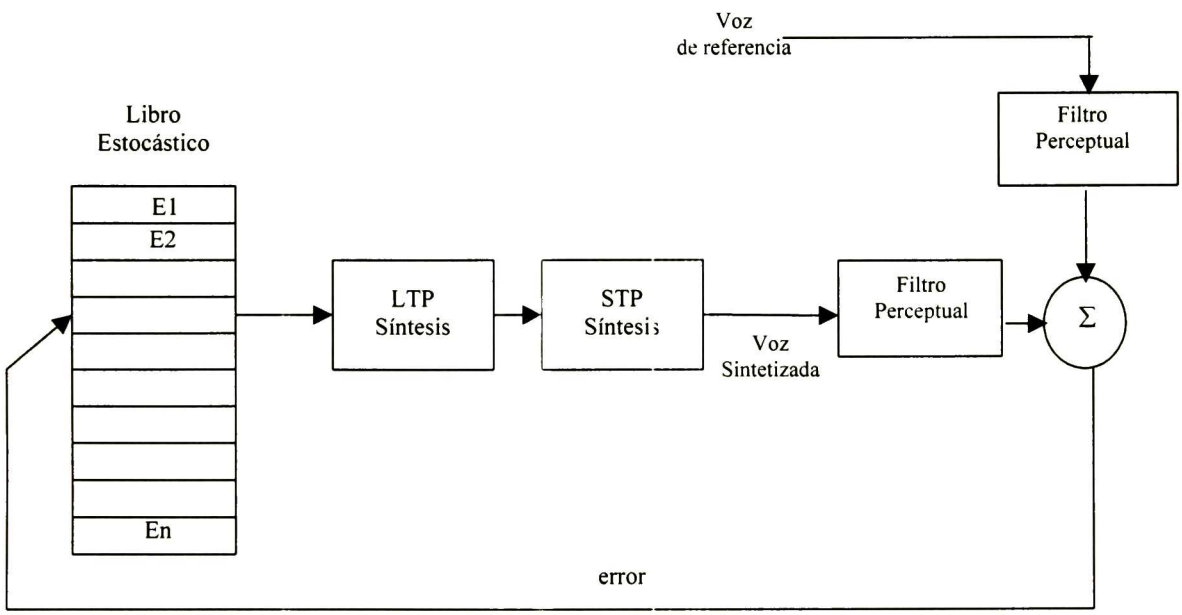

Figura 6. Diagrama a bloques codificador CELP.

Capítulo III. Descripción del MELP a 2400 bps

3.1 Introducción al codificador MELP a 2400 bps

El modelo LPC 10E para codificación de voz a 2400 bps [8], antiguo estándar a esta velocidad, tiene como inconveniente la calidad de voz que produce, la cual además de sonar áspera y con poca naturalidad, contiene distorsiones como golpeteos y zumbidos, por lo que se requiere de personas entrenadas para descifrar el mensaje. Debido a lo anterior y a los nuevos avances en el área de la codificación paramétrica, se decidió sustituir al modelo LPC 10E por otro modelo que además de trabajar a la misma velocidad, producierá voz de buena calidad, eliminará las principales distorsiones del antiguo modelo y suene natural. Muchos modelos fueron propuestos [13], siendo el codificador MELP el elegido para sustituir al codificador LPC 10E como nuevo estándar federal de los E.U.A. para codificación de voz a 2400 bps [1].

El codificador MELP es un nuevo modelo de codificación paramétrica, que utiliza el modelo de predicción lineal y una excitación formada por mezclas de ruido banco y pulsos periódicos. Este codificador puede trabajar a velocidades de transmisión desde 4800 bps hasta 1700 bps [2],[11], produciendo una señal de voz de buena calidad. En relación con el modelo tradicional LPC 10E, el MELP añade 5 nuevas características: Una excitación mixta, las magnitudes de Fourier de los primeros 10 armónicos del residual de predicción, una bandera de aperiodicidad, un filtro de realce espectral adaptable y un filtro de dispersión de pulso. En este capítulo se describen detalladamente los principios en los que se basa el nuevo modelo de codificación y su implementación. La implementación realizada está basada en el "draft" del estándar [12] (ya que la versión definitiva aun no está terminada) y fue realizada en *Matlab*. Cada uno de los procesos del algoritmo hará referencia a su correspondiente función en *Matlab*, misma que está debidamente documentada. En el apéndice B se describen todas las funciones implementadas, sus entradas y salidas.

3.2 Descripción del Algoritmo

Debido a la naturaleza no estacionaria de la señal de voz, los parámetros de ésta son obtenidos por tramas, ya que en intervalos de tiempo de entre 10 a 40 ms la señal de voz sí puede considerarse como un proceso estacionario. En el "draft" se utilizan tramas de duración 22.5 ms, equivalente a 180 muestras de la señal de voz, siendo la frecuencia de muestreo 8000 muestras/segundo. No obstante para otras implementaciones del modelo, la duración de la trama puede variar [11]. Las muestras de voz consideradas en el "draft" están restringidas a valores pico entre –32768 y 32767. mismas que se pueden representar con 16 bits por muestra. Los códigos digitales de las muestras corresponden a un cuantificador uniforme.

Para lograr la velocidad de transmisión de 2400 bps considerando la duración por trama de 22.5 ms, se requiere codificar al conjunto de parámetros en 54 bits por trama, los cuales forman la ***trama codificada***. Los parámetros para cada trama dependen de la información de sonoridad de ésta, esta información de sonoridad es determinada por medio de un análisis multibandas.

Los parámetros enviados en una trama (con información de sonoridad sonora) son el *pitch*, la ganancia, las magnitudes de Fourier de los primeros 10 armónicos del residual de predicción, la decisión de sonoridad para las bandas de paso, los coeficientes LPC's codificados como LSF's, una bandera de aperiodicidad y un bit de sincronía. Cuando la información de sonoridad indique que una trama es no sonora en todas sus bandas de frecuencia, la información de las magnitudes de Fourier, la decisión de sonoridad para las bandas de paso y la bandera de aperiodicidad no es necesaria en el decodificador para la generación de la excitación, en su lugar se introducen códigos de Hamming con el fin de proteger a los parámetros más importantes contra errores de canal.

3.2.1 Codificador

Los parámetros que se requieren en el decodificador, son obtenidos a través de la trama codificada generada en el codificador. La trama codificada contiene los códigos de los parámetros, está formada por 54 bits y es dependiente de la información de sonoridad. En la Tabla 1, se muestra el número de bits asignado para cada uno de los parámetros, al final de esta sección se presenta el orden de su acomodo en la trama codificada. En esta parte se explica el mecanismo de obtención de dichos parámetros y cómo son codificados.

Parámetros	Sonora [bits]	No sonara [bits]
LPC's codificados como LSF's *(LSF$_i$)*	25	25
Magnitudes de Fourier *(CM$_i$)*	8	
Ganancia *(CG$_i$)*	8	8
Pitch (Información de sonoridad) *(P_{q3})*	7	7
Decisiones de sonoridad multibanda *(CV_{bpi})*	4	
Bandera de aperiodicidad *(A_F)*	1	-
Códigos de Hamming *(FEC$_i$)*	-	13
Bit de sincronía *(SINC)*	1	1
Total de bits por trama	**54**	**54**

Tabla 1. Número de bits asignado a cada parámetro dependiendo de la información de sonoridad.

Un diagrama del codificador MELP se presenta en la figura 1, en éste se ven claramente los bloques dedicados a la extracción de cada uno de los parámetros. La señal de entrada en el diagrama corresponde a una versión filtrada de la señal de voz,

el filtro que se aplica es un filtro Chebyshev tipo II pasa-altas de cuarto orden, con frecuencia de corte de 60 Hz y rizo de 30 dB, éste se aplica con el fin de remover cualquier componente de DC que contenga la señal de voz (función *cheb02.m* de *Matlab*).

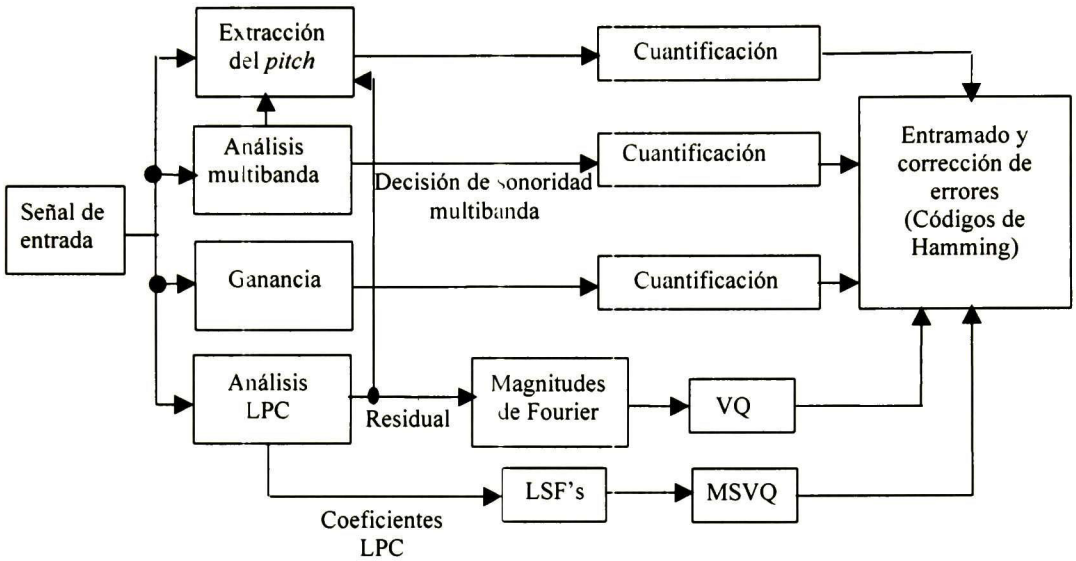

Figura 1. Diagrama del codificador MELP.

Definición del tamaño y la dinámica de la memoria de entrada

Debido a que se realiza un procesamiento por tramas de la señal de voz, es necesario definir una memoria que contenga las muestras más recientes de la señal, a partir de las cuales son extraídos los parámetros. El tamaño y la dinámica de esta memoria están gobernados por la ecuación (ε) (Ver más adelante *Extracción del primer estimado del pitch*), donde S_i es un elemento de la memoria (muestra de la señal de entrada). S_0 se denomina la ultima muestra de la trama actual y sirve como referencia durante la mayor parte del proceso de extracción de parámetros.

$$C_\tau(m,n) = \sum_{k=\lfloor -\tau/2 \rfloor -80}^{\lfloor -\tau/2 \rfloor +79} S_{k+m} S_{k+n} \qquad (\varepsilon)$$

El tamaño de la memoria depende directamente de los valores límite de S_{k+0} y $S_{k+\tau}$ en la ecuación (ε). Los valores de k dependen de la variable τ, la cual como se verá puede tomar valores enteros entre 20 y 160. Tomando el caso cuando τ=20, de la ecuación (ε) se ve claramente que k toma valores desde –90 hasta 69 por lo que: S_{k+0} toma valores desde S_{-90} hasta S_{69} y $S_{k+\tau}$ toma valores desde S_{-70} hasta S_{89}. Si tomamos el caso cuando τ=160, de la ecuación (ε) k toma valores desde -160 hasta -1 por lo que: S_{k+0} toma valores desde S_{-160} hasta S_{-1} y $S_{k+\tau}$ toma valores desde S_0 hasta S_{159}. Si

se hace un análisis para los valores intermedios de la variable τ, se verá que los valores correspondientes de S_i en la memoria, caen dentro de los límites obtenidos para los casos de τ analizados.

Del análisis realizado en el párrafo anterior se observa que los valores de S_i, que deben considerarse en la memoria, corresponden a valores de i desde –160 hasta 159, por lo que se puede definir a la memoria como un arreglo de 320 muestras S_i de la siguiente manera (figura 2):

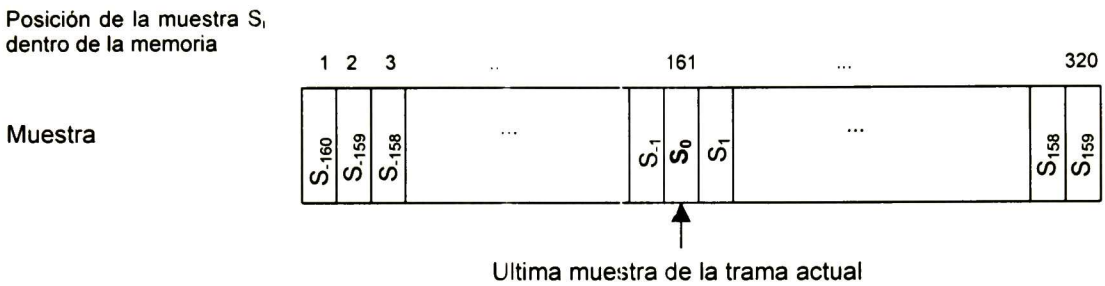

Figura 2. Esquema general de la estructura de la memoria.

Debido a que las tramas consideradas en la implementación son de 180 muestras (25 ms), es posible definir a la memoria en función de las muestras correspondientes a cada trama, donde las muestras S_{-160} hasta S_0 corresponden a 161 muestras de la trama actual y las muestras S_1 hasta S_{159} corresponden a 159 muestras de la trama siguiente. Se puede ver que la memoria se extiende en el pasado y futuro de la última muestra de la trama actual S_0. En la figura 3 se esquematiza a la memoria en función de las muestras de las tramas (M_h donde h = 1 ... 180). Esta memoria se define en el programa *principal.m* de la implementación.

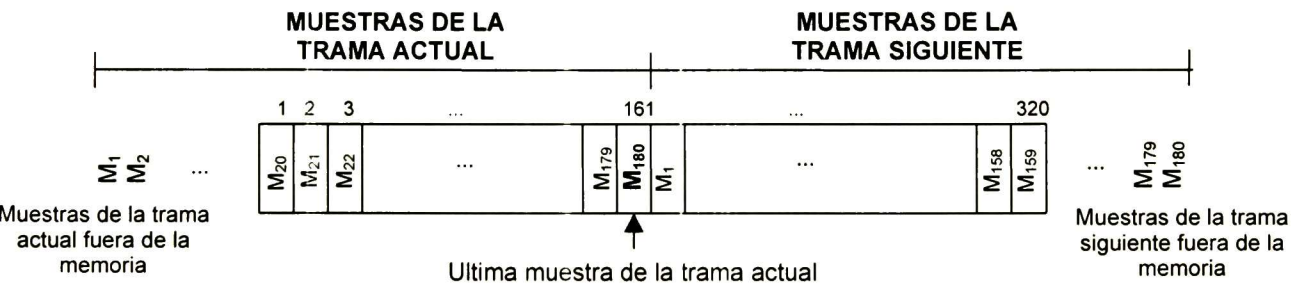

Figura 3. Esquema de la memoria de entrada en función de las muestras de las tramas.

Una vez definida la estructura de la memoria, es posible definir la dinámica de actualización de muestras en la memoria como se muestra en la figura 4, donde j varía

desde 1 hasta el número de tramas. Los parámetros son extraídos por tramas a partir de la memoria como se explica en los siguientes apartados de esta sección.

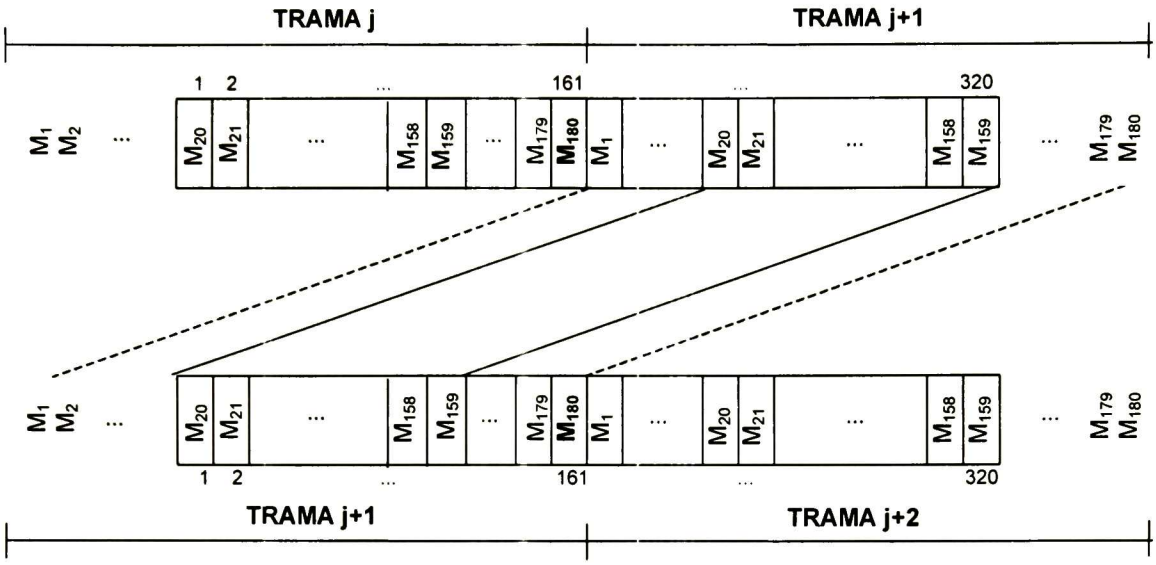

Figura 4. Diagrama de la dinámica de la memoria de entrada.

Extracción del primer estimado del pitch

El ***pitch* se define como el menor tiempo en el que la forma de onda de la señal de entrada (o una versión escalada de la misma) se repite.** Existen muchos métodos para calcular este parámetro, el "draft" se basa en un modelo de similaridad con resolución infinita [14].

El *pitch* es un parámetro que está limitado por la naturaleza de la voz, es decir, el valor del parámetro está limitado a un rango de valores, típicamente de 10 a 180 muestras (entre 1.25 ms a 22.5 ms) con frecuencia de muestreo 8000 muestras/segundo, en el "draft" se restringe el rango, a valores entre 20 y 160 muestras (entre 2.5 ms a 20 ms) ya que valores fuera de este rango son poco probables.

En el método utilizado, primero se obtiene un estimado entero del *pitch* y posteriormente se hace un refinamiento. La referencia [14] está basada en la definición inicial del *pitch* utilizando un modelo de similaridad, para ello, toma dos ventanas consecutivas de la señal de voz de longitud Υ y para diferentes longitudes (Υ) el valor de Υ que maximice una función de correlación entre estas dos ventanas se denomina como el valor entero del *pitch*, figura 5. Debido a que no se realiza un muestreo síncrono con el *pitch*, es difícil que el valor obtenido del Υ óptimo sea el valor exacto del

pitch (puesto que Υ es un valor entero), por ello, se hace un refinamiento utilizando una fórmula de interpolación lineal, la cual se explica más adelante en el proceso de obtención del *pitch* fraccional.

En el "draft", para obtener el valor entero del *pitch*, las muestras de la memoria son primero filtradas por un filtro Butterworth pasa-bajas de sexto orden con frecuencia de corte de 1 KHz. A diferencia del trabajo realizado en [14], las ventanas definidas de la señal de voz tienen una longitud fija de 160 muestras (ecuación (2)). El valor de τ que maximice la función de autocorrelación $r(\tau)$ de la ecuación (1), es declarado como el valor entero del *pitch* P_1(como Υ en [14]). En la figura 6, se esquematizan las ventanas en la memoria para diferentes valores de τ. Cuando se obtiene el primer estimado del *pitch*, el valor de τ se restringe a valores entre 40 y 160 (para la determinación del *pitch* final y los refinamientos, el valor de τ extiende su rango a valores entre 20 y 160). El resultado obtenido en esta parte (P_1) es utilizado posteriormente. Lo anterior se encuentra implementado en la función *pitchentero.m*.

