UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACION DEL MODELO PSICOACUSTICO PARA CODIFICACION DIGITAL DE SEÑALES DE AUDIO

INFORME DE SUFICIENCIA

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

PRESENTADO POR:
JOSE ISRAEL MEDINA RIVERO

PROMOCIÓN 1999 – II

LIMA – PERU
2005

DEDICATORIA

A mis padres, quienes me orientaron por el camino de la superación y el buen ejemplo, por su amor y sacrificio realizados, guía de mi formación profesional y a mis hermanas por su apoyo invalorable.

A Dios, quien me ha dado la vida y por permitirme disfrutar de una familia maravillosa.

IMPLEMENTACION DEL MODELO PSICOACUSTICO PARA CODIFICACION DIGITAL DE SEÑALES DE AUDIO

SUMARIO

En este informe se realiza una investigación, estudio e implementación del modelo psicoacústico para codificación digital de señales de audio tomando como base el estándar de Codificación de Audio MPEG Capa 3. Este trabajo tiene tres propósitos principales: primero, el estudio y análisis de un sistema de codificación de Audio MPEG así como de los bloques codificador y decodificador; segundo, el diseño y estudio del modelo psicoacústico de acuerdo al estándar ISO/IEC 11172-3 usando MATLAB como herramienta de desarrollo; y tercero, brindar las bases y conceptos fundamentales para futuras investigaciones en el campo del procesamiento de señales de audio.

INDICE

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

PROLOGO

Las señales digitales se han convertido en los últimos años, en el mejor medio para transmitir señales de audio, reemplazando a las analógicas debido a su fácil procesamiento y rápida distribución. En los últimos años, nuevas aplicaciones de audio han sido usadas para redes, sistemas de computo, telefonía y radiodifusión; así, la necesidad de un eficiente sistema de compresión se hacía necesaria, surgiendo de esta manera diferentes algoritmos entre los que se mencionan al estándar de Audio MPEG capa 3 como principal alternativa.

El Capitulo I nos introduce al estudio de los sistemas de compresión de audio, la búsqueda y necesidad de un sistema óptimo de compresión, el porque del estándar MPEG Capa 3 (MP3) como principal medio de transmisión de audio vía Internet, asimismo se brinda una introducción a los estándares de codificación del grupo MPEG: MPEG-1, MPEG-2, MPEG-4, MPEG-7.

En el Capitulo II empieza el desarrollo del codificador y su funcionalidad, se empieza el estudio y descripción del proceso de codificación mediante un diagrama de bloques con las principales partes constituyentes del sistema, tales como el banco

de filtros polifásicos, el uso de la transformada discreta de coseno modificada (MDCT), la aplicación del modelo psicoacústico que simula el patrón de sonidos percibidos por los humanos y que aprovecha la incapacidad del oído humano de escuchar el ruido de cuantización, asimismo se estudian los lazos de control de distorsión y control de tasa de cuantización no uniforme, para luego iniciar el estudio del codificador Huffman y de sus características.

En el Capitulo III, se describe el decodificador MPEG Capa 3, y su funcionalidad mediante el estudio de un diagrama de bloques, el que se subdivide en tres secciones muy bien definidas: Decodificación de los Flujos de bits, Cuantización Inversa, y Mapeo de Frecuencia a Tiempo. Asimismo se describe el formato de la trama MPEG capa 3.

En el capitulo IV se tratan otras técnicas de compresión utilizadas actualmente, así, se brinda información sobre el estándar de Sony ATRAC, sistema Dolby y una breve explicación de los algoritmos usados. Asimismo se indican diferentes maneras de medir la calidad del sonido codificado.

En el capitulo V, se inicia el desarrollo y simulación del modelo psicoacústico usando MATLAB como herramienta de desarrollo, la explicación del programa y las conclusiones de los resultados obtenidos.

CAPITULO I

COMPRESION DE SEÑALES DE AUDIO

1.1 Compresión

Durante los últimos 15 años, las señales de audio digital han reemplazado a las señales analógicas, debido a que las señales digitales ofrecen mayores ventajas comparadas con las analógicas.

Las señales de audio digitales brindan una mejor conservación, fácil procesamiento y distribución. El formato común de todos los sistemas de audio digital es la Modulación por Código de Pulso PCM, esta muestrea el sonido a una tasa fija con bits seteados por la señal de audio. Recientemente nuevas aplicaciones de audio digital han sido usadas para redes de comunicación, radiodifusión, multimedia, y sistemas de computo los cuales enfrentan diversos problemas tales como el ancho de banda del canal, capacidad limitada de almacenamiento y costo. La calidad de audio de CD usa una frecuencia de muestreo de 44.1 Khz, con 16 bits de cuantización, es decir, dos canales estéreo requieren 1'411,200 bits por segundo, esto significa que para transmitir una señal de audio en formato PCM con calidad de CD

sobre una red de comunicación, necesitaremos mas de 1.4 Mbps de ancho de banda. Una canción de 4 minutos de duración requeriría 40 Mbytes de espacio en disco, por lo tanto, ¿cómo transmitir señales de audio por Internet a bajas velocidades o almacenarlas sin que ocupen demasiados volúmenes de disco?. Como resultado de esta interrogante surgió la solución: **compresión de datos.** En comparación con la compresión de video digital y la compresión de voz, el audio digital es relativamente complejo. El oído humano tiene una sensitividad sobre un rango dinámico de 100dB, en contraste, la capacidad visual del sistema humano es mas alta que la resolución del sistema de televisión común.

Para solucionar la necesidad de compresión de audio, y debido a la complejidad que ello requiere, se han propuesto varios métodos, estos métodos pueden ser separados en dos categorías: codificación de transformación y codificación de sub-banda. La codificación de transformación usa transformaciones unitarias para el análisis tiempo a frecuencia. Estos algoritmos típicamente logran una alta resolución espectral estimada con un buen compromiso de resolución temporal.

1.2 Por qué MPEG Capa 3

MPEG Capa 3 surgió como la principal herramienta para audio en Internet., los siguientes factores fueron definitivamente muy útiles para su desarrollo.

1.2.1 Estándar Abierto

MPEG se define como un estándar abierto. Las especificaciones están disponibles (previo pago) para todo el mundo. Mientras hay un número de patentes

que cubren la codificación y decodificación de audio MPEG, todas los poseedores de patentes han declarado que ellos autorizarán el uso de sus patentes en términos justos y razonables para todo el mundo. Ejemplos de códigos fuentes están disponibles para ayudar a los desarrolladores a comprender las especificaciones. Como el formato esta bien definido, no existen problemas con la interoperabilidad de equipo o software de diferentes proveedores, excepto de algunas excepciones, con implementaciones incompletas.

1.2.2 Disponibilidad de codificadores y decodificadores

Codificadores basados en hardware y software DSP han estado disponibles por muchos años, en un primer momento por la demanda para uso profesional en difusión.

1.2.3 Soporte en Tecnología

Mientras la compresión de audio es vista como una tecnología que permite el desarrollo de otras, existen otras que contribuyeron al desarrollo MP3, como:

- El uso ampliamente extendido de tarjetas de sonido en computadoras;
- Computadoras que llegan a ser lo bastante poderosas para ejecutar software de decodificadores de audio y codificadores en tiempo real;
- Acceso rápido a Internet por universidades y negocios;
- Disponibilidad de grabadoras de CD-ROM y CD- Audio

MPEG Capa-3 tuvo la suerte de ser la tecnología correcta, disponible en el momento justo. De esta manera, la búsqueda de codificadores de audio progresó, y

códigos con una mejor eficiencia de compresión estuvieron disponibles. De estos, la Codificación de Audio Avanzada MPEG-2 (ACC) fue desarrollada como la sucesora de MPEG-1.

1.3 Estándares de Codificación de audio MPEG

El grupo de trabajo MPEG, llamado formalmente como ISO/IEC JTC/SC29/WG11, pero más conocido por Grupo experto en imágenes en movimiento, fue establecido por la ISO/IEC en 1988 para desarrollar estándares genéricos (útiles para diferentes aplicaciones) para la representación codificada de imágenes en movimiento, audio asociado y sus combinaciones, así MPEG emprendió la estandarización de las técnicas de compresión para audio y video, siendo originalmente su principal meta la codificación de video junto con la codificación de audio para almacenamiento digital. De esta manera, el estándar de codificación de audio MPEG encontró su rumbo dentro de diversas aplicaciones, incluyendo:

- Radiodifusión digital de audio (Eureka-147 DAB, WorldSpace, ARIB, DRM);
- Transmisiones ISDN para radiodifusión y propósitos de distribución y enlace.
- Almacenamiento de archivos en radiodifusión
- Sonido para televisión digital (DVB, Video CD, ARIB);
- Flujos de Internet (Microsoft Netshow, Apple Quicktime);
- Dispositivos de audio portátiles (mpman, mplayer3, Rio, Lyra, YEPP y muchos mas);

- Almacenamiento e intercambio de archivos de música en computadoras.

Los formatos de compresión de audio mas ampliamente usados son Audio MPEG Capa 2 y Capa 3 y Dolby AC-3.

1.3.1 MPEG-1

MPEG-1 es el nombre para la primera fase del trabajo MPEG, empezó en 1988. Este trabajo fue terminado con la adopción del estándar de la ISO/IEC IS11172 a finales de 1992, la parte de codificación de audio de este estándar (IS-11172-3) describe un sistema de codificación genérico, diseñado para satisfacer las demandas de muchas aplicaciones. El estándar de Audio MPEG-1 consiste de tres modos de operación llamadas "capas", que incrementándose en complejidad y rendimiento se llaman Capa-1, Capa-2 y Capa-3, de las cuales, la Capa-3, con la más alta complejidad, fue diseñada para alcanzar la calidad mas alta de sonido a baja tasa de bits (alrededor de 128 Kbps para una señal estero típica).

1.3.2 MPEG-2

MPEG-2 inicia la segunda fase del estándar MPEG. Introduce muchos conceptos nuevos, dentro de la codificación de video, incluyendo soporte para señales de video entrelazados. La principal área de aplicación para MPEG-2 es la televisión digital. El estándar original MPEG-2 (IS13818-3) fue terminado en 1994 y consistió de dos extensiones de Audio MPEG:

- Codificación de audio multicanal, incluyendo la configuración del canal 5.1 mas conocido como sonido de cinema, esta extensión multicanal es hecha

para permitir compatibilidad con versiones anteriores, permitiendo a los decodificadores estéreo MPEG-1 reproducir una mixtura de todos los canales disponibles.

- Codificación a frecuencias de muestreo mas bajas, esta extensión añade frecuencias de muestreo de 16 Khz, 22.05 Khz a MPEG-1, muestreando frecuencias de 32 Khz, 44,1 Khz y 40 Khz, mejorando la eficiencia de la codificación con menores tasas de bits.

1.3.3 MPEG-2 AAC

A inicios de 1994, las pruebas de verificación mostraron que los nuevos algoritmos de codificación, sin compatibilidad con MPEG-1, prometían una sustancial mejora en la eficiencia de la codificación. Como resultado se definió un nuevo trabajo que finalmente llevó a la definición del nuevo estándar MPEG de codificación de audio, MPEG-2 Codificación avanzada de audio (AAC). El estándar fue terminado en 1997 (IS 13818-7), AAC es un esquema de codificación de audio de segunda generación para codificación de señales estéreo y señales multicanal, soportando frecuencias de muestreo desde 8 Khz a 96 Khz y un número de canales de audio en el rango de 1 a 48.

1.3.4 MPEG-3

Originalmente, MPEG fue planeado para definir la codificación de video para aplicaciones HDTV en una fase futura, para ser llamado MPEG-3. Sin embargo, tiempo después se cambió indicando que las herramientas diseñadas para video MPEG-2 cumplían también los requerimientos de HDTV, y el grupo MPEG concibió

la idea de desarrollar un estándar especial MPEG-3. Algunas veces MPEG Capa 3 ("MP3") es llamado MPEG-3.

1.3.5 MPEG-4

MPEG-4 intenta llegar a ser el mayor estándar en el mundo multimedia. La primera versión fue terminada a finales de 1988 (IS 14496-3), y la segunda versión a finales de 1999, a diferencia de MPEG-1 y MPEG-2, el énfasis en MPEG-4 esta en lograr nuevas funcionalidades que en mejorar la eficiencia de la compresión. Los terminales móviles como los usuarios estacionarios, acceso a base de datos, comunicaciones, y nuevos tipos de servicios interactivos serán las mayores aplicaciones para MPEG-4. El nuevo estándar facilita el crecimiento de la interacción y solapamiento entre los hasta ahora mundos separados de la computación, la masa electrónica (TV y Radio) y las telecomunicaciones. Audio MPEG-4 consiste de una familia de algoritmos de codificación de audio, abarcando un rango desde codificación de voz de baja tasa de bits (debajo de 2 Kbps) hasta codificación de audio de alta calidad a 64 Kbps por canal. La codificación Genérica de Audio desde medias a altas tasas de bits es hecha por AAC.

1.3.6 MPEG-7

A diferencia de MPEG-1, MPEG-2 y MPEG-4, MPEG-7 no define algoritmos de compresión. MPEG-7 contiene información estándar para búsqueda de información multimedia, filtraje, administración y procesamiento.

CAPITULO II

CODIFICACION DE AUDIO MPEG CAPA 3

2.1 Algoritmo de Codificación MPEG Capa 3

En esta sección describiremos el codificador de audio MPEG capa 3 y su funcionalidad. La descripción del proceso de codificación se basa en el diagrama de bloques de la figura 2.1. la señal de entrada, que procede de un canal simple PCM pasa a través de un banco de filtros polifásicos. Este banco de filtros divide la señal de entrada en 32 sub-bandas de frecuencia igualmente espaciadas. Después de este proceso, las muestras de cada sub-banda están aún en el dominio del tiempo. Una transformada discreta coseno modificada (MDCT) es entonces usada para mapear las muestras en cada sub-banda al dominio de la frecuencia. Mientras tanto la señal de entrada después de la transformación FFT pasa a través de un modelo psicoacústico que determina la razón de la energía de la señal al enmascaramiento umbral para cada sub-banda. El bloque de control de distorsión usa la razón de señal a máscara (SMR) del modelo psicoacústico para decidir como asignar el numero total de bits de código disponible para la cuantización de las señales de sub-banda con el objeto de minimizar el ruido de cuantización. Las muestras de sub-banda cuantizada son

codificadas usando la codificación de Huffman para decrementar la entropía de las muestras. Finalmente el último bloque lleva las muestras codificadas de Huffman y la información secundaria en paquetes de acuerdo al estándar de audio MPEG.

En las siguientes secciones, describiremos la operación y funcionalidad en detalle de cada etapa del diagrama de bloques.

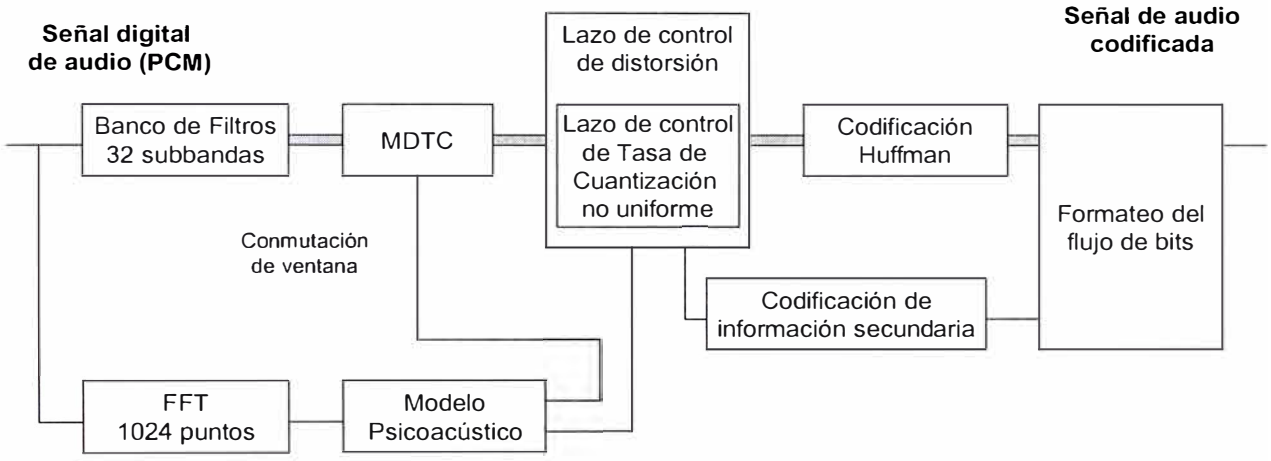

Figura 2.1 Diagrama de bloques del codificador de audio MPEG Capa 3

2.1.1 Análisis del Banco de Filtros Polifásicos

El primer paso en el proceso de codificación es el filtraje de la señal de audio a través del banco de filtros. El análisis del banco de filtros polifásicos, divide la señal de audio en 32 sub-bandas igualmente espaciadas en frecuencia y decima las muestras de sub-banda por un factor de 32 con una buena resolución de tiempo y una razonable resolución de frecuencia. La decimación resulta en un número agregado de sub-bandas muestreadas que son iguales a la señal original pero que sin embargo,

introduce algo de distorsión debido al traslapamiento de sub-bandas adyacentes, este fenómeno es conocido como "aliasing".

En una trama, una secuencia de 1152 muestras de audio PCM son filtradas de tal modo que cada sub-banda contiene 36 muestras de audio. De la siguiente ecuación se deriva la salida del banco de filtros:

$$St[i] = \sum_{k=0}^{63} \sum_{j=0}^{7} M[i][k] * (C[k+64j] * x[k+64j]) \qquad (2.1)$$

Donde:

i es el índice de la sub-banda y varia de 0 a 31.

St[i] es la muestra de salida del filtro por sub-banda *i* en el tiempo *t*, donde *t* es un múltiplo entero de 32 intervalos muestreados de audio.

C[n] es uno de los 512 coeficientes de la ventana de análisis definida en el estándar.

x[n] es una muestra de audio de entrada leída desde un buffer de 512 muestras, y

$$M[i][k] = \cos\left[\frac{(2i+1)(k-16)\pi}{64}\right] \qquad (2.2)$$

son los coeficientes de la matriz de análisis.

Manipulando la ecuación 2.1 en una ecuación de convolución de un filtro (2.3) para un análisis más conveniente:

$$St[i] = \sum_{n=0}^{511} x[t-n] * Hi[n] \qquad (2.3)$$

Donde:

x(t) es una muestra de audio en el tiempo *t*,

$$Hi[n] = h[n] * \cos\left[\frac{(2 * i + 1) * (n - 16) * \pi}{64}\right] \tag{2.4}$$

con:

h[n]= -C[n], si la parte entera de (*n*/64) es impar.

C[n] en otro caso, para *n*=0 a 511.

De esta forma cada sub-banda del banco de filtros, tiene su propia respuesta al filtro pasa-bajo $H_i[n]$. Aunque esta forma es más conveniente para el análisis, no es una solución eficiente: calculando la cantidad de operaciones que se requieren resulta, 32x512=16384 multiplicaciones y 32x511 sumas para calcular las salidas de los 32 filtros.

Los coeficientes de *h[n]* son el prototipo filtro pasa-bajo para el banco de filtros, como se muestra en la figura 2.2. La modulación del filtro prototipo (*h[n]*) con un termino coseno (*M[i][k]*) resulta en un desplazamiento del filtro. *Hi[n]* representa los bancos de filtros que desplazan la respuesta pasa-bajos a la banda de frecuencia apropiada, así, son llamados banco de filtros "polifásicos". Estos filtros tienen frecuencias centrales en múltiplos impares de $\pi/(64T)$ y cada uno tiene un ancho de banda de $\pi/(32T)$ donde *T* es el periodo del muestreo de la señal de audio.

Por ejemplo si el periodo del muestreo *T* es 31.25ms (32Khz de frecuencia de muestreo), la respuesta en frecuencia de los filtros polifásicos tendrán una frecuencia

central de 250 Hz y un ancho de banda de 500 Hz mientras 2π señala la frecuencia del muestreo como indica la figura 2.3.

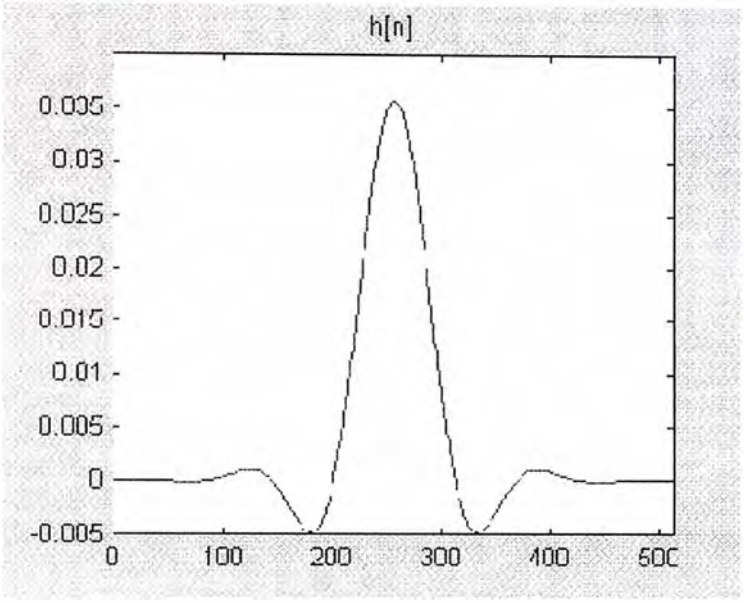

Figura 2.2 H[n], El prototipo Filtro Pasa-Bajo para el Banco de Filtros Polifásicos

En la figura 2.3, el sobrecruce de los filtros polifásicos adyacentes es evidente para compresión de audio, debido a que será introducido el solapamiento o "alias" por este sobrecruce y decimación.

La frecuencia de la señal cerca de las sub-bandas laterales nominales genera salidas en dos filtros polifásicos adyacentes. La figura 2.4 muestra como un tono puro sinusoidal, que tiene frecuencias cerca de las sub-bandas laterales, aparece en la salida de los dos filtros polifásicos. Esta desventaja será cancelada usando series mariposa (butterfly) con un apropiado diseño de un banco de filtros en la parte de codificación / decodificación.

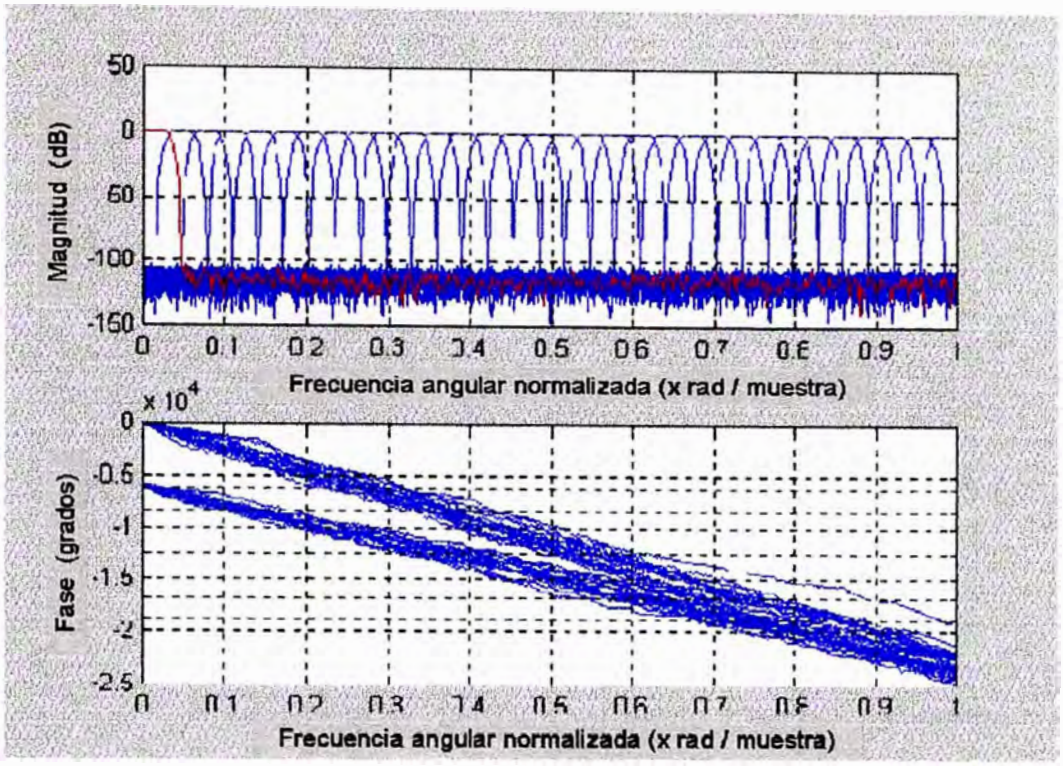

Figura 2.3 Respuesta en Frecuencia del Banco de Filtros Polifásicos

Las muestras de la salida en cada sub-banda aún en el dominio del tiempo, serán procesadas por medio de un bloque MDCT el que transfiere las muestras del dominio del tiempo al dominio de la frecuencia. La figura 2.5 ilustra el análisis del banco de filtros polifásicos y sus detalles.