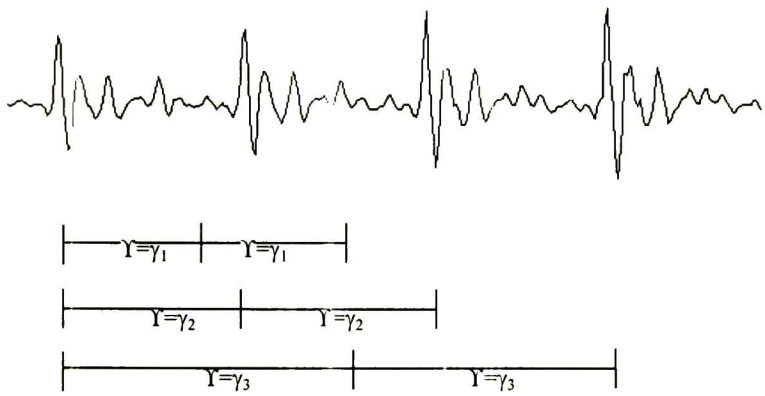

Figura 5. Diagrama de las ventanas de voz para diferentes longitudes (Υ)[14].

$$r(\tau) = \frac{C_\tau(0,\tau)}{\sqrt{C_\tau(0,0)C_\tau(\tau,\tau)}} \qquad (1)$$

Donde

$$C_\tau(m,n) = \sum_{k=\lfloor -\tau/2 \rfloor -80}^{\lfloor -\tau/2 \rfloor +79} S_{k-m} S_{k+n} \qquad (2)$$

Puede verse a partir de la ecuación (1) que m y n en la ecuación (2) únicamente pueden tomar valores 0 ó τ, es por ello, que la figura 6 para definir las ventanas en la memoria, considera el caso $C_\tau(0, \tau)$, puesto que contiene tanto a S_{k+0} como a $S_{k+\tau}$, que son las dos posibles ventanas.

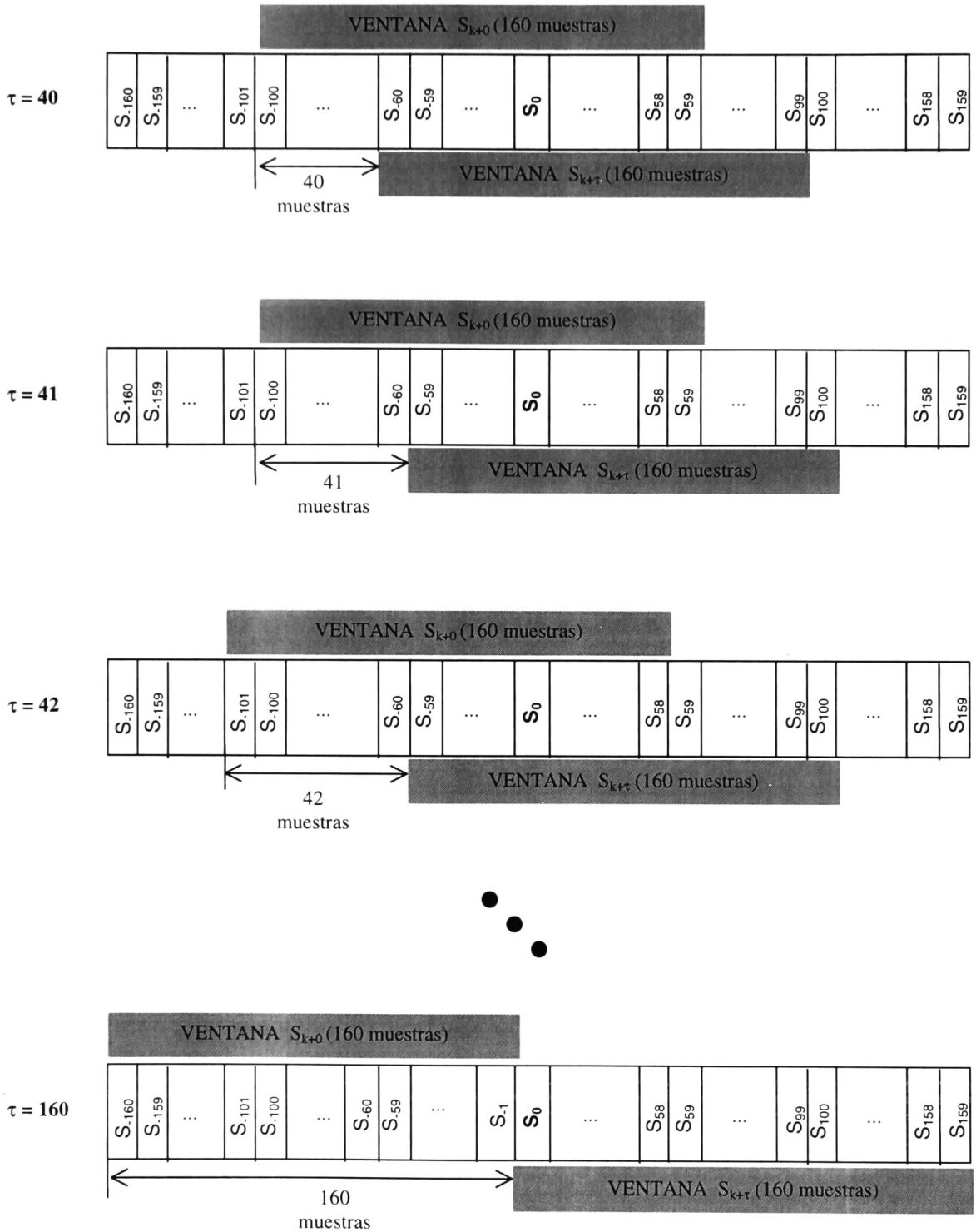

Figura 6. Ventanas para el análisis de extracción del pitch.

Extracción y codificación de la decisión de sonoridad para las bandas de paso y la bandera de aperiodicidad.

La decisión de sonoridad en las bandas de paso se realiza en base a la energía de sonoridad en 5 bandas frecuenciales no uniformemente distribuidas. Las bandas definidas son de 0 a 500 Hz, de 500 a 1000 Hz, de 1000 a 2000 Hz, de 2000 a 3000 Hz y de 3000 a 4000 Hz. Las muestras contenidas en la memoria son filtradas por 5 filtros, correspondientes a cada una de las 5 bandas definidas, los filtros utilizados son Butterworth pasa-bandas de sexto orden. La salida de estos filtros es procesada para obtener la información de la energía de sonoridad en cada una de estas bandas.

Antes de analizar las bandas superiores se analiza la banda de 0 a 500 Hz, a partir de la cual se obtiene un refinamiento del *pitch* entero P_1 (obtenido anteriormente). Además, la información de sonoridad de las bandas superiores depende directamente de la información de sonoridad en la primera banda.

El refinamiento del *pitch*, considera los valores del *pitch* entero de las tramas actual (P_1) y anterior (P_{1a}) como posibles candidatos para obtener el valor refinado del *pitch* (P_2). Para cada uno de estos candidatos (P_1 y P_{1a}) se utiliza la ecuación (1) para buscar un entero óptimo utilizando las muestras de la memoria (S_i) filtradas por el filtro de 0 a 500 Hz. El rango de τ comprende del valor del candidato menos 5 al valor del candidato más 5 (cuando el candidato es $P_1 \Rightarrow P_1 - 5 \leq \tau \leq P_1 + 5$ y cuando el candidato es $P_{1a} \Rightarrow P_{1a} - 5 \leq \tau \leq P_{1a} + 5$), para cada caso se restringen los valores inferior y superior de τ a valores entre 20 y 160 muestras respectivamente. Cuando la trama que se está procesando es la primera, no existe aún un valor para P_{1a}, en este caso se puede utilizar una condicional que indique que se utilizaran 2 candidatos a partir de la segunda trama. Otra manera para resolver el problema es inicializar a la variable P_{1a} con el valor del *pitch* entero actual P_1.

Cada uno de los enteros óptimos (encontrados para los candidatos P_1 y P_{1a} en el análisis anterior) es introducido al proceso del *pitch* fraccional (refinamiento del *pitch*), en el cual se utiliza la señal de salida del filtro de 0 a 500 Hz. Para cada valor de *pitch* introducido, el proceso de extracción del *pitch* fraccional regresa un valor refinado del *pitch* y su correspondiente valor de autocorrelación (obtenidos en base a una fórmula de interpolación lineal). Aquel candidato que genere el mayor valor de autocorrelación es elegido como el segundo estimado de *pitch* P_2 y su valor de autocorrelación se declara como la energía de sonoridad de la primera banda de paso V_{bp1}(función *vbp1.m* de la implementación). El nuevo estimado del *pitch* (P_2) es utilizado para extraer las energías de sonoridad de las bandas restantes y en el cálculo del *pitch* final y de la ganancia.

Para obtener el resto de las energías de sonoridad, la memoria es filtrada por los filtros correspondientes a cada una de las bandas restantes (función *bancobut.m* de *Matlab*), cada una de las salidas es rectificada completamente y pasada por un filtro de

alizamiento. Estas nuevas señales se introducen al proceso del *pitch* fraccional junto con el valor del segundo estimado del *pitch* P_2, el valor de autocorrelación correspondiente a cada una de las bandas es decrementado por 0.1 para compensar por una deformación observada experimentalmente debida al alizamiento de las señales [12], dando como resultado la energía de sonoridad para las bandas restantes (V_{bp2}, V_{bp3}, V_{bp4} y V_{bp5}). El filtro de alizamiento consiste en un cero en DC en cascada con un par de polos (complejos conjugados) a 150 Hz con magnitud 0.97. El filtro de alizamiento está definido en la función *filtroalisa.m* de la implementación. La obtención de las energías de sonoridad para cada una de las bandas superiores está implementada en la función *vbpi.m*.

Cuando la energía de la primera banda de paso es menor o igual a 0.6 ($V_{bp1} \leq 0.6$), la trama de voz que se está analizando se declara completamente no sonora y por tanto el resto de las bandas se declaran no sonoras también. Cuando $V_{bp1} > 0.6$ la primera banda de paso se declara sonora y se analiza el resto de las bandas de la siguiente manera: Si $V_{bpi} \leq 0.6$ la banda i se declara no sonora, de otra manera si, $V_{bpi} > 0.6$ la banda i se declara sonora, donde i = 2 ... 5.

La codificación para la decisión de sonoridad de la primera banda de paso está implícita en la codificación del *pitch* (lo cual se verá más adelante), la codificación de la decisión de sonoridad para las bandas restantes está definida como sigue: Cuando $V_{bp1} \leq 0.6 \Rightarrow CV_{bpi} = 0$ de otra forma, cuando $V_{bp1} > 0.6 \Rightarrow CV_{bpi} = 0 \Leftrightarrow V_{bpi} \leq 0.6 \wedge CV_{bpi} = 1 \Leftrightarrow V_{bpi} > 0.6$, donde CV_{bpi} es el código para la decisión de sonoridad de la banda de paso i (i = 2 ... 5). Existe una excepción, cuando $CV_{bpk} = 0$ (k = 2 ... 4) $\Rightarrow CV_{bp5} = 0$ independientemente del valor de V_{bp5}, lo anterior puede deberse a que la energía de sonoridad de la señal de voz normalmente se encuentra en las bandas frecuenciales bajas, y la última banda únicamente contiene la energía de los armónicos de las bandas inferiores, por lo que si existe energía en esta banda y en las bandas inferiores no, es muy probable que se deba a energía por una señal de ruido. Los valores de CV_{bpi} pueden cambiar en el proceso de obtención del "peakiness" La codificación de las decisiones de sonoridad está implementada en las funciones *qvbpi.m y codevbp.m*, en las cuales ya se considera la información del "peakiness"

El valor de la bandera de aperiodicidad (A_F) depende del valor de la energía de sonoridad de la primera banda de paso. Como se verá en el decodificador, esta bandera sirve para modelar las transiciones entre sonidos sonoros y no sonoros o entre sonidos sonoros. Cuando se cumple que $V_{bp1} < 0.5$ el valor de A_F es 1 de otra manera es 0, de lo anterior puede verse que A_F será 1 en regiones donde la energía de sonoridad de la primera banda de paso es baja.

Proceso de extracción del pitch fraccional

Debido a que no se realiza un muestreo síncrono con el *pitch*, es difícil que el estimado entero del *pitch* sea el valor exacto, por ello se puede decir, que el valor

exacto del *pitch* (T_f) es un valor estimado entero del *pitch* (T) más un error (Δ), ecuación (3):

$$T_f = T + \Delta \qquad (3)$$

donde T siempre es un valor entero y Δ es una fracción. Las entradas a este proceso son un estimado del *pitch* (si no es entero debe de redondearse al entero más próximo) y una señal (muestras de la memoria que pueden estar filtradas o no).

En el método utilizado, se usa una fórmula de interpolación que supone la existencia de un máximo de la función de autocorrelación de la señal de entrada en el intervalo entre T y T+1. En el "draft" para determinar la probabilidad de que el máximo caiga en el intervalo de T-1 a T ó que caiga en el intervalo de T a T+1, se calcula $C_T(0,T-1)$ y $C_T(0,T+1)$ utilizando la ecuación (2). Si $C_T(0,T-1) > C_T(0,T+1)$ entonces es más probable que el máximo de la función de autocorrelación se encuentre en el intervalo de T-1 a T y el valor de T es decrementado en 1 debido a la suposición que hace la fórmula de interpolación de que el máximo está en el intervalo entre T y T+1. Si por el contrario $C_T(0,T+1) > C_T(0,T-1)$ entonces es más probable que el máximo se encuentre en el intervalo de T a T+1 y el valor de T permanece sin cambios ya que satisface la suposición inicial.

El valor de Δ se obtiene en base a una fórmula de interpolación lineal y es el valor que maximiza la función $r(\tau+\Delta)$ para un τ dado, donde las muestras en el instante $\tau+\Delta$ corresponden a valores interpolados de las muestras de la señal. La fórmula de interpolación de Δ [12] que maximiza $r(\tau+\Delta)$ es:

$$\Delta = \frac{C_T(0,T+1)C_T(T,T) - C_1(0,T)C_T(T,T+1)}{C_T(0,T+1)[C_T(T,T) - C_T(T,T+1)] + C_T(0,T)[C_T(T+1,T+1) - C_T(T,T+1)]} \qquad (4)$$

La respectiva función de autocorrelación [12] está dada por:

$$r(T+\Delta) = \frac{(1-\Delta)C_T(0,T) + \Delta C_T(0,T+1)}{\sqrt{C_1(0,0)[(1-\Delta)^2 C_T(T,T) + 2\Delta(1-\Delta)C_T(T,T+1) + \Delta^2 C_T(T+1,T+1)]}} \qquad (5)$$

En la ecuación (3), el valor obtenido para el *pitch* refinado T_f es restringido a valores entre 20 y 160, mientras que de la ecuación (4) debido a que Δ puede tomar valores fuera del rango entre 0 y 1, el valor de Δ es limitado a valores entre –1 y 2. El valor $r(T+\Delta)$ es utilizado para determinar la energía de sonoridad de cada una de las bandas de paso (V_{bpi}, i = 1 ... 5). Las señales utilizadas corresponden a las salidas de los filtros de cada una de las bandas de paso.

Para evitar que pueda haber una indeterminación al aplicar las expresiones (4) y (5) se puede hacer una validación para cuando el denominador de estas expresiones

sea cero. El proceso del refinamiento del *pitch*, es muy utilizado a lo largo del proceso de codificación y está implementado en la función *pitchfrac.m*.

Extracción y codificación de los coeficientes de predicción lineal

Para obtener los coeficientes de predicción lineal correspondientes a cada trama, se aplica una ventana de Hamming de 200 muestras, esta ventana se centra en la última muestra de la trama actual (por lo que se consideran 100 muestras de la trama actual y 100 de la siguiente) figura 7 Los coeficientes son extraídos por el método de autocorrelación utilizando la recursión de Levinson-Durbin, un análisis para un filtro de orden 10 es realizado. De acuerdo con el "draft", un coeficiente de expansión de 0.994 es aplicado a cada uno de los coeficientes obtenidos (a_i, para i = 1 ... 10), por lo que cada coeficiente es multiplicado por 0.994^i (la obtención de los coeficientes a_i se encuentra implementado en la función *lpcmio.m*). Los coeficientes a_i obtenidos son convertidos a coeficientes LSF's (*Line Spectral Frequencies*) ya que son más robustos en la transmisión. El método utilizado para convertir los coeficientes LPC's a LSF´s se basa en el trabajo realizado en [15], en el cual se hace uso de los polinomios Chebyshev. En el apéndice A se explica detalladamente el proceso de extracción de los coeficientes LPC's y su conversión a coeficientes LSF´s.

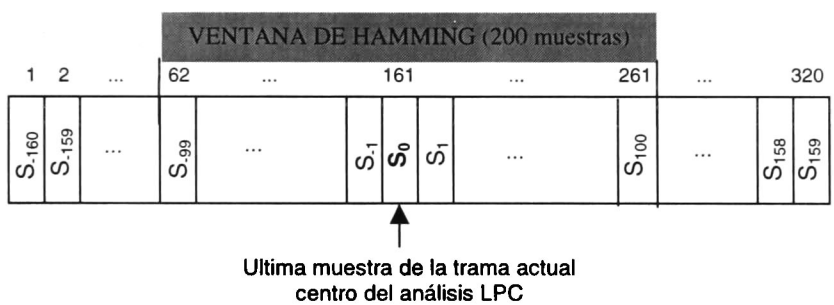

Figura 7. Ventana del análisis de predicción lineal en la memoria.

Una vez que los coeficientes LPC's han sido convertidos a coeficientes LSF's (función *convelsf.m*), éstos se acomodan en orden ascendente asegurando una separación mínima de 50 Hz. En el "draft" se presenta un pseudo código del proceso que asegura la separación mínima entre dos coeficientes LSF's adyacentes, lo cual se encuentra implementado en la función *sepamin.m*. El orden ascendente de los coeficientes LSF's se asegura en la función *convelsf.m*.

Una vez que los coeficientes han sido acomodados de manera ascendente asegurando una separación mínima entre ellos, éstos son cuantificados utilizando una técnica de cuantificación vectorial en multi-etapas (*MSVQ, Multi-Stage Vector Quantization*), lo cual está basado en el trabajo realizado en [16]. En el "draft" para cuantificar los coeficientes LSF's se utilizan 4 etapas, la primera y más importante consta de 128 vectores mientras que las otras 3 etapas contienen 64 vectores cada

una. Los libros de código para las cuatro etapas se encuentran en la función *apendiceC.m*.

En la cuantificación vectorial en multi-etapas se busca encontrar el conjunto de vectores (uno de cada etapa), que al sumarse minimicen la distancia entre los LSF's (f_i, i = 1 ... 10) y el resultado de la suma. En el "draft" se utiliza como distancia la distancia euclidiana pesada (6), d^2 La suma del conjunto de vectores que minimicen la expresión (6) se considera como el valor cuantificado de los coeficientes LSF's (f_{qi}, i = 1 ... 10).

$$d^2 = \sum_{i=1}^{10} w_i (f - f_{ji})^2 \qquad (6)$$

Donde el peso de la distancia (w_i) está definido como:

$$w_i = \begin{cases} P(f_i)^{0.3} & 1 \le i \le 8 \\ 0.64P(f_i)^{0.3} & i = 9 \\ 0.16P(f_i)^{0.3} & i = 10 \end{cases} \qquad (7)$$

Para obtener los valores de $P(f_i)$ de la ecuación (7) es necesario evaluar la potencia del filtro de análisis en la frecuencia f_i. Sea G(z) el filtro de análisis, ecuación (8):

$$G(Z) = 1 + a_1 Z^{-1} + a_2 Z^{-2} + ... + a_{10} Z^{-10} \qquad (8)$$

Haciendo $Z = e^{j\omega}$, la ecuación (8) se expresaría como:

$$G(\omega) = 1 + a_1 e^{-j\omega} + a_2 e^{-2j\omega} + ... + a_{10} e^{-10j\omega} \qquad (9)$$

Si obtenemos un valor de ω para cada valor de f_i con ayuda de la expresión (10), es posible evaluar la expresión (9) para cada uno de los valores de estos valores (ω_i), elevando al cuadrado se obtiene el valor de la potencia del filtro de análisis en la frecuencia f_i ($P(f_i)$), ecuación (11).

$$\omega_i = \frac{2\pi f_i}{8000} \qquad (10)$$

$$P(f) = |G(\omega_i)|^2 \qquad (11)$$

En la cuantificación vectorial en multi-etapas para obtener el conjunto de vectores óptimos (uno para cada etapa), es necesario realizar una búsqueda exhaustiva, es

decir, se requiere la verificación de todas las posibles combinaciones de conjuntos de vectores. Debido a la complejidad que produce el realizar una búsqueda exhaustiva, en la implementación realizada, se utiliza un proceso de búsqueda subóptimo. En este proceso se eligen las ocho mejores trayectorias para cada etapa, es decir, se comienza eligiendo los 8 mejores vectores de la primera etapa (correspondientes a las 8 mejores trayectorias para la primera etapa), para la segunda etapa se toma a cada uno de estos 8 vectores y se buscan las 8 mejores trayectorias (las cuales deben de estar formadas por uno de los 8 mejores vectores de la primera etapa y con un vector de la segunda etapa). En la tercera se toman las 8 mejores trayectorias, antes definidas, y se actualizan tomando en cuenta los vectores de la tercera etapa, lo mismo sucede para la cuarta etapa. A diferencia de una búsqueda exhaustiva, el proceso de búsqueda subóptimo únicamente toma en cuenta las ocho mejores combinaciones (trayectorias) que se van generando, lo que permite obtener un buen estimado de cuantificación con una complejidad mucho menor.

Los índices de los vectores de la mejor trayectoria son codificados y enviados al decodificador. La primera etapa es codificada con 7 bits ya que son 128 vectores (LSF_1), cada una de las otras etapas está codificada en 6 bits debido a que contienen 64 vectores cada una (LSF_2, LSF_3 y LSF_4). El proceso de cuantificación vectorial en multi-etapas está implementado en la función *msvq.m*, por otro lado el proceso que se encarga de obtener las ocho mejores trayectorias para cada etapa se encuentra en la función *mejorocho.m*.

Cálculo del residual de predicción y del "peakiness"

El residual de predicción se utiliza para obtener el valor del "peakiness" y en el proceso de extracción del *pitch* final. El residual de predicción (r_n) se obtiene al filtrar las muestras de la señal de voz (S_i) de la memoria por el filtro de análisis G(z) de la ecuación (8), cuyos coeficientes fueron encontrados en la sección anterior.

El valor del "peakiness" se obtiene como la razón entre las normas L1 y L2 del residual de predicción sobre una ventana de 160 muestras del mismo, ecuación (12). La ventana para el análisis del "peakiness", está centrada en la salida del filtro de análisis que corresponde a la última muestra de la trama actual.

$$peakiness := \frac{L2}{L1} = \frac{\sqrt{\frac{1}{160}\sum_{n=1}^{160} r_n^2}}{\frac{1}{160}\sum_{n=1}^{160} |r_n|} \qquad (12)$$

Los valores de las energías de sonoridad de las 3 primeras bandas de paso pueden variar, dependiendo del valor que tome la variable *peakiness* de la ecuación (12). Si *peakiness* > 1.34 $\Rightarrow$ CV_{bp1} = 1 y si *peakiness* > 1.6 $\Rightarrow$ CV_{bpk} = 1 (k = 1,2 y 3).

La obtención del "peakiness" y del residual de predicción se encuentra en la función *lpcmio.m* de *Matlab*.

Extracción y codificación de la ganancia

El parámetro de ganancia se obtiene dos veces para cada trama, por lo cual se generan dos valores de ganancia, uno para la primera mitad de la trama (G_1) y otro para la segunda mitad (G_2). Los valores de G_1 y G_2 están en dB y son obtenidos a partir de la expresión (13), en la cual el término 0.01 del argumento sirve para evitar que el argumento sea cero o muy próximo a cero. L_v en la ecuación (13) es una ventana de longitud adaptativa (dependiente del segundo estimado del *pitch* P_2 y de la energía de sonoridad en la primera banda de paso V_{bp1}) y S_n son muestras de la memoria.

$$G_i = 10\log_{10}\left[0.01 + \frac{1}{L_v}\sum_{n=1}^{L_v} S_n^2\right] \quad i = 1 \acute{o} 2 \qquad (13)$$

Las muestras S_n de la ecuación (13) que se consideran para el cálculo de la ganancia G_1 no coinciden con las muestras S_n consideradas para el cálculo de G_2 (es decir, se consideran dos ventanas diferentes de la memoria). Lo anterior se debe a que el análisis para el cálculo de la ganancia G_1 está centrado 90 muestras antes de la última muestra de la trama actual, la ventana considerada se centra en la muestra S_{-89} de la memoria. En contra parte, el análisis para el cálculo de G_2 está centrado en la última muestra de la trama actual S_0.

Un pseudo código para obtener la longitud de la ventana L_v para la ecuación (13) se presenta a continuación:

```
Entradas: Vbp1 y P2   // energía de sonoridad de la primera banda de paso y el segundo
                      // estimado del pitch
Salida: Lv            // Longitud de la ventana para la determinación de las ganancias
                      // G1 y G2
P = entero(P2)        // Se obtiene el valor entero más próximo de P2
if Vbp1 > 0.6
  for (k=1,k≤7,k++)   // Búsqueda del menor múltiplo de P2 que sea mayor que 120
    M = P*k           // M es el múltiplo de P2
    if M > 120
      Lv = M          // El valor de M que sea el menor múltiplo de P2 > 120 declara
      break           // como la longitud de la ventana Lv
    end if
  end for
  if Lv > 320         // Si el valor de Lv excede 320 se declara una nueva ventana
    Lv = entero(Lv /2)// con longitud el entero más próximo a Lv/2
  end if
```

else
  L_v = 120 // Longitud de la ventana cuando $V_{bp1} \leq 0.6$
end if // (valor predeterminado)

Debido a que el centro del análisis para G_1 se centra en la muestra S_{-89} y a que la longitud de la ventana L_v puede tomar valores entre 120 y 320, existen casos (de L_v) en los que las muestras S_n para el análisis de G_1 de la ecuación (13) caen fuera de la memoria definida inicialmente, por ello es necesario definir una nueva memoria que contenga todas las muestras necesarias para realizar el análisis correspondiente a G_1. En base a un análisis realizado se determinó que la memoria requerida para el cálculo de la ganancia G_1 debe contener muestras de las tramas anterior, actual y siguiente. Esta memoria se esquematiza en la figura 8, en función de las muestras de las tramas (M_h, h = 1 ... 180).

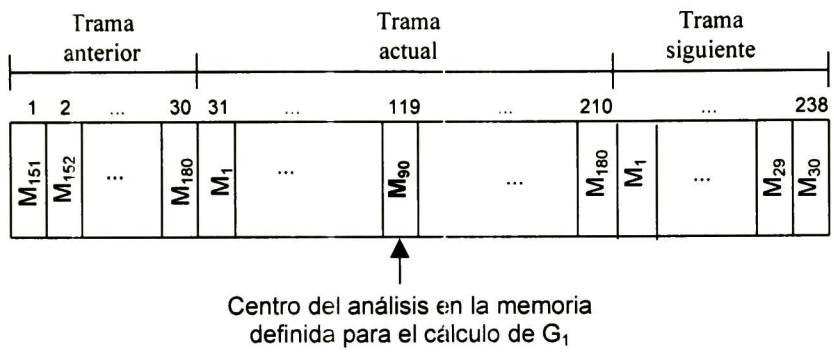

Figura 8. Esquema de la memoria para el cálculo de G_1.

La memoria para el cálculo de G_1 se define en el programa *principal.m*, junto con la memoria general antes definida. La obtención de los valores G_1 y G_2 está implementada en la función *ganancia.m*.

Una vez obtenidos los valores para G_1 y G_2 éstos son codificados en 8 bits. La ganancia G_2 se codifica con un codificador uniforme de 32 niveles en 5 bits (CG_2) (utilizando la función *cuantizador.m*), los límites inferior y superior del codificador son 10 y 77 dB respectivamente. La ganancia G_1 se codifica en 3 bits (CG_1) utilizando un algoritmo adaptable cuyo pseudo código se encuentra en el "draft" [12]. En el algoritmo de codificación de G_1 se considera a la ganancia G_2 de las tramas actual (G_2) y anterior (G_{2a}). Cuando $|G_2 - G_{2a}| < 5$ dB y $|G1-(G_2 - G_{2a})/2| < 3$ dB, entonces se considera a la trama en estado estable y se utiliza un código especial (los tres bits en cero) para indicar este estado en el decodificador. Si no se cumple lo anterior G_1 es codificada utilizando un cuantificador uniforme cuyos límites superior e inferior dependen de los valores de G_2 y G_{2a} [12], limitándose a valores entre 10 y 77 dB. La codificación de los parámetros de ganancia se realiza en la función *qgana.m*.

Obtención y codificación del pitch final

La señal residual antes obtenida es filtrada por un filtro Butterworth de sexto orden pasa-bajas con frecuencia de corte de 1000 Hz, a partir de esta señal se hace una búsqueda de un *pitch* entero utilizando la ecuación (1). El rango de τ considerado, comprende del valor entero más próximo a P_2 menos 5 al valor entero más próximo a P_2 más 5. Una vez encontrado este valor entero, se realiza un refinamiento del *pitch* utilizando el proceso del *pitch* fraccional, los valores introducidos son el valor del *pitch* encontrado y la señal residual, el resultado obtenido produce un valor tentativo del *pitch* final P_3, el cual puede cambiar en un proceso que verifica que el valor de *pitch* obtenido no sea un múltiplo del valor de *pitch* real (función *doblchk.m*) y cuyo pseudo código se presenta en el "draft" [12].

En el "draft" se presenta un pseudo código del algoritmo para calcular el *pitch* final [12]. De igual manera se define una función dependiente del valor de autocorrelación $r(P_3)$ y de la ganancia G_2, que calcula un promedio de los valores finales del *pitch* de las ultimas tres tramas (función *pavg.m* de *Matlab*), por lo que debe de existir una memoria que contenga los 3 valores más recientes del *pitch* final P_{3aaa}, P_{3aa} y P_{3a}, donde P_{3aaa} y P_{3aa} son los valores del *pitch* final de tres tramas y dos tramas atrás respectivamente y P_{3a} es el valor final del *pitch* de la trama anterior.

El valor promedio del *pitch* se calcula antes de realizar el cálculo del *pitch* final y se utiliza como entrada a dicho proceso. Debido a que en las primeras tramas no existe un valor definido para P_{3aaa}, P_{3aa} y P_{3a}, estos valores se pueden inicializar con el valor del segundo estimado del *pitch* para la trama actual (P_2) y el correspondiente valor de autocorrelación $r(P_{3a})$ puede inicializarse con el valor $r(P_2)$ (lo anterior se puede observar en el programa *principal.m* de la implementación). La obtención del valor del *pitch* final se encuentra implementado en la función *pitchfinal.m.*

En la codificación del *pitch* se codifica también la información de la energía de sonoridad para la primera banda de paso, ya que se asigna un código especial de puros ceros cuando $V_{bp1} \leq 0.6$, de otra manera el *pitch* es codificado utilizando un cuantificador uniforme (función *cuantizador.m*) de 99 niveles cuyos límites inferior y superior son 20 y 160 respectivamente. Se utilizan 7 bits para la codificación del *pitch* (P_{q3}), lo que genera 128 palabras de código, 99 de estas palabras se utilizan para la codificación de los niveles del cuantificador (aquellas palabras que contengan más de dos 1's), una de las palabras corresponde al código especial para cuando $V_{bp1} \leq 0.6$ y las 28 palabras del código restantes corresponden a aquellas palabras que contengan uno o dos valores en 1 dentro de la palabra de código, estas palabras se utilizan para la detección y posible corrección de errores de canal. La asignación de los códigos se presenta en la referencia [12], la codificación del *pitch* se encuentra implementada en la función *qpitch.m* de *Matlab*.

Cálculo y codificación de las magnitudes de Fourier de los primeros 10 armónicos del residual de predicción

La función de las magnitudes de Fourier de los primeros 10 armónicos del residual de predicción, es tratar de reconstruir una versión del residual en el decodificador. Se extraen las magnitudes de los armónicos del residual ya que corresponden a picos de energía en el espectro y por tanto contienen más información que fuera de los armónicos.

El residual de predicción, a partir del cual se extraen los armónicos, corresponde a la salida de un filtro de análisis, cuando su entrada son las muestras de la memoria inicialmente definida. Los coeficientes a_i del filtro de análisis son obtenidos a partir del valor cuantificado de los LSF's. Una vez obtenido el residual de predicción (320 muestras de la memoria), éste es multiplicado por una ventana de Hamming de 200 muestras centrada en la última muestra de la trama actual. Una transformada rápida de Fourier (*FFT*)de 512 puntos es aplicada a estas 200 muestras "ventaneadas" del residual (el cual es rellenado con ceros para completar las 512 muestras). Lo anterior se hace con el fin de obtener un espectro de mayor resolución. Debido a que el resultado de la *FFT* consiste de valores complejos, es necesario obtener sus magnitudes a partir de las cuales son extraídas las magnitudes de los primeros 10 armónicos del residual de predicción.

Para extraer estas magnitudes, el "draft" utiliza un algoritmo de extracción de picos, el cual estima la posición de cada armónico y obtiene el valor máximo dentro de una ventana centrada en dicho estimado. La estimación de la posición del armónico se hace en función del valor cuantificado del *pitch* final P_{q3}, el ancho de la ventana de búsqueda es $512/P_{q3}$. Los estimados de las posiciones de cada armónico se obtienen a partir de la ecuación (14).

$$P_i = \frac{512i}{P_{q3}} \quad i = 1...10 \qquad (14)$$

Donde P_i indica la posición estimada para cada uno de los armónicos.

El número de armónicos (NA) está limitado al menor valor entre 10 y $P_{q3}/4$. Una vez obtenidas las magnitudes para los armónicos (M_f), éstas son normalizadas para tener un valor RMS de 1, ecuaciones (15) y (16).

$$RMS = \sqrt{\frac{1}{NA}(M_f)^2} \qquad (15)$$

$$M_{fn} = \frac{M_f}{NA} \qquad (16)$$

Donde M_f en (15) y (16) es un vector de longitud NA que contiene las magnitudes de Fourier y M_{fn} corresponde al vector de las magnitudes de Fourier normalizadas. Si NA < 10 el vector M_{fn} es rellenado con unos hasta que contenga 10 elementos.

Las magnitudes de Fourier M_{fn} son cuantificadas utilizando cuantificación vectorial. El libro de códigos utilizado contiene 256 vectores (función *apendiceD.m*)y se elige aquel que minimice la distancia euclidiana pesada entre el vector sin cuantificar (M_{fn}) y el vector de cuantificación (M_{fq}), ecuación (17).

$$d^2 = \sum_{i=1}^{10} w_i (M_{fn_i} - M_{fq_i})^2 \qquad (17)$$

Donde i = 1 ... 10 y el peso w_i está dado por la ecuación (18) [12]. El w_i definido enfatiza las bajas frecuencias (perceptualmente más importantes) sobre las altas frecuencias.

$$w_i := \left[\frac{117}{25 + 75\left[1 + 1.4\left(\frac{f_i}{1000}\right)^2\right]^{0.69}} \right]^2 \quad i = 1 \ldots 10 \qquad (18)$$

f_i en la ecuación (18) está dado por 8000i/60 [Hz] y corresponde al i-*ésimo* armónico para un *pitch* predeterminado de 60 muestras. Una vez encontrado el vector óptimo, el índice (CM_f) correspondiente es codificado en 8 bits para ser enviado al decodificador. La obtención y codificación de las magnitudes de Fourier de los primeros 10 armónicos se realiza en la función *fourier1.m*.

Protección contra errores de canal

Cuando la decisión de sonoridad de la primera banda de paso indique que la trama es completamente no sonora ($V_{bp1} \leq 0.6$), las magnitudes de Fourier, la bandera de aperiodicidad y la decisión de sonoridad para las bandas restantes no son necesarias en el decodificador, puesto que se utilizan valores predeterminados. Una opción para aprovechar esta circunstancia, es definir al MELP como un codificador con velocidad de transmisión variable, sin embargo, este tipo de codificación tiene muchas desventajas (y por ello no es muy utilizado). Otra opción es utilizar los bits del código de los parámetros que no se necesitan para proteger a los parámetros más importantes contra errores de canal, obteniendo por consiguiente un codificador con velocidad de transmisión fija, más robusto al ruido.

En el MELP implementado son 13 los bits de los parámetros que no se utilizan cuando ($V_{bp1} \leq 0.6$), dando lugar a que se puedan introducir un código de Hamming$_{(8,4)}$ y tres códigos de Hamming$_{(7,4)}$. El código (8,4) protege a los 4 bits más significativos del índice de la primera etapa para la codificación de los coeficientes LSF's, los 3 bits restantes de este índice se protegen con uno de los códigos (7,4), los cuatro bits mas significativos de la ganancia G_2 se protegen con otro de los códigos (7,4) y finalmente el bit restante de la ganancia G_2 y los tres bits de la ganancia G_1 se protegen con el último código (7,4). Las matrices generadoras se definen en el "draft" y se presentan en las expresiones (19) y (20) (para los códigos (7,4) y (8,4) respectivamente).

$$G_{(7,4)} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \qquad (19)$$

$$G_{(8,4)} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \qquad (20)$$

Para obtener los bits de paridad correspondientes a la protección de errores se multiplica el vector de los parámetros por la matiz generadora dando como resultado un vector con los bits de paridad, ecuación (22). Los vectores de bits de los parámetros que se van a proteger son de 4 elementos y se definen a continuación en la ecuación (21).

$$\begin{aligned}
&\text{Para código Hamming}_{(8,4)} \quad v_1 = [LSF_1(7) \quad LSF_1(6) \quad LSF_1(5) \quad LSF_1(4)] \\
&\text{Para código Hamming}_{(7,4)} \quad v_2 = [LSF_1(3) \quad LSF_1(2) \quad LSF_1(1) \quad 0] \\
&\text{Para código Hamming}_{(7,4)} \quad v_3 = [G_2(5) \quad G_2(4) \quad G_2(3) \quad G_2(2)] \\
&\text{Para código Hamming}_{(7,4)} \quad v_4 = [G_2(1) \quad G_1(3) \quad G_1(2) \quad G_1(1)] \qquad (21)
\end{aligned}$$

Donde el valor entre paréntesis en los parámetros representa el número de bit, siendo 1 el bit menos significativo.

$$\begin{aligned}
FEC_1 &= v_1 G_{(8,4)}^T = [FEC_1(1) \quad FEC_1(2) \quad FEC_1(3) \quad FEC_1(4)] \\
FEC_2 &= v_2 G_{(7,4)}^T = [FEC_2(1) \quad FEC_2(2) \quad FEC_2(3)] \\
FEC_3 &= v_3 G_{(7,4)}^T = [FEC_3(1) \quad FEC_3(2) \quad FEC_3(3)] \\
FEC_4 &= v_4 G_{(7,4)}^T = [FEC_4(1) \quad FEC_4(2) \quad FEC_4(3)] \qquad (22)
\end{aligned}$$

La posición en la trama codificada de cada uno de los bits de paridad de la expresión (22), se presenta en la Tabla 2. Referencias que contiene información acerca de los códigos de Hamming son [17][18]. Lo anterior se encuentra implementado en la función *fec.m* (FEC: *Forward Error Correction*).