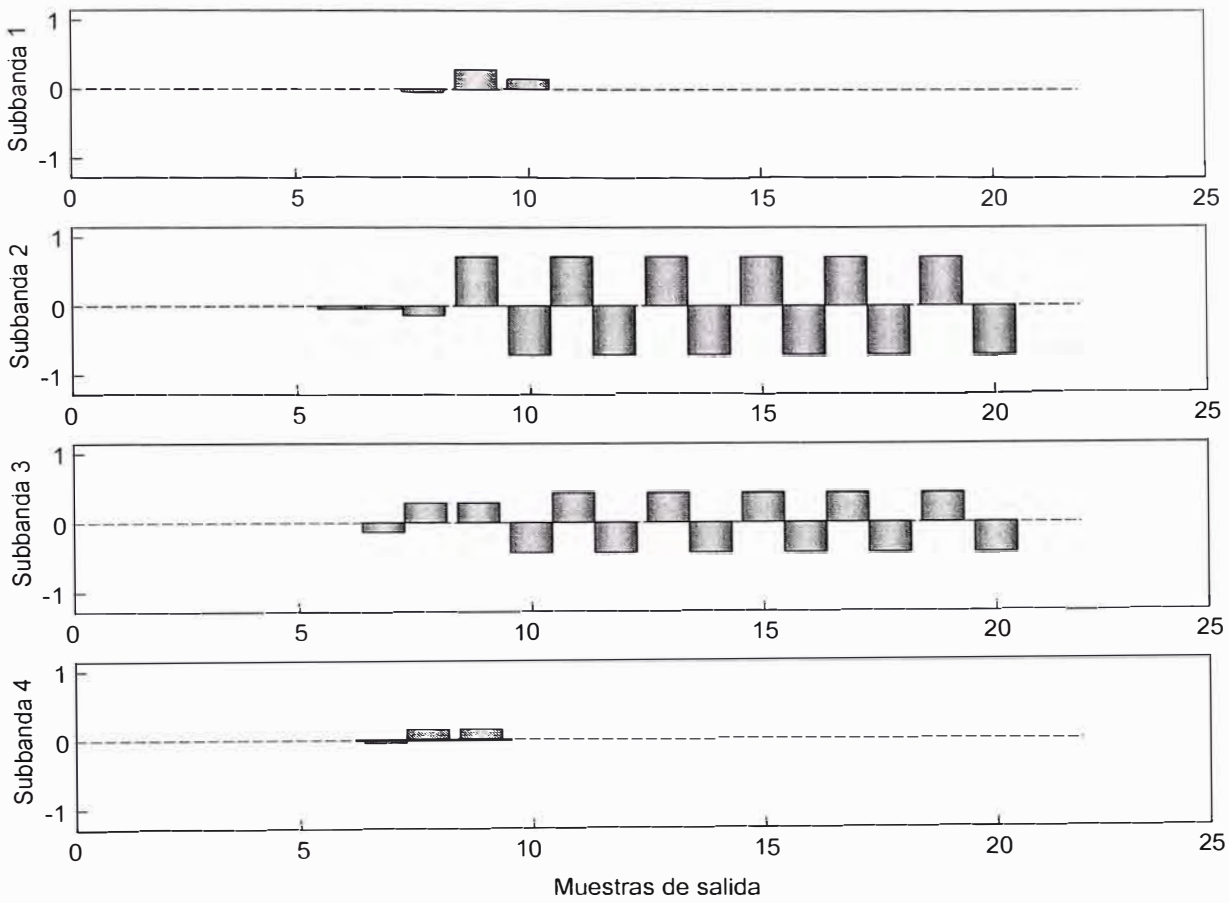

Figura 2.4 Entrada sinusoidal puede producir salidas no cero para dos sub-bandas

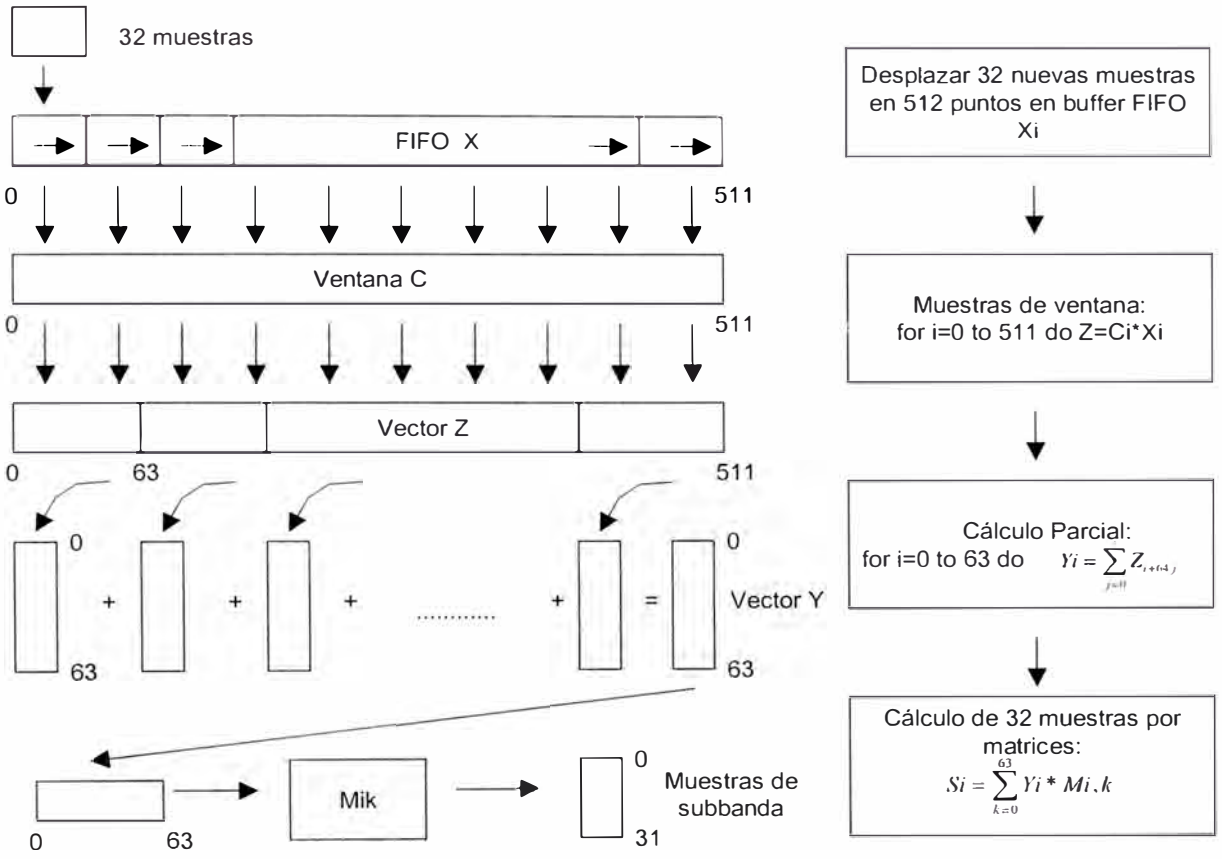

Figura 2.5 Diagrama y procedimiento del análisis del banco de filtros polifásicos

2.1.2 MDCT y Reducción del Solapamiento (Alias)

a. Transformación Discreta Coseno Modificada

En este proceso las 32 sub-bandas son mapeadas en una Transformada Discreta Coseno Modificada. Esta transformación mejorará la resolución de frecuencia por sub-banda. La ecuación 2.5 muestra la formula para la transformación MDCT:

$$X_i = \sum_{k=0}^{n-1} z_k \cos\left(\frac{\pi}{2n}\left(2k+1+\frac{n}{2}\right)(2i+1)\right) \quad \text{, para } i = 0\ldots\frac{n}{2}-1 \qquad (2.5)$$

Antes del cálculo de la MDCT, cuatro funciones de ventana se aplican a las muestras de sub-banda. La capa 3 MPEG/Audio especifica dos longitudes diferentes de bloque MDCT: un bloque largo de 18 muestras o uno corto de 6. El enventanado usa cualquiera de ellos dependiendo de la dinámica en cada sub-banda. Si las muestras sub-banda en una sub-banda determinada muestran un comportamiento estacionario, se usa la ventana larga (tipo 0). Si las muestras de la sub-banda contienen transitorios, se aplica una ventana corta (tipo 2) para subdividir la salidas sub-banda en frecuencia, con el objeto de mejorar la resolución en el tiempo. El mecanismo de conmutación ayuda a prevenir la aparición del fenómeno de pre-eco, (transitorios con elevados errores de cuantización). Las otras dos ventanas usadas para manejar las transiciones de largo a corto o corto a largo son llamadas ventana de arranque (tipo 1) y ventana de parada (tipo 3). Note que la longitud del bloque corto es un tercio del bloque largo. En modo bloque corto, tres bloques cortos reemplazan a un bloque largo tal que el numero de muestras MDCT para una trama de muestras de audio es incambiable respecto a la selección del tamaño del bloque. Para una trama de muestras de audio, el MDCT puede tener la misma longitud de bloque (largo o corto) o tener un bloque en modo mixto. En el modo de bloque mixto el MDCT usa una ventana larga para las dos sub-bandas mas bajas y una ventana corta para las 30 sub-bandas superiores. Este modo proporciona una mejor resolución de frecuencia para las frecuencias más bajas sin sacrificio de la resolución de tiempo para las frecuencias más altas.

Las funciones de las ventanas se explican a continuación y se muestran en la figura 2.6.

a) bloque tipo=0 (ventana larga)

$$z_i = x_i sen\left(\frac{\pi}{36}\left(i+\frac{1}{2}\right)\right) \quad \text{,para } i = 0\ldots35 \tag{2.6}$$

b) bloque tipo=1 (ventana de inicio o arranque)

$$z_i = \begin{cases} x_i sen\left(\frac{\pi}{36}\left(i+\frac{1}{2}\right)\right) & 0\ldots17 \\ x_i & 18\ldots23 \\ x_i sen\left(\frac{\pi}{12}\left(i-18+\frac{1}{2}\right)\right) & 24\ldots29 \\ 0 & 30\ldots35 \end{cases} \text{, Para } i = \tag{2.7}$$

c) bloque tipo =3 (ventana de parada)

$$z_i = \begin{cases} 0 & 0\ldots5 \\ x_i sen\left(\frac{\pi}{12}\left(i-18+\frac{1}{2}\right)\right) & 6\ldots11 \\ x_i & 12\ldots17 \\ x_i sen\left(\frac{\pi}{36}\left(i+\frac{1}{2}\right)\right) & 18\ldots35 \end{cases} \text{, Para } i = \tag{2.8}$$

d) bloque tipo =2 (ventana corta)

$$z_i = x_i sen\left(\frac{\pi}{12}\left(i+\frac{1}{2}\right)\right) \quad \text{,para } i = 0\ldots11 \tag{2.9}$$

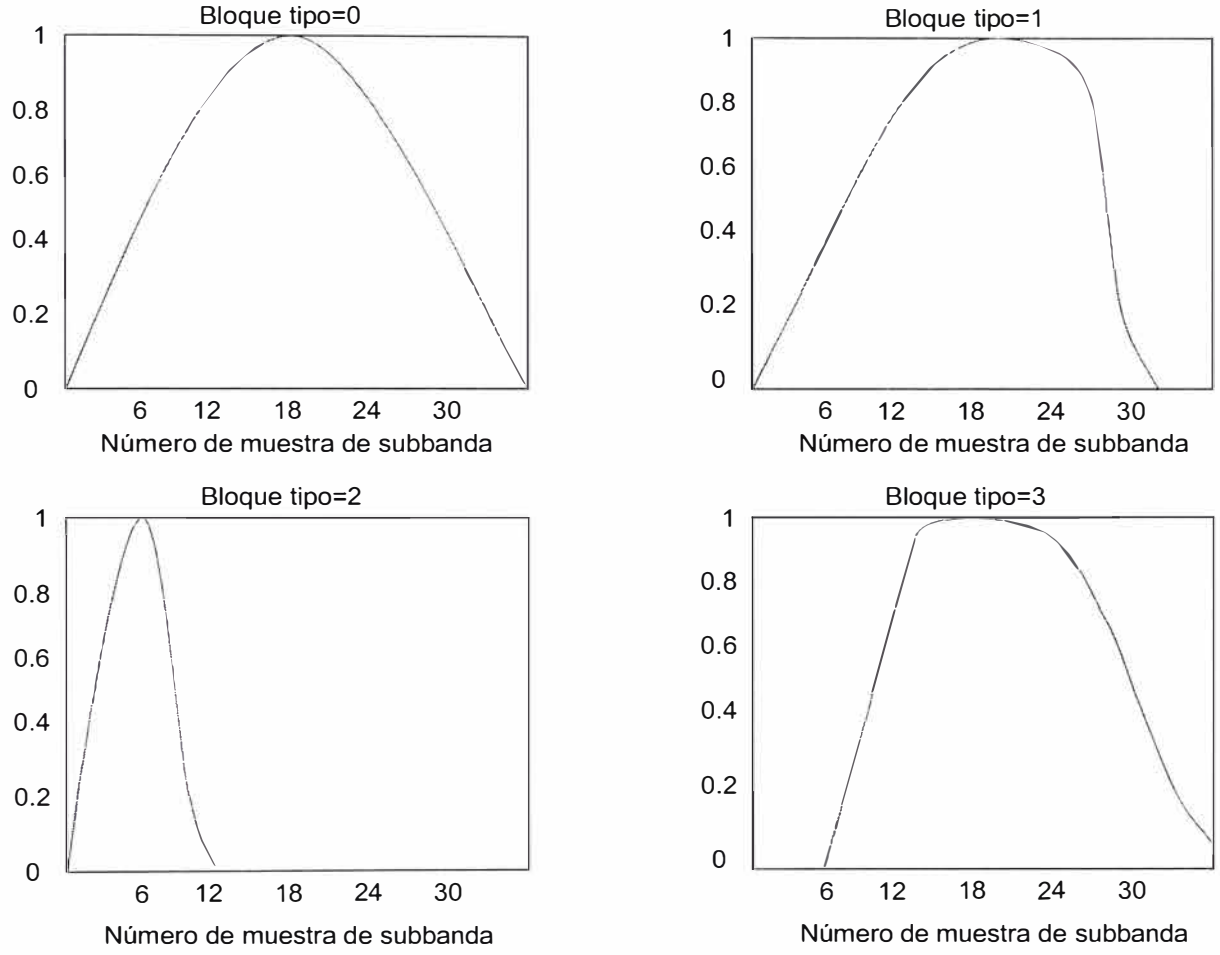

Figura 2.6 Ilustración de los cuatro tipos de ventana aplicables

b. Reducción de Solapamiento (Alias)

Antes de que pasen las líneas de frecuencia, se introduce una reducción del solapamiento o "aliasing" en el análisis del banco de filtros polifásicos. El

solapamiento se remueve en esta primera etapa con el objeto de reducir la cantidad de información a transmitir. La reducción se obtiene por medio del cálculo de una serie mariposa (butterfly), ver figura 2.7. Las constantes cs_i y ca_i son proporcionadas por el estándar internacional. Las operaciones mariposa con pesos apropiados cancelan el solapamiento o "alias" causado por el traslapamiento de dos sub-bandas adyacentes.

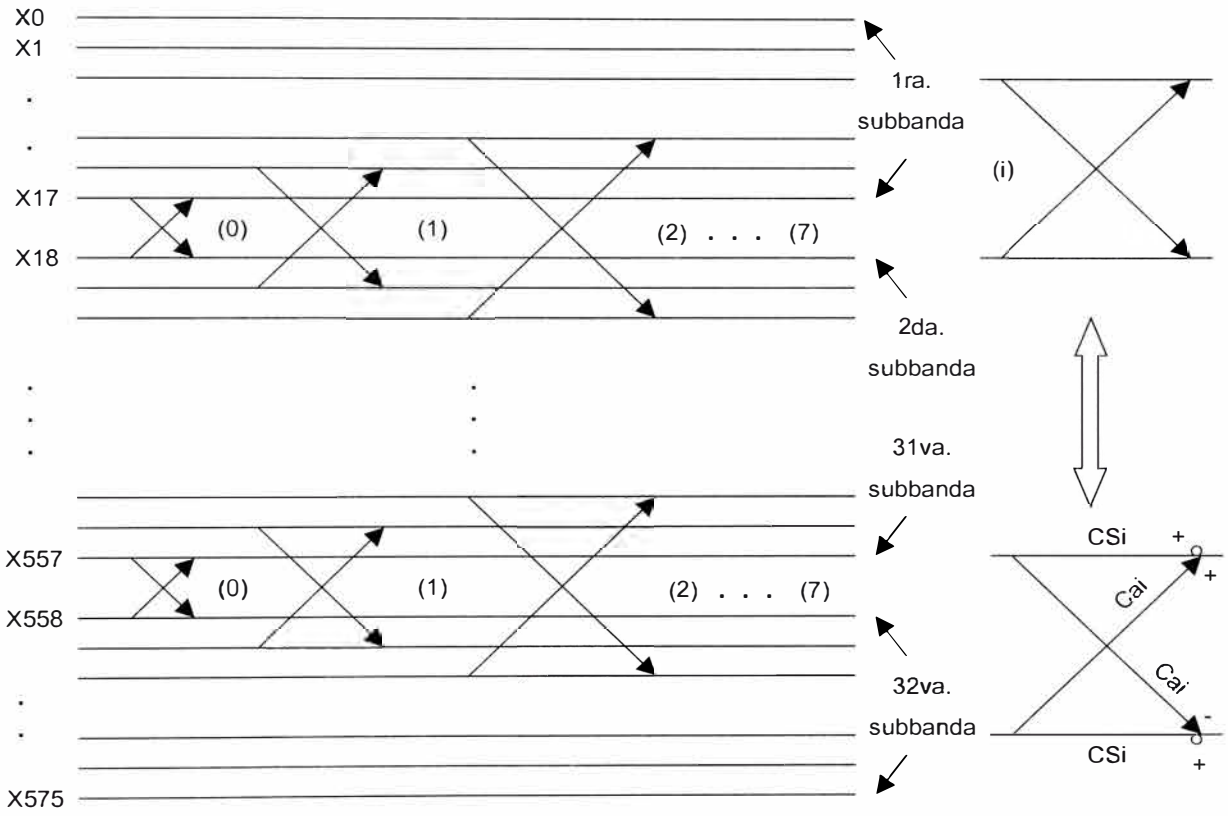

Figura 2.7 Ilustración de reducción del alias mariposa

2.1.3 Modelo Psicoacústico

Es evidente que mientras podemos escuchar un sonido muy silencioso como la caída de una aguja, y fácilmente un sonido ruidoso como la de un aeroplano

despegando, nos es imposible distinguir la caída de una aguja si escuchamos al aeroplano al mismo tiempo. Este fenómeno muestra que nuestro oído se adapta a las variaciones dinámicas del sonido, por lo que no podemos escuchar algunos tonos.

El modelo psicoacústico es un patrón que simula los sonidos percibidos por los humanos. El modelo se usa en el codificador solo para decidir que partes de las señales de audio son acústicamente irrelevantes y que partes no lo son, y remover las partes no audibles. Así, se toma ventaja de la incapacidad del oído humano de escuchar el ruido de cuantización bajo condiciones de enmascaramiento auditivo. Este enmascaramiento es una propiedad perceptual del sistema auditivo humano que ocurre cuando existe la presencia de una señal fuerte de audio en una vecindad temporal o espectral de señales de audio imperceptibles. Los resultados del modelo psicoacústico son usados en el bloque MDCT y en el bloque de cuantización no uniforme. El enmascaramiento auditivo consiste de tres principios básicos:

a. Mínimo umbral auditivo

Este umbral, también conocido como umbral absoluto, corresponde al sonido de intensidad más débil que se puede escuchar en un ambiente silencioso. El mínimo umbral auditivo no tiene un comportamiento lineal; se representa por una curva de Intensidad SPL – Sound Pressure Level (expresado en dB) contra Frecuencia (Hz), que posee niveles mínimos entre 2 y 5 Khz, los cuales corresponden a la parte más sensitiva del oído humano. Por lo tanto, en los sistemas de compresión de audio que sacan provecho de la psicoacústica, no es necesario codificar los sonidos situados bajo este umbral (el área por debajo de la curva), ya que éstos no serán percibidos.

En la figura 2.8, los tonos en las inmediaciones de 3000 Hz requieren menos intensidad para ser escuchados, como resultado, su umbral se expresa como 0 dB y todos los demás valores se expresan con relación a este. El valor absoluto auditivo se llama también umbral en reposo o en silencio. Este umbral auditivo absoluto varía con la frecuencia y cubre un rango dinámico de mas de 60 dB, como se muestra en la figura.

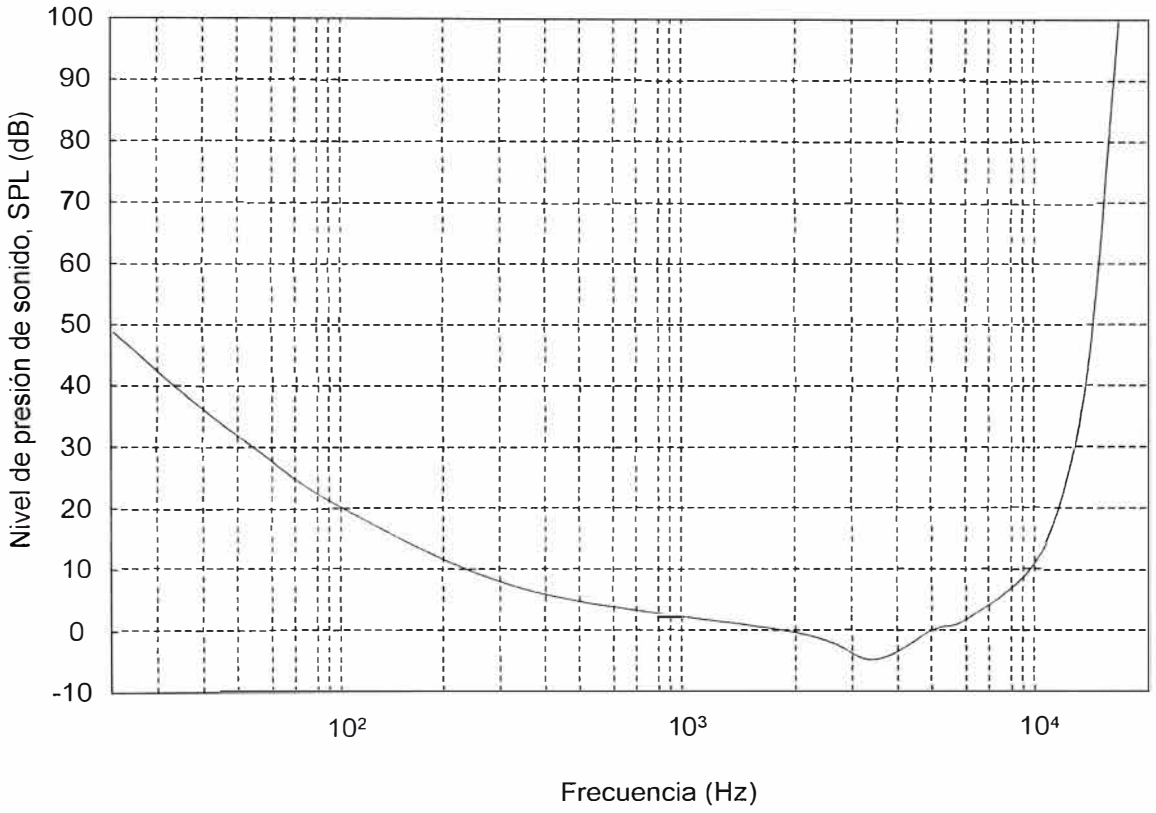

Figura 2.8 El umbral absoluto auditivo

b. Enmascaramiento en frecuencia.