Entramado de los bits (generación de la trama codificada)

Una vez que los códigos de los parámetros han sido obtenidos, éstos son acomodados de acuerdo a la Tabla 2, para formar la trama codificada. La trama se codifica de manera diferente, dependiendo de la información de sonoridad de la primera banda de paso. En el MELP no se realiza una distinción de sonoridad para la trama, la distinción de sonoridad se realiza en 5 bandas de frecuencia, por simplicidad se denominará como una trama sonora a aquella trama que cumple $V_{bp1} > 0.6$, por otro lado una trama se denomina como no sonora cuando no cumple la condición anterior. El ordenamiento y empaquetado de los códigos de los parámetros en la trama codificada se encuentra en la función *cadebit.m*.

Bit	*Sonora*	*No sonora*	*Bit*	*Sonora*	*No sonora*	*Bit*	*Sonora*	*No sonora*
1	$CG_2(1)$	$CG_2(1)$	**19**	$LSF_1(7)$	$LSF_1(7)$	**37**	$CG_1(1)$	$CG_1(1)$
2	CV_{bp2}	$FEC_1(1)$	**20**	$LSF_4(6)$	$LSF_4(6)$	**38**	CV_{bp4}	$FEC_1(3)$
3	$P_{q3}(1)$	$P_{q3}(1)$	**21**	$P_{q3}(4)$	$P_{q3}(4)$	**39**	CV_{bp3}	$FEC_1(2)$
4	$LSF_2(1)$	$LSF_2(1)$	**22**	$LSF_1(6)$	$LSF_1(6)$	**40**	$LSF_2(2)$	$LSF_2(2)$
5	$LSF_3(1)$	$LSF_3(1)$	**23**	$LSF_1(5)$	$LSF_1(5)$	**41**	$LSF_3(4)$	$LSF_3(4)$
6	$CG_2(4)$	$CG_2(4)$	**24**	$LSF_2(6)$	$LSF_2(6)$	**42**	$LSF_2(3)$	$LSF_2(3)$
7	$CG_2(5)$	$CG_2(5)$	**25**	CV_{bp5}	$FEC_1(4)$	**43**	$LSF_3(3)$	$LSF_3(3)$
8	$LSF_3(6)$	$LSF_3(6)$	**26**	$LSF_1(4)$	$LSF_1(4)$	**44**	$LSF_3(2)$	$LSF_3(2)$
9	$CG_2(2)$	$CG_2(2)$	**27**	$LSF_1(3)$	$LSF_1(3)$	**45**	$LSF_4(4)$	$LSF_4(4)$
10	$CG_2(3)$	$CG_2(3)$	**28**	$LSF_2(5)$	$LSF_2(5)$	**46**	$LSF_4(3)$	$LSF_4(3)$
11	$P_{q3}(5)$	$P_{q3}(5)$	**29**	$LSF_4(5)$	$LSF_4(5)$	**47**	A_F	$FEC_4(3)$
12	$LSF_3(5)$	$LSF_3(5)$	**30**	$CM_f(1)$	$FEC_4(1)$	**48**	$LSF_4(2)$	$LSF_4(2)$
13	$P_{q3}(6)$	$P_{q3}(6)$	**31**	$LSF_1(2)$	$LSF_1(2)$	**49**	$CM_f(5)$	$FEC_3(3)$
14	$P_{q3}(2)$	$P_{q3}(2)$	**32**	$LSF_2(4)$	$LSF_2(4)$	**50**	$CM_f(4)$	$FEC_3(2)$
15	$P_{q3}(3)$	$P_{q3}(3)$	**33**	$CM_f(8)$	$FEC_2(3)$	**51**	$CM_f(3)$	$FEC_3(1)$
16	$LSF_4(1)$	$LSF_4(1)$	**34**	$CM_f(7)$	$FEC_2(2)$	**52**	$CM_f(2)$	$FEC_4(2)$
17	$P_{q3}(7)$	$P_{q3}(7)$	**35**	$CM_f(6)$	$FEC_2(1)$	**53**	$CG_1(3)$	$CG_1(3)$
18	$LSF_1(1)$	$LSF_1(1)$	**36**	$CG_1(2)$	$CG_1(2)$	**54**	SINC	SINC

CG_i (i = 1 ó 2): Códigos de las ganancias G_1 (3 bits) y G_2 (5 bits).
CM_f : Código del índice para las magnitudes de Fourier (8 bits, solo se trasmiten si $V_{bp1}>0.6$).
P_{q3} : Código del pitch (7 bits).
A_F : Código de la bandera de aperiodicidad (1 bit, solo se trasmite si $V_{bp1}>0.6$).
SINC : Bit de sincronía (1 bit).
LSF_i (i = 1, ..., 4) : Códigos de los índices de la cuantificación vectorial en multi-etapas, LSF_1 (7 bits), LSF_2 (6 bits), LSF_3 (6 bits) y LSF_4 (6 bits).
CV_{bpi} (i = 2, ..., 5) : Códigos de la decisión de sonoridad para las 4 bandas superiores de frecuencia, se utiliza un bit para cada CV_{bpi} (4 bits en total, solo se trasmiten si $V_{bp1}>0.6$).
FEC_i (i = 1, ..., 4) : Códigos de Hamming para protección contra errores de canal, FEC_1 código $Hamming_{(8,4)}$ (4 bits), FEC_2, FEC_3 y FEC_4 códigos $Hamming_{(7,4)}$ (3 bits c/u, solo se trasmiten si $V_{bp1}\leq 0.6$).
En la tabla el valor entre paréntesis indica el número de bit, donde 1 es el bit menos significativo

Tabla 2. Orden de los bits en la trama codificada de los códigos de los parámetros.

3.2.2 Decodificador

En el decodificador es donde se pueden apreciar las cinco nuevas características en referencia al modelo tradicional LPC 10E (Excitación mixta, bandera de aperiodicidad, magnitudes de Fourier, filtro de realce espectral adaptable y filtro de dispersión de pulso). Un diagrama del decodificador MELP propuesto en el "draft" [12], se presenta en la figura 9. El nuevo modelo (MELP) supone que el principal defecto del antiguo modelo (LPC 10E) es la excitación que se introduce en el filtro de síntesis, ya que, como se dijo en el capítulo II, se realiza una discriminación de la trama de voz en sonora y no sonora, modelando la excitación sonora como un simple tren de pulsos y la no-sonora como ruido blanco. Las principales distorsiones (golpeteos y zumbidos) de la voz sintetizada en el modelo tradicional se atribuyen al modelo de excitación [2].

Estudios realizados [2][5], indican que una trama de voz contiene mezclas de pulsos y ruido, y proponen un modelo de excitación con mezclas de pulsos y ruido. Esta mezcla se realiza en base a decisiones de sonoridad en el dominio de la frecuencia. La excitación mixta es la característica más importante del nuevo modelo, ya que elimina las principales distorsiones del modelo tradicional [2][19], lo cual como se mencionó es el principal defecto del modelo tradicional.

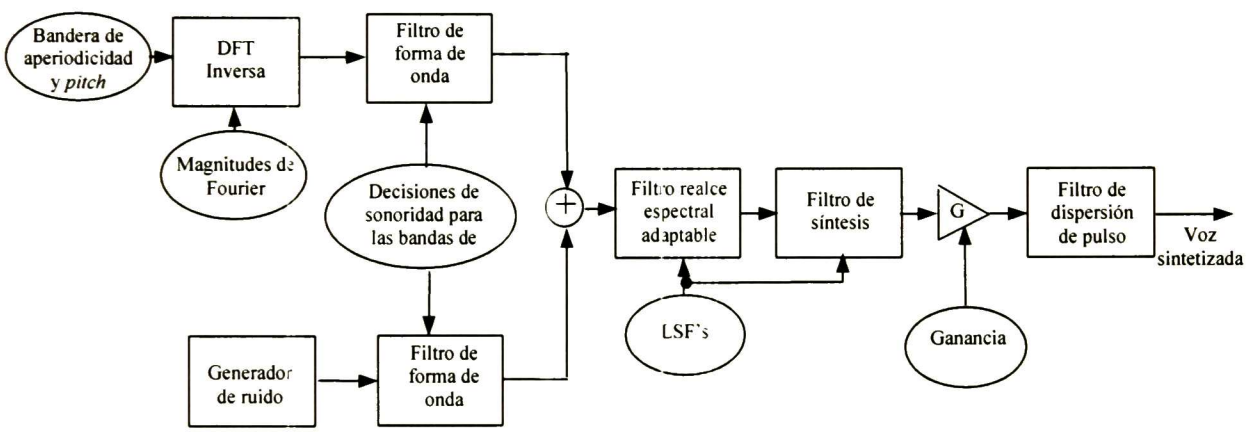

Figura 9. Diagrama del decodificador MELP

Para generar la excitación mixta se utilizan dos filtros de forma de onda (para las excitaciones periódica y ruidosa (figura 9)). Los filtros utilizados en [12] son filtros digitales de respuesta al impulso finita (*FIR*) de orden 31 cuyos coeficientes dependen de la información de la decisión de sonoridad de las bandas de paso. El filtro de forma de onda para la excitación periódica permite que la excitación periódica afecte únicamente a bandas de frecuencia dominadas por energía periódica, mientras que el filtro de forma de onda para el ruido realiza lo mismo pero aplicado a la excitación ruidosa. Estos dos filtros se complementan para formar un espectro plano de excitación,

las salidas de estos filtros son sumadas y forman la excitación mixta. En la figura 10 se presentan los espectros de potencia de la respuesta al impulso de cada filtro cuando las decisiones de sonoridad indican que las bandas 1, 2 y 5 contienen energía periódica.

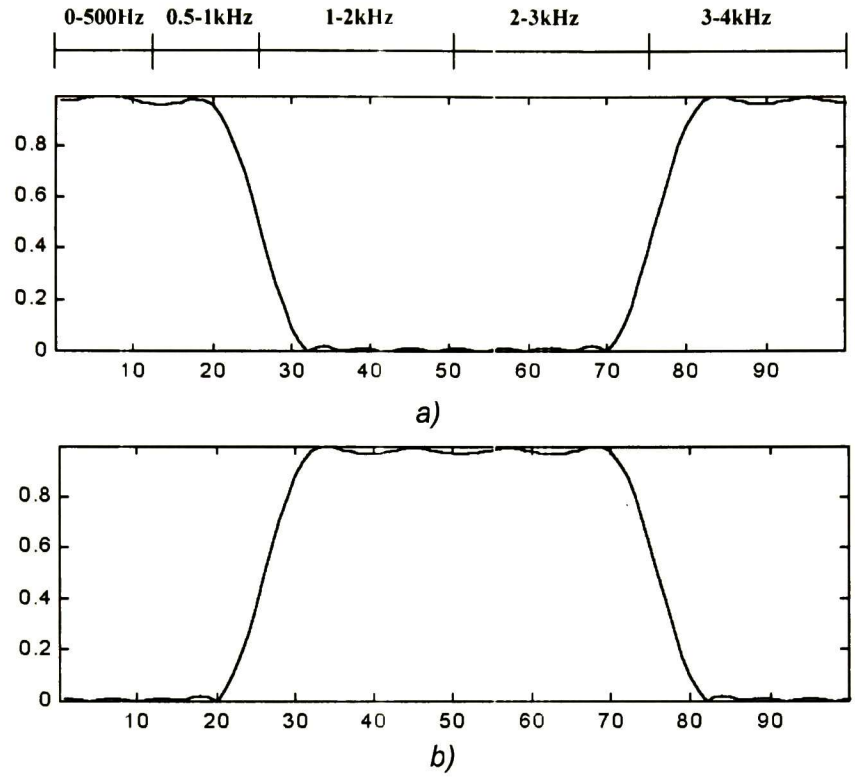

Figura 10. Gráficas de los espectros de potencia de la respuesta al impulso de los filtros de forma de onda, cuando las decisiones de sonoridad indican que las bandas 1,2 y 5 contienen energía periódica a)Filtro de forma de onda para excitación periódica y b) Filtro de forma de onda para excitación de ruido

En las regiones de transición entre sonidos sonoros y no sonoros (o entre sonidos sonoros), no se puede decir que existe una periodicidad en la señal, pero tampoco se puede decir que existe una señal dominada por ruido. Con el fin de modelar estas regiones, el MELP considera una bandera de aperiodicidad, que normalemnte está activa en este tipo de regiones. Esta bandera tiene como función indicar que se debe variar aleatoriamente la posición de los pulsos (dentro de un rango) de la señal periódica, modelando de esta manera el comporamiento de la voz en este tipo de regiones.

En el nuevo modelo se utiliza la técnica de predicción lineal para modelar el tracto vocal, debido a que los polos del filtro de síntesis coinciden en gran medida con la

posición de los formantes de la señal de voz. Sin embargo, el filtro de síntesis por si solo, no es capaz de reproducir las caracteristicas de las resonancias de los formantes, por ello se utiliza un filtro de realce espectral adaptable cuya función es ayudar al filtro de síntesis en estas regiones ayudando a producir una señal de voz que suene con mayor naturalidad [2]. El filtro de realce espectral consiste en un filtro de respuesta al impulso infinita (*IIR*) de polos y ceros, cuyos coeficientes se obtienen a partir de los coeficientes de predicción lineal, ecuación (31).

Si al filtro de síntesis se le introduce el residual de predicción generado en el análisis, se obtendría a la salida la señal de voz original, pero, debido a que no es posible mandar el residual de predicción (manteniendo una velocidad de 2400 bps), las magnitudes de Fourier de los primeros 10 armónicos del residual de predicción son enviadas con el fin de generar un estimado del residual en el decodificador. Estas magnitudes únicamente son enviadas cuando la primera banda de paso contiene energía periódica.

Finalmente, la última característica en el modelo, es un filtro de dispersión de pulso, cuya función es ayudar a las regiones lejanas a los formantes, distibuyendo de mejor manera la energía de la señal en el dominio del tiempo. Lo anterior se ve reflejado en la forma de onda de la señal de voz (en el dominio del tiempo). El filtro de dispersión de pulso es un filtro de respuesta finita al impulso de orden 65 y se encuentra implementado en la función *pulsedisp.m* de *Matlab*. El programa que engloba al proceso de decodificación se llama *decodificador.m*. El resto de está sección explica detalladamente la implementación del decodificador.

Desentramado de la trama codificada

Los códigos de los parámetros son extraídos por tramas, a partir de la trama codificada, con ayuda de la Tabla 2. El primer parámetro en ser extraído es el *pitch* ya que éste, contiene la información de sonoridad de la primera banda de paso y el significado de algunos bits de la trama codificada dependen directamente de esta información.

Cuando el código del *pitch* corresponde al codigo especial (puros ceros) indica que $V_{bp1} \leq 0.6$, por lo tanto los códigos Hamming son extraídos para detectar y corregir errores de canal en los parametros protegidos. Los códigos de $\text{Hamming}_{(7,4)}$ son capaces de detectar y corregir un error mientras que el código de $\text{Hamming}_{(8,4)}$ puede detectar errores dobles corrigiendo uno. El funcionamiento de detección y corrección se explica en [17] y está implementado en la función *fecinve.m*. Si existen errores no corregibles se declara a la trama como insegura, en el caso contrario el resto de los parámetros es extraído de la trama codificada.

Si el código del *pitch* no corresponde al código especial antes mencionado, se analiza la palabra de código correspondiente al *pitch*, para verificar posibles errores de canal. Cuando la palabra de código contiene un solo uno, se declara que hubo un error

en el canal y se considera a la trama como no sonora en la primera banda de paso y por tanto se siguen los pasos considerados cuando el valor extraído, es el código especial (puros ceros). Si la palabra de código del *pitch* extraído contiene únicamente dos unos (existencia de un error de canal), se declara a la trama como insegura. Finalmente si la palabra de código del *pitch* no cae en ninguna de las categorias anteriores el resto de los parámetros son extraídos.

Si no se declaró trama insegura, los códigos de los parámetros son extraídos a partir de la trama codificada (función *cadehamm.m* de la implementación), después se obtienen los parámetros a partir de los códigos (el valor del *pitch* se extrae en la función *decopitch.m* de la implementación). Cuando una trama se declara insegura se aplica un mecanismo de repetición de trama en el cual los parámetros considerados para la decodificación corresponden a los parámetros de la trama anterior.

Cuando el código del *pitch* indique que la trama es completamente no sonora, se utilizan valores predeterminados para el *pitch*, las magnitudes de Fourier de los primeros 10 armónicos del residual de predicción, la bandera de aperiodicidad y la decisión de sonoridad de las bandas de paso, estos valores son:
El pitch se pone en 50
Las 10 magnitudes de Fourier (M_l) se ponen en 1
El valor de la decisión de sonoridad (CV_{bpi}) se pone en 0 ($i = 2,...,4$)
La bandera de aperiodicidad (A_F) se pone en 1

Debido a que en el proceso de síntesis de voz, se realiza una interpolación de parámetros (ver más adelante *Proceso de interpolación de parámetros*) entre los valores de las tramas actual y previa, es necesario realizar una inicialización de los parámetros de la trama anterior cuando se comienza el proceso de decodificación, ya que en la primera trama no se dipone de estos valores. La inicialización se realiza en el programa *decodificador.m*.

Extracción de la ganancia

Por la manera en cómo se codificó a los parámetros de ganancia G_1 y G_2 es posible detectar y corregir algunos errores en G_2. La ganancia G_2 es decodificada a partir del índice del nivel de cuantificación, para lo cual se considera un cuantificador uniforme de 32 niveles con límites inferior y superior de 10 y 77 dB respectivamente. Para obtener el valor de G_1 se necesita considerar a la ganancia G_2 de las tramas actual (G_2) y previa (G_{2a}), cuando la palabra de código corresponde al código especial (puros ceros) la ganancia G_1 es el promedio de las ganancias G_2 y G_{2a}. Si el código recibido no es el código especial, la ganancia G_1 se decodifica a partir del índice del nivel de cuantificación, considerando un cuantificador uniforme de 7 niveles con límites inferior y superior dependiantes de los valores de G_2 y G_{2a} y limitados a un rango de entre 10 y 77 dB respectivamente. Un pseudo código del proceso de decodificación se encuentra en [12] y su programación en *gdeco.m*.

Atenuación de ruido

Con el fin de reducir la calidad ruidosa en las regiones de silencio, éstas son detectadas y un factor de atenuación de ruido (en dB) es aplicado a las ganacias G_1 y G_2. Para ello se define un estimado de ruido de fondo G_n el cual se actualiza de acuerdo a la expresión (23). El valor de G_n se utiliza posteriormente para calcular una señal de probabilidad, necesaria para la obtención del filtro de realce espectral adaptable.

$$G_n = \begin{cases} G_n + C_{up} & G_i > G_n - C_{up} \\ G_n - C_{down} & G_i < G_n - C_{down} \\ G_i & De\ otra\ manera \end{cases} \qquad i = 1\ ó\ 2 \qquad (23)$$

En la ecuación (23), C_{up} := 0.0337435 y C_{down} := 0.135418. El valor de G_n se limita a valores entre 10 y 80, al principio del proceso este valor es inicializado con el valor de la ganancia G_1. El factor de atenuación aplicado a las ganacias G_1 y G_2, depende del estimado de ruido de fondo G_n y del propio valor de la ganancia G_i (i = 1 ó 2) y está definido como:

$$G_{att} = -10\log_{10}(1 - 10^{0.1[G_n + 3 - G_i]}) \qquad (24)$$

Debido a que en la implemantación el valor de G_{att}, ocasionalmente producía valores imaginarios, se validó el argumento del logaritmo de la siguiente manera: Si $0.1[Gn+3-Gi] \geq 0 \Rightarrow$ se sustituye este valor por 0.0001. El valor de G_n que se utiliza en la ecuación (24), se limita a un máximo de 20 dB y el factor de atenuación (G_{att}) aplicado a las ganancias G_i se limita a 6 dB como máximo. El valor utilizado de las ganancias, corresponde al valor atenuado G_{iate} (i = 1 ó 2) de la expresión (25). Por simplicidad se les seguirá llamando G_1 y G_2.

$$G_{iate} = G_i - G_{att} \qquad (25)$$

Interpolación de Parámetros y actualización del apuntador de inicio de ventana de síntesis (t_c)

La síntesis de voz en el decodificador se realiza síncronamente con el *pitch*, es decir, se sintetizan señales de la longitud del *pitch* y se van concatenado para formar la señal de voz sintetizada completa. Debido a lo anterior es necesario definir un apuntador, que indique el desplazamiento de la nueva ventana de síntesis (correspondiente al nuevo período de *pitch*) dentro de la trama. Como una trama está formada por 180 muestras, se define al apuntador como una variable (t_0) que toma valores en el rango entre 0 y 179, donde el valor de 0 indica que la nueva ventana a sintetizar no contiene ningun desplazamiento a partir del inicio de la trama y el valor de

179 indica que el inicio de la nueva ventana de síntesis está desplazado al final de la trama.