También llamado enmascaramiento simultáneo, funciona de manera que un sonido en determinada frecuencia puede enmascarar o disminuir el nivel de otro sonido en las frecuencias adyacentes, siempre y cuando el nivel del sonido

enmascarante sea más alto (un sonido más intenso, más fuerte) que el nivel del sonido adyacente. Tomando un ejemplo de enmascaramiento umbral para el enmascarador de banda angosta SPL=80dB, en la figura 2.9: el enmascarador es la señal S_o y cualquier energía de señal bajo el borde de este enmascaramiento umbral será enmascarado por la presencia de S_o. Las señales más débiles S_2 y S_3 son completamente inaudibles. Esto es porque los niveles de presión de sus sonidos individuales están todos por debajo del umbral de enmascaramiento. La señal S_1 es parcialmente enmascarada y la porción perceptible de la señal esta sobre la curva de enmascaramiento. Entonces es posible incrementar el ruido de cuantización en la sub-banda que contiene a la señal S_1 hasta el nivel AB, lo que significa que muy pocos bits serán necesarios para representar la señal en esta sub-banda.

Sin ningún enmascarador, una señal puede también ser inaudible, si su intensidad (nivel de presión de sonido) esta por debajo del umbral absoluto. El modelo psicoacústico proporciona al bloque de cuantización no uniforme la información acerca de como cuantificar las líneas de frecuencia. La cuantización de las líneas de frecuencia se adapta a las limitaciones de percepción del oído humano.

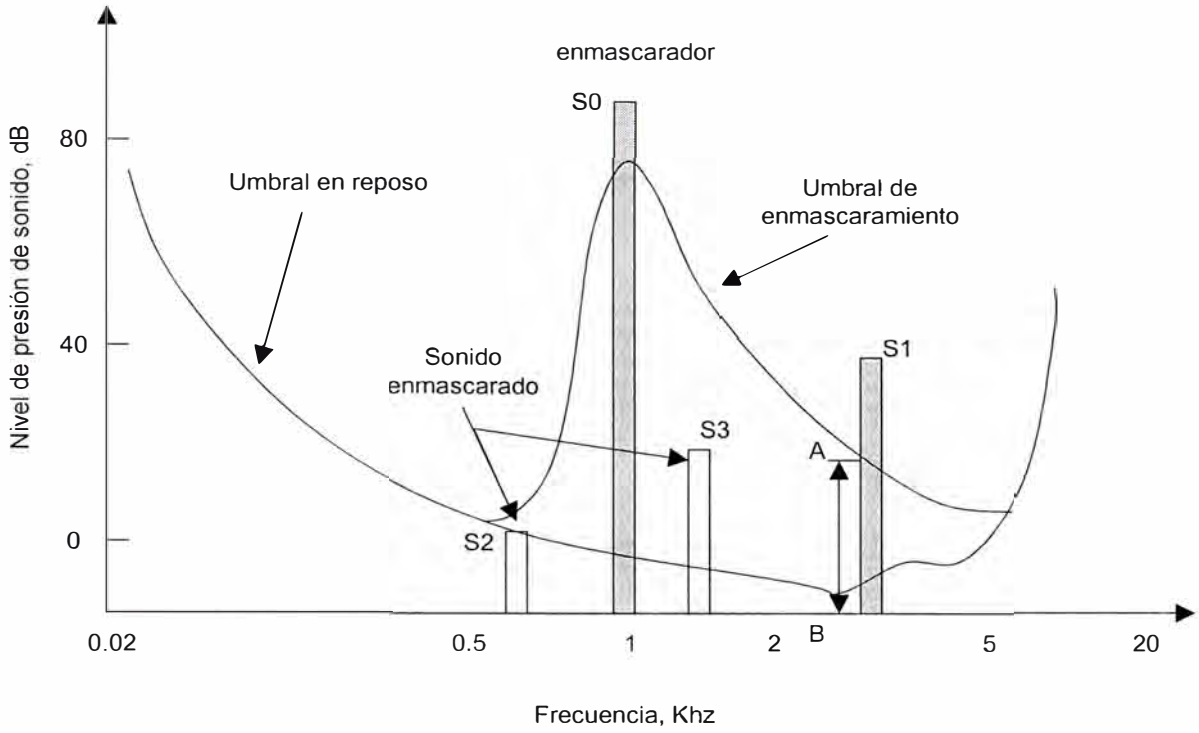

Figura 2.9 Umbral de enmascaramiento de frecuencia y Umbral en reposo

c. Enmascaramiento temporal.

Llamado también, enmascaramiento no simultáneo. Se presenta cuando un tono suave está muy cercano en el dominio del tiempo (unos cuantos milisegundos) a un tono fuerte. Si se está escuchando un tono suave y aparece un tono fuerte, el tono suave será enmascarado por el tono fuerte, antes de que el tono fuerte efectivamente aparezca (pre-enmascaramiento). Posteriormente, cuando el tono fuerte desaparece, el oído necesitará un pequeño intervalo de tiempo (entre 50 y 300 ms) para que se pueda seguir escuchando el tono suave (post-enmascaramiento).

Con el post-enmascaramiento no hay problemas, pero el pre-enmascaramiento sugiere que un tono será enmascarado por otro tono, antes de que el tono enmascarador realmente aparezca, atentando contra el buen juicio de cualquier oyente. Para este fenómeno, se han presentado dos explicaciones:

1. El cerebro integra el sonido sobre un período de tiempo, y procesa la información por ráfagas en la corteza auditiva.
2. Simplemente, el cerebro procesa los sonidos fuertes más rápido que los sonidos suaves.

Sin importar el mecanismo, el caso es que el pre-enmascaramiento temporal en verdad existe, así sea exageradamente pequeño (se ha calculado con un valor aproximado de 30 ms).

En un sonido cualquiera, se presentan ambos tipos de enmascaramiento. El enmascaramiento en frecuencia es mucho más importante que el enmascaramiento temporal; aunque en ciertos dispositivos para compresión de audio se tienen en cuenta ambos tipos de enmascaramiento, con lo cual se logra una mejor compresión de datos. Superponiendo ambas gráficas en una sola que presente tres ejes, se podrá observar una curva bajo la cual están todos los sonidos que no pueden ser escuchados.

Note en la figura 2.10 que el post-enmascaramiento tiene una duración más larga que el pre-enmascaramiento, el post-enmascaramiento continúa por mas de 160 ms.

después del enmascarador, mientras que el pre-enmascaramiento actúa solo 20ms. antes del enmascarador.

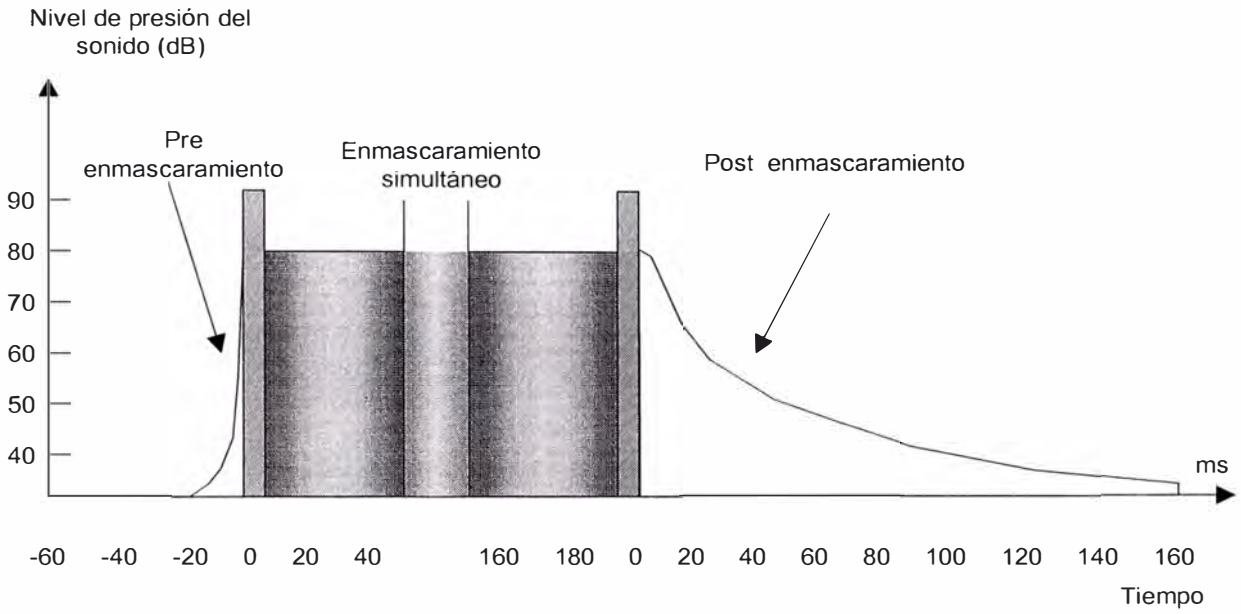

Figura 2.10 Umbral de enmascaramiento temporal

d. Las bandas críticas y el Bark.

Bandas críticas.

Estudios de la discriminación en frecuencia del oído han demostrado que en las bajas frecuencias, tonos con unos cuantos Hertz de separación pueden ser distinguidos; sin embargo, en las altas frecuencias para poder discriminar los tonos se necesita que estén separados por cientos de Hertz. En cualquier caso, el oído responde al estimulo más fuerte que se presente en sus diferentes regiones de frecuencia; a este comportamiento se le da el nombre de bandas críticas. Los estudios muestran que las bandas críticas son mucho más estrechas en las bajas frecuencias que en las altas; el 75% de las bandas críticas están por debajo de los 5 Khz, lo que

implica que el oído recibe más información en las bajas que en las altas frecuencias. Las bandas críticas tienen un ancho de aproximadamente 100 Hz para las frecuencias de 20 a 400 Hz; este ancho aumenta de manera logarítmica a medida que aumenta la frecuencia. Se ha comprobado que el ancho de las bandas críticas se puede aproximar con la fórmula:

$$\textit{Ancho de la banda crítica}\ (Hz) = 24.7\ (4.37F + 1) \qquad (2.10)$$

F es la frecuencia central en Khz.

Las bandas críticas son comparables a un analizador de espectro con frecuencia central variable. Más importante aún es el hecho de que las bandas críticas no son fijas; son continuamente variables en frecuencia y cualquier tono audible creará una banda crítica centrada en él. Apreciado desde otro punto de vista, el concepto de la banda crítica es un fenómeno empírico: una banda crítica es el ancho de banda al cual las respuestas subjetivas cambian abruptamente.

El Bark.

El Bark (en honor al físico alemán George Heinrich Barkhausen) es la unidad de frecuencia perceptual; específicamente, un Bark mide la tasa de banda crítica, o sea, una banda crítica tiene un ancho de un Bark. La escala Bark relaciona la frecuencia absoluta (en Hz) con las frecuencias medidas perceptualmente (el caso de las bandas críticas). Usando el Bark, un sonido en el dominio de la frecuencia puede ser convertido a sonido en el dominio psicoacústico. De esta manera, un tono puro (representado por una componente en el dominio de la frecuencia) puede ser representado como una curva de enmascaramiento psicoacústico. Eberhard Zwicker

modeló el oído con 24 bandas críticas arbitrarias para frecuencias por debajo de 15 Khz, con una banda adicional que ocupa la región entre 15 y 20 Khz. El Bark (ancho de una banda crítica) puede calcularse con las siguientes fórmulas:

$$1 \text{ bark (Hz)} \cong f/100 \qquad \text{para } f < 500 \text{ Hz} \qquad (2.11)$$

$$1 \text{ bark (Hz)} \cong 9+4\log(f/1000) \qquad \text{para } f > 500 \text{ Hz} \qquad (2.12)$$

f = frecuencia.

Para determinar el ancho de una banda crítica, se pueden usar estas fórmulas o la fórmula mostrada anteriormente.

e. Modelo Psicoacústico I

El cálculo del modelo psicoacústico tiene que ser adoptado a la correspondiente capa. Este ejemplo es valido para las capas I y II. El modelo puede ser adoptado a la capa III. No hay diferencias relevantes en la aplicación del modelo psicoacústico I a la capa I o II.

Capa I: Una nueva asignación de bit es calculada para cada bloque de 12 muestras por sub-banda o 384 muestras de entrada PCM.

Capa II: Una nueva asignación de bit es calculada para los tres bloques totalizando 36 muestras por sub-banda correspondientes a 3x384 (1152) muestras de entrada PCM.

El cálculo de la relación de señal a máscara se basa en los siguientes pasos:

Paso 1:

Cálculo de la FFT para conversión de tiempo a frecuencia.

Paso 2:

Determinación de los niveles de presión de sonido en cada sub-banda.

Paso 3:

Determinación del umbral en reposo (umbral absoluto)

Paso 4:

Encontrar las componentes tonales y no tonales de la señal de audio

Paso 5:

Decimación de los enmascaradores, para obtener solo los enmascaradores relevantes.

Paso 6:

Cálculo de los umbrales de enmascaramiento individuales.

Paso 7:

Determinación del umbral de enmascaramiento global.

Paso 8.

Determinación del Umbral de enmascaramiento mínimo en cada sub-banda

Paso 9:

Cálculo de la relación de señal a mascara en cada sub-banda.

Paso 1: Análisis FFT

El umbral de enmascaramiento se deriva de una densidad espectral de potencia estimada, que es calculada por una FFT de 512 puntos para la capa I, o por una FFT de 1024 puntos para las capas II y III. La FFT se calcula directamente de la señal de entrada PCM, procesada previamente por una ventana de Hanning.

Para una coincidencia en tiempo entre la localización de bit y las correspondientes muestras de sub-banda, las muestras PCM a las que se le aplicarán la FFT tienen que ser retrasadas.

1. El retardo del filtro sub-banda es de 256 muestras, correspondiente a 5.3 ms en 48 Khz de tasa de muestreo, lo que corresponde a un desplazamiento de ventana de 256 muestras.
2. La ventana de Hanning debe coincidir con las muestras de sub-banda de la trama, para la capa I esto adiciona un desplazamiento adicional de 64 muestras.

Datos técnicos para la FFT:

	Capa I	Capa II
Longitud de la transformada	512 muestras	1024 muestras
Tamaño de ventana si *fs*=48 Khz.	10.67ms	21.3ms
Tamaño de ventana si *fs*=44.1 Khz.	11.6ms	23.2ms
Tamaño de ventana si *fs*=32 Khz.	16ms	32ms
Resolución de frecuencia	*fs*/512	*fs*/1024

Ventana de Hanning, *h(i)*

$h(i)=0.5*\{1-\cos[2\pi*(i)/(N-1)]\}$, $0<=i<=N-1$ (2.13)

Densidad espectral de potencia *X(k)*:

$$X(k) = 10\log\left|\frac{1}{N}\sum_{l=0}^{N-1} h(l)s(l)e^{-2jkl\pi/N}\right|^2 \quad \text{dB} \quad , \quad k=0...N/2 \qquad (2.14)$$

Una normalización al nivel de referencia SPL de 96dB debe ser hecha de tal manera que el máximo valor corresponda a 96dB.

Paso 2: Determinación del Nivel de presión de sonido

El nivel de presión de sonido (*Lsb*) en cada sub-banda *n* esta definido por:

$$Lsb(n) = MAX[X(k), 20*log(scfmax(n)*32768)-10] \text{ dB} \quad (2.15)$$

X(k) en cada sub-banda *n*

Donde *X(k)* es el nivel de presión de sonido de la línea espectral con índice *k* de la FFT con la máxima amplitud en el rango de frecuencia correspondiente a la sub-banda *n*. La expresión *scfmax(n)* es el factor de escala en la capa I, y en la capa II el máximo de los 3 factores de escala por sub-banda *n* en una trama. Los –10dB corrigen la diferencia entre el nivel pico y el RMS.

Paso 3: Considerando el Umbral en reposo

El Umbral en reposo *LTq(k)*, llamado también Umbral absoluto esta disponible en las tablas "Frecuencias, Tasa de banda crítica y Umbral Absoluto". Estas tablas dependen de la tasa de muestreo de la señal PCM de entrada. Los valores están disponibles para cada muestra en el dominio de la frecuencia donde el umbral de enmascaramiento es calculado.

Un offset que depende de la tasa de bits de transferencia se usa para el umbral absoluto. Este offset es de –12 dB para tasas de transferencia >=96 Kbps y 0 dB para <96 Kbps por canal.

Paso 4: Calculando las componentes tonales y no tonales

La tonalidad de una componente de enmascaramiento tiene influencia sobre el umbral de enmascaramiento. Por esta razón es importante discriminar entre las componentes tonales y no tonales. Para calcular el umbral de enmascaramiento global es necesario derivar las componentes tonales y no tonales del espectro FFT.

El ancho de banda de las bandas críticas varía con la frecuencia central desde casi 0.1 Khz. en las bajas frecuencias a un ancho de banda de 4 Khz. en las frecuencias altas. Para determinar si un máximo local puede ser una componente tonal se examina un rango de frecuencia *df* alrededor del máximo local. El rango de frecuencia *df* esta dado por:

Tasa de muestreo: 32 Khz		
Capa I:	df = 125 Hz	0 kHz $< f <=$ 4.0 Khz
	df = 187.5 Hz	4.0 kHz $< f <=$ 8.0 Khz
	df = 375 Hz	8.0 kHz $< f <=$ 15.0 Khz
Capa II:	df = 62.5 Hz	0 Khz $< f <=$ 3.0 Khz
	df = 93.75 Hz	3.0 Khz $< f <=$ 6.0 Khz
	df = 187.5 Hz	6.0 Khz $< f <=$ 12.0 Khz
	df = 375 Hz	12.0 Khz $< f <=$ 24.0 Khz
Tasa de muestreo: 44.1Khz		
Capa I:	df = 172.266 Hz	0 Khz $< f <=$ 5.512 Khz
	df = 281.25 Hz	5.512 Khz $< f <=$ 11.024 Khz

	df = 562.50 Hz	11.024 Khz < f <= 19.982 Khz
Capa II:	df = 86.133 Hz	0 Khz < f <= 2.756 Khz
	df = 129.199 Hz	2.756 Khz < f <= 5.512 Khz
	df = 258.398 Hz	5.512 Khz < f <= 11.024 Khz
	df = 516.797 Hz	11.024 Khz < f <= 19.982 Khz

Tasa de muestreo: 48 Khz

Capa I:	df = 187.5 Hz	0 Khz < f <= 6.0 Khz
	df = 281.25 Hz	6.0 Khz < f <= 12.0 Khz
	df = 562.50 Hz	12.0 Khz < f <= 24.0 Khz
Capa II:	df = 93.750 Hz	0 Khz < f <= 3.0 Khz
	df = 140.63 Hz	3.0 Khz < f <= 6.0 Khz
	df = 281.25 Hz	6.0 Khz < f <= 12.0 Khz
	df = 562.50 Hz	12.0 Khz < f <= 24.0 Khz

Para calcular las líneas espectrales $X(k)$ tonales y no tonales, se siguen los pasos:

(i) Etiquetando el máximo local

Una línea espectral $X(k)$ es etiquetada como máximo local si:

$$X(k) > X(k-1) \quad \text{y} \quad X(k) >= X(k+1) \tag{2.16}$$

(ii) Listando las componentes tonales y cálculo del nivel de presión de sonido

Un máximo local se coloca en la lista de componentes tonales si:

$$X(k) - X(k+j) >= 7 \text{ dB}, \tag{2.17}$$

Donde j se escoge de acuerdo a:

Capa I:

j = -2, +2	para $2 < k < 63$
j = -3, -2, +2, +3	para $63 <= k < 127$
j = -6,..., -2, +2,..., +6	para $127 <= k <= 250$

Capa II:

j = -2, +2	para $2 < k < 63$
j = -3, -2, +2, +3	para $63 <= k < 127$
j = -6,..., -2, +2,..., +6	para $127 <= k < 255$
j = -12,..., -2, +2,..., +12	para $255 <= k <= 500$

Si $X(k)$ es una componente tonal, los siguientes parámetros son listados:

Indice k de la línea espectral.

Nivel de presión de sonido $Xtm(k)=X(k-1)+X(k)+X(k+1)$, en dB

Flag tonal.

Luego, todas las líneas espectrales en el rango de frecuencia examinado se fijan a -8 dB.

(iii) Listando las componentes no tonales y cálculo de la potencia

Las componentes no tonales (ruido) se calculan del resto de líneas espectrales. Para calcular las componentes no tonales de estas líneas espectrales *X(k)*, las bandas críticas *z(k)* se determinan usando las tablas definidas en el estándar. En la capa I se definen 23 bandas críticas para una tasa de muestreo de 32 Khz, 24 bandas críticas para 44.1Khz y 25 bandas criticas se usan para 48 Khz. En la capa II, 24 bandas criticas son usadas para una tasa de muestreo de 32 Khz, y 26 bandas críticas para 44.1 Khz y 48 Khz. Dentro de cada banda crítica se suman las energías de las líneas espectrales para formar el nivel de presión de sonido de la nueva componente no tonal correspondiente a esa banda crítica.

Los siguientes parámetros son listados:

- Índice *k* de la línea espectral más cercana a la media geométrica de la banda crítica.
- Nivel de presión de sonido *Xnm(k)* en dB.
- Flag no tonal.

Paso 5: Decimación de las Componentes de enmascaramiento tonales y no-tonales

La decimación es un procedimiento que se usa para reducir el número de enmascaradores, los que son considerados para el cálculo del umbral de enmascaramiento global. Se siguen las siguientes consideraciones:

(i) Las componentes tonales *Xtm(k)* o no tonales *Xnm(k)* son consideradas para el cálculo del umbral de enmascaramiento solo si:

$$Xtm(k) >= LTq(k) \quad \text{ó} \quad Xnm(k) >= LTq(k) \tag{2.18}$$

En esta expresión, *LTq(k)* es el umbral absoluto (o umbral en reposo) en la frecuencia de índice *k*. Estos valores se dan en tablas.

(ii) La decimación de dos o más componentes tonales con una distancia de menos de 0.5 Bark: mantener la componente con más alta energía, y remover las componentes más pequeñas de la lista de componentes tonales. Para esta operación se usa un deslizamiento de ventana en el dominio de la banda crítica con un ancho de 0.5 Bark.

El índice *j* se usa para indicar las componentes de enmascaramiento relevantes tonales o no tonales de la lista decimada combinada.

Paso 6: Cálculo de los umbrales de enmascaramiento individuales

Los umbrales de enmascaramiento individuales para las componentes tonales y no tonales están dados por las siguientes expresiones:

$$LTtm[z(j),z(i)] = Xtm[z(j)] + avtm[z(j)] + vf[z(j),z(i)] \text{ dB} \quad (2.19)$$

$$LTnm[z(j),z(i)] = Xnm[z(j)] + avnm[z(j)] + vf[z(j),z(i)] \text{ dB} \quad (2.20)$$

En esta formula *LTtm* y *LTnm* son los umbrales de enmascaramiento individuales en la banda crítica en Bark. Los valores en dB pueden ser positivos o negativos. El término *Xtm[z(j)]* es el nivel de presión de sonido de la componente de enmascaramiento con el índice *j* en la correspondiente banda crítica *z(j)*. El término *av* se llama índice de enmascaramiento y *vf* la función de enmascaramiento de la

componente enmascarante *Xtm[z(j)]*. El índice de enmascaramiento *av* es diferente para los enmascaradores tonales y no tonales (*avtm* y *avnm*).

Para enmascaradores tonales:

$$avtm = -1.525 - 0.275 * z(j) - 4.5 \text{ dB} \quad (2.21)$$

y, para enmascaradores no tonales:

$$avnm = -1.525 - 0.175 * z(j) - 0.5 \text{ dB} \quad (2.22)$$

La función de enmascaramiento *vf* de un enmascarador esta caracterizado por los declives mas altos y más bajos, lo que depende de la distancia en Bark $dz = z(i) - z(j)$ al enmascarador. En esta expresión *i* es el índice de la línea espectral en la cual la función de enmascaramiento es calculada y *j* corresponde al enmascarador. La función de enmascaramiento es la misma para las componentes tonales y no tonales y esta dada por:

$$\begin{aligned}
vf &= 17 * (dz + 1) - (0.4 * X[z(j)] + 6) \text{ dB} && \text{para } -3 <= dz < -1 \text{ Bark} \\
vf &= (0.4 * X[z(j)] + 6) * dz \text{ dB} && \text{para } -1 <= dz < 0 \text{ Bark} \\
vf &= -17 * dz \text{ dB} && \text{para } 0 <= dz < 1 \text{ Bark} \\
vf &= -(dz - 1) * (17 - 0.15 * X[z(j)]) - 17 \text{ dB} && \text{para } 1 <= dz < 8 \text{ Bark}
\end{aligned} \quad (2.23)$$

En estas expresiones *X[z(j)]* es el nivel de presión del sonido de la j-ésima componente de enmascaramiento en dB.