Cuando el valor de t_0 es mayor que 180 (el tamaño de la trama) indica el paso a una nueva trama y el valor de t_0 debe de ser actualizado para trabajar con esta nueva trama. El proceso de actualizacion de t_0 es el siguiente: Sea *L* la longitud de la ventana que ha sido sintetizada y sea t_0 la variable que indica el inicio de la nueva ventana de síntesis dentro de la trama *k*. Si $L + t_0 < 180 \Rightarrow t_0 = t_0 + L$, lo que indica que la nueva ventana de síntesis inicia dentro de la trama *k*, por el contrario si $L + t_0 \geq 180 \Rightarrow t_0 = (t_0 + L)$ 180 y $k = k+1$, lo que indica que la nueva ventana de síntesis inicia en la trama $k+1$ con t_0 muestras de desplazamiento.

Todos los parámetros son linealmante interpolados para cada ventana que se vaya a sintetizar. Los parámetros son: los coeficientes LSF's, las ganancias, el *pitch*, la bandera de aperiodicidad, los coeficientes de los filtros de forma de onda, las magnitudes de Fourier, y un coeficiente μ utilizado para definir el filtro de realce espectral adaptable. El procedimiento de interpolación de parámetros es el siguiente, sea *X* un parámetro cualquiera, el valor interpolado del parámatro *X* es:

$$X_{\text{int}} = \text{int}^* X + (1 - \text{int})^* X_a \qquad (26)$$

Donde *int* es un factor de interpolación, X_a es el valor del parámetro *X* en la trama anterior, y X_{int} es el valor interpolado del parámetro *X*. Debido a que la ganancia es calculada dos veces por trama, su interpolación depende del valor de t_0, es decir, si $t_0 < 90 \Rightarrow X = G_1$ y $X_a = G_{2a}$ (ganancia G_2 de la trama anterior) y si $t_0 > 90 \Rightarrow X = G_2$ y $X_a = G_1$, donde *X* y X_a corresponden a los valores de la ecuación (26). En general el factor de interpolación se define como, ecuación (27):

$$\text{int} = \frac{t_0}{180} \qquad (27)$$

Existen dos excepciones en el proceso de interpolación, la primera excepción es cuando la frecuencia fundamental de la señal periódica aumenta drásticamente (donde la frecuencia fundamental es el inverso del valor de *pitch*), lo cual ocurre cuando la ganancia G_1 es más grande por más de 6 dB que la ganancia G_{2a} y el valor del *pitch* de la trama actual es menor que la mitad del *pitch* de la trama anterior. Cuando esto sucede el *pitch* no se interpola, tomando el nuevo valor (el *pitch* de la trama actual). La segunada excepción corresponde a un cambio drástico en los valores de ganancia, ésto ocurre cuando la ganancia G_2 de la trama actual difiere por más de 6 dB de la ganancia G_2 de la trama anterior (G_{2a}). Cuando esto pasa, el factor de interpolación para los coeficientes LSF's, el coeficiente μ y el *pitch*, está dado por la ecuación (28). Lo anterior se hace debido a que el parámetro de ganancia se obtiene dos veces por trama (G_1 y G_2), y por tanto sigue una mejor trayectoria de interpolación.

$$\text{int} = \frac{G_{\text{int}} - G_{2a}}{G_2 - G_{2a}} \qquad (28)$$

El valor G_{int} en la ecuación (28) es el valor interpolado de la ganancia; para obtenerlo se utiliza el factor de interpolación definido en la expresión (27). G_{2a} es el valor de la ganancia G_2 de la trama anterior. La interpolación de los parametros se realiza en la función *interpola.m* de *Matlab*.

Definición de la longitud de la ventana de síntesis

La longitud de la ventana de síntesis (L_{vs}) depende del valor del *pitch* interpolado y del valor de la bandera de aperiodicidad interpolada. Si la bandera de aperiodicidad interpolada es cero, entonces la longitud de la ventana de síntesis será el entero más próximo al *pitch* interpolado. Por otro lado si la bandera de aperiodicidad interpolada es uno, la longitud de la ventana será el entero más próximo del valor del *pitch* interpolado más una variable aleatoria, la cual hace variar el valor del *pitch* dentro de un rango del 25% de su valor.

Generación de la excitación mixta

La excitación mixta consiste en la suma de una excitación periódica y una excitación de ruido. Para una ventana de síntesis el número de elementos de cada excitación corresponde a la longitud de la ventana de síntesis (L_{vs}) definida anteriormente.

La excitación periódica en la ventana de síntesis $e_p(n)$, n = 0, 1, ..., L_{vs}–1, consiste en un pulso obtenido a través de una transformada inversa de Fourier discreta de longitud L_{vs} (ecuación (29)), en la cual se introduce la información de las magnitudes de Fourier, dentro del vector M(k), k=0, ..., L_{vs}–1 como se muetra en la expresión (30). Las magnitudes de Fourier son extraídas a partir del índice CM_f cuando se trata de una trama sonora, de otra manera como se dijo al inicio, se utiliza un vector predeterminado formado con puros unos. La excitación mixta se obtiene en la función *exciperi.m*.

$$e_p(n) = \frac{1}{L_{vs}} \sum_{k=0}^{L_{vs}-1} M(k) e^{j 2\pi n k / L_{vs}} \qquad (29)$$

Donde

$$M(k) = [0 \quad M_{f1} \cdots M_{f10} \quad 1 \quad 1 \quad \ldots \quad 1 \quad 1 \quad M_{f10} \cdots M_{f1}] \qquad (30)$$

La exitación de ruidio se obtiene se obtiene a partir de un generador de números aleatorios entre –1732 y 1732.

Una vez que se obtienen las excitaciones, éstas son filtradas por el filtro de forma de onda correspondiente (filtros FIR de orden 31 cuyos coeficientes se generan con la función *shaping.m*, la cual hace uso de la función *apendiceA.m*). Las salidas de estos filtros son sumadas para generar la excitación mixta.

Filtro de realce espectral Adaptable

La excitación mixta es filtrada por un filtro de realce espectral adaptable, este filtro, como se dijo, sirve para ayudar al filtro de síntesis a modelar las resonancias de los formantes. El filtro de realce espectral adaptable se define en la ecuación (31). La salida de este filtro se introduce al filtro de síntesis.

$$H_{ase}(Z) = \frac{G(\alpha Z^{-1})}{G(\beta Z^{-1})}(1 + \mu Z^{-1}) \qquad (31)$$

$G(\alpha Z^{-1})$ y $G(\beta Z^{-1})$ en la ecuación (31) corresponden a G(Z) de la ecuación (8) evaluada en αZ^{-1} y βZ^{-1} respectivamente. μ, β y α están definidos como:

$$\mu = (\max(0.5k_1, 0))p$$
$$\beta = 0.8p$$
$$\alpha = 0.5p \qquad (32)$$

Donde k1 es el primer coeficiente de reflexión, calculado a partir de los coeficientes LSF's (función *reflecoef.m*), y p es una señal de probabilidad definida en el "draft", ecuación (33). El valor de la probabilidad p se limita a valores entre 0 y 1.

$$p = \frac{G_{\text{int}} - G_n - 12}{18} \qquad (33)$$

Síntesis de la señal de voz

La señal que sale del filtro de realce espectral adaptable es filtrada por un filtro todo polos, cuyos coeficientes corresponden a los coeficientes de predicción lineal obtenidos a partir de los coeficientes LSF's interpolados para esa ventana de síntesis. La salida de este filtro es escalada por el factor de ganancia de la ecuación (34) y posteriormente es filtrada por el filtro de dispersión de pulso (*pulsedisp*.m de *Matlab*). El filtro de realce espectral, el de síntesis y la aplicación del factor de escalamiento se llevan a cabo en el programa *decodificador.m*.

$$S_{gain} = \frac{10^{G_{int}/20}}{\sqrt{\frac{1}{L_{vs}}\sum_{n=1}^{L_{vs}} \hat{S}_n^2}} \qquad (34)$$

S_n de la expresión (34) corresponde a la salida del filtro de síntesis.

Capítulo IV. Evaluación del Algoritmo

4.1 Introducción

En los capítulos anteriores se ha hablado mucho de la calidad de voz que se obtiene con las diferentes técnicas de codificación. Sin embargo, el utilizar frases como "*buena calidad*" "*calidad pobre*" etc. puede ser muy ambiguo, por ello es necesario definir pruebas para medir, de una manera más estándar, las características cualitativas de la señal de voz producida por un codificador determinado.

La señal de voz tiene al ser humano como receptor final, es por ello que, un codificador de voz debe buscar que la voz producida sea por lo menos inteligible para el humano, de otra manera no tendría utilidad alguna. La señal de voz tiene como objetivo transmitir información al ser humano, podría decirse, que el "mensaje" es la parte más importante de la información, pero además del "mensaje" la señal de voz transmite información sobre el estado de ánimo de la persona, de su identidad, de su sexo, de su edad, etc.

Cuando en un codificador es difícil determinar la información que no está contenida en el "mensaje", pero éste ha sido extraído completamente, entonces se puede afirmar que el codificador produce voz inteligible, pero no se puede afirmar que produce voz de buena calidad, entendiendo a la calidad como un conjunto de cualidades de la voz y siendo la inteligibilidad una de estas cualidades. Debido a lo anterior, un codificador de voz puede producir voz inteligible pero de una calidad no buena, sin embargo, no es posible obtener un codificador de buena calidad con una inteligibilidad pobre. Existen pruebas enfocadas a verificar la extracción del "mensaje" (sin importar el resto de la información contenida en la voz), estas pruebas son conocidas como pruebas de inteligibilidad. Cuando una prueba engloba otros aspectos de la señal de voz, se le conoce como prueba de calidad.

Es posible definir un conjunto de pruebas basadas en las opiniones y percepciones del receptor final (el ser humano), este tipo de pruebas son aplicadas a un conjunto de personas las cuales califican, de acuerdo a lo escuchado, ciertos aspectos de la voz reproducida. A este tipo de pruebas se les conoce como pruebas subjetivas, ya que dependen de las opiniones y percepciones de personas, las cuales no siempre son iguales. Una desventaja de las pruebas subjetivas es que los resultados pueden depender de las indicaciones proporcionadas a las personas evaluadoras. Para evitar situaciones como la anterior, se han propuesto una gran cantidad de pruebas basadas en un tratamiento matemático, en estas pruebas se realiza una manipulación matemática de las señales de voz original y sintetizada (voz reproducida por el decodificador), con el fin de realizar una comparación objetiva entre ellas, a este tipo de pruebas se les conoce como pruebas objetivas.

Para evaluar la implementación del MELP a 2400 bps, se realizaron pruebas objetivas y subjetivas. El algoritmo se probó bajo condiciones de canal ideal y ruidoso. Los canales que se utilizaron fueron un canal binario simétrico y un canal Gaussiano, para el primer caso la probabilidad de error en la prueba fue de 0.006 y para el segundo caso se utilizó ruido Gaussiano con desviación estándar de 0.21. Los errores de canal fueron introducidos a la trama codificada con ayuda de las funciones *bschan* y *gausschan*.

Las pruebas subjetivas que se realizaron fueron la DRT y MOS (por sus siglas en inglés *"Diagnostic Rhyme Test" y "Mean Opinion Score"* respectivamente). La prueba DRT mide la inteligibilidad de la voz producida en el decodificador, mientras que la prueba MOS mide la calidad de la misma. La prueba objetiva implementada mide la distancia entre los espectros de la señales de voz original y sintetizada. La descripción detallada de las pruebas se presenta en el resto del capítulo junto con los resultados obtenidos. Un panorama de los métodos de evaluación para codificadores de voz puede encontrarse en [20].

4.2 Pruebas subjetivas

Las pruebas subjetivas se basan en la opinión y percepción de un grupo de personas. En esta trabajo se utilizan las voces de 6 hablantes (3 mujeres y 3 hombres), las cuales son procesadas por la implementación en *Matlab* del MELP, ocho personas evalúan la inteligibilidad y calidad, de las palabras y frases escuchadas. Las frases y palabras utilizadas en la prueba están fonéticamente balanceadas. En primera parte de la prueba subjetiva, se hace una prueba DRT para evaluar la inteligibilidad, después en la segunda parte se evalúa la calidad con una prueba MOS.

4.2.1 Prueba DRT *(Diagnostic Rhyme Test)*

Los sonidos sonoros que se pueden producir en el idioma Español, corresponden a tres tipos de apertura del tracto vocal, en el tipo 1 (sonido, "*a*") el tracto tiene una apertura semi-cerrada, en el tipo 2 (sonidos, "*e*" y "*o*") se tiene una apertura semi-abierta y finalmente en el tipo 3 (sonidos, "*u*" e "*i*") se tiene una apertura completa del tracto [23], por lo que al realizar una prueba de rima, es necesario mantener un equilibrio entre las diferentes aperturas del tracto.

En la prueba DRT se utilizan 60 pares de palabras, cada par de palabras rima y difiere únicamente por un fonema, en total se tienen 20 pares de palabras para cada tipo de apertura (ninguno de ellos es repetido). Cada uno de los hablantes se encarga de pronunciar 10 pares de palabras (4 palabras corresponden a un tipo de apertura, 3 a otro y las otras 3 palabras al otro tipo). Los pares procesados son reproducidos y 8 personas (los evaluadores) se encargan de escribir las palabras escuchadas.

La calificación DRT es el porcentaje de palabras entendidas, lo cual se obtiene a partir de la ecuación (1). Los resultados de esta prueba, junto con los resultados de las otras pruebas, se presentan al final del capítulo.

$$DRT(\%) = \frac{100(N_T - N_E)}{N_I} \qquad (1)$$

En la expresión (1), N_T es el número total de palabras y N_E es el número de errores.

4.2.2 Prueba MOS *(Mean Opinion Score)*

La prueba MOS se utiliza para medir la calidad de la voz que produce el codificador. Para ello se utilizan 48 palabras fonéticamente balanceadas (8 palabras por cada hablante), 12 frases (2 frases por hablante) y una poesía que contienen todos los fonemas del idioma Español por lo menos una vez. Las frases utilizadas están fonéticamente balanceadas en conjunto, es decir, se busca que aparezcan todos los fonemas entre todas las frases. La prueba MOS califica la voz reproducida por el decodificador, de acuerdo a la escala siguiente:

Escala	Calidad de Voz
1	Insatisfactoria
2	Pobre
3	Regular
4	Buena
5	Excelente

En la prueba MOS implementada además de calificar la calidad de voz, las frases y palabras escuchadas debían ser escritas. Los resultados son presentados al final del capítulo.

4.3 Prueba objetiva

Como se mencionó anteriormente, las pruebas objetivas buscan realizar una comparación matemática entre las señales original y sintetizada. Una medida objetiva buena, debe coincidir de alguna manera con las pruebas subjetivas, ya que las pruebas subjetivas están basadas en las opiniones del receptor final.

Algunas pruebas objetivas se basan en obtener la distancia entre las formas de onda de las señales original y reconstruida, sin embargo, este tipo de pruebas no son útiles cuando se habla de codificadores paramétricos, ya que en este tipo de codificadores no se busca reconstruir la forma de onda de la señal de voz sino sus características espectrales. Debido a lo anterior se definió una medida que considera la

distancia entre las muestras del espectro de las señales original y reconstruida, donde la distancia considerada es el error cuadrático medio.

El proceso para obtener la medida objetiva es el siguiente: Se aplica una ventana de Hamming de 200 muestras tanto a la señal de voz original como a la reconstruida, las señales "ventaneadas" son rellenadas con ceros hasta obtener dos vectores de 512 muestras cada uno, una transformada rápida de Fourier (*FFT*) de 512 muestras es aplicada a cada vector para obtener las muestras del espectro de las señales original y sintetizada, finalmente se utiliza la ecuación (2) para obtener el error cuadrático medio entre los dos espectros.

$$err_j = \frac{1}{512}\sum_{k=1}^{512}(U(k) - U_s(k))^2 \qquad (2)$$

U y U_s en la expresión (2) corresponden a las muestras espectrales, de las señales original y reconstruida respectivamente. El error err_j es el error cuadrático medio para una ventana de 200 muestras de voz. Para obtener el error total *E* es necesario promediar los errores err_j de todas las ventanas. La expresión para el error total *E* en decibeles se presenta en la ecuación (3), donde N_v es el número total de ventanas de 200 muestras, de la señal de voz original.

$$E = 10\log_{10}\left(\frac{1}{N_v}\sum_{j=1}^{N_v} err_j\right) \quad [dB] \qquad (3)$$

Mientras más negativo sea el valor de *E* en la expresión (3) las señales original y sintetizada serán espectralmente más parecidas, lo que indica que el error es más cercano a cero.

Los resultados obtenidos se presentan el final del capítulo. La medida objetiva está implementada en la función *objetiva.m* de *Matlab*.

4.4 Resultados de las pruebas

Los resultados de las pruebas objetivas y subjetivas para condiciones de canales ideal y ruidoso, se presentan en las Tablas 1 y 2. En estas tablas puede observarse que los resultados de las pruebas objetivas coinciden en gran medida con los resultados de las pruebas subjetivas, además se puede ver que existe una degradación perceptual de la señal sintetizada cuando se trabaja bajo condiciones de canal ruidoso. Sin embargo, los resultados obtenidos en las pruebas MOS indican un índice relativamente alto de calidad, lo anterior también se ve reflejado en los resultados expuestos en la Tabla 2.

Condición	Prueba DRT	Prueba MOS	Prueba Objetiva
Canal ideal	72.8 %	3.59	-11.75 dB
Canal Gaussiano	62.1 %	3.35	-10.47 dB
Canal binario simétrico	60.4 %	3.46	-10.88 dB

Tabla 1. Resultados de las pruebas objetivas y subjetivas.

Los resultados de la Tabla 1 son reportados en [21]. Otras pruebas (DAM por sus siglas en Inglés *"Diagnostic Acceptability Measure"* DRT y MOS) han sido aplicadas al nuevo estándar [19],[24], dichas pruebas se realizaron en idioma Inglés.

De acuerdo a los trabajos realizados en [19] y [24], los resultados de las pruebas MOS obtenidas en este trabajo concuerdan con los reportados obtenidos en dichas referencias. Sin embargo, los resultados de las pruebas DRT obtenidos aquí difieren por 19 puntos en relación con los reportados en [24], esto puede deberse al método utilizado en este trabajo, para realizar las pruebas DRT.

La manera tradicional para efectuar las pruebas DRT sugiere una simple comparación entre un par de palabras que riman entre sí y difieren por solo una consonante, en una hoja de respuestas se encuentran las opciones, de las cuales únicamente hay que seleccionar una [20]. La desventaja del método anterior es que los evaluadores pueden intentar adivinar una palabra no entendida. En la prueba DRT aplicada en este trabajo, se le pedía a los evaluadores que escribieran la palabra entendida y no se les brindaba ninguna opción.

Los índices de inteligibilidad aumentan cuando en vez de palabras aisladas se utilizan frases. De las frases utilizadas en la prueba MOS se obtuvieron los resultados de la Tabla 2. La razón del aumento de inteligibilidad, se debe a que, cuando se emplean frases el evaluador tiene un contexto, lo que le permite obtener información adicional de una palabra que no entienda.