Si $dz < -3$ Bark, ó $dz >= 8$ Bark, el enmascaramiento no es considerado (*LTtm* y *LTnm* son fijados a –8dB fuera de este rango)

Paso 7: Cálculo del Umbral de enmascaramiento global LTg

El umbral de enmascaramiento global *LTg(i)* en la i-ésima muestra de frecuencia se deriva de los valores mas altos y bajos de los umbrales de enmascaramiento individuales de cada uno de los enmascaradores tonales y no tonales, y en adición del umbral en reposo *LTq(i)*. El umbral de enmascaramiento global se calcula sumando las correspondientes energías a los umbrales de enmascaramiento individual y al umbral absoluto.

$$LTg(i) = 10\log\left[10^{LTq(i)/10} + \sum_{j=1}^{m} 10^{LTtm(j,i)/10} + \sum_{j=1}^{n} 10^{LTnm(j,i)/10}\right] \qquad (2.24)$$

El número total de enmascaradores tonales esta dado por *m*, y el número total de enmascaradores no tonales esta dado por *n*. Para un *i* dado, el rango de *j* puede ser reducido para solo abarcar estas componentes de enmascaramiento que están entre –8 y +3 Bark desde *i*. Fuera de este rango *LTtm* y *LTnm* son –8dB.

Paso 8: Determinación del umbral de enmascaramiento mínimo

El mínimo nivel de enmascaramiento *LTmin(n)* en la sub-banda *n* se determina por la siguiente expresión:

$$LTmin(n) = MIN[LTg(i)] \text{ dB} \qquad (2.25)$$

f(i) en sub-banda *n*

Donde *f(i)* es la frecuencia de la i-ésima muestra de frecuencia. Los *f(i)* son calculados en tablas.

Paso 9: Cálculo de la relación de señal a máscara

La relación de señal a mascara, es calculada para cada sub-banda.

$$SMRsb(n) = Lsb(n)\text{-}LTmin(n) \text{ dB} \quad (2.26)$$

f. Modelo Psicoacústico II

En el modelo psicoacústico II el proceso de generación del umbral tiene tres entradas:

1. Un desplazamiento para el proceso del cálculo umbral, *iblen*, donde 384<*iblen*<640. Este *iblen* debe permanecer constante sobre cualquier examen particular del proceso del cálculo umbral.
2. Las nuevas muestras *iblen* de la señal, con las muestras retrasadas (en el banco de filtros o en el cálculo psicoacústico) tal que la ventana del cálculo psicoacústico este centrada en la ventana de tiempo de la aplicación.
3. La tasa de muestreo. Hay un conjunto de tablas proporcionadas para los estándares de muestreo. Las tasas de muestreo como *iblen*, deben necesariamente permanecer constante sobre la implementación del proceso de cálculo del umbral.

En la capa 2, las tasas de enmascaramiento psicoacústico deben ser calculadas dos veces durante cada proceso de codificación de trama.

Comentarios de la notación.

w : indica que el cálculo es indexado por frecuencia en el dominio espectral FFT, un índice 1 corresponde al termino DC y un índice de 513 a una línea espectral en la frecuencia de Nyquist.

b : indica que el cálculo es indexado en el dominio de la partición del cálculo umbral, en el caso en el que el cálculo incluye una convolución o suma *bb* se usará como una variable de suma. La numeración de la partición empieza en 1.

n : indica que el cálculo es indexado en el dominio del bit codificador. Un índice de 1 corresponde a la banda más baja en el banco de filtro sub-banda.

La función de dispersión

Es calculado por el siguiente método:

$$tmpx = 1.05\,(j-i\,), \tag{2.27}$$

Donde *i* es el valor Bark de la señal que es dispersada, *j* es el valor Bark de la banda que es dispersada, y *tmpx* es una variable temporal.

$$x = 8\ minimum\ ((tmpx-0.5)^2-2(tmpx-0.5),0) \tag{2.28}$$

Donde *x* es una variable temporal, y *minimum (a,b)* es una función que devuelve los valores más negativos de a ó b.

$$tmpy = 15.811389+7.5(tmpx+0.474)-17.5(1.0+(tmpx+0.474)^2)^{0.5} \tag{2.29}$$

Donde *tmpy* es otra variable temporal.

Si ($tmpy$<-100) entonces {$sprdngf\ (i,j)$=0} en caso contrario {$sprdngf(i,j)=10^{(x+tmpy)/10}$ }

Pasos en el cálculo Umbral

Los siguientes pasos son necesarios para calcular el SMRn usado en el codificador:

a. Reconstruir las 1024 muestras de la señal de entrada.

b. Calcular el espectro complejo de la señal de entrada.

c. Calcular y predecir r y f

$$\hat{r}w = 2.0rw(t-1) - rw(t-2) \tag{2.30}$$

$$\hat{f}w = 2.0fw(t-1) - fw(t-2) \tag{2.31}$$

donde t representa el número de bloque actual, t-1 indexa los datos del bloque previo, y t-2 indexa los datos del bloque de cálculo umbral anterior a ese.

d. Calcular la medida no predecible cw

$$cw = (((rw\cos fw - \hat{r}w\cos \hat{f}w)^2 + \ldots$$

$$\ldots (rw\sin fw - \hat{r}w\sin \hat{f}w)^2)^{0.5}) / (rw + abs(\hat{r}w)) \tag{2.32}$$

e. Calcular la energía en las particiones de cálculo umbral.

La energía en cada partición eb es:

$$eb = \sum_{w=wlowb}^{whighb} rw^2 \tag{2.33}$$

y el valor no predecible, cb, es:

$$cb = \sum_{w=wlowb}^{whighb} rw^2\, cw \tag{2.34}$$

f. combinar la energía particionada y el valor no predecible con la función de dispersión.

$$ecbb = \sum_{bb=1}^{bmax} ebb * sprdngf(bvalbb, bvalb) \tag{2.35}$$

$$cth = \sum_{bb=l}^{bmax} cbb * sprdngf(bvalbb, bvalb) \quad (2.36)$$

Puesto que *cth*. Es asignado por la señal de energía, debe ser renormalizado a *cbb*.

$$cbb = cth/ecb \quad (2.37)$$

Al mismo tiempo debido a la naturaleza no normalizada de la función de dispersión, *ecbb* debería ser renormalizada y la energía normalizada *enb*, calculada.

$$enb = ecbb * rnormb \quad (2.38)$$

El coeficiente de normalización es:

$$rnormb = 1/(\sum_{bb=0}^{bmax} sprdngf(bvalbb, bvalb)) \quad (2.39)$$

g. Convertir *cbb* a *tbb*

$$tbb = -0.299 - 0.43 \log_e(cbb) \quad (2.40)$$

Cada *tbb* es limitado al rango de 0 *tbb* 1.

h. Calcular el SNR en cada partición

NMTb = 5.5dB para todo *b*. *NMTb* es el valor para el tono de enmascaramiento de ruido (en dB) para la partición. La seña requerida para la tasa de ruido, *SNRb*, es:

$$SNRb = \text{maximum}(minvalb,\ tbb * TMNb + (1 - tbb) NMTb) \quad (2.41)$$

Donde *maximum* (a,b) es una función que devuelve el valor menos negativo de a ó b.

i. Calcular la tasa de potencia

La tasa de energía, *bcb*, es:

$$bcb = 10^{-SNRb/10} \quad (2.42)$$

j. Cálculo del umbral de energía actual.

$$nbb = enb \; bcb \tag{2.43}$$

k. Dispersión de la energía umbral sobre las líneas FFT, produciendo *nbw*

$$nbw = nbb / (whighb - wlowb + 1) \tag{2.44}$$

l. Incluir los umbrales absolutos, lo que produce el umbral de energía final, *thrw*.

$$thrw = \max(nbw, absthrw) \tag{2.45}$$

m. Control de pre-eco

n. Calcular las tasas de señal a máscara, SMRn

Las tablas del estándar muestran:

1. El índice, *n*, de la partición del codificador.
2. El índice mas bajo *wlown*, de la partición del codificador.
3. El índice mas alto, *whighn* de la partición del codificador.
4. El ancho de índice, *widthn*, donde *widthn*=1 para una banda de factor de escala psicoacústicamente angosta, y *widthn*=0 para una banda de factor de escala psicoacústicamente ancha. Una banda de factor de escala psicoacústicamente angosta es una cuyo ancho es menor que aproximadamente 1/3 de la banda crítica.

La energía en la banda del factor de escala *epartn es:*

$$epartn = \sum_{w=wlown}^{whighn} rw^2 \tag{2.46}$$

Entonces, si (*widthn* = 1), el nivel de ruido en la banda del factor de escala, *npartn* se calcula como:

$$npartn = \sum_{w=wlown}^{whighn} thrw \tag{2.47}$$

en caso contrario,

$$npartn = \text{minimum}(thr_{wlown}, \ldots, thr_{whighn}) * (whighn - wlown + 1) \quad (2.48)$$

Donde, en este caso, *minimum* (a,...,z) es una función que devuelve el argumento positivo más pequeño de los argumentos a...z.

Los ratios que serán enviados al codificador, *SMRn*, son calculados como:

$$SMRn = 10 \log_{10} (epart_n / npart_n) \quad (2.49)$$

2.1.4 Cuantización No Uniforme

El bloque de cuantización no uniforme, que recibe la línea de frecuencia del bloque MDCT y de la ventana de conmutación, enmascara la información del modelo psicoacústico, ejecutando las técnicas de cuantización y codificación Huffman. Por este bloque salen los datos codificados que el sistema auditivo humano puede escuchar y su correspondiente información secundaria.

El lazo (loop) de cuantización no uniforme es la parte que consume mas tiempo en los algoritmos de codificación de audio MPEG capa 3. Ello, depende de la variación de la señal de audio y no tiene un tiempo determinado de ejecución, las partes de la señal que tienen mayores variaciones necesitarán mas tiempo de codificación.

La descripción del módulo del lazo de Capa 3 se subdivide en tres niveles: El nivel principal o "Programa de lazos de trama", que llama a una subrutina denominada "Lazo de iteración exterior" la cual invoca a la subrutina "Lazo de iteración interior".

El modulo de loop se muestra en la figura 2.11, cuantiza un vector de datos de entrada de líneas espectrales en un proceso iterativo de acuerdo a varias solicitudes. El lazo interior cuantiza los datos de entrada e incrementa el tamaño de los pasos del cuantizador hasta que los datos de salida puedan ser codificados con la cantidad disponible de bits. Después de completada la labor del lazo interior, un lazo exterior verifica la distorsión de cada banda del factor de escala, si la distorsión permitida se excede, se amplifica el factor de escala y se llama nuevamente al lazo interior.

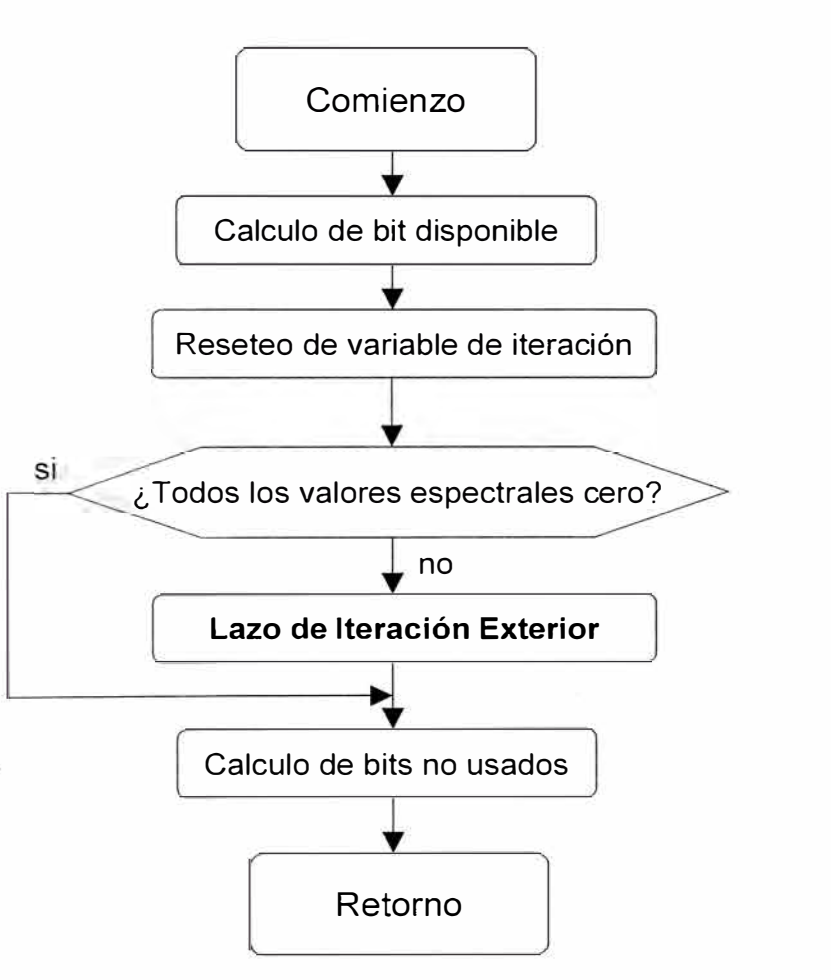

Figura 2.11 Programa de lazo de trama de Audio / MPEG Capa 3

Lazo de Iteración Exterior (lazo de control de distorsión)

El lazo de iteración exterior controla el ruido de cuantización que se produce por la cuantización de las líneas en el dominio de la frecuencia en el lazo de iteración interior. La coloración del ruido es realizada por la multiplicación de las líneas en las bandas del factor de escala con los factores de escala actuales antes de hacer la cuantización. Si el ruido de cuantización excede el umbral de enmascaramiento, el factor de escala para esta banda es ajustada para reducir el ruido de cuantización. El lazo exterior se ejecuta hasta que el ruido actual este por debajo del umbral de enmascaramiento para cada banda del factor de escala. La figura 2.12 muestra el diagrama de flujo del lazo exterior.

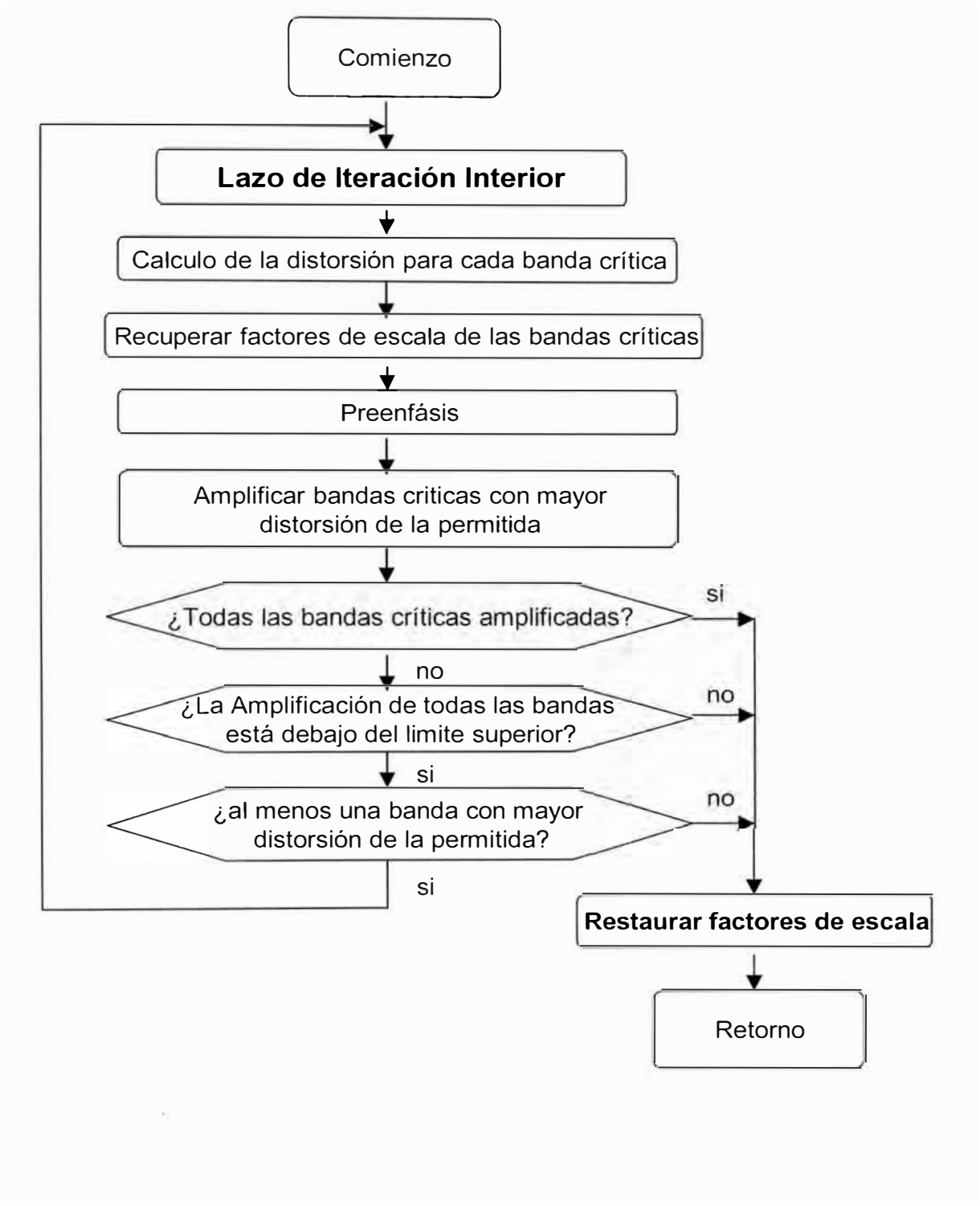

Figura 2.12 Lazo de Iteración Exterior Audio / MPEG Capa 3

Lazo de Iteración Interior (lazo de control de tasa de bits)

El lazo de Iteración Interior realiza la cuantización actual de los datos en el dominio de la frecuencia y prepara la operación de formateo. Las tablas de código Huffman asignan códigos de palabras cortas para pequeños valores cuantizados. Si el número total de bits resultante de la operación de codificación Huffman, excede al número de bits disponibles para codificar una trama, entonces puede ser corregido ajustando la ganancia global que resulta en un tamaño de paso de cuantización más grande, dirigiendo a un valor cuantizado más pequeño. Esta operación se repite con diferentes tamaños de pasos de cuantización hasta que el numero de bits resultante requeridos por la codificación Huffman sea bastante pequeña. La figura 2.13 muestra el diagrama de flujo del lazo interior.

Excepto por el escalamiento, la cuantización y codificación Huffman, la selección de la tabla Huffman, subdivisión del rango *big_value* en sub-regiones y la selección del paso cuantizador también tiene lugar en esta etapa.

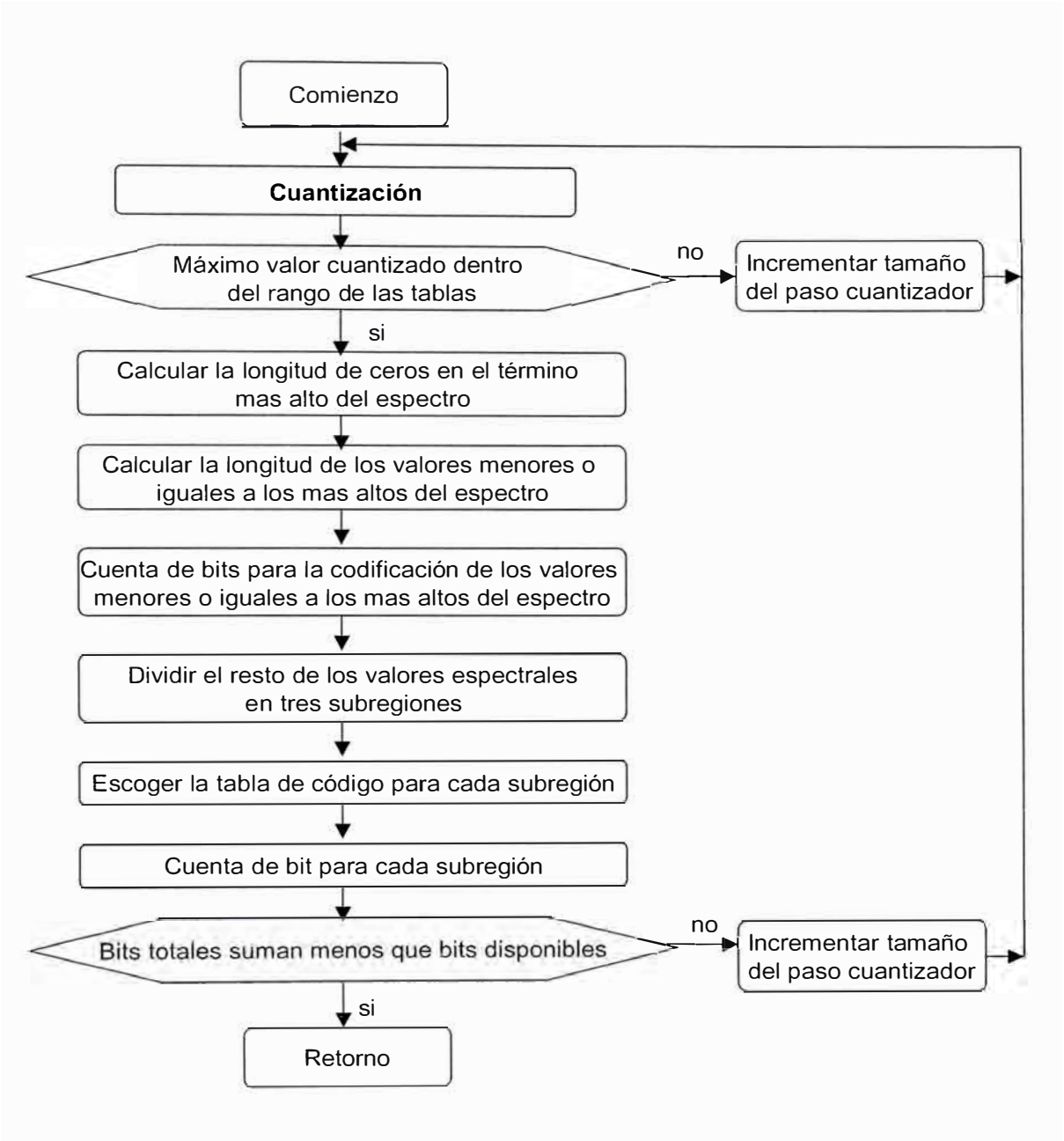

Figura 2.13 Lazo de Iteración Interior Audio / MPEG Capa 3

2.1.5 Codificación Huffman

En este bloque una codificación de las líneas de frecuencia cuantizada se logra usando el algoritmo de codificación de una de las 32 tablas estáticas de Huffman. La

codificación Huffman proporciona pérdidas de compresión muy bajas y por lo tanto reduce la cantidad de datos a ser transmitidos sin pérdida de calidad de la señal.

La técnica más popular para eliminar la redundancia de la codificación es la codificación Huffman. En este tipo de codificación la entropía se basa en una distribución estática del grupo de valores de datos. Desde los datos estáticos se establece una tabla de sustitución que cubre todos los valores de datos. En esta tabla, los valores con una alta probabilidad de estar presentes se asocian con palabras de códigos cortos y los datos menos probables se asocian con códigos más largos. Por lo tanto la codificación Huffman es una codificación de longitud variable (VLC).

La capa 3 de audio MPEG delimita las líneas de frecuencia en tres secciones y adopta un valor de ESCAPE en una de las tres secciones en el proceso de codificación por dos razones:

a. Con el objeto de incrementar la frecuencia excepto para el bloque en modo corto MDCT. Para el bloque corto hay tres conjuntos de valores de ventana en una sub-banda, así el ordenamiento es por frecuencia, luego por ventana, y finalmente por factor de escala. Este orden resulta ventajoso puesto que los valores grandes tenderán a estar en las frecuencias mas bajas y los largos que van de cero o casi cero tenderán a estar en las frecuencias más altas.
b. Cuando un número grande de símbolos esta por ser codificado, la construcción del código óptimo de Huffman es una tarea no trivial.

Con el beneficio de la primera razón, el codificador delimita las líneas de frecuencia en tres regiones distintas. Esto habilita al codificador para codificar cada región con un conjunto diferente de tablas de Huffman. Las tres regiones se llaman "*rzero*", "*count1_region*" y "*big_value_region*".

Comenzando en las frecuencias mas altas, el codificador identifica los flujos continuos de todos los valores cero como una región, "*rzero*". Esta región no tiene que ser codificada porque su tamaño puede ser deducido del tamaño de las otras dos regiones. Sin embargo, podría contener un número par de ceros porque las otras regiones codifican sus valores en agrupamientos pares.

Una segunda región, "*count1_región*", comprende un flujo continuo de valores consistentes de –1, 0 ó 1. En esta región, dos tablas de Huffman, codifican 4 valores a la vez, así el numero de valores en esta región puede ser múltiplo de 4.

Finalmente, una tercera región, que cubre el resto de los valores, se llama "*big values_region*". Para esta región, 30 tablas de Huffman codifican los valores en pares, esta región es además subdividida en tres sub-regiones, tal que cada una tiene su propia tabla de Huffman. En la región "*big_values_region*", debido a la desventaja de la segunda razón, un valor "ESCAPE" se introduce con el objeto de mejorar la eficiencia de la codificación antes de codificar las líneas de frecuencia. En esta región los valores que exceden 15 se representan con el número 15 y el resto con el valor ESCAPE. Dependiendo del tamaño del valor ESCAPE un número de bits,

llamados *linbits*, se asignan para representar este valor de acuerdo a la siguiente ecuación:

Valor ESCAPE = valor cuantizado – 15, si el valor cuantizado >15 (2.50)

Linbits = longitud de palabra (valor ESCAPE) (2.51)

La figura 2.14 muestra la relación de las tres regiones codificadas de Huffman y el factor de escala. La tabla 2.1 lista las características de las 32 tablas de Huffman del estándar de audio MPEG capa 3.

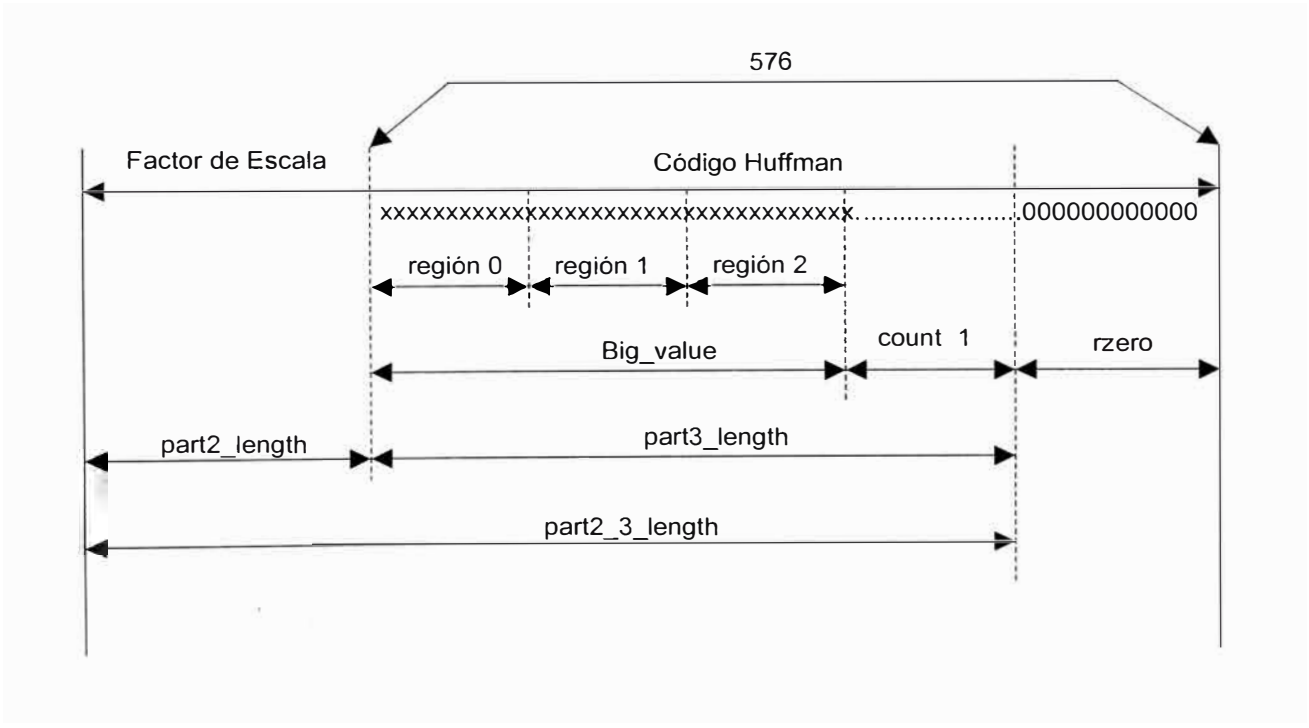

Figura 2.14 Organización de datos principales

Indice Tabla	Valor Max.	Indice Tabla	Valor Max.	Linbits	Valor max.*	Region usada
A	1	B	1	No		count_1
0	0	16	15	1	16	
1	1	17	15	2	19	
2	2	18	15	3	23	
3	2	19	15	4	31	
4	no usada	20	15	6	79	
5	3	21	15	8	271	
6	3	22	15	10	1039	
7	5	23	15	13	8207	Valor grande
8	5	24	15	4	31	
9	5	25	15	5	47	
10	7	26	15	6	79	
11	7	27	15	7	143	
12	7	28	15	8	271	
13	15	29	15	9	527	
14	no usada	30	15	11	2016	
15	15	31	15	13	8207	

* significa la adición del valor máximo y valor ESCAPE

Tabla 2.1 Características de las 32 tablas de Huffman

2.1.6 Formación del Flujo de Bits

El último bloque del proceso de codificación produce el flujo de bits de audio MPEG capa 3. Huffman codifica las líneas de frecuencia, la información secundaria y la cabecera de la trama y los ensambla para formar el flujo de bits. El flujo de bits es particionado en tramas cada una de las cuales representan 1152 muestras de audio. La cabecera indica cual es la tasa de bits y la frecuencia de muestreo que es usada para la codificación de audio. La información secundaria indica el tipo de bloque, tablas Huffman, ganancia de sub-banda y factores de sub-banda que son seleccionadas.

Bit Reservorio

En este bloque se usa un método llamado "bit reservorio" para rellenar las variaciones de tiempo del codificador en código de bits. El codificador puede donar bits al reservorio cuando necesite menos del número promedio de bits para codificar una trama. Luego, cuando el codificador necesite mas del número promedio de bits para codificar una trama, puede pedir prestado bits del mecanismo de reservorio. El codificador solo puede pedir prestado bits donados por tramas anteriores, no puede pedir prestado de futuras tramas. El flujo de bits de audio MPEG capa 3 usa un puntero de 9 bits, llamado *main_data_begin* en cada trama de información secundaria que apunta a la dirección del primer byte de arranque de audio para esa trama. Un ejemplo de cómo los datos principales son distribuidos se muestra en la figura 2.15.

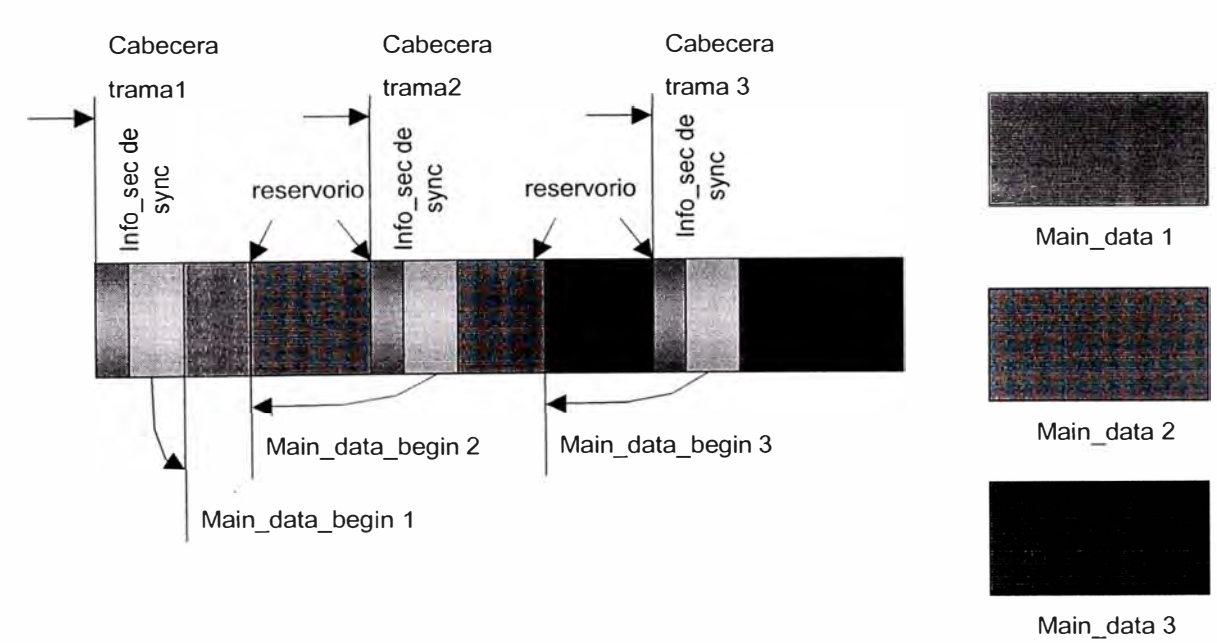

Figura 2.15 Ejemplo de bit reservorio

Las tramas de audio

El audio en un flujo MPEG-1 está organizado de tal manera que cada fragmento del audio codificado (llamado trama) sea decodificable por sí mismo, con una posible excepción para la Capa III. La trama está constituida por las muestras de audio y por la información secundaria. Esta última sirve de control, además de proporcionar información del archivo. Para la Capa III, la trama está constituida por:

1 trama = 1152 muestras de audio + información de la trama

A la salida del banco de filtros polifásico, las muestras de audio se dividen por sub-bandas de la manera mostrada en la figura 2.16. Como se observa, cada sub-banda aporta 12 muestras para un total de 384 muestras de audio, en la Capa I; mientras que para la Capa III, cada sub-banda aporta 36 muestras de audio para un total de 1152 muestras sub-banda por trama.

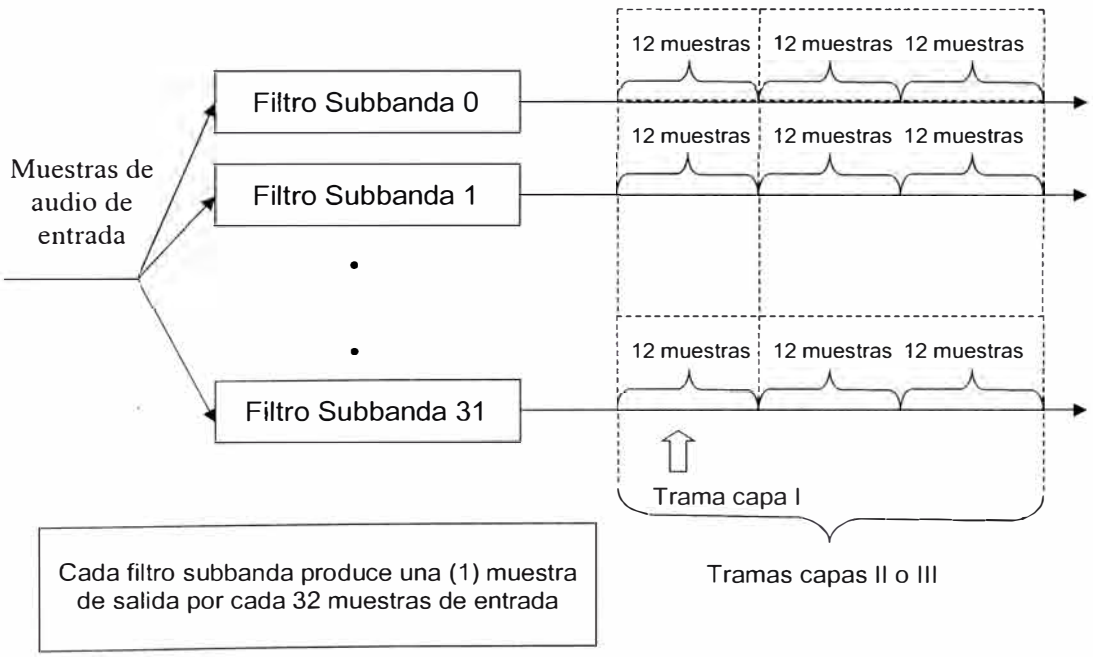

Figura 2.16 Formación de las tramas de audio

Una trama es un bloque de datos con su propio encabezado e información de audio. En el caso de las Capas I ó II, las tramas son elementos totalmente independientes, así que se puede extraer cualquier fragmento de datos del archivo MPEG y decodificarlo correctamente. Sin embargo, en el caso de la Capa III, las tramas no son siempre totalmente independientes, debido al posible uso del *bit reservorio*, que es una especie de búfer, las tramas son a menudo dependientes unas de otras. En el peor caso, se pueden necesitar hasta nueve tramas antes de poder realizar la decodificación de una sola. En la figura 2.17 se muestra una idea general de este proceso.

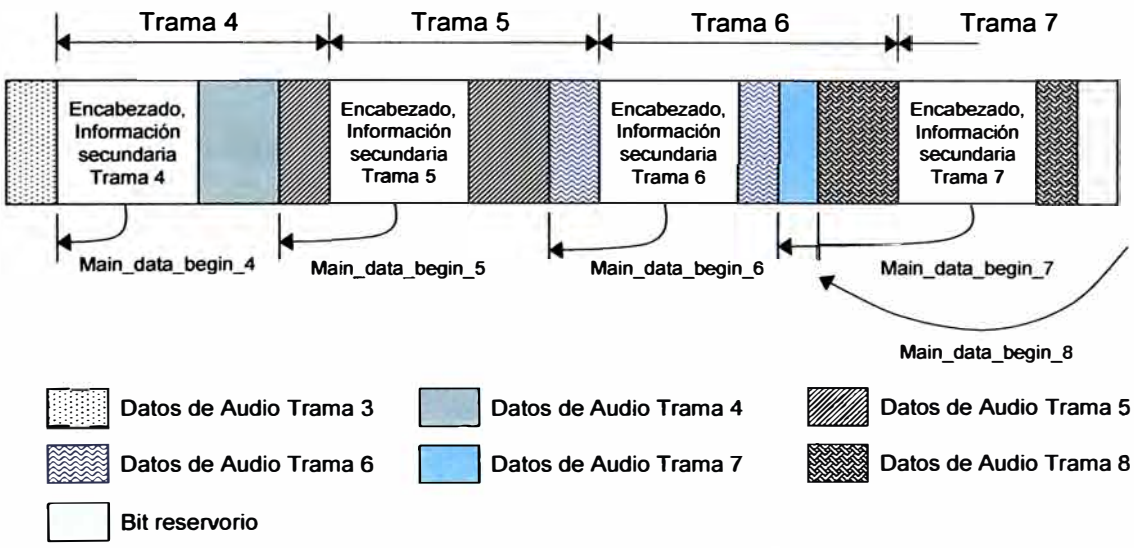

Figura 2.17 Las tramas de audio

Main_data_begin es un puntero de ajuste negativo, incluido dentro de la información secundaria, que indica el inicio de la información de audio dentro de cada trama. Por

ejemplo, *main_data_begin_4* es igual a cero, indicando que los datos de audio empiezan inmediatamente después de la información secundaria. Para indicar que el audio de la trama 5 se inicia en la trama 4, se especifica *main_data_begin_5* como un ajuste negativo que indica el desplazamiento en bytes hacia la izquierda para encontrar el primer dato de audio de la trama 5.

En el ejemplo se aprecia como cada trama permite el uso del *bit reservorio*. En el caso de la trama 7, el proceso empieza codificando la información de audio de su propia trama; como los datos requieren muy pocos bits, y la trama 6 tenía espacio disponible, entonces todos los datos de audio de la trama 7 se incluyen en la trama 6, pero la trama 6 sigue con espacio para bit reservorio, que se usa para datos de la trama 8; por lo que gracias al bit reservorio, la trama 6 incluye los datos de audio de tres (3) tramas: las tramas 6, 7 y 8. El audio de la trama 8 se reparte entre las tramas 6 y 7; sin embargo, éste no alcanza a ocupar todo el espacio disponible en la trama 7, así que el bit reservorio de la trama 7 se usa para la trama 9, y así sucesivamente, teniendo en cuenta que los datos de audio de una determinada trama no pueden estar desplazados más de nueve (9) tramas.

Este caso puede ocurrir en una señal de audio MPEG-1 estéreo, si la frecuencia de muestreo es 48 Khz y la tasa de transferencia deseada es 32 Kbps. En este caso, cada trama consume 768 bits, donde 304 bits (32 bits para el encabezado, 16 bits para el chequeo de errores, 256 bits para la información secundaria) son fijos. Por lo tanto, quedan 464 bits disponibles para los datos codificados con Huffman, y debido a que el valor de *main_data_begin* puede apuntar máximo 511 bytes (4088 bits)

hacia atrás, entonces es posible que *main_data_begin* apunte sobre más de ocho (8) tramas (no se cuenta ninguno de los bits usados para el encabezado y la información secundaria de ninguna trama).

También es importante mencionar que el bit reservorio sólo puede originarse de tramas que ya han sido codificadas; para este búfer no es posible usar tramas para las que todavía no se haya hecho la repartición de los bits disponibles (repartición de ruido).

El formato que tiene cada trama se muestra en la siguiente figura, en la cual se puede ver el encabezado de trama que posee 32 bits (cuatro bytes) de longitud; los primeros 12 bits siempre se ponen en '1', se llaman "*FRAME SYNC*", y se usan para sincronización de la trama.

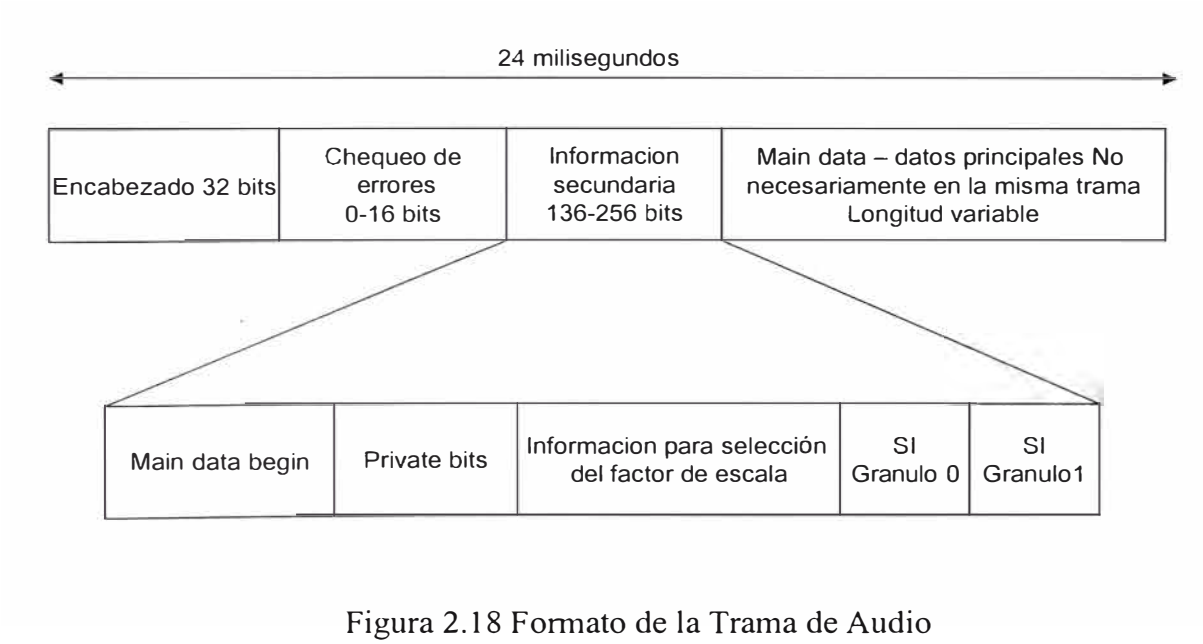

Figura 2.18 Formato de la Trama de Audio

Las tramas pueden tener opcionalmente un CRC para chequeo de errores. Su longitud es de 16 bits, y si existe, se pone después del encabezado. Volviendo a calcular el CRC se puede comprobar si la trama ha sido alterada durante la transmisión del flujo de bits de audio MP3. A continuación sigue la información secundaria (*Side Information*) que indica cómo se realizó la codificación, y por lo tanto, cómo debe realizarse la decodificación. En el último bloque sí vienen los datos de audio (*main data*), repartidos entre dos (2) gránulos.

Dentro de la información secundaria, que usa 136 bits en modo monofónico y 256 bits en los otros modos, se incluye el *main_data_begin*, que es el puntero ya visto. Los bits privados están a disposición del usuario. Después viene la información que indica cuál combinación de factores de escala se está usando (*scfsi*, *scalefactor selection information*). Los últimos dos sub-bloques corresponden a la información secundaria (*Side Info*, SI) para cada uno de los dos gránulos (sub-tramas) en los que se divide una trama.

El último bloque, *main data*, es el que lleva la información de audio; las muestras MDCT codificadas con Huffman, repartidas entre dos gránulos. Cada gránulo contiene información de 576 muestras de audio (exactamente la mitad de la información total de la trama). Además, en este mismo bloque se incluyen los factores de escala de la trama y la información auxiliar, siendo esta última opcional.

CAPITULO III

DECODIFICACION DE AUDIO MPEG CAPA 3

3.1 Algoritmo de Decodificación MPEG Capa 3

En esta sección describiremos el decodificador y su funcionalidad. El proceso de decodificación se basa en el diagrama de bloques de la figura 3.1. El decodificador tiene tres partes principales: "Decodificación de los flujos de bits", "Cuantización Inversa" y "Mapeo de Frecuencia a Tiempo".

El flujo de entrada codificado se pasa a través de la prımera etapa para sincronizar y extraer la línea de frecuencia cuantizada e información adicional de cada trama. La etapa de cuantización inversa, decuantiza las líneas de frecuencia de acuerdo a la salida de la etapa anterior. Finalmente, la última etapa realiza un conjunto de operaciones inversas de la MDCT y del análisis del banco de filtros polifásico en el codificador. Su salida es la señal de audio en formato PCM.

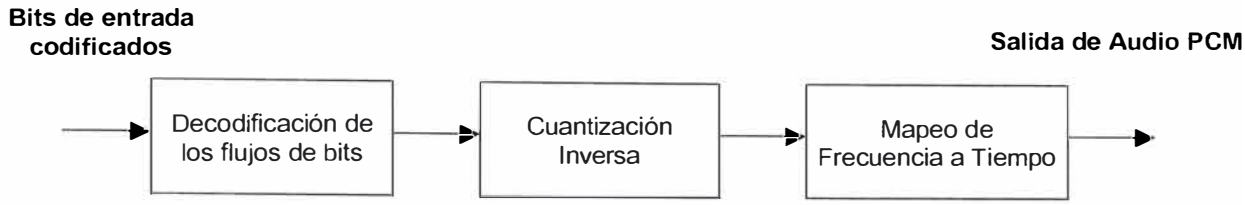

Figura 3.1 Diagrama de bloques del decodificador de Audio / MPEG Capa 3

3.2 Decodificación de Flujos de Bits

Esta parte efectúa la sincronización y extracción de las líneas de frecuencia cuantizada e información adicional de cada trama. Por supuesto, se necesita sincronizar donde la trama comienza y donde termina. El diagrama de bloques correspondiente se muestra en la figura 3.2.