Condición	Porcentaje de Inteligibilidad
Canal ideal	95.1 %
Canal Gaussiano	85.0 %
Canal binario simétrico	90.2 %

Tabla 2. Porcentaje de inteligibilidad cuando se utilizan frases.

La calidad y el desempeño del codificador MELP bajo condiciones de canales ruidoso e ideal puede ser comparado con otras técnicas de codificación a diferente velocidad utilizando los resultados presentes en [22],[9].

4.5 Comparación con un codificador MELP en código *"C"*

Con el fin de verificar que la implementación en *MATLAB* cumpliera con los requerimientos del estándar, se utilizó un código desarrollado por *Texas Instruments* para verificar la interoparabilidad.

Se implementaron dos funciones en *MATLAB* que permiten introducir una trama de parámetros generada por el codificador de *MATLAB* en el decodificador del código en *"C"* y viceversa. La voz sintetizada generada al introducir una trama codificada en el decodificador del otro lenguaje (ej. introducir una trama codificada en *MATLAB* en el decodificador de "C") contiene algunas distorsiones, sin embargo es inteligible y de buena calidad.

Capítulo V. Conclusiones

Se realizó una implementación del codificador MELP siguiendo las especificaciones del "draft" del estándar, ya que la versión definitiva aún no está terminada.

Con el fin de verificar los requerimientos de interoperabilidad, se utilizó una implementación en código *"C"* desarrollada por *Texas Instruments* en 1996. Se implementaron dos funciones para convertir un archivo *".mlp"* con formato en *"C"* en un archivo *".mlp"* con formato en *Matlab* y viceversa (*los archivos ".mlp" contienen a la trama codificada*), estas funciones se describen en el apéndice B, junto con el resto de las funciones de la implementación.

Se implementó una prueba objetiva que considera la distancia en el dominio de la frecuencia entre las señales original y sintetizada.

Los resultados de las pruebas subjetivas *MOS* (por sus siglas en Inglés *Mean Opinion Score*) indican que la implementación en *Matlab* satisface los requerimientos de calidad del nuevo estándar (*ver capítulo IV*) [19],[24]. Sin embargo, cuando un archivo *".mlp"* de *Matlab* es convertido a un archivo *".mlp"* con formato en *"C"* la voz producida por el decodificador de la implementación en *"C"* contiene algunas distorsiones. No obstante la voz reproducida es inteligible y de buena calidad. Lo anterior también ocurre en el caso contrario en el que un archivo *".mlp"* del código en *"C"* es convertido a un archivo *".mlp"* con formato en *Matlab.*

La voz producida por la implementación en *Matlab* del codificador MELP fue comparada con la voz producida por un codificador tradicional LPC 10E. De lo anterior se observó que la voz sintetizada producida por el MELP es de mucho mejor calidad, además de que no requiere de entrenamiento para la captación del mensaje ó la distinción de algunas cualidades propias de la voz del locutor.

El aumento en la calidad producida por el nuevo modelo, conlleva a un aumento en la complejidad del algoritmo y por lo tanto para implementaciones en tiempo real del nuevo modelo, se necesitan procesadores digitales de señales más rápidos y poderosos.

Se puede realizar una comparación de la complejidad entre los codificadores MELP y LPC 10E a partir de las referencias [22] y [25], en las cuales se reporta que el LPC 10E requiere de 7 MIPS [22] y que el MELP requiere de 20 MIPS [25], para su implementación en tiempo real.

Para facilitar el uso del programa en *Matlab*, se implementó una interfaz gráfica para el usuario, para activarla únicamente hay que teclear ***melp*** en la pantalla principal de *Matlab*. En el apéndice B se describe esta interfaz.

La ventaja del uso de *Matlab* como lenguaje de programación del algoritmo es la facilidad para realizar modificaciones al programa, además, sinedo un lenguaje de muy alto nivel la comprensión de las subrutinas es mucho mas sencilla. Como principal desventaja se encuentra el tiempo de procesamiento, sin embargo, esta desventaja puede no ser muy significativa si se hace uso de un compilador que convierta el código en *Matlab* a código en *C*. En la Tabla 1 se muestran los tiempos de procesamieto utilizando una computadora con un procesador pentium a 133 MHz con 16 Mbytes en RAM.

La tendencia de los codificadores paramétricos se basa en el modelo de predicción lineal como modelo del tracto vocal (ya que es capaz de modelarlo de buena manera con relativamente poca información). Sin embargo, se buscan mejores modelos de la excitación, entre los cuales figuran el modelo de excitación en multi-bandas y la excitación armónica (generada por un conjunto de osciladores senoidales) entre otros.

El trabajo que se puede realizar en un futuro con el nuevo modelo de codificación (MELP) puede dividirse en dos: La implementación del algoritmo en un sistema de tiempo real ó investigación para mejorar el algoritmo. En la parte de implementación se puede pensar en la transmisión de voz por redes, ya que el trafico que se generaría es mucho menor que el generado por otros algoritmos de codificación, también puede pensarse en sistemas de correo de voz.

En la parte de investigación, se pueden buscar formas para disminuir la complejidad del algoritmo, hacerlo mas robusto en condiciones de canal ruidoso, bajar su velocidad de transmisión ó mejorar el modelo con el fin de mejorar la calidad.

Duración del archivo	Duración del Análisis (MIN:SEG)	Duración de la síntesis (MIN:SEG)	Duración total (MIN:SEG)
1 seg.	3:09	1:20	4:29
2 seg.	6:23	2:24	8:47
3 seg.	9:32	3:30	13:02
4 seg.	12:40	4:39	17:19
5 seg.	15:47	5:54	21:41
10 seg.	32:06	13:10	45:16

Tabla 1. Tiempos de procesamiento de archivos de diferente duración, utilizando un procesador pentium a 133 MHz con 16 Mbytes en RAM.

APÉNDICES

Apéndice A. LPC

El método de predicción lineal ha sido ampliamente utilizado en muchos campos, pero es en el procesamiento digital de la señal de voz donde ha tenido un gran éxito. Lo anterior se debe principalmente a que el método de predicción lineal es capaz de generar un buen modelo del tracto vocal con muy pocos parámetros.

El método de predicción lineal estima una muestra de voz como una combinación lineal de su pasado, removiendo la correlación existente entre muestras contiguas (análisis predictor de término corto). También es posible realizar un predictor de "pitch" removiendo la correlación existente entre muestras alejadas (análisis predictor de término largo). En este apéndice nos enfocaremos en el predictor de término corto, en [9] se puede profundizar en el predictor de término largo.

Al proceso de extracción de parámetros se le conoce como análisis, mientras que al proceso de reconstrucción de la señal se le conoce como síntesis. Este último utiliza un filtro todo polos cuyos coeficientes corresponden a los coeficientes LPC extraídos en el análisis.

1. Determinación de los coeficientes LPC

Como se mencionó anteriormente en el método de predicción lineal, se obtiene un estimado de una muestra de la señal de voz, como una combinación lineal de su pasado. Para obtener los coeficientes óptimos se realiza el siguiente procedimiento, al cual se le conoce como el método de autocorrelación. Sea $s_e(n)$ una señal de voz estimada, expresada como una combinación de su pasado,

$$s_e(n) = \sum_{j=1}^{p} \alpha_j s(n-j) \quad (1)$$

Donde *p* el número de muestras del pasado, utilizadas para estimar la nueva muestra, *s(n)* es la señal original y α son los coeficientes de predicción lineal. Para obtener los coeficientes α óptimos se realiza el siguiente procedimiento:

Sea *e(n)* la diferencia entre las señales original y estimada,

$$e(n) = s(n) - s_e(n) \quad (2)$$

sustituyendo (1) en (2) obtenemos que,

$$e(n) = s(n) - \sum_{j=1}^{p} \alpha_j s(n-j) \quad (3)$$

El error cuadrático medio se obtiene a partir del valor esperado del error *e(n)* al cuadrado,

$$E = \mathrm{E}\{e(n)^2\} = \mathrm{E}\left\{\left[s(n) - \sum_{j=1}^{p} \alpha_j s(n-j) \right]^2 \right\} \qquad (4)$$

Donde E, en la ecuación (4), es el operador esperanza. Derivando (4) con respecto a α_k e igualando cero (para obtener los α's óptimos) se tiene que:

$$\frac{dE}{d\alpha_k} = \frac{d}{d\alpha_k}\mathrm{E}\left\{\left[s(n) - \sum_{j=1}^{p} \alpha_j s(n-j) \right]^2 \right\} = \mathrm{E}\left\{ \frac{d}{d\alpha_k}\left[s(n) - \sum_{j=1}^{p} \alpha_j s(n-j) \right]^2 \right\} \qquad (5)$$

$$= \mathrm{E}\left\{ -2\left[s(n) - \sum_{j=1}^{p} \alpha_j s(n-j) \right] s(n-k) \right\} = \mathrm{E}\left\{ \left[s(n) - \sum_{j=1}^{p} \alpha_j s(n-j) \right] s(n-k) \right\} = 0 \quad k = 1,\ldots,p$$

Suponiendo que la señal de voz es un proceso estacionario, la expresión (5) puede expresarse como,

$$\sum_{j=1}^{p} \alpha_j \phi_n(k,j) = \phi_n(k,0) \quad k = 1,\ldots,p \qquad (6)$$

Donde,

$$\phi_n(k,j) = \mathrm{E}\{s(n-k)s(n-j)\} \qquad (7)$$

Debido a que la voz no es un proceso estacionario en largos períodos de tiempo, se utilizan ventanas de voz de corta duración, ya que en intervalos cortos si se le puede considerar como un proceso estacionario, por lo tanto se puede remplazar el operador esperanza en (7) por una sumatoria con límites finitos, por lo que la expresión (7) podría expresarse como:

$$\phi_n(k,j) = \mathrm{E}\{s(n-k)s(n-j)\} = \sum_{m} s_n(m-k)s_n(m-j) \qquad \begin{matrix} k = 1,\ldots,p \\ j = 0,\ldots,p \end{matrix} \qquad (8)$$

Si suponemos que $s_n(m)$ es cero fuera del intervalo $0 \leq m \leq N-1$, donde N es la longitud de la ventana, cuando se trata de estimar las muestras en el intervalo $N \leq m \leq N+p$, se tratan de predecir muestras con valor cero. La muestra predecida no va a ser exactamente cero, por lo que el error de predicción tampoco va a ser cero. Para eliminar este tipo de errores en el método de autocorrelación se consideran ventanas de longitud $N+p$, por lo que la expresión (8) en función de la ventana recién definida quedaría como:

$$\phi_n(k,j) = \sum_{m=0}^{N+p-1} s_n(m-k)s_n(m-j) \qquad \begin{matrix} k=1,...,p \\ j=0,...,p \end{matrix} \qquad (9)$$

o bien

$$\phi_n(k,j) = \sum_{m=0}^{N-1-(k-j)} s_n(m)s_n(m+k-j) \qquad \begin{matrix} k=1,...,p \\ j=0,...,p \end{matrix} \qquad (10)$$

Es decir, el valor de $\phi_n(k,j)$ corresponde al valor de autocorrelación de término corto,

$$\phi_n(k,j) = R_n(|k-j|) \qquad \begin{matrix} k=1,...,p \\ j=0,...,p \end{matrix} \qquad (11)$$

de manera que la expresión (6), puede escribirse en función de la función de autocorrelación como:

$$\sum_{j=1}^{p} \alpha_j R_n(|k-j|) = R_n(k) \quad k=1,...,p \qquad (12)$$

Cuya forma matricial es,

$$\begin{bmatrix} R_n(0) & R_n(1) & \cdots & R_n(p-2) & R_n(p-1) \\ R_n(1) & & & & R_n(p-2) \\ \vdots & & & & \vdots \\ R_n(p-2) & & & & R_n(1) \\ R_n(p-1) & R_n(p-2) & \cdots & R_n(1) & R_n(0) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{p-1} \\ \alpha_p \end{bmatrix} = \begin{bmatrix} R_n(1) \\ R_n(2) \\ \vdots \\ R_n(p-1) \\ R_n(p) \end{bmatrix} \qquad (13)$$

la cual tiene una estructura *Toeplitz*. En (13), se tiene un sistema de ecuaciones del tipo $Ax = b$, donde la solución para x son los coeficientes de predicción lineal óptimos. Para resolver un sistema como el anterior, únicamente hay que invertir la matriz A y multiplicar ambos lados de la expresión $Ax = b$ por la inversa de A, obteniendo $x = A^{-1}b$. Sin embargo, existe un método de resolver el sistema anterior que toma en cuenta la

estructura *Toeplitz* de la matriz, el método es conocido como el Algoritmo de *Levinson-Durbin*. El Algoritmo de *Levinson-Durbin* es un algoritmo recursivo para resolver el sistema de ecuaciones de la expresión (13), en este algoritmo al calcular los coeficientes LPC para un filtro de orden *p*, es necesario calcular los coeficientes LPC para todos filtros de orden inferior a *p*, por lo que al final del proceso se tienen los coeficientes para los *p* filtros. El proceso parte de la condición inicial,

$$E_n(0) = R_n(0) \qquad (14)$$

donde E_n representa el error cuadrático medio mínimo y R_n es el valor de la función de autocorrelación definido en (11).

Para valores de *k* entre 1 y *p* se tiene que,

$$K_k = \frac{\left[R_n(k) - \sum_{j=1}^{k-1} \alpha_j(k-1) R_n(k-j) \right]}{E_n(k-1)} \qquad k = 1,\ldots,p \qquad (15)$$

donde K_k se conoce como un coeficiente de reflexión y los coeficientes α están dados como:

$$\begin{aligned} \alpha_k(k) &= K_k \\ \alpha_j(k) &= \alpha_j(k-1) - K_k \alpha_{k-j}(k-1) \end{aligned} \qquad j = 1,\ldots,k-1 \qquad (16)$$

El error se actualiza para cada etapa de la siguiente manera:

$$E_n(k) = (1 - K_k^2) E_n(k-1) \qquad (17)$$

Al final del proceso, cuando *k=p*, los coeficientes de la expresión (16) son los coeficientes de predicción lineal que minimizan el error cuadrático medio de (4), y por lo tanto son los utilizados para procesar la señal de voz.

2. Conversión de los coeficientes LPC's a coeficientes LSF's

Una vez que se han obtenido los coeficientes LPC´s, con ayuda del algoritmo de Levinson-Durbin, éstos son convertidos a coeficientes LSF's (*Line Spectral Frequencies*). Existen muchos métodos para obtener los LSF's, en este apéndice se explica un método que hace uso de los polinomios de Chebyshev [15], ya que el algoritmo implementado en este trabajo utiliza este método. Los coeficientes LSF's se obtienen a partir de un par de polinomios $F_1(z)$ y $F_2(z)$, los cuales están en función de un polinomio $A(z)$ de orden *p*, formado con los coeficientes LPC's.

$$F_1(z) = A(z) + z^{-(p+1)}A(z^{-1})$$
$$F_2(z) = A(z) - z^{-(p+1)}A(z^{-1}) \qquad (18)$$

Donde

$$A(z) = 1 - \sum_{i=1}^{p} a(i)z^{-i} \qquad (19)$$

a(i) en la ecuación (19), corresponde a los coeficientes de predicción (LPC's) obtenidos en la primera parte. Los polinomios $F_1(z)$ y $F_2(z)$ son polinomios simétrico y antisimétrico respectivamente, sus raíces están ubicadas sobre el circulo unitario y son conocidas como los coeficientes LSF's. De acuerdo con [15] los polinomios $F_1(z)$ y $F_2(z)$ tienen raíces en z = 1 y/o en z = -1, dependiendo del valor de *p*. Estas raíces pueden removerse de la siguiente manera:

$$G_1(z) = \frac{F_1(z)}{1+z^{-1}} \quad y \quad G_2(z) = \frac{F_2(z)}{1-z^{-1}} \qquad p \ par$$
$$G_1(z) = F_1(z) \quad y \quad G_2(z) = \frac{F_2(z)}{1-z^{-2}} \qquad p \ impar \qquad (20)$$

Donde $G_1(z)$ y $G_2(z)$ son polinomios simétricos de orden par $2M_1$ y $2M_2$ respectivamente, donde M_1 y M_2 están dados por,

$$M_1 = \frac{p}{2} \quad y \quad M_2 = \frac{p}{2} \qquad p \ par$$
$$M_1 = \frac{p+1}{2} \quad y \quad M_2 = \frac{p-1}{2} \qquad p \ impar \qquad (21)$$

Debido a que las raíces tanto de $G_1(z)$ como de $G_2(z)$ son valores complejos conjugados, únicamente interesan las raíces de estos polinomios que se encuentran en el semicírculo superior, donde los LSF's son las posiciones angulares ω_i de las raíces, para valores de i = 1, ..., *p*. Un total de *p* coeficientes LSF's son enviados, por lo que $G_1(z)$ contribuye con M_1 raíces y $G_2(z)$ con M_2, es decir, $M_1 + M_2 = p$. En el caso del algoritmo implementado tanto M_1 como M_2 son 5, ya que se utilizó un análisis de predicción lineal de orden 10 ($p = 10$). Para este caso especifico los polinomios $G_1(z)$ y $G_2(z)$, cada uno de orden 10, tienen la siguiente forma,

$$G_1(z) = 1 + g_1(1)z^{-1} + g_1(2)z^{-2} + \cdots + g_1(5)z^{-5} + g_1(4)z^{-6} + \cdots + g_1(1)z^{-9} + z^{-10}$$
$$G_2(z) = 1 + g_2(1)z^{-1} + g_2(2)z^{-2} + \cdots + g_2(5)z^{-5} + g_2(4)z^{-6} + \cdots + g_2(1)z^{-9} + z^{-10} \quad (22)$$

Evaluando z en $e^{j\omega}$ para las expresiones de (22) y realizando una expansión en series de cosenos se tiene que:

$$G_1(e^{j\omega}) = e^{-j\omega M_1} G_1'(\omega)$$
$$G_2(e^{j\omega}) = e^{-j\omega M_2} G_2'(\omega) \quad (23)$$

donde,

$$G_1'(\omega) = 2\cos(5\omega) + 2g_1(1)\cos(4\omega) + \cdots + 2g_1(4)\cos(\omega) + g_1(5)$$
$$G_2'(\omega) = 2\cos(5\omega) + 2g_2(1)\cos(4\omega) + \cdots + 2g_2(4)\cos(\omega) + g_2(5) \quad (24)$$

Las raíces de las expresiones en (24), corresponden a las posiciones angulares de los LSF's. Un método para obtenerlas utiliza una formulación en series de Chebyshev. Si se hace el mapeo $x=cos(\omega)$ entonces,

$$cos(m\omega)=T_m(x) \quad (25)$$

donde $T_m(x)$ es un polinomio de Chebyshev de orden m, en función de x. Además, por definición de los polinomios de Chebyshev, éstos satisfacen la recursión,

$$T_k(x)=2\,xT_{k-1}(x)-T_{k-2}(x) \quad (26)$$

con condiciones iniciales $T_0(x)=1$ y $T_1(x)= x$.

El método para convertir los coeficientes LPC's a LSF's, obtiene los polinomios de la expresión (24) en función de la variable x, con ayuda de las expresiones (26) y (25), por lo que la expresión (24) quedaría como:

$$G_1'(x) = 2T_5(x) + 2g_1(1)T_4(x) + 2g_1(2)T_3(x) + 2g_1(3)T_2(x) + 2g_1(4)T_1(x) + g_1(5)$$
$$G_2'(x) = 2T_5(x) + 2g_2(1)T_4(x) + 2g_2(2)T_3(x) + 2g_2(3)T_2(x) + 2g_2(4)T_1(x) + g_2(5) \quad (27)$$

desarrollando los polinomios de Chebyshev, la expresión (27) tendría la siguiente forma:

$$G_1'(x) = C_1(1)x^5 + C_1(2)x^4 + C_1(3)x^3 + C_1(4)x^2 + C_1(5)x^1$$
$$G_2'(x) = C_2(1)x^5 + C_2(2)x^4 + C_2(3)x^3 + C_2(4)x^2 + C_2(5)x^1 \qquad (27a)$$

Las raíces de estos polinomios ($G'_1(x)$ y $G'_2(x)$) están intercaladas en el intervalo $-1 \leq x \leq 1$, comenzando siempre, a partir de 1, con una raíz de $G'_1(x)$. Lo anterior se puede observar en la figura 1, en la cual también se observa que tanto $G'_1(x)$ como $G'_2(x)$ tienen 5 raíces (x_i), que era lo que se esperaba.