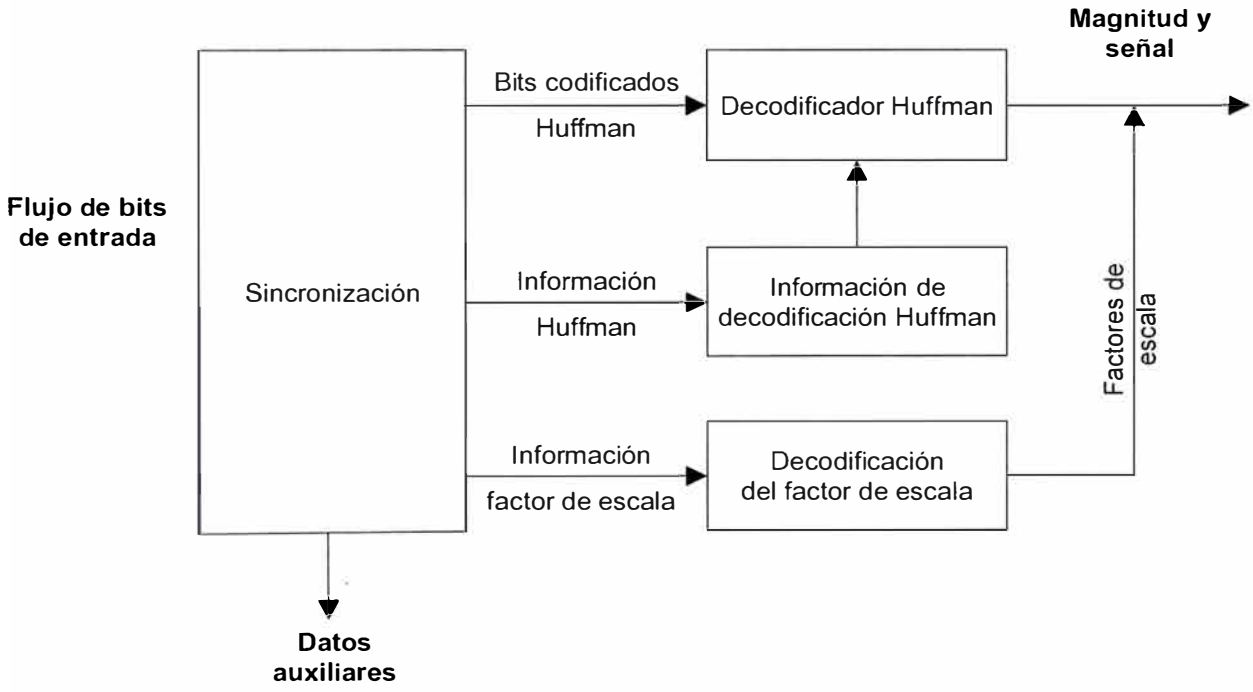

Figura 3.2 Diagrama de bloques del decodificador de bits

3.2.1 Sincronización

El propósito de este bloque es recibir el flujo de bits de entrada, identificar su contenido y pasar la información a los bloques subsiguientes del decodificador.

3.2.2 Formato del flujo de bits

El contenido de un flujo de bits de audio MPEG se organiza en tramas, cada una contiene información para reconstruir las muestras de audio. Una trama consiste de cuatro partes: cabecera, información secundaria, datos principales y datos auxiliares.

a. Cabecera

La cabecera contiene datos de sincronización (12 bits) e información del sistema. Para ser capaz de detectar el comienzo de una nueva trama, cada trama comienza con 12 bits de sincronización. El resto de la cabecera describe el tipo de trama, la capa que se usa, la tasa de bits usada para la transmisión, la frecuencia del muestreo de la señal de audio original, si la señal de audio es de canal único o de doble canal y otras informaciones adicionales. La figura 3.3 muestra el formato de la cabecera MPEG de audio capa 3.

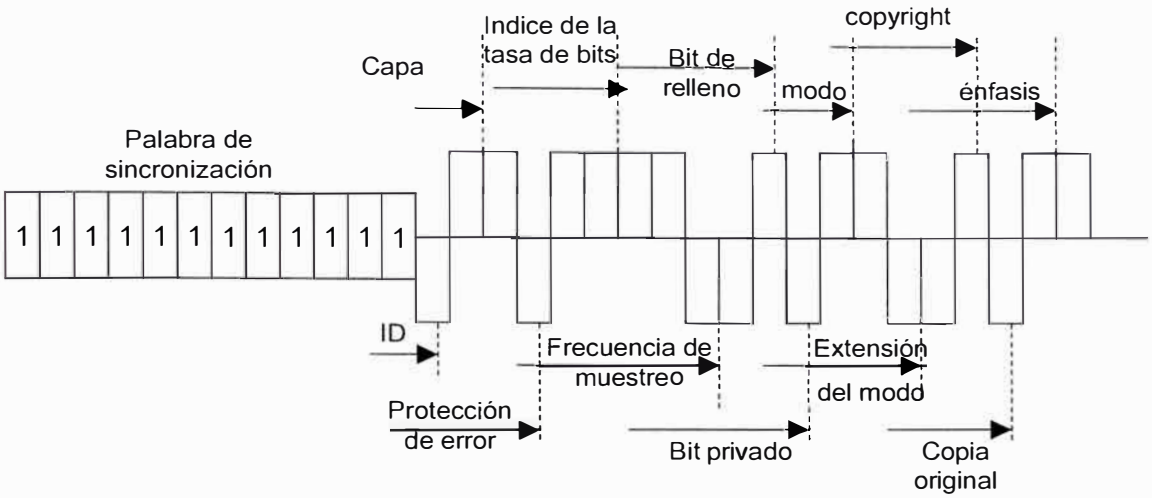

Figura 3.3 Formato de la cabecera Audio / MPEG Capa 3

b. Información secundaria

La información secundaria contiene información necesaria para decodificar los datos principales, esta etapa contiene información relativa a las tablas de Huffman usadas durante el proceso de decodificación, e información de los factores de escala. También contiene información acerca de donde comienzan los datos principales debido al bit reservorio descrito en la sección 2.1.6.

c. Data principal

Esta sección contiene el valor del factor de escala codificado y los datos de Huffman codificados. Ver la figura 2.15.

d. Datos auxiliares

Es posible incluir datos auxiliares en cada trama. El formato de estos datos es definido por el usuario y puede ser usado por ejemplo, para el titulo, el artista, etc. Los datos auxiliares se colocan entre el final de los datos principales de una trama, y el comienzo de los datos principales de la siguiente trama.

3.2.3 Decodificación Huffman

Puesto que la decodificación Huffman es un método VLC, un código de palabra única en medio de los códigos de bits no puede ser identificado sin empezar a decodificar desde un punto conocido en los bits de código, que es el inicio del código de la palabra.

3.2.4 Información de Decodificación Huffman

Este bloque sirve para configurar todos los parámetros necesarios para llevar a cabo una decodificación correcta del código Huffman. La primera tarea es recolectar los datos de información secundaria, la que describe las características del código Huffman de los bits, donde encontrarlos dentro de la trama, para decidir cual tabla de Huffman usar. Además, este bloque debe asegurarse que todas las líneas de frecuencia sean generadas sin hacer caso de cuantas líneas de frecuencia son descritas en los bits de códigos Huffman. Cuando aparecen menos de 576 líneas de frecuencia, este bloque debe llevar a cabo un relleno de bits ceros para compensar la falta de datos.

3.2.5 Decodificador del Factor de Escala

Su función es decodificar los factores de escala codificados, cuya entrada es información del factor de escala, por ejemplo, la primera parte de los datos principales. La salida de este bloque se usará en el bloque de cuantización inversa.

3.3 Cuantización Inversa

El propósito de este bloque es restablecer los datos de las líneas de frecuencia generadas por el bloque MDCT en el codificador. Esta basado en las líneas de frecuencia escalada cuantizada, reconstruida del bloque de decodificación Huffman y el factor de escala reconstruida en el bloque de decodificación del factor de escala. Este factor se calcula con la siguiente ecuación.

$$xr[i] = sign(is[i]) * abs(is[i])^{\frac{4}{3}} * \frac{2^{\frac{1}{4}(global\_gain-210-8sbg[i])}}{2^{scalefact\_multiplier*(sf[i]+preflag*pt[i])}} \qquad (3.1)$$

Donde:

is[i] es la línea de frecuencia reconstruida por el decodificador Huffman, *global_gain*, *sbg[i]*, *scalefac_multiplier*, *sf[i]*, *preflag*, *pt[i]* provienen de la etapa de decodificación del factor de escala.

3.4 Mapeo de Frecuencia a Tiempo

Permite reproducir la señal de audio de la línea de frecuencia decuantizada. Esta parte contiene varios sub-bloques como se muestra la figura 3.4.

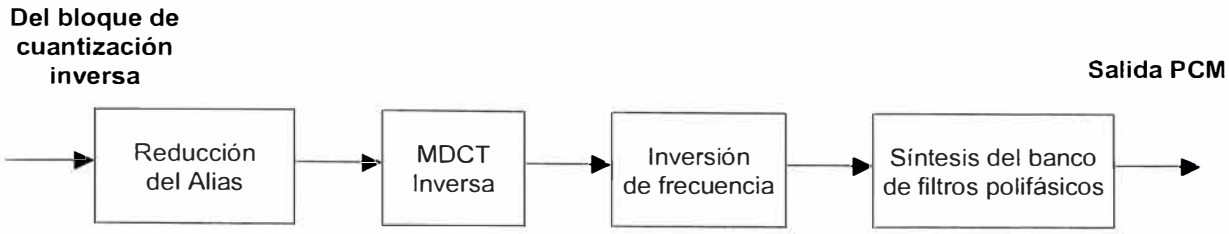

Figura 3.4 Mapeo de Frecuencia a Tiempo

3.4.1 Reducción del Solapamiento (Alias)

En el bloque MDCT del codificador, se describió la aplicación de una reducción de solapamiento o "alias". Con el objeto de obtener una correcta reconstrucción del banco de filtros polifásicos, en los algoritmos provenientes de etapas anteriores, el proceso de solapamiento debe ser añadido nuevamente al proceso de decodificación.

3.4.2 MDCT inverso

Las líneas de frecuencia provenientes del bloque de reducción del solapamiento o "alias" son procesadas a través del bloque IMDCT. La expresión analítica se muestra en la ecuación 3.2.

$$x_i = \sum_{k=0}^{\frac{n}{2}-1} X_k \cos\left(\frac{\pi}{2n}\left(2i+1+\frac{n}{2}\right)(2k+1)\right) \quad \text{, para } i = 0 \ldots n-1 \tag{3.2}$$

Donde:

X_k es la línea de frecuencia.

n es igual a 12 para ventana corta, y 36 para ventana larga.

3.4.3 Inversión de frecuencia

Con el objeto de compensar la decimación usada en el banco de filtros polifásicos, cada muestra de tiempo impar de cada sub-banda impar se multiplica por –1.

3.4.4 Síntesis del Banco de Filtros Polifásicos

Cada vez, 32 muestras provenientes de cada una de las 32 sub-bandas, son aplicadas al banco de filtros polifásicos y son calculadas 32 muestras de audio. La figura 3.5 muestra el algoritmo y el procedimiento del banco de filtros polifásicos.

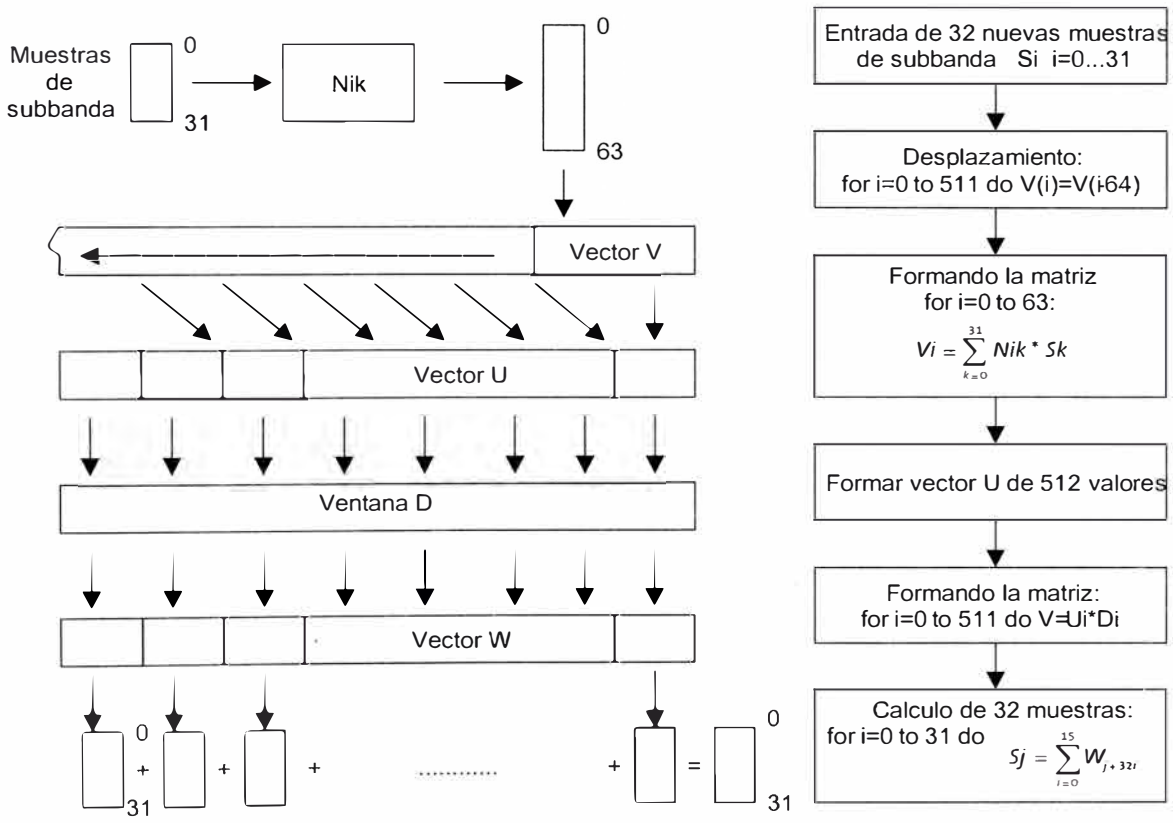

Figura 3.5 Diagrama y procedimiento de síntesis del banco de filtros polifásicos

4.2.1 Algoritmo

El sonido es primero dividido en tres bandas de frecuencia con frecuencias, 0-5.5125 Khz, 5.5125-11.025 Khz y 11.025-22.05 Khz. Después de esto las bandas son transformadas al dominio de la frecuencia, usando una Transformada Discreta de Coseno Modificada (MDCT). Al igual que el algoritmo MPEG Capa 3, ATRAC usa una longitud de bloque adaptivo para evitar los efectos de pre-eco en señales temporales. Un bloque largo tiene 11.6 ms y un bloque corto tiene 1.45 ms o 2.9 ms dependiendo de la banda de frecuencia. La banda de alta frecuencia usa bloques cortos de 1.45 ms, y las dos bandas de frecuencias mas bajas usan 2.9 ms., a diferencia del algoritmo de Capa 3, el tamaño de los bloques puede ser seleccionado individualmente para cada banda. Después del bloque MDCT los valores espectrales producidos son cuantizados y guardados por un proceso de asignación de bits, lo cual es también muy similar al algoritmo MPEG.

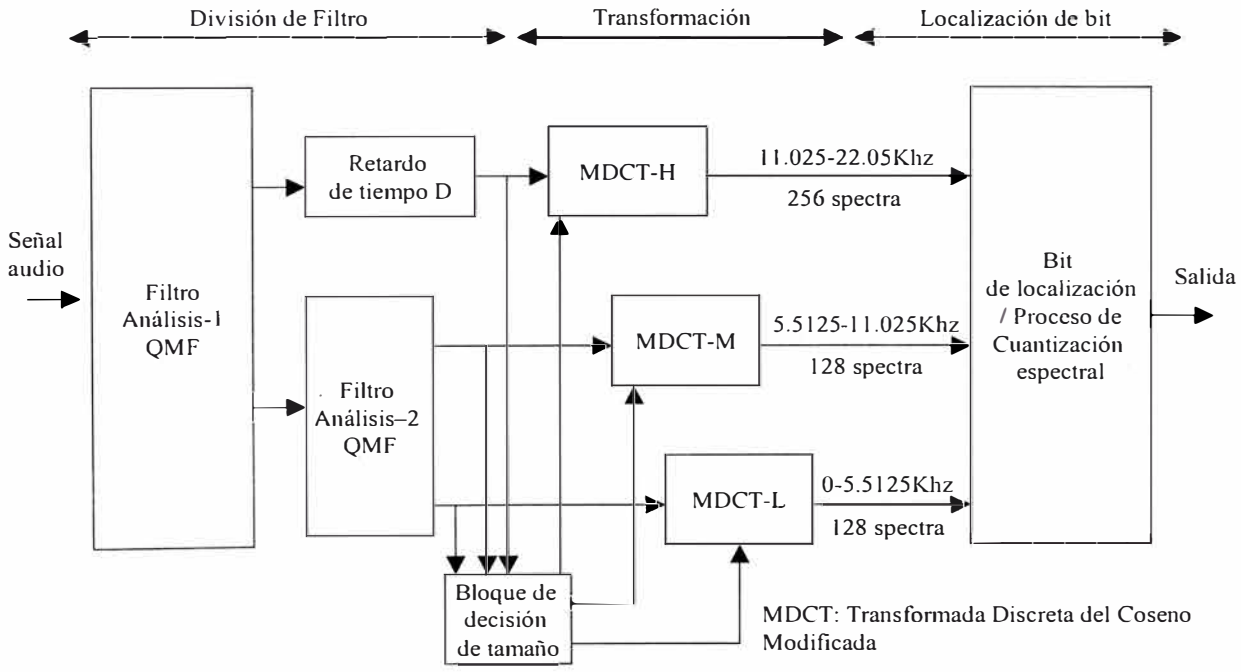

Figura 4.1 Diagrama de Bloques de Decodificación Sony ATRAC

CAPITULO IV

OTRAS TÉCNICAS DE COMPRESIÓN

4.1 Introducción

Existen otras técnicas de compresión para audio de alta fidelidad que las que MPEG ha desarrollado. Sony ATRAC (usado para sistemas MiniDisc), Dolby AC-3 y TwinVQ son algunos de ellos. Estos tienen similares técnicas de compresión, pero difieren en algunas áreas de la codificación, mencionaremos algunas de ellas.

4.2 Sony ATRAC

A inicios de los 90's Sony se dio cuenta que había una necesidad para un sistema de audio digital, portátil y regrabable. Las investigaciones resultaron en el primer sistema MiniDisc presentado en 1992, usando su algoritmo de compresión ATRAC. ATRAC, que es un acrónimo de Codificación Acústica Adaptiva Transformada, opera exclusivamente en muestras de 16 bits con una frecuencia de muestreo de 44,1 Khz y provee una tasa de compresión de 1:5.

4.3 Dolby AC-3

En 1989 Dolby introdujo su sistema de compresión de audio AC-2, siendo uno de los pioneros en el campo de la compresión de audio, este sistema surgió después como AC-3, heredando muchas de las ideas de AC-2. El sistema AC-3 es comúnmente usado en cinemas, pero también ha sido adoptado por el estándar Norte Americano HDTV y es usado como uno de los estándares en DVD.

4.3.1 Algoritmo

El flujo de bits AC-3 consiste de tramas de sincronización. Una trama consiste de 6 bloques de audio con 256 muestras, haciendo un total de 1536 muestras por trama (32 ms a 48 Khz). La información no es compartida entre tramas diferentes, haciendo posible decodificar una trama independientemente, sin información alguna de otra.

Primero, hay una detección de fase temporal, para definir las banderas (flags) de los bloques de conmutación. La señal de entrada es segmentada en un árbol jerárquico con diferentes longitudes de bloques y es luego examinada por detección temporal. La señal es transformada al dominio de la frecuencia usando MDCT de manera similar a otros algoritmos. El MDCT es una transformación de 512 puntos o dos de 256, dependiendo de las banderas (flags) de conmutación. Los valores espectrales son luego divididos en 50 bandas, no igualmente espaciadas, pero tratando de igualar al ancho de banda critico de escucha. Los valores espectrales son luego suministrados al algoritmo de asignación de bits, analizando el audio y la codificación con respecto a los efectos de enmascaramiento. Los valores codificados consisten de un exponente y una mantisa. Los exponentes son valores de 5 bits, y son

transmitidos usando valores delta basados en el exponente previo. Hay 3 diferentes estrategias de exponentes. Los bits de la mantisa dependen del análisis de enmascaramiento de audio.

4.4 Twin VQ

Twin VQ (Transform-domain Weighted Interleave Vector Cuantización) es una codificación de transformación como MP3, AAC o AC-3. Ello como AAC será usado en el venidero estándar MPEG-4 y usa algunas herramientas de ACC como la predicción del intertramado, pero la codificación de música es completamente diferente. El estándar Twin VQ usa una biblioteca de patrones la que es preparada con anticipación y comparada contra los bits en el flujo de audio. Los patrones estándar que proveen la igualdad mas cerrada es seleccionada, y un número asociado con ese patrón es transmitido como la compresión de código.

4.5 No Todos los Decodificadores son creados iguales

Los estándares MPEG no indican la implementación de un codificador de audio, en un caso extremo uno podría evitar el modelo perceptual, decidir no usar los factores de escala (y por lo tanto el lazo de iteración exterior), y realizar un simple lazo de iteración interior. Tal codificador sería muy veloz (potencialmente mucho mas veloz que cualquier codificador producido actualmente), estaría cumpliendo con el estándar, igualmente produciría una buena calidad de audio para algunas señales, pero sonaría muy mal con una gran selección de música. Mientras tal proyecto sería fácil de implementar, es mucho más difícil construir un codificador que ofrezca una calidad de audio muy alta en todos los tipos de música y en todas las pruebas. En

MPEG, las pruebas siempre han estado dirigidas a verificar el rendimiento del codificador en los peores escenarios. Los actuales codificadores MP3 muestran diferencias muy marcadas en su capacidad de producir de una manera consistente una alta calidad de sonido a bajas tasas de bits.

4.6 Como medir la calidad del sonido

Básicamente hay tres métodos. Las pruebas de audición, los métodos de medición objetiva simple y las técnicas de medición perceptual.

4.6.1 Pruebas de audición

Por citar, la escala grande y las pruebas de audición controlada son aún los únicos métodos disponibles para comparar el rendimiento de los diferentes algoritmos de codificación y los diferentes codificadores. Con el ingreso de un mayor número de difusores y el grupo de audio MPEG, la ITU-R ha desarrollado un muy elaborado conjunto de reglas para las pruebas de audición, estas pruebas dirigidas a probar a los codificadores bajo las peores condiciones, por ejemplo los probadores intentan encontrar sonidos difíciles de codificar, y entonces evalúan el rendimiento del codificador con este material. Este procedimiento esta basado en la observación que en muchos casos de hecho llega a ser audible y detectable después de un extensivo periodo de entrenamiento. Puesto que el uso de equipo basado en tecnología de compresión de audio (tales como reproductores de audio portátiles), por si mismo constituye un entrenamiento extensivo, todos pueden llegar a ser expertos oyentes en un periodo de tiempo. Por lo tanto desde el principio, los codificadores deberían ser diseñados para satisfacer los requerimientos de calidad de los oyentes expertos.

4.6.2 Técnicas de medición objetivas simples

Las personas intentan lograr una medición de la calidad del codificador observando los parámetros como la relación de señal a ruido o el ancho de banda de la señal decodificada. Los paradigmas básicos de los codificadores perceptuales de audio confían en la mejora de la calidad subjetiva por modelamiento del ruido de cuantización sobre la frecuencia (y tiempo), principalmente un SNR el cual sea lo mas bajo posible sin modelar el ruido, estas medidas definen el propósito completo de la codificación perceptual. Otra aproximación es observar la salida codificada para ciertas señales de prueba de entrada, tales como transitorios o señales multitono. Mientras los resultados de tales pruebas pueden indicar a los expertos mucho acerca del codificador bajo prueba, es muy arriesgado confiar únicamente en tales resultados.

4.6.3 Técnicas de Medición Perceptual

Por 15 años ha habido intensa investigación aplicando el modelo psicoacústico para la predicción de la calidad y audición del sonido; mientras las pruebas de escucha y audición resultan obsoletas e insuficientes, las técnicas de medición perceptual han progresado hasta el punto donde son un suplemento muy útil para las pruebas de audición, en algunos casos, los están reemplazando. El ITU-R Grupo de Tarea 10/4 trabajó en la estandarización de las técnicas de medición perceptual los que finalmente elaboraron una recomendación para un sistema llamado PEAQ (Evaluación Perceptual de la Calidad de Audio - Perceptual Evaluation of Audio Quality). Esta recomendación define un sistema multimodo basado en la colaboración de los laboratorios que trabajan en las técnicas de medición perceptual.

CAPITULO V

DISEÑO DE PROGRAMA USANDO MATLAB

5.1 Simulación Modelo Psicoacústico usando MATLAB

Esta es la implementación del modelo psicoacústico capa 1 para MATLAB (Fuente: Fabien Petitcolas - Universidad de Cambridge – Inglaterra). El archivo principal es Prueba_MPEG.m. El programa es didáctico y permite apreciar como trabaja el formato MPEG. Esta implementación se basa en ISO/IEC 11172-3

Archivos usados:

Archivo inicio	
Prueba_MPEG.m	
Archivos anexos	
Tabla_umbral_absoluto.m	Analisis_filtro_sub-banda.m
Tabla_ventana_analisis.m	Factores_escala.m
Tabla_limites_banda_critica.m	Analisis_FFT.m
Nivel_presion_sonido.m	Umbral_enmasc_global.m

Hallar_componentes_tonales.m Decimacion.m Umbral_enmasc_individual.m	Umbral_enmasc_minimo.m Common.m Mod.m

5.1.1 Descripción de los archivos del programa

Prueba_MPEG.m

En este archivo se construye una señal de muestra (señal *x*) a la que se le aplicará el modelo psicoacústico, se utiliza una función de muestreo *fs* de 44100 muestras/seg, así mismo se empiezan a cargar primero las tablas definidas en el estándar: Tablas de umbral absoluto, tablas de límites de banda crítica, tabla de ventana de análisis las cuales son necesarias para el cálculo.

Seguidamente se realiza el ingreso del vector *x* al banco de filtros polifásicos, cuya salida es la matriz S. A continuación se lleva a cabo el Análisis del filtro sub-banda. La capa 1 usa 12 muestras por sub-banda. La matriz S es una matriz de 12 filas por 32 columnas, representan las 32 sub-bandas y 12 muestras por sub-banda (total 384 muestras), en le programa esta dado por la siguiente expresión:

```
for i = 0:11,
  S = [S; Analisis_filtro_subbanda(x, OFFSET + 32 * i, C)];
end
```

Seguidamente se calculan los 32 factores de escala (uno por sub-banda).

```
scf = Factores_escala(S);
```

A continuación se realiza el análisis psicoacústico. Para ello se calcula la transformada rápida de Fourier para la conversión del dominio del tiempo al dominio de la frecuencia usando la siguiente expresión:

X = Analisis_FFT(x, OFFSET);

X representa el espectro auditivo (DEP normalizada)

Luego se determina el nivel de presión de sonido en cada sub-banda mediante la siguiente expresión:

Lsb = Nivel_presion_sonido(X, scf);

Seguidamente se procede a calcular las componentes tonales (como las senoidales) y no-tonales (como ruido) de la señal

[Flags Tonal_list Non_tonal_list] = Hallar_componentes_tonales(X, TH, Map, CB);

Se realiza la decimación de los enmascaradores: es decir eliminar todos los enmascaradores irrelevantes:

[Flags Tonal_list Non_tonal_list] = Decimacion(X, Tonal_list, Non_tonal_list, Flags, TH, Map);

Se calculan los umbrales de enmascaramiento individual:

[LTt, LTn]=Umbrales_enmasc_individual(X,Tonal_list, Non_tonal_list, TH, Map);

A continuación se calcula el umbral de enmascaramiento global:

LTg = Umbral_enmasc_globall(LTq, LTt, LTn);

Se determina el umbral de enmascaramiento mínimo en cada sub-banda.

LTmin = Umbral_enmasc_minimo(LTg, Map);

A continuación se calcula la relación de señal a máscara mediante la siguiente expresión:

SMRsb = Lsb - LTmin;

Por último se grafican el mínimo Umbral de enmascaramiento y el umbral de enmascaramiento global.

Tabla_umbral_absoluto.m

[TH, Map, LTq] = Tabla_umbral_absoluto(Layer, fs, bitrate).

Este archivo devuelve las frecuencias, la razón de banda crítica y el umbral absoluto en TH. Map contiene un mapeo entre la línea de frecuencia k y el índice para el TH o tablas LTq. LTq contiene solamente el umbral en reposo. LTq(k) está definido en las tablas del estándar, sus valores están disponibles para cada muestra en el domino de la frecuencia donde el umbral de enmascaramiento es calculado. Estos valores dependen de la capa, la frecuencia de muestreo fs (Hz) y la tasa de bits de transferencia "bitrate" (kbits/s). La conversión de frecuencia f en Barks se realiza usando (f * bitrate/fs).

Se usa un offset dependiendo de la tasa de transferencia para el Umbral Absoluto, este offset es de -12dB para tasa de transferencia >=- 96 Kbps y 0dB para tasas de

transferencia< 96 Kbps por canal. En el se usa una tasa de 128 Kbps. ATH=3 (ver archivo common.m).

Tabla_limites_banda_critica.m

CB = Tabla_limites_banda_critica(Layer, fs)

Devuelve el índice en la tabla de umbral absoluto para los límites de banda crítica definidas en las tablas de umbral absoluto para tasa de muestreo fs = 44100 (Hz). Estos valores dependen de la capa y de la frecuencia de muestreo fs.

Revisar también Tabla umbral absoluto.m

Tabla ventana analisis.m

Inicializa la ventana de análisis usada en el análisis de sub-banda. Los coeficientes C son dados en las tablas del estándar. Contrariamente al estándar, aquí las muestras mas recientes tendrán un índice mas alto que la primera muestra, formándose una matriz columna (usando el comando flipud de MATLAB)

C(1)=0.000000477

C(512)=0

Revisar también el archivo Analisis_filtro_subbanda.m

Analisis_filtro_subbanda.m

S = Analisis_filtro_subbanda(Input, n, C)

Devuelve las 32 sub-bandas muestreadas S(i) definidas en el estándar es decir una muestra para cada sub-banda. Input es la señal de entrada de audio x, n es el índice

en la entrada donde las 32 nuevas muestras están localizadas. C es la ventana de análisis definida en el estándar (definidas en el archivo Tabla_ventana_analisis.m).

Se construye un vector de entrada X de 512 elementos. La muestra mas reciente esta en la posición 512 mientras los elementos más antiguos están en la posición 1. Se completa con ceros si la señal de entrada no existe.

```
.....................................................
|       480 muestras       | 32 muestras |
   n-480                    n          n+31
```

La salida del banco de filtros S se genera por la siguiente expresión:

```
S = zeros(1, 32);
for i = 1 : 32,
  for k = 1 : 64,
    S(i) = S(i) + M(i, k) * Y(k);
  end
end
```

La cual produce una matriz de 1 fila por 32 columnas S=[S(1) ....S(32)], es decir una muestra para cada una de las 32 sub-bandas.

Revisar también el archivo Tabla_ventana_analisis.m

Factores_escala.m

```
scf = Factores_escala(S)
```

Se cargan las tablas de los factores de escala de acuerdo al estándar definido en la norma.

Para cada sub-banda S(i, :), el máximo valor absoluto de las 12 muestras S(i, 1) ... S(i, 12) se determina. Se escoge el siguiente valor mas grande del factor de escala en la tabla de factores de escala.

Este archivo genera 32 valores en scf(i) uno para cada sub-banda.

Analisis FFT.m

X = Analisis_FFT(Input, n)

Calcula el espectro auditivo usando la Transforma Rápida de Fourier.

El espectro X se expresa en dB. El tamaño de la transformada es 512 y es centrada sobre las 384 muestras (12 muestras por sub-banda) usadas por el análisis de sub-banda. La primera de las 384 muestras es indexada en n:

```
.....
|     | 384 muestras  |     |
n-64   n              n+383   n+447
```

Una ventana de Hanning se aplica antes de calcular la FFT.:

```
h = sqrt(8/3) * hanning(512, 'periodic');
```

Se calcula la Densidad espectral de potencia. X será matriz de 512 filas y 1 columna:

```
X = max(20 * log10(abs(fft(s .* h)) / FFT_SIZE), MIN_POWER);
```

Finalmente, se realiza una normalización del nivel de presión del sonido de tal manera que el valor máximo del espectro sea de 96dB; el número de dB añadidos es almacenado en la salida Delta

```
Delta = 96 - max(X);
X = X +  Delta;
```

Nivel_presion_sonido.m

```
X = Nivel_presion_sonido(X, scf)
```

El nivel de presión del sonido Lsb es calculado para cada sub-banda, mediante la siguiente expresión:

```
for i = 1:N_SUBBAND,
  local_max = Xmin;
  for j = 1:n,
    local_max = max(X((i - 1) * n + j), local_max);
  end
  Lsb(i) = max(local_max, 20 * log10(scf(i) * 32768) - 10);
end
```

Lsb (i) es calculado para cada una de las 32 sub-bandas, los -10 dB en la expresión corrigen la diferencia entre el valor pico y el RMS. X es la densidad espectral de potencia normalizada y scf son los 32 factores de escala (uno por banda).

Hallar_componentes_tonales.m

```
[Flags, Tonal_list, Non_tonal_list] = Hallar_componentes_tonales(X, TH, Map, CB)
```

Identifica y lista las componentes tonales y no-tonales de la señal de audio.

Se asume en esta implementación que la frecuencia de muestreo fs es 44100 Hz.

Se etiqueta la máxima local: Una línea espectral es etiquetada como máxima si X(k) > X(k-1) y X(k) >= X(k+1) y si el índice k se encuentra dentro del rango k > 2 & k <= 250

```
local_max_list = [];
counter = 1;
for k = 2:FFT_SIZE / 2 - 1,
  if (X(k) > X(k-1) & X(k) >= X(k+1) & k > 2 & k <= 250)
    local_max_list(counter, INDEX) = k;
    local_max_list(counter, SPL) = X(k);
    counter = counter + 1;
  end
end
```

Se listan las componentes tonales y se calcula el nivel de presión del sonido. Un máximo local se coloca en la lista de componentes tonales si X(k) - X(k + j) >= 7dB donde j se escoge de acuerdo a:

Capa I:

j=-2,+2	para 2<k<63
j=-3,-2,+2,+3	para 63<=k<127
j=-6,...,-2,+2,...,+6	para 127<=k<=250

```
Tonal_list = [];
counter = 1;
```

```
if not(isempty(local_max_list))
        for i = 1:length(local_max_list(:, 1)),
          k = local_max_list(i, INDEX);
          is_tonal = 1;
```

Para la Capa I se examinan las frecuencias vecinas:

```
          if (2 < k & k < 63)
            J = [-2 2];
          elseif (63 <= k & k < 127)
            J = [-3 -2 2 3];
          elseif (127 <= k & k < 250)
            J = [-6:-2, 2:6];
          else
            is_tonal = 0;
          end

          for j = J,
            is_tonal = is_tonal & (X(k) - X(k + j) >= 7);
          end
```

Si X(k) es una componente tonal entonces los siguientes datos son listados

- número de índice k de la línea espectral
- nivel de presión del sonido: Xtm(k)=X(k-1)+X(k)+X(k+1)
- fijar flag tonal

Se listan las componentes no tonales y se calcula la potencia, todas las líneas espectrales que no han sido examinadas durante el proceso de búsqueda anterior son sumadas para formar la componente no tonal.

```
Non_tonal_list = [];
counter = 1;
for i = 1:length(CB(:, 1)) - 1,,
Para cada banda crítica, se calcula la potencia en las componentes no-tonales
  power  = MIN_POWER;
Suma Parcial:
  weight = 0;
Usada para calcular la media geométrica de la banda crítica:
  for k = TH(CB(i), INDEX):TH(CB(i + 1), INDEX) - 1,
En cada banda crítica
    if (Flags(k) == NOT_EXAMINED),
      power    = 10 * log10(10^(power / 10) + 10^(X(k) / 10));
      weight     = weight + 10^(X(k) / 10) * (TH(Map(k), BARK) - TH(CB(i),
BARK)');
      Flags(k) = IRRELEVANT;
    end
  end
```

El numero de índice para la componente tonal es el índice mas cercano a la media geométrica de la banda crítica. Para cada sub-banda se calcula:

- índice de la componente no-tonal
- nivel de presión de sonido de este componente

En esta parte se procede a realizar las primeras gráficas al hallar las componentes locales máximas en función del índice de frecuencia k, así como las graficas de las componentes tonales y no tonales.

Decimacion.m

[DFlags, DTonal_list, DNon_tonal_list] = Decimacion(X, Tonal_list, Non_tonal_list, Flags, TH, Map)

Las componentes que están por debajo del umbral auditivo o son menores que la mitad del ancho de banda crítica de un componente vecino son eliminadas. Estas variables contienen la lista de banderas (flags), componentes tonales y no-tonales después de la decimación.

Las componentes Tonales o no-tonales son considerados si son menores que el umbral absoluto de escucha encontrado en TH(:, ATH), definido en el archivo Tabla umbral_absoluto.m

Se procede a eliminar las componentes tonales que son menores que la mitad (0.5 Bark) del ancho de la banda crítico de una componente vecina (segunda parte del caso tonal).

Este archivo también gráfica las componentes tonales y no tonales, después de aplicar la decimación.

Umbrales_enmasc_individual.m

[LTt, LTn] = Umbrales_enmasc_individual(X, Tonal_list, Non_tonal_list, TH, Map)

Calcula el efecto de enmascaramiento de las componentes tonales y no tonales en las frecuencias espectrales vecinas. La potencia del enmascarador es sumada con el índice de enmascaramiento y la función de enmascaramiento.

LTt: umbral de enmascaramiento para las componentes tonales

LTn: umbral de enmascaramiento para las componentes no-tonales.

Los Umbrales de enmascaramiento individual para componentes tonales y no-tonales son fijadas a -infinito puesto que la función de enmascaramiento tiene atenuación infinita fuera de -3 y +8 Barks, esto significa que la componente no tiene efecto de enmascaramiento en frecuencias fuera de esos rangos.

Solamente un subconjunto de muestras son consideradas para el cálculo de Umbral de enmascaramiento global. El número de estas muestras depende de la tasa de muestreo y de la capa de codificación. Toda la información necesaria está en TH la que contiene las frecuencias, tasa de banda crítica y umbral absoluto. Para cada componente tonal el Umbral de enmascaramiento individual:

$LTtm = Xtm[z(j)] + avtm[z(j)] + vf[z(j),z(i)]$ dB

La Función de enmascaramiento depende de la distancia en Bark al enmascarador:

$dz=z(i)-z(j)$

Para cada componente no-tonal el Umbral de enmascaramiento individual:

LTnm = Xnm[z(j)] + avnm[z(j)] + vf[z(j),z(i)] dB

La Función de enmascaramiento para componentes tonales depende de la distancia en Bark al enmascarador: dz=z(i)-z(j) se define en el siguiente condicional if:

```
if (-3 <= dz & dz < -1)
  vf = 17 * (dz + 1) - (0.4 * X(j) + 6);
elseif (-1 <= dz & dz < 0)
  vf = (0.4 * X(j) + 6) * dz;
elseif (0 <= dz & dz < 1)
  vf = -17 * dz;
elseif (1 <= dz & dz < 8)
  vf = - (dz - 1) * (17 - 0.15 * X(j)) - 17;
end
  LTt(k, i) = Tonal_list(k, SPL) + avtm + vf;
```

Función de enmascaramiento para cada componente no tonal se define en el siguiente condicional if:

```
if (-3 <= dz & dz < -1)
  vf = 17 * (dz + 1) - (0.4 * X(j) + 6);
elseif (-1 <= dz & dz < 0)
  vf = (0.4 * X(j) + 6) * dz;
elseif (0 <= dz & dz < 1)
  vf = -17 * dz;
```

```
elseif (1 <= dz & dz < 8)
   vf = - (dz - 1) * (17 - 0.15 * X(j)) - 17;
end
LTn(k, i) = Non_tonal_list(k, SPL) + avnm + vf;
```

Seguidamente se grafican los umbrales de enmascaramiento individual de las componentes tonales y no-tonales sobre la gráfica ya existente.

Umbral_enmasc_global.m

LTg = Umbral_enmasc_global(LTq, LTt, LTn)

Calcula el umbral de enmascaramiento global para el subconjunto de líneas de frecuencia definidas en las tabla del estándar. Es la suma en la escala de amplitud normal cuadrada del espectro del umbral de enmascaramiento individual y del umbral absoluto, donde:

LTq : umbral de enmascaramiento absoluto

LTt : umbral de enmascaramiento individual para componentes tonales

LTn : umbral de enmascaramiento individual para componentes no-tonales

El enmascaramiento global es calculado por el subconjunto de frecuencias definidas en tablas. Ellos son la suma de las energías correspondientes a los umbrales de enmascaramiento individual (LTt, LTn) y al umbral en reposo (LTq).

La contribución de las componentes tonales se calcula usando el condicional if y for:

```
  if not(isempty(LTt))
      for j = 1:m,
        temp = temp + 10^(LTt(j, i) / 10);
      end
  end

Contribución de las componentes de ruido (no tonales):
  for j = 1:n,
    temp = temp + 10^(LTn(j, i) / 10);
  end

  LTg(i) = 10 * log10(temp);
end
```

Umbral enmasc minimo.m

LTmin = Umbral_enmasc_minimo(LTg, Map)

Encuentra el valor mínimo del umbral de enmascaramiento global para cada sub-banda.

Revisar también Umbral_enmasc_global.m

5.1.2 Scripts de los archivos del programa

a. Prueba.MPEG.m

```
function Prueba_MPEG

%
%  Nota:
```

```
%  Este programa se basa en el estandard ISO/IEC 11172-3,
%  Codificación para imágenes en movimiento y audio digital Parte 3 Grupo MPEG
%-------------------------------------------------------------------------------
Common;

% Construir una señal de muestra
fs = 44100;
f = [1500 1700 1800 2000 10000 11000 12000 13000];
a = [  1   2   1  .5   2  1.5   3   1];
t = 0:1/fs:1;
x = randn(size(t));
for i = 1:length(f),
  x = x + a(i) * sin(2 * pi * f(i) * t);
end
x = x / max(abs(x));
%[x fs] = wavread('svega');

% Cargar tablas definidas en el estándar.
[TH, Map, LTq] = Tabla_umbral_absoluto(1, fs, 128); % Umbral en reposo
CB = Tabla_limites_banda_critica(1, fs); % límites de banda crítica
C = Tabla_ventana_analisis; % ventana de análisis

% Proceso de ingreso del vector x. Longitud de x=44101
% la matriz S es la salida del banco de filtros
for OFFSET = 1:384:length(x);
  S = [];

%%% Análisis del filtro sub-banda. La capa 1 usa 12 muestras por sub-banda.

  % Análisis del filtro de sub-banda. la matriz S será una
  % matriz de 12 filas por 32 columnas, representan las 32 sub-bandas y
  % 12 muestras por sub-banda (total 384 muestras)
  for i = 0:11,
    S = [S; Analisis_filtro_subbanda(x, OFFSET + 32 * i, C)];
  end

  % Cálculo del Factor de escala
  % genera los 32 factores de escala (uno por sub-banda)
  scf = Factores_escala(S);

%%% Análisis Psicoacústico.

  % Calcular la FFT para conversión de tiempo a frecuencia.
  % X es el espectro auditivo (DEP normalizada)
  X = Analisis_FFT(x, OFFSET);

  % Determinar el nivel de presión de sonido en cada sub-banda
  Lsb = Nivel_presion_sonido(X, scf);

  % Encontrar las componentes tonales (como las senoidales) y no-tonales (como ruido)
  % de la señal
```

```
  [Flags Tonal_list Non_tonal_list] = Hallar_componentes_tonales(X, TH, Map, CB);

  % Decimar los enmascaradores: eliminar todos los enmascaradores irrelevantes
  [Flags Tonal_list Non_tonal_list] = ...
    Decimacion(X, Tonal_list, Non_tonal_list, Flags, TH, Map);

  % Calcular los umbrales de enmascaramiento individual
  [LTt, LTn] = ...
    Umbrales_enmasc_individual(X, Tonal_list, Non_tonal_list, TH, Map);

  % Calcular el umbral de enmascaramiento global
  LTg = Umbral_enmasc_global(LTq, LTt, LTn);

  if (DRAW)
    disp('Umbral de enmascaramiento Global');
    hold on;
    plot(TH(:, INDEX), LTg, 'k--');
    hold off;
    title('Componentes de enmascaramiento y umbrales de enmascaramiento.');
  end

  % Determinar el umbral de enmascaramiento mínimo en cada sub-banda
  LTmin = Umbral_enmasc_minimo(LTg, Map);
  if (DRAW)
    figure; plot(LTmin); title('Umbral de enmascaramiento mínimo');
    xlabel('Número de sub-banda'); ylabel('dB'); pause;
  end

  % Calcular la relación de señal a máscara
  SMRsb = Lsb - LTmin;
end
```

b. Tabla umbral absoluto.m

```
function [TH, Map, LTq] = Tabla_umbral_absoluto(Layer, fs, bitrate)
%[TH, Map,  LTq] = Tabla_umbral_absoluto(Layer, fs, bitrate)
%
%   Devuelve las frecuencias, la razon de banda crítica y el umbral absoluto
%   en TH. Map contiene un mapeo entre la linea de frecuencia k y un número indice
%   para el TH o tablas LTq. LTq contiene solamente el umbral en reposo
%   LT_q(k) definido en las tablas del estándar. los valores estan
%   disponibles para cada muestra en el domino de la frecuencia donde el umbral de
%   enmascaramiento es calculado.
%   Estos valores dependen de la capa, la frecuencia de muestreo fs (Hz)
%   y la tasa de bits de transferencia 'bitrate' kbits/s.
%
%   La conversion de frecuencia f en barks se realiza usando (f * bitrate / fs)
%
%
%-------------------------------------------------------------------------------
```

```
Common;

i f(Layer == 1)
  i f(fs == 44100)
    % Frequ  encia(Hz) | Tasa de banda Critica (z)| Umbral Absolu  to(dB)
    TH = [
        86.13  0.850  25.87 ;  172.27  1.694  14.85 ;
       258.40  2.525  10.72 ;  344.53  3.337   8.50 ;
       430.66  4.124   7.10 ;  516.80  4.882   6.11 ;
       602.93  5.608   5.37 ;  689.06  6.301   4.79 ;
       775.20  6.959   4.32 ;  861.33  7.581   3.92 ;
       947.46  8.169   3.57 ; 1033.59  8.723   3.25 ;
      1119.73  9.244   2.95 ; 1205.86  9.734   2.67 ;
      1291.99 10.195   2.39 ; 1378.13 10.629   2.11 ;
      1464.26 11.037   1.83 ; 1550.39 11.421   1.53 ;
      1636.52 11.783   1.23 ; 1722.66 12.125   0.90 ;
      1808.79 12.448   0.56 ; 1894.92 12.753   0.21 ;
      1981.05 13.042  -0.17 ; 2067.19 13.317  -0.56 ;
      2153.32 13.577  -0.96 ; 2239.45 13.825  -1.38
      2325.59 14.062  -1.79 ; 2411.72 14.288  -2.21 ;
      2497.85 14.504  -2.63 ; 2583.98 14.711  -3.03 ;
      2670.12 14.909  -3.41 ; 2756.25 15.100  -3.77 ;
      2842.38 15.283  -4.09 ; 2928.52 15.460  -4.37 ;
      3014.65 15.631  -4.60 ; 3100.78 15.795  -4.78 ;
      3186.91 15.955  -4.91 ; 3273.05 16.110  -4.97 ;
      3359.18 16.260  -4.98 ; 3445.31 16.405  -4.92 ;
      3531.45 16.547  -4.81   3617.58 16.685  -4.65 ;
      3703.71 16.820  -4.43 ; 3789.84 16.951  -4.17 ;
      3875.98 17.079  -3.87 ; 3962.11 17.204  -3.54 ;
      4048.24 17.327  -3.19 ; 4134.38 17.447  -2.82 ;
      4306.64 17.680  -2.06 ; 4478.91 17.904  -1.32 ;
      4651.17 18.121  -0.64 ; 4823.44 18.331  -0.04 ;
      4995.70 18.534   0.47 ; 5167.97 18.730   0.89
      5340.23 18.922   1.23 ; 5512.50 19.108   1.51
      5684.77 19.288   1.74 ; 5857.03 19.464   1.93
      6029.30 19.635   2.11 ; 6201.56 19.801   2.28
      6373.83 19.963   2.46 ; 6546.09 20.120   2.63
      6718.36 20.273   2.82 ; 6890.63 20.421   3.03 ;
      7062.89 20.565   3.25 ; 7235.16 20.705   3.49 ;
      7407.42 20.840   3.74 ; 7579.69 20.971   4.02 ;
      7751.95 21.099   4.32 ; 7924.22 21.222   4.64 ;
      8096.48 21.341   4.98 ; 8268.75 21.457   5.35 ;
      8613.28 21.676   6.15 ; 8957.81 21.882   7.07 ;
      9302.34 22.074   8.10 ; 9646.88 22.253   9.25 ;
      9991.41 22.420  10.54 ; 10335.94 22.575  11.97 ;
     10680.47 22.721  13.56 ; 11025.00 22.857  15.31 ;
     11369.53 22.984  17.23 ; 11714.06 23.102  19.34 ;
     12058.59 23.213  21.64 ; 12403.13 23.317  24.15 ;
     12747.66 23.414  26.88 ; 13092.19 23.506  29.84 ;
     13436.72 23.592  33.05 ; 13781.25 23.673  36.52 ;
     14125.78 23.749  40.25 ; 14470.31 23.821  44.27 ;
```

```
    14814.84  23.888  48.59  ; 15159.38  23.952  53.22  ;
    15503.91  24.013  58.18  ; 15848.44  24.070  63.49  ;
    16192.97  24.124  68.00  ; 16537.50  24.176  68.00  ;
    16882.03  24.225  68.00  ; 17226.56  24.271  68.00  ;
    17571.09  24.316  68.00  ; 17915.63  24.358  68.00  ;
    18260.16  24.398  68.00  ; 18604.69  24.436  68.00  ;
    18949.22  24.473  68.00  ; 19293.75  24.508  68.00  ;
    19638.28  24.541  68.00  ; 19982.81  24.573  68.00
  ];

  N = length(TH(:, 1)); % devuelve el numero de frecuencias (nº de filas) de la matriz
(106)

  % Convierte frequencias a muestras indexadas(106 muestras).
  for i = 1:N,
    TH(i, INDEX) = round(TH(i, INDEX) / 44100 * 512);
  end

  % Genera un mapeo entre las FFT_SIZE / 2 muestras de la señal de entrada
  % y los N coeficientes de la tabla de umbral absoluto.

  % Límites (1 -1) (232 -512/2) (2 -
  for j = 1:TH(1, INDEX),
    Map(j) = 1;
  end
  for j = TH(N, INDEX):FFT_SIZE/2,
    Map(j) = N;
  end
  % Todos los demás (de la tabla)
  for i = 2:N-1,
    for j = TH(i, INDEX):TH(i+1, INDEX) - 1,
      Map(j) = i;
    end
  end

  % Se usa un offset dependiendo del bit rate para el Umbral Absoluto
  % Este offset es de -12dB para bit rates >=- 96kbits/s y 0dB
  % para bit rates (tasa de transferencia)< 96 kbits/s por canal.
  % en este ejemplo usamos una tasa de 128 kbps. ATH=3 (ver common.m).
  if (bitrate >= 96)
    for i = 1:N,
      TH(i, ATH) = TH(i, ATH) - 12;
    end
  end

  LTq = TH(:, ATH);
 else
  error('Frequencia no soportada.');
 end
else
 error('Capa no soportada');
```

```
end
```