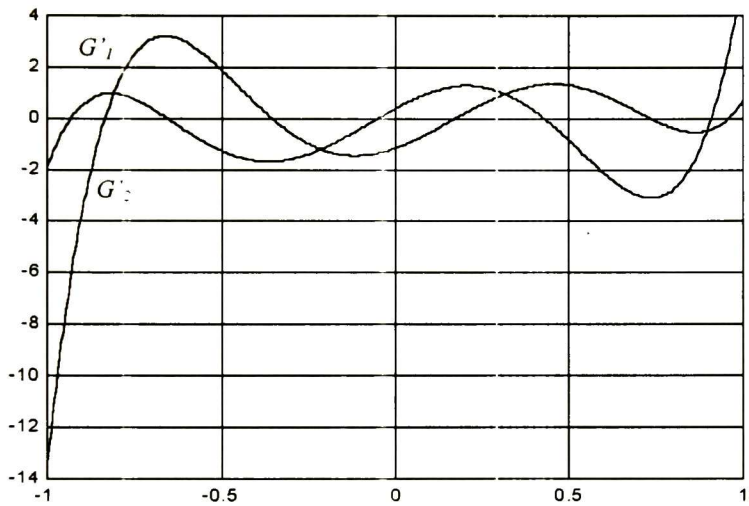

Figura 1. Gráficas de $G'_1(x)$ y $G'_2(x)$.

Los valores de los coeficientes LSF's se obtienen con ayuda de la expresión (28).

$$LSF_i = \frac{8000\omega_i}{2\pi} \quad i = 1,...,10 \qquad (28)$$

donde

$$\omega_i = \cos^{-1}(x_i) \qquad (29)$$

y x_i corresponde a las raíces de los polinomios $G'_1(x)$ y $G'_2(x)$.

A partir de (20) y (22), es posible poner a los coeficientes $g_1(h)$ y $g_2(h)$, h=1,...,5, en función de los coeficientes $a(i)$ del polinomio $A(z)$, expresión (30).

$$
\begin{aligned}
g_1(1) &= [a(1) + a(10)] - 1 \\
g_1(2) &= [a(2) + a(9)] - g_1(1) \\
g_1(3) &= [a(3) + a(8)] - g_1(2) \\
g_1(4) &= [a(4) + a(7)] - g_1(3) \\
g_1(5) &= [a(5) + a(6)] - g_1(4) \\
g_2(1) &= [a(1) - a(10)] + 1 \\
g_2(2) &= [a(2) - a(9)] + g_2(1) \\
g_2(3) &= [a(3) - a(8)] + g_2(2) \\
g_2(4) &= [a(4) - a(7)] + g_2(3) \\
g_2(5) &= [a(5) - a(6)] + g_2(4)
\end{aligned}
\qquad (30)
$$

Las raíces x_i se pueden obtener al dejar los polinomios $G'_1(x)$ y $G'_2(x)$ en función de los coeficientes a_i, sustituyendo los valores de la expresión (30) en la expresión (27). En la referencia [15] se sugiere un método numérico de baja complejidad para implementar la extracción de las raíces x_i. Lo anterior se encuentra implementado en la función *convelsf* de *Matlab*.

3. Recuperación de los coeficientes LPC's

Para recuperar los coeficientes LPC's, se sigue el proceso inverso por lo que las LSF's recibidos son primero convertidos a las raíces x_i con la expresión siguiente:

$$x_i = \cos\left(\frac{2\pi LSF_i}{8000}\right) \qquad (31)$$

A partir de estas raíces se pueden obtener los polinomios de la expresión (27a) y a partir de los coeficientes de estos polinomios es posible obtener los coeficientes de la expresión (27) de la siguiente manera:

$$
\begin{aligned}
g_1(1) &= C_1(2)/16 & g_2(1) &= C_2(2)/16 \\
g_1(2) &= [C_1(3) + 40]/8 & g_2(2) &= [C_2(3) + 40]/8 \\
g_1(3) &= [C_1(4) + 16g_1(1)]/4 & g_2(3) &= [C_2(4) + 16g_2(1)]/4 \\
g_1(4) &= [C_1(5) + 6g_1(2) - 10]/2 & g_2(4) &= [C_2(5) + 6g_2(2) - 10]/2 \\
g_1(5) &= [C_1(6) + 2g_1(3) - 2g_1(1)] & g_2(5) &= [C_2(6) + 2g_2(3) - 2g_2(1)]
\end{aligned}
\qquad (32)
$$

Una vez que se han obtenido los coeficientes de la expresión (32) es posible regresar a los coeficientes LPC's (a_i) aplicando la expresión (33).

$$
\begin{aligned}
a_1 &= [g_1(1) + g_2(1)]/2 \\
a_2 &= [g_1(2) + g_2(2) + g_1(1) - g_2(1)]/2 \\
a_3 &= [g_1(3) + g_2(3) + g_1(2) - g_2(2)]/2 \\
a_4 &= [g_1(4) + g_2(4) + g_1(3) - g_2(3)]/2 \\
a_5 &= [g_1(5) + g_2(5) + g_1(4) - g_2(4)]/2 \\
a_6 &= [g_1(5) - g_2(5) + g_1(4) + g_2(4)]/2 \\
a_7 &= [g_1(4) - g_2(4) + g_1(3) + g_2(3)]/2 \\
a_8 &= [g_1(3) - g_2(3) + g_1(2) + g_2(2)]/2 \\
a_9 &= [g_1(2) - g_2(2) + g_1(1) + g_2(1)]/2 \\
a_{10} &= [g_1(1) - g_2(1) + 2]/2
\end{aligned}
\qquad (33)
$$

Apéndice B Software

La implementación en *Matlab* está dividida en 4 partes, el codificador, el decodificador, funciones de propósito general y la interfaz gráfica con el usuario En este apéndice se describe la función de cada programa de la implementación, así como sus entradas y salidas. Cada uno de los programas está debidamente documentado.

Instalación del programa

La implementación del MELP se realizó en *Matlab versión 5.0* por lo que es necesario, que el dispositivo en donde se vaya a instalar el software tenga instalada esta versión de *Matlab*. Para que se pueda hacer uso del software, es necesario definir las siguientes carpetas en la unidad C:\.

C:\melp\coder\	// Ubicación de las funciones del codificador.
C:\melp\decoder\	// Ubicación de las funciones del decodificador.
C:\melp\gui\	// Ubicación de las funciones de la interfaz gráfica con el usuario.
C:\melp\miselaneos\	// Ubicación de las funciones de propósito general.
C:\melp\entrada\	// Ubicación de los archivos *".wav"* de entrada.
C:\melp\salida\	// Ubicación de los archivos *".wav"* de salida.
C:\melp\melpfiles\	// Ubicación de los archivos *".mlp"*

Los archivos *".mlp"* contienen a la trama codificada y son generados como salida del programa ***principal*** en el codificador, estos archivos sirven como entrada en el programa ***decodificador***.

Los archivos *".wav"* deben de tener el siguiente formato para un correcto desempeño del codificador:

Frecuencia de muestreo: 8000 Hz.
Bits por muestra: 16 bits, mono.
Formato: PCM, uniforme.

Es importante que las trayectorias definidas anteriormente, sean también dadas de alta en Matlab con el fin de poder acceder a ellas, de otra manera no se podrá correr el algoritmo.

Funciones del codificador

Las señales de entrada para las funciones del codificador, se refieren una memoria (*"buffer"*) con 320 muestras de la señal de salida de la función ***cheb02***, estas muestras pueden estar filtradas de alguna otra manera o pueden no estarlo (dependiendo de la función). Existen dos excepciones, una es la señal para la ganancia G_1, para la cual se utiliza una memoria (con 238 muestras de la señal de salida de la

función ***cheb02***) definida en el programa principal y en el capítulo 3 de esta tesis. La segunda excepción se refiere a la señal de entrada a la función ***cheb02***, en este caso la señal de entrada corresponde a todas las muestras de la señal de voz.

apendiceC
Entradas :
Salidas : Libros de código para la cuantificación vectorial en multi-etapas, *MSVQ* (*stg1,stg2,stg3,stg4*).
Descripción : Contiene todos los libros de código para la cuantificación de los coeficientes LSF's.

apendiceD
Entradas:
Salidas : Libro de código para la cuantificación vectorial, *VQ* (*mf*).
Descripción : Contiene el libro de código para la cuantificación de las magnitudes de Fourier.

bancobut
Entradas: Señal de entrada (*buffer*).
Salidas : Señal filtrada por el banco de filtros (*pvbp2,pvbp3,pvbp4,pvbp5*).
Descripción : Banco de filtros Butterworth de 6^{th} orden, pasa-bandas. Se utilizan 4 bandas 500-1000, 1000-2000, 2000-3000 y 3000-4000 [Hz].

cadebit
Entradas: Índices codificados de los parámetros (*indexmf,msvqtrayec,codepitch,vbp,aflag,gan,sincr*).
Salidas : Trama codificada (*bitstream*).
Descripción : Toma los códigos en decimal de los parámetros para generar la trama codificada (binario). Hace uso de la función ***fec***.

codevbp
Entradas: Valor cuantificado de las decisiones de sonoridad (*qvbp1,qvbp2,qvbp3,qvbp4,qvbp5*).
Salidas : Código de las decisiones de sonoridad (*vbp*).
Descripción : Asignación de un código decimal para las decisiones de sonoridad de las bandas de paso.

convelsf
Entradas: Coeficientes LPC's, $a_i = [1\ a_1\ a_2\ \ldots\ a_9\ a_{10}]$ (*ai*).
Salidas : Coeficientes LSF's (*lsfcoef*)
Descripción : Conversión de los coeficientes LPC's a coeficientes LSF's utilizando polinomios de Chebyshev.

cuantizador
Entradas: Número de niveles, límites superior e inferior y valor a cuantificar (*bajo,alto,niveles,valor*).
Salidas : Valor cuantificado e índice del cuantificador (*index,qvalor*).
Descripción : Cuantificador uniforme utilizado para cuantificar el valor de las ganancias y el valor del "pitch".

cheb02
Entradas: Señal de voz (*señal*).
Salidas : Señal filtrada (*vozent*).
Descripción : Filtro Chebyshev tipo II, pasa-altas, fc=60Hz y rizo=30dB. Remueve componentes de DC que pueda contener la señal de entrada. Se aplica un factor de 2^{15} a la señal de entrada para obtener un valor nominal.

doblchk
Entradas: Señal de entrada, un valor de "pitch" y un umbral definido en el proceso de extracción del "pitch" final (para este ultimo, ver el *"draft"* ó el programa de la función ***pitchfinal***) (*señal,pitch,dth*).
Salidas : Valor del "pitch" verificado y su respectivo valor de autocorrelación (*pitchver,rpitchv*).
Descripción : Verifica que el valor del "pitch" obtenido no sea un múltiplo del valor del "pitch" real.

fec
Entradas: Códigos de las ganancias y de la primera etapa del *MSVQ* (*stg1,g1,g2*).
Salidas : Códigos de Hamming para proteger a los parámetros de entrada (*fec1,fec2,fec3,fec4*).
Descripción : Obtención de los códigos de Hamming para protección contra errores de canal. Se utiliza cuando una trama es no sonora en todas sus bandas de frecuencia.

filtroalisa
Entradas: Señal de entrada (*buffer*).
Salidas : Señal filtrada (*salisada*).
Descripción : Filtro de alisamiento utilizado en el proceso extracción de las decisiones de sonoridad.

fourier1

Entradas: Valor cuantificado del "pitch", valor cuantificado de los LSF's y señal de entrada (*qpitch,qlsf, buffer*).

Salidas : Índice de la cuantificación, y valores cuantificado y sin cuantificar de las magnitudes de Fourier, (*indexmf,qfourier,fourier*).

Descripción Obtención y cuantificación de las magnitudes de Fourier de los primeros 10 armónicos del residual de predicción. Se utiliza ***lsfinverso*** para construir un filtro de análisis a partir de los LSF's cuantificados.

ganancia

Entradas: Señal para G_2, Señal para G_1, valor de la autocorrelación de la primera banda de paso y el segundo estimado del "pitch" (*buffer,bufferg1,vbp1,p2*).

Salidas : Valores de las ganancias (*g1,g2*).

Descripción : Cálculo de las ganancias G_1 y G_2.

lpcmio

Entradas: Señal de entrada y el estado anterior del filtro de análisis (*buffer,stA*).

Salidas Coeficientes del filtro LPC, residual de predicción, el "*peakiness*" y el estado actualizado del filtro (*ai,resi,peak,stA*).

Descripción : Para cada trama se obtienen los coeficientes de predicción, el residual y el valor del "*peakiness*", el cual se utiliza en la determinación de las decisiones de sonoridad de las bandas de paso (función ***qvbpi***).

lsfinverso

Entradas: Coeficientes LSF's (*lsf*).

Salidas . Coeficientes LPC's (*lpc*).

Descripción : Función inversa a ***convelsf,*** se obtienen un conjunto de coeficientes LPC's a partir de un conjunto de coeficientes LSF's.

mejorocho

Entradas: Un arreglo que contiene las 8 mejores trayectorias, la trayectoria que se está analizando, el error de la trayectoria que se está analizando y un vector con los errores de las 8 mejores trayectorias (*index,trayecto,error,err*).

Salidas : Un arreglo que contiene las 8 mejores trayectorias y un vector con los errores de las 8 mejores trayectorias (*index,err*).

Descripción : Determinación de las 8 mejores trayectorias en la cuantificación de los LSF's (*MSVQ*). Cuando una trayectoria es mejor que alguna de las trayectorias contenidas en el arreglo de las 8 mejores trayectorias, ésta toma su lugar.

msvq

Entradas: Coeficientes LSF's sin cuantificar y coeficientes LPC's (*lsfcoef,ai*).

Salidas : Arreglo con las 8 mejores trayectorias, el valor cuantificado de los LSF's y los índices de la mejor trayectoria (*index,qlsf,msvqtrayec*).

Descripción : Cuantificación de los coeficientes LSF's. Se realiza una cuantificación vectorial en multi-etapas (*MSVQ*), para cada etapa se asigna un índice. Se utiliza un *MSVQ* de 4 etapas. Se utiliza a la función ***mejorocho***.

pavg

Entradas: Valores del "pitch" final de las ultimas 3 tramas , valor de autocorrelación de la ultima trama y valor de la ganancia G_2 de la trama actual (*p31,p32,p3,rdp3,g2*).

Salidas : Valor promedio del "pitch" (*pavg*).

Descripción : Cálculo del valor promedio del "pitch" a partir de los tres últimos valores de éste.

pitchentero

Entradas: Señal de entrada (*buffer*).

Salidas : Estimado entero del "pitch" (*p1*).

Descripción : Se obtiene un valor estimado entero del "pitch"

pitchfrac

Entradas: Señal de entrada y un estimado del "pitch" (*señal,pitch*).

Salidas : Valor refinado del "pitch" y su correspondiente valor de autocorrelación (*pfr,rdpfr*).

Descripción : Refinamiento del valor del "pitch" a partir de un estimado del mismo.

pitchfinal

Entradas: Valor promedio del "pitch", segundo estimado del "pitch", señal de entrada y el residual de predicción (*pavg,p2,buffer,resi*).

Salidas : Valor final del "pitch" y su correspondiente valor de autocorrelación (*p3,rdp3*).

Descripción : Cálculo del "pitch" final a partir del residual y de la señal de entrada. Esta función hace uso de la función ***doblchk***.

principal

Entradas: Nombre del archivo *".wav "* de entrada, una bandera que indica si se desea generar un archivo *".mlp"* de salida y si es necesario el nombre del archivo *".mlp"* de salida (*wavin,genera,mlpfile*).

Salidas : Trama codificada (*bitstream*) y archivo *".mlp"* de salida.

Descripción : Programa principal del codificador, a partir de este programa se llaman a todas las funciones del codificador. Los archivos *".mlp"* contienen a la trama codificada y son utilizados como entrada en el programa ***decodificador***.

qgana

Entradas: Ganancia G_1, ganancia G_2 y ganancia G_2 de la trama anterior (*g1,g2,g2a*).

Salidas : Valores cuantificados de G_1 y G_2 y los códigos para las ganancias G_1 y G_2 (*qg1,qg2, index1,index2*).

Descripción : Codificador de las ganancias G_1 y G_2. Se utiliza la función ***cuantizador***.

qpitch

Entradas: Valor del "pitch" final y de la energía de sonoridad en la primera banda de paso (*pitch,vbp1*).

Salidas : Valor cuantificado del "pitch", índice del cuantificador y el código del "pitch" (*qpitch,index,pcode*).

Descripción : Cuantificación y codificación del "pitch" final, se utiliza la función ***cuantizador*** y la tabla definida en el "draft" para la asignación del código.

qvbpi

Entradas: Valores de las energías de sonoridad de las bandas de paso y el valor del *"peakiness"* (*vbp1,vbp2,vbp3,vbp4,vbp5,peak*).

Salidas : Valores cuantificados de las energías de sonoridad de las bandas de paso y el código de las decisiones de sonoridad (*qvbp1,qvbp2,qvbp3,qvbp4,qvbp5,vbp*).

Descripción : Obtención de los valores cuantificados de las energías de sonoridad de las bandas de paso (*vbpi* (i = 1, ..., 5) puede tomar valores entre 0 y 1, *qvbpi* puede tomar valores 0 ó 1), y del código en decimal de las decisiones de sonoridad de las 4 bandas de paso superiores, para lo ultimo se hace uso de la función ***codevbp***.

sepamin

Entradas: Coeficientes LSF´s (*lsf*).

Salidas : Coeficientes LSF´s con una separación mínima de 50 Hz entre cada uno (*lsfsep*).

Descripción : Proceso para asegurar una separación mínima entre cada coeficiente LSF de 50 Hz.

vbp1

Entradas: Señal de entrada y valores del "pitch" de la trama actual y anterior (*buffer,pitch,pitcha*).

Salidas : Segundo estimado del "pitch", la energía de sonoridad de la primera banda de paso y la bandera de aperiodicidad (*p2,vbp1,af1ag*).

Descripción : Cálculo del segundo estimado del "pitch", de la energía de sonoridad de la primera banda de paso y de la bandera de aperiodicidad. Se toman dos candidatos para encontrar el segundo estimado del "pitch" (los valores del "pitch" de las tramas actual y anterior) y aquel que genere un mejor valor de autocorrelación en el procedimiento de "pitch" fraccional se toma como el segundo estimado del "pitch".

vbpi

Entradas: Salidas de ***bancobut*** y el segundo estimado del "pitch" (*pvbp2,pvbp3,pvbp4,pvbp5,p2*).

Salidas : Eergías de sonoridad de las 4 bandas de paso superiores (*vbp2,vbp3,vbp4,vbp5*).

Descripción : Determinación de las energías de sonoridad de las 4 bandas de paso superiores, lo anterior se realiza con la función ***pitchfrac***, utilizando las señales que salen de la función ***bancobut***

Funciones del decodificador

apendiceA

Entradas:

Salidas : Coeficientes de los filtros de forma de onda (*sh*).

Descripción : Contiene los coeficientes de las 5 bandas de paso, estos coeficientes se utilizan para generar los filtros de forma de onda (función ***shaping***). La salida *sh* es una matriz de (31X5). Se tienen 31 coeficientes para cada banda de paso.

apebdiceCd

*Ver **apendiceC** en el Codificador.*

apendiceDd

*Ver **apendiceD** en el Codificador.*

atenua

Entradas: Valores de las ganancias G_1 y G_2 y un estimador de ruido de fondo G_n (*g1,g2,gn*).

Salidas : Valor de las ganancias atenuadas y un par de estimadores de ruido de fondo (*g1at,g2at, gn1,gn2*).

Descripción : Atenuación del ruido, realizada en función de una atenuación de ganancias en regiones silenciosas.

cadehamm

Entradas: Trama codificada (*bitstream*).

Salidas : Códigos de todos los parámetros y una bandera que indica la existencia de algún error incorregible de canal (*indexmf,msvqtrayec,codepitch,vbp,aflag,gan,sincr,hammflag*).

Descripción : Extracción de los códigos de los parámetros a partir de la trama codificada. Función inversa a ***cadebit*** del codificador. Utiliza a la función ***fecinve*** para detectar y corregir errores de canal.

decodificador

Entradas: Nombre del archivo *".mlp"*, una bandera que indica si se desea generara una archivo *".wav"* de salida y si es necesario el nombre del archivo *".wav"* de salida (*mlpfile,genewav,wavout*).

Salidas : Señal sintetizada (*señal*) y archivo *".wav"* de salida.

Descripción : Programa principal del decodificador, en este programa se realiza la síntesis y el realce espectral adaptable. A partir de este programa se llaman a todas las funciones del decodificador.

decopitch

Entradas: Código del "pitch" (*codepitch*).

Salidas : Valor del "pitch" y una bandera que indica la existencia de errores de canal (*pitch,flagsure*).

Descripción : Determinación del valor del "pitch" a partir de su código, para ello se utiliza la tabla definida en el "draft".

exciperi

Entradas: Valor interpolado del "pitch", valor interpolado de las magnitudes de Fourier, valor interpolado de los coeficientes del filtro de forma de onda para la excitación periódica y su estado (*tint,mfint,pulsecffint,stPE*).

Salidas : Excitación periódica con longitud del valor de *tint* y valor actualizado del filtro de forma de onda para la excitación periódica (*periodica,stPE*).

Descripción : Generación de la excitación periódica a partir de las magnitudes de Fourier, la excitación ya está filtrada por el filtro de forma de onda y al sumarse con la excitación de ruido de la misma longitud, produce la excitación mixta (función ***decodificador***).

fecinve

Entradas: Códigos de Hamming para corrección de errores, las ganancias (G_1 y G_2) y el índice de la primera etapa de la cuantificación (*MSVQ*) de los LSF's (*fec1,fec2,fec3,fec4,g1,g2,stg1*).

Salidas : Los códigos verificados de las ganancias (G_1 y G_2) y del índice stg_1, y el valor de la bandera que indica si se detectaron errores de canal (*g1,g2,stg1,hammflag*).

Descripción : Detección y corrección de errores de canal en los parámetros protegidos por los códigos de Hamming, cuando la trama de voz es no sonora en las 5 bandas de frecuencia. Esta función utiliza a la función ***sindrome*** para corrección de errores.

gdeco

Entradas: Vector de dos elementos que contiene a los códigos de las ganancias (G_1 y G_2), valor de la ganancia G_2 de la trama anterior y un error cuya función se describe en el "draft" (*gan,g2a,errorg*).

Salidas : Valor de las ganancias (G y G_2) y el error *errorg* actualizado (*g1,g2,errorg*).

Descripción : Decodificación de las ganancias (G_1 y G_2).

interpola

Entradas: Apuntador del inicio de la ventana de síntesis, los valores anterior y actual del parámetro a interpolar y un factor de interpolación (*t0,anterior,actual,inte*).

Salidas : Parámetro interpolado (*interpolado*).

Descripción : Función para la interpolación de parámetros.

jitter

Entradas: Valor del "pitch" y la bandera de aperiodicidad (*pitch,aflag*).

Salidas : Valor del "pitch" con una variación (*pitchjitt*).

Descripción : Varia el valor del *pitch* hasta un 25 % de su valor cuando *aflag* es 1, de otra manera no hay variación alguna.

lsfinversod

Ver ***lsfinverso*** *en el Codificador.*

pulsedisp

Entradas: Señal de entrada y el estado del filtro de dispersión (*señal,stP*).
Salidas : Señal filtrada por el filtro de dispersión de pulso y el nuevo estado del filtro (*salida,stP*).
Descripción : Filtrado de la señal sintetizada por el filtro de dispersión de pulso. La señal que entra a este filtro en el programa ***decodificador*** tiene la misma longitud de la excitación mixta.

reflecoef

Entradas: Coeficientes LPC's (*lpc*).
Salidas : Primer coeficiente de reflexión (*k1*).
Descripción : Se obtiene el primer coeficiente de reflexión a partir de los coeficientes LPC's, este coeficiente se utiliza para definir el filtro de realce espectral adaptable, en el programa ***decodificador***.

sepamind

Ver ***sepamin*** *en el Codificador.*

shaping

Entradas: Bandera con la Información de sonoridad de la primera banda de paso y el código de las decisiones de sonoridad de las 4 bandas superiores (*unvoiced,vbp*).
Salidas : Coeficientes de los filtros de forma de onda para la excitación periódica y de ruido (*pulsecff,noisecff*).
Descripción : Obtención de los coeficientes de los filtros de forma de onda en base a la información de sonoridad de las bandas de paso, se hace uso de la función ***apendiceA***.

sindrome

Entradas: El vector de error y el tipo de código Hamming ((8,4) ó (7,4)),(*error,code*).
Salidas : Valor del síndrome (*sindro*).
Descripción : El valor de *sindro* es utilizado para corregir errores en los parámetros protegidos.

Funciones de propósito general

En esta parte del apéndice se definen las funciones en *Matlab* para desarrollar las pruebas en condiciones de canal ruidoso, la medida objetiva considerada y las funciones para la interoperabilidad del código en *Matlab*, con el código en *C* (*versión 1.2*) desarrollado por Texas Instruments en 1996.

bschan

Entradas: Nombres de los archivos *".mlp"* de entrada y de salida y la probabilidad de error, (*mlpfile,pe,mlpout*).
Salidas : Nueva trama codificada con ruido de canal (*nueva*) y archivo *".mlp"* de salida.
Descripción : Canal binario simétrico, genera un archivo de salida *".mlp"* a partir de un archivo *".mlp"* de entrada. El archivo de salida se puede utilizar en el programa ***decodificador***.

gausschan

Entradas: Nombres de los archivos *".mlp"* de entrada y de salida y la desviación estándar, (*mlpfile,mlpout,var*).
Salidas : Nueva trama codificada con ruido de canal (*nueva*) y archivo *".mlp"* de salida.
Descripción : Canal Gaussiano, genera un archivo de salida *".mlp"* a partir de un archivo *".mlp"* de entrada. El archivo de salida se puede utilizar en el programa ***decodificador***.

objetiva

Entradas: Señal de voz de referencia (original) y señal de voz sintetizada (*refe,señal*).
Salidas : Distancia en dB entre las señales de entrada (*error*).
Descripción : Obtención de la medida objetiva (entre dos señales), definida en el capítulo 4 de esta tesis.

read_

Entradas: Nombre de un archivo binario (*fname*).
Salidas : Variable con el contenido del archivo binario (*X*).
Descripción : Escribe el contenido de un archivo binario en la variable *X*. Es la función inversa de ***write_***.

write_

Entradas: Nombre del archivo binario y variable (*fname,X*).
Salidas : Archivo binario.
Descripción : Escribe el contenido de la variable *X* en un archivo binario. Función inversa a ***read_***.

c2matl

Entradas: Nombre del archivo *".mlp"* con el formato del código *C*, bandera que indica si se desea generar un archivo *".mlp"* de salida, Nombre del archivo *".mlp"* con formato del código en *Matlab* (*infile,genera,outfile*).
Salidas : Trama codificada en formato *Matlab* (*bitstream*) y archivo *".mlp"* con formato en *Matlab*.
Descripción : Programa para convertir una trama codificada con formato en código *C* a una trama codificada con formato en código *Matlab*. Función inversa a ***matl2c***.

matl2c

Entradas: Trama codificada con formato en *Matlab* y nombre del archivo *".mlp"* de salida con formato en código *C* (*bitsream,outfile*).
Salidas : Archivo *".mlp"* con formato en código *C*.
Descripción : Programa para convertir una trama codificada con formato en código *Matlab* a una trama codificada con formato en código *C*. Función inversa a ***c2matl***, puede ser necesario el uso de la función ***mlp2bit***.

mlp2bit

Entradas: Nombre del archivo *".mlp"* con formato en *Matlab* (*mlpfile*).
Salidas : Trama codificada con formato en *Matlab* (*bitstream*).
Descripción : Recuperación de la trama codificada en la variable *bitstream* a partir de un archivo *".mlp"* con formato en *Matlab*. Esta función se puede utilizar para generar la entrada de la trama codificada a la función ***matl2c***, a partir de un archivo *".mlp"* ya existente.

Interfaz gráfica con el usuario

Con el fin de facilitar el uso del programa en *Matlab*, se diseñó una interfaz gráfica para el manejo del programa por el usuario. Para activar a la interfaz el usuario únicamente tiene que teclear ***melp*** en la pantalla principal de *Matlab*, al hacer esto el usuario verá aparecer en la pantalla un menú como el que se ilustra en la figura 1, este menú es el menú principal y a partir de él, se pueden correr todas las opciones del algoritmo. En la figura 2 se muestran las ventanas de las opciones del menú principal.

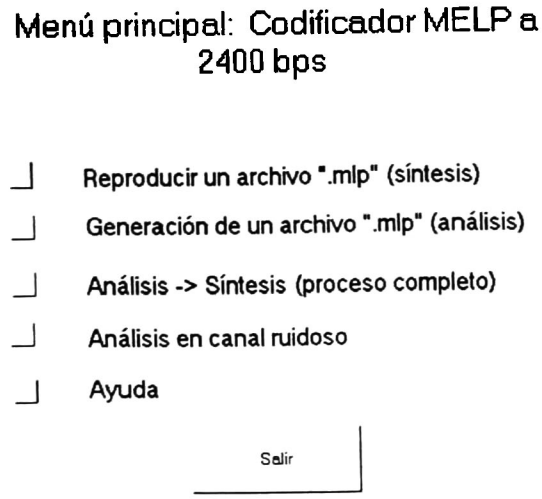

Figura 1. Menú principal de algoritmo de codificación de voz a 2400 bps MELP

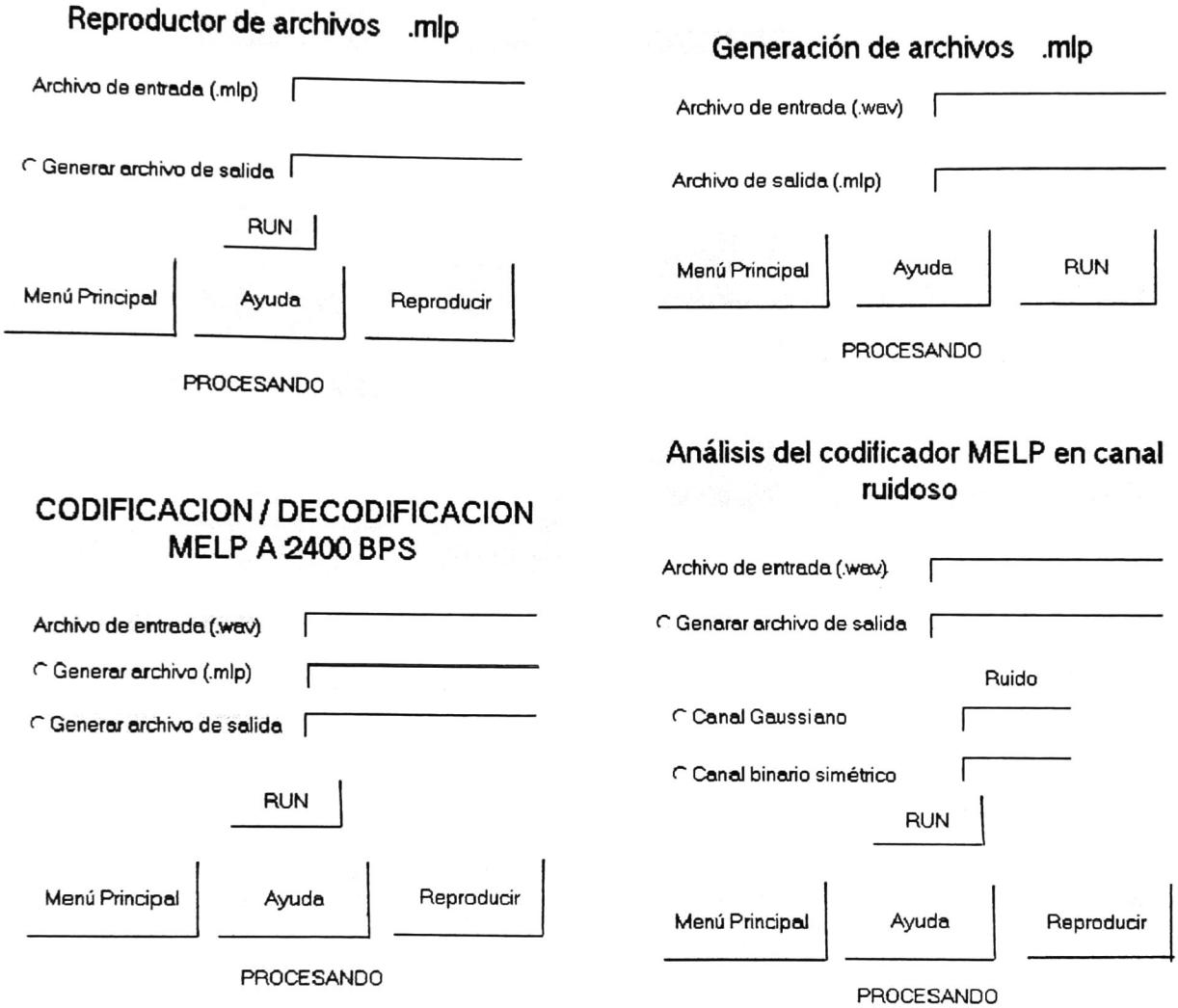

Figura 2. Ventanas de las diferentes opciones del menú principal.

***NOTA:** Las extensiones (".wav" o .mlp") deben de incluirse en los nombres de los archivos.*

Referencias

[1] Alan V. McCree, Lynn M. Supplee, Ronald P Cohn, John S. Collura "MELP: The New Federal Standard at 2400 bps, IEEE International Conference on Acoustics, Speech and Signal Processing, Munich 1997

[2] Alan V. McCree and T. P Barnwell III, "A Mixed Excitation LPC Vocoder Model for Low Bit Rate Speech Coding," IEEE Transactions on Speech and Audio Processing, Vol. 3, No. 4, July 1995, pp. 242-250.

[3] P Lupini, H. Hassein and V. Cuperman, "A 2400 kb/s CELP Speech Codec With Class-Dependent Structure," IEEE International Conference on Acoustics, Speech and Signal Processing, 1993, pp. 143-146.

[4] Simon Haykin, "Adaptive Filter Theory, Prentice Hall Information and Systems Sciences Series, Third Edition, 1996.

[5] D. W. Griffin and J. S. Lim, "Multiband Excitation Vocoder," IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 36, No. 8, August 1988, pp. 1223-1235.

[6] V. Cuperman, P. Lupini and B. Bhattacharya, "Spectral Excitation Coding of Speech at 2400 kb/s,' IEEE International Conference on Acoustics, Speech and Signal Processing, 1995 pp. 496-499.

[7] P Zolfaghari and T. Robinson, "A Formant Vocoder Based on Mixtures of Gaussians,' IEEE International Conference on Acoustics, Speech and Signal Processing, Munich 1997. pp. 1575-1578.

[8] B. S. Atal and S. L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," The Journal of the Acoustical Society of America, April 1997, pp. 637-655.

[9] A. M. Kondoz, "Digital Speech Coding for Low Bit Rate Communication Systems, Wiley Series in Communication and Distributed Systems, 1994.

[10]T. E. Tremain, "The Government Standard Linear Predictive Coding Algorithm: LPC-10," Speech Technology, April 1982, pp. 40-49.

[11]Alan V. McCree and Juan Carlos De Martin, "A 1.7 kb/s MELP Coder with Improved Analysis and Quantization,' IEEE International Conference on Acoustics, Speech and Signal Processing, 1998 pp. 593-596.

[12]"Specifications for the Analog to Digital Conversion of Voice by 2,400 Bit/Second Mixed Excitation Linear Prediction,' Federal Information Processing Standards Publication (MELP).

[13]M. A. Kohler, L. M. Supplee, T. E. Tremain, "Progress Towards a New Government Standard 2400 bps Voice Coder," IEEE International Conference on Acoustics, Speech and Signal Processing, 1995 pp. 488-491.

[14]Y Medan, E. Yair, and D. Chazan, "Super Resolution Pitch Determination of Speech Signals, IEEE Transactions on Signal Processing, Vol. 39, No. 1, January 1991, pp. 40-48.

[15]P. Kabal and R. P. Ramachandran, "The Computation of Line Spectral Frequencies Using Chebyshev Polynomials, IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-34, No. 6, December 1986, pp. 1419-1426.

[16]W. P LeBlanc, B. Battacharya, S. A. Mahmoud, and V. Cuperman, "Efficient Search and Design Procedures for Robust Multi-Stage VQ of LPC Parameters for 4 kb/s Speech Coding, IEEE Transactions on Speech and Audio Processing, Vol. 1, No. 4, October 1993, pp. 373-385.

[17]Bernard Sklar, "Digital Communications, Fundamentals and Applications, Prentice Hall Publication, 1988.

[18]R. W. Hamming, "Error Detecting and Error Correcting Codes, Bell System Technical Journal, No. 26, April 1950, pp.147-160.

[19] Alan V. McCree and T. P. Barnwell III, "Implementation and Evaluation of A 2400 bps Mixed Excitation LPC Vocoder," IEEE International Conference on Acoustics, Speech and Signal Processing, Vol. 2, 1993, pp. 159-162.

[20]J. R. Deller, J. G. Proakis and J. H. Hansen, "Discrete-Time Processing of Speech Signals," Macmillan Publishing Company, 1993.

[21]Alejandro Leñero, Arturo Veloz and José Luis Ponce, "Implementation and Simulation of the New U.S. Federal Standard for Speech Coding at 2400 bps (MELP) Under Ideal and Noisy Channel Conditions Using MATLAB, International Symposium on Information Theory and its Applications, Mexico D.F., 1998.

[22]Andreas S. Spanias, "Speech Coding: A Tutorial Review," Proceedings of the IEEE, Vol. 82, October 1994, pp. 1541-1582.

[23]Gilberto Sánchez, "Manual de Fonética y Fonología" Ed. Trillas, Mexico D.F., 1962.

[24]M. A. Kohler, "A Comparison of The New 2400 bps MELP Federal Standard with Other Standard Coders," IEEE International Conference on Acoustics, Speech and Signal Processing, April 1997.

[25]Alan V. McCree, K. Truong, E. B. George, T. P. Barnwell and V. Viswanathan, "A 2.4 kbits/s MELP Coder Candidate for the New U.S. Federal Standard, Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 1996, pp. 200-203.

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: "Programación y simulación del estandar MELP para codificación de Voz a 2400 bps. del Sr. Alejandro Leñero Beracoechea , el día 16 de noviembre de 1998.

Dr. José Luis Leyva Montiel
Profesor Investigador 3A
CINVESTAV DEL IPN
Unidad Guadalajara.

Dr. Manuel Mauricio Lara Barrón
Profesor Adjunto
Departamento de Ingeniería Eléctrica
Sección Comunicaciones
CINVESTAV DEL IPN,
México, D.F.

Dr. Deni Librado Torres Román
Profesor Titular
CINVESTAV DEL IPN,
Unidad Guadalajara.