c. Tabla_limites_banda_critica.m

```
function CB = Tabla_limites_banda_critica(Layer, fs)
%CB = Tabla_limites_banda_critica(Layer, fs)
%
%   Devuelve el índice en la tabla de umbral absoluto para los limites de banda
%   critica definidas el estandar, para capa en tasa de muestreo fs (Hz).
%   Estos valores dependen de la capa y de la frecuencia de muestreo fs
%   Ver también Tabla_umbral_absoluto

%-------------------------------------------------------------------------------

if (Layer == 1)
  if (fs == 44100)
    CB = [1, 2, 3, 5, 6, 8, 9, 11, 13, 15, 17, 20, 23, 27, 32, 37, 45, 50, 55, 61, 68, 75, 81, 93,
106]';
   else
     error('Frequencia no soportada.');
   end
 else
   error('Capa no soportada.');
 end
```

d. Tabla_ventana_analisis.m

```
function C = Tabla_ventana_analisis
%C = Tabla_ventana_analisis
%
%   Inicializa la ventana de análisis usada en el analisis de sub-banda.
%   Los coeficientes C son dados en el estándar.
%
%   Ver también Analisis_filtro_subbanda

%-------------------------------------------------------------------------------

% Matlab inicia los índices en uno por lo que necesitamos un arreglo para C(0) = 0.0;
                C(  1)=-0.000000477; C(  2)=-0.000000477; C(  3)=-0.000000477 ;
C(  4)=-0.000000477; C(  5)=-0.000000477; C(  6)=-0.000000477; C(  7)=-0.000000954 ;
C(  8)=-0.000000954; C(  9)=-0.000000954; C( 10)=-0.000000954; C( 11)=-0.000001431 ;
C( 12)=-0.000001431; C( 13)=-0.000001907; C( 14)=-0.000001907; C( 15)=-0.000002384 ;
C( 16)=-0.000002384; C( 17)=-0.000002861; C( 18)=-0.000003338; C( 19)=-0.000003338 ;
C( 20)=-0.000003815; C( 21)=-0.000004292; C( 22)=-0.000004768; C( 23)=-0.000005245 ;
C( 24)=-0.000006199; C( 25)=-0.000006676; C( 26)=-0.000007629; C( 27)=-0.000008106 ;
C( 28)=-0.000009060; C( 29)=-0.000010014; C( 30)=-0.000011444; C( 31)=-0.000012398 ;
C( 32)=-0.000013828; C( 33)=-0.000014782; C( 34)=-0.000016689; C( 35)=-0.000018120 ;
C( 36)=-0.000019550; C( 37)=-0.000021458; C( 38)=-0.000023365; C( 39)=-0.000025272 ;
C( 40)=-0.000027657; C( 41)=-0.000030041; C( 42)=-0.000032425; C( 43)=-0.000034809 ;
C( 44)=-0.000037670; C( 45)=-0.000040531; C( 46)=-0.000043392; C( 47)=-0.000046253 ;
C( 48)=-0.000049591; C( 49)=-0.000052929; C( 50)=-0.000055790; C( 51)=-0.000059605 ;
```

C(52)=-0.000062943; C(53)=-0.000066280; C(54)=-0.000070095; C(55)=-0.000073433 ;
C(56)=-0.000076771; C(57)=-0.000080585; C(58)=-0.000083923; C(59)=-0.000087261 ;
C(60)=-0.000090599; C(61)=-0.000093460; C(62)=-0.000096321; C(63)=-0.000099182 ;
C(64)= 0.000101566; C(65)= 0.000103951; C(66)= 0.000105858; C(67)= 0.000107288 ;
C(68)= 0.000108242; C(69)= 0.000108719; C(70)= 0.000108719; C(71)= 0.000108242 ;
C(72)= 0.000106812; C(73)= 0.000105381; C(74)= 0.000102520; C(75)= 0.000099182 ;
C(76)= 0.000095367; C(77)= 0.000090122; C(78)= 0.000084400; C(79)= 0.000077724 ;
C(80)= 0.000069618; C(81)= 0.000060558; C(82)= 0.000050545; C(83)= 0.000039577 ;
C(84)= 0.000027180; C(85)= 0.000013828; C(86)=-0.000000954; C(87)=-0.000017166 ;
C(88)=-0.000034332; C(89)=-0.000052929; C(90)=-0.000072956; C(91)=-0.000093937 ;
C(92)=-0.000116348; C(93)=-0.000140190; C(94)=-0.000165462; C(95)=-0.000191212 ;
C(96)=-0.000218868; C(97)=-0.000247478; C(98)=-0.000277042; C(99)=-0.000307560 ;
C(100)=-0.000339031; C(101)=-0.000371456; C(102)=-0.000404358; C(103)=-
0.000438213 ;
C(104)=-0.000472546; C(105)=-0.000507355; C(106)=-0.000542164; C(107)=-
0.000576973 ;
C(108)=-0.000611782; C(109)=-0.000646591; C(110)=-0.000680923; C(111)=-
0.000714302 ;
C(112)=-0.000747204; C(113)=-0.000779152; C(114)=-0.000809669; C(115)=-
0.000838757 ;
C(116)=-0.000866413; C(117)=-0.000891685; C(118)=-0.000915051; C(119)=-
0.000935555 ;
C(120)=-0.000954151; C(121)=-0.000968933; C(122)=-0.000980854; C(123)=-
0.000989437 ;
C(124)=-0.000994205; C(125)=-0.000995159; C(126)=-0.000991821; C(127)=-
0.000983715 ;
C(128)= 0.000971317; C(129)= 0.000953674; C(130)= 0.000930786; C(131)= 0.000902653
;
C(132)= 0.000868797; C(133)= 0.000829220; C(134)= 0.000783920; C(135)= 0.000731945
;
C(136)= 0.000674248; C(137)= 0.000610352; C(138)= 0.000539303; C(139)= 0.000462532
;
C(140)= 0.000378609; C(141)= 0.000288486; C(142)= 0.000191689; C(143)= 0.000088215
;
C(144)=-0.000021458; C(145)=-0.000137329; C(146)=-0.000259876; C(147)=-
0.000388145 ;
C(148)=-0.000522137; C(149)=-0.000661850; C(150)=-0.000806808; C(151)=-
0.000956535 ;
C(152)=-0.001111031; C(153)=-0.001269817; C(154)=-0.001432419; C(155)=-
0.001597881 ;
C(156)=-0.001766682; C(157)=-0.001937389; C(158)=-0.002110004; C(159)=-
0.002283096 ;
C(160)=-0.002457142; C(161)=-0.002630711; C(162)=-0.002803326; C(163)=-
0.002974033 ;
C(164)=-0.003141880; C(165)=-0.003306866; C(166)=-0.003467083; C(167)=-
0.003622532 ;
C(168)=-0.003771782; C(169)=-0.003914356; C(170)=-0.004048824; C(171)=-
0.004174709 ;
C(172)=-0.004290581; C(173)=-0.004395962; C(174)=-0.004489899; C(175)=-
0.004570484 ;

C(176)=-0.004638195; C(177)=-0.004691124; C(178)=-0.004728317; C(179)=-0.004748821 ;
C(180)=-0.004752159; C(181)=-0.004737377; C(182)=-0.004703045; C(183)=-0.004649162 ;
C(184)=-0.004573822; C(185)=-0.004477024; C(186)=-0.004357815; C(187)=-0.004215240 ;
C(188)=-0.004049301; C(189)=-0.003858566; C(190)=-0.003643036; C(191)=-0.003401756 ;
C(192)= 0.003134727; C(193)= 0.002841473; C(194)= 0.002521515; C(195)= 0.002174854 ,
C(196)= 0.001800537; C(197)= 0.001399517; C(198)= 0.000971317; C(199)= 0.000515938 ,
C(200)= 0.000033379; C(201)=-0.000475883; C(202)=-0.001011848; C(203)=-0.001573563 ;
C(204)=-0.002161503; C(205)=-0.002774239; C(206)=-0.003411293; C(207)=-0.004072189 ;
C(208)=-0.004756451; C(209)=-0.005462170; C(210)=-0.006189346; C(211)=-0.006937027 ;
C(212)=-0.007703304; C(213)=-0.008487225; C(214)=-0.009287834; C(215)=-0.010103703 ;
C(216)=-0.010933399; C(217)=-0.011775017; C(218)=-0.012627602; C(219)=-0.013489246 ;
C(220)=-0.014358521; C(221)=-0.015233517; C(222)=-0.016112804; C(223)=-0.016994476 ;
C(224)=-0.017876148; C(225)=-0.018756866; C(226)=-0.019634247; C(227)=-0.020506859 ;
C(228)=-0.021372318; C(229)=-0.022228718; C(230)=-0.023074150; C(231)=-0.023907185 ;
C(232)=-0.024725437; C(233)=-0.025527000; C(234)=-0.026310921; C(235)=-0.027073860 ;
C(236)=-0.027815342; C(237)=-0.028532982; C(238)=-0.029224873; C(239)=-0.029890060 ;
C(240)=-0.030526638; C(241)=-0.031132698; C(242)=-0.031706810; C(243)=-0.032248020 ;
C(244)=-0.032754898; C(245)=-0.033225536; C(246)=-0.033659935; C(247)=-0.034055710 ;
C(248)=-0.034412861; C(249)=-0.034730434; C(250)=-0.035007000; C(251)=-0.035242081 ;
C(252)=-0.035435200; C(253)=-0.035586357; C(254)=-0.035694122; C(255)=-0.035758972 ;
C(256)= 0.035780907; C(257)= 0.035758972; C(258)= 0.035694122; C(259)= 0.035586357 ,
C(260)= 0.035435200; C(261)= 0.035242081; C(262)= 0.035007000; C(263)= 0.034730434 ,
C(264)= 0.034412861; C(265)= 0.034055710; C(266)= 0.033659935; C(267)= 0.033225536 ,
C(268)= 0.032754898; C(269)= 0.032248020; C(270)= 0.031706810; C(271)= 0.031132698 ,
C(272)= 0.030526638; C(273)= 0.029890060; C(274)= 0.029224873; C(275)= 0.028532982

C(276)= 0.027815342; C(277)= 0.027073860; C(278)= 0.026310921; C(279)= 0.025527000
,
C(280)= 0.024725437; C(281)= 0.023907185; C(282)= 0.023074150; C(283)= 0.022228718
,
C(284)= 0.021372318; C(285)= 0.020506859; C(286)= 0.019634247; C(287)= 0.018756866
,
C(288)= 0.017876148; C(289)= 0.016994476; C(290)= 0.016112804; C(291)= 0.015233517
,
C(292)= 0.014358521; C(293)= 0.013489246; C(294)= 0.012627602; C(295)= 0.011775017
,
C(296)= 0.010933399; C(297)= 0.010103703; C(298)= 0.009287834; C(299)= 0.008487225
,
C(300)= 0.007703304; C(301)= 0.006937027; C(302)= 0.006189346; C(303)= 0.005462170
,
C(304)= 0.004756451; C(305)= 0.004072189; C(306)= 0.003411293; C(307)= 0.002774239
,
C(308)= 0.002161503; C(309)= 0.001573563; C(310)= 0.001011848; C(311)= 0.000475883
,
C(312)=-0.000033379; C(313)=-0.000515938; C(314)=-0.000971317; C(315)=-
0.001399517 ;
C(316)=-0.001800537; C(317)=-0.002174854; C(318)=-0.002521515; C(319)=-
0.002841473 ;
C(320)= 0.003134727; C(321)= 0.003401756; C(322)= 0.003643036; C(323)= 0.003858566
,
C(324)= 0.004049301; C(325)= 0.004215240; C(326)= 0.004357815; C(327)= 0.004477024
,
C(328)= 0.004573822; C(329)= 0.004649162; C(330)= 0.004703045; C(331)= 0.004737377
,
C(332)= 0.004752159; C(333)= 0.004748821; C(334)= 0.004728317; C(335)= 0.004691124
,
C(336)= 0.004638195; C(337)= 0.004570484; C(338)= 0.004489899; C(339)= 0.004395962
,
C(340)= 0.004290581; C(341)= 0.004174709; C(342)= 0.004048824; C(343)= 0.003914356
,
C(344)= 0.003771782; C(345)= 0.003622532; C(346)= 0.003467083; C(347)= 0.003306866
,
C(348)= 0.003141880; C(349)= 0.002974033; C(350)= 0.002803326; C(351)= 0.002630711
,
C(352)= 0.002457142; C(353)= 0.002283096; C(354)= 0.002110004; C(355)= 0.001937389
,
C(356)= 0.001766682; C(357)= 0.001597881; C(358)= 0.001432419; C(359)= 0.001269817
,
C(360)= 0.001111031; C(361)= 0.000956535; C(362)= 0.000806808; C(363)= 0.000661850
,
C(364)= 0.000522137; C(365)= 0.000388145; C(366)= 0.000259876; C(367)= 0.000137329
,
C(368)= 0.000021458; C(369)=-0.000088215; C(370)=-0.000191689; C(371)=-
0.000288486 ;
C(372)=-0.000378609; C(373)=-0.000462532; C(374)=-0.000539303; C(375)=-
0.000610352 ;

C(376)=-0.000674248; C(377)=-0.000731945; C(378)=-0.000783920; C(379)=-
0.000829220
C(380)=-0.000868797; C(381)=-0.000902653; C(382)=-0.000930786; C(383)=-
0.000953674 ;
C(384)= 0.000971317; C(385)= 0.000983715; C(386)= 0.000991821; C(387)= 0.000995159
,
C(388)= 0.000994205; C(389)= 0.000989437; C(390)= 0.000980854; C(391)= 0.000968933
,
C(392)= 0.000954151; C(393)= 0.000935555; C(394)= 0.000915051; C(395)= 0.000891685
,
C(396)= 0.000866413; C(397)= 0.000838757; C(398)= 0.000809669; C(399)= 0.000779152
,
C(400)= 0.000747204; C(401)= 0.000714302; C(402)= 0.000680923; C(403)= 0.000646591
,
C(404)= 0.000611782; C(405)= 0.000576973; C(406)= 0.000542164; C(407)= 0.000507355
,
C(408)= 0.000472546; C(409)= 0.000438213; C(410)= 0.000404358; C(411)= 0.000371456
,
C(412)= 0.000339031; C(413)= 0.000307560; C(414)= 0.000277042; C(415)= 0.000247478
,
C(416)= 0.000218868; C(417)= 0.000191212; C(418)= 0.000165462; C(419)= 0.000140190
,
C(420)= 0.000116348; C(421)= 0.000093937; C(422)= 0.000072956; C(423)= 0.000052929
,
C(424)= 0.000034332; C(425)= 0.000017166; C(426)= 0.000000954; C(427)=-0.000013828
,
C(428)=-0.000027180; C(429)=-0.000039577; C(430)=-0.000050545; C(431)=-
0.000060558 ;
C(432)=-0.000069618; C(433)=-0.000077724; C(434)=-0.000084400; C(435)=-
0.000090122 ;
C(436)=-0.000095367; C(437)=-0.000099182; C(438)=-0.000102520; C(439)=-
0.000105381 ;
C(440)=-0.000106812; C(441)=-0.000108242; C(442)=-0.000108719; C(443)=-
0.000108719 ;
C(444)=-0.000108242; C(445)=-0.000107288; C(446)=-0.000105858; C(447)=-
0.000103951 ;
C(448)= 0.000101566; C(449)= 0.000099182; C(450)= 0.000096321; C(451)= 0.000093460
,
C(452)= 0.000090599; C(453)= 0.000087261; C(454)= 0.000083923; C(455)= 0.000080585
,
C(456)= 0.000076771; C(457)= 0.000073433; C(458)= 0.000070095; C(459)= 0.000066280
,
C(460)= 0.000062943; C(461)= 0.000059605; C(462)= 0.000055790; C(463)= 0.000052929
,
C(464)= 0.000049591; C(465)= 0.000046253; C(466)= 0.000043392; C(467)= 0.000040531
,
C(468)= 0.000037670; C(469)= 0.000034809; C(470)= 0.000032425; C(471)= 0.000030041
,
C(472)= 0.000027657; C(473)= 0.000025272; C(474)= 0.000023365; C(475)= 0.000021458

C(476)= 0.000019550; C(477)= 0.000018120; C(478)= 0.000016689; C(479)= 0.000014782

C(480)= 0.000013828; C(481)= 0.000012398; C(482)= 0.000011444; C(483)= 0.000010014

C(484)= 0.000009060; C(485)= 0.000008106; C(486)= 0.000007629; C(487)= 0.000006676

C(488)= 0.000006199; C(489)= 0.000005245; C(490)= 0.000004768; C(491)= 0.000004292

C(492)= 0.000003815; C(493)= 0.000003338; C(494)= 0.000003338; C(495)= 0.000002861

C(496)= 0.000002384; C(497)= 0.000002384; C(498)= 0.000001907; C(499)= 0.000001907

C(500)= 0.000001431; C(501)= 0.000001431; C(502)= 0.000000954; C(503)= 0.000000954

C(504)= 0.000000954; C(505)= 0.000000954; C(506)= 0.000000477; C(507)= 0.000000477

C(508)= 0.000000477; C(509)= 0.000000477; C(510)= 0.000000477; C(511)= 0.000000477

```
% Arreglo: insertar el coeficiente nulo al comienzo C(1)=0.0... C(512)=0.000000477
C = [0.0 C];

% Aquí las muestras mas recientes tendrán un indice mas alto
% que la primera muestra, formándose un matriz columna (usando el comando flipud)
% C(1)=0.000000477
%  .
%  .
% C(512)=0

C = flipud(C(:)); % reflejo de una matriz
```

e. Analisis_filtro_subbanda.m

```
function S = Analisis_filtro_subbanda(Input, n, C)
% S = Analisis_filtro_subbanda(Input, n, C)
%   Devuelve las 32 sub-bandas muestreadas S(i) definidas en el estandár
%   es decir una muestra para cada sub-banda
%   Input es la señal de entrada de audio x
%   n es el indice en la entrada donde las 32 nuevas muestras estan localizadas.
%   C es la ventana de análisis definida en en el estándar
%
%   Ver también Tabla_ventana_analisis

%---------------------------------------------------------------------------
Common;

% Longitud de input (señal de entrada) =44101
nmax = length(Input);

% Verificar parámetros de entrada
if (n + 31 > nmax | n < 1)
```

```
  error('Indice de analisis inesperado.');
end

% Construir un vector de entrada X de 512 elementos. la muestra mas reciente
% esta en la posición 512 mientras los elementos mas antiguos estan en la posición 1.
% Completar con ceros si la señal de entrada no existe.
% ..........................................................
%       |       480 muestras      | 32 muestras  |
%       n-480                     n          n+31
X = Input(max(1, n - 480):n + 31); % / 32768
X = X(:);
X = [zeros(512 - length(X), 1); X]; % X de 512 filas 1 columna (480 ceros y 32 valores)

% Vector ventana X por vector C. Esto produce el Buffer Z.
Z = X .* C;

% Calculo parcial: 64 Yi coeficientes
Y = zeros(1, 64);
for i = 1 : 64,
  for j = 0 : 7,
    Y(i) = Y(i) + Z(i + 64 * j);
  end
end

% Calcular los coeficientes del banco de filtros
% M es una matriz de 32 filas por 64 columnas
for i = 0 : 31,
  for k = 0 : 63,
    M(i + 1, k + 1) = cos((2 * i + 1) * (k - 16) * pi / 64);
  end
end

% Calcular las 32 sub-bandas muestredas S(i). Salida del banco de filtros
% genera una matriz de 1 fila por 32 columnas S=[S(1) ....S(32)]
S = zeros(1, 32);
for i = 1 : 32,
  for k = 1 : 64,
    S(i) = S(i) + M(i, k) * Y(k);
  end
end
```

f. Factores_escala.m

```
function scf = Factores_escala(S)
%scf = Factores_escala(S)
%   Para cada sub-banda S(i, :), el maximo valor absoluto de 12
%   muestras S(i, 1) ... S(i, 12) se determina. Es escogido el siguiente valor
%   mas grande del factor de escala en la tabla de factores de escala
%
%
```

```
%-------------------------------------------------------------------------

% Tabla de acuerdo al estándar

Table_scf = [
  2.00000000000000; 1.58740105196820; 1.25992104989487; 1.00000000000000;
  0.79370052598410; 0.62996052494744; 0.50000000000000; 0.39685026299205;
  0.31498026247372; 0.25000000000000; 0.19842513149602; 0.15749013123686;
  0.12500000000000; 0.09921256574801; 0.07874506561843; 0.06250000000000;
  0.04960628287401; 0.03937253280921; 0.03125000000000; 0.02480314143700;
  0.01968626640461; 0.01562500000000; 0.01240157071850; 0.00984313320230;
  0.00781250000000; 0.00620078535925; 0.00492156660115; 0.00390625000000;
  0.00310039267963; 0.00246078330058; 0.00195312500000; 0.00155019633981;
  0.00123039165029; 0.00097656250000; 0.00077509816991; 0.00061519582514;
  0.00048828125000; 0.00038754908495; 0.00030759791257; 0.00024414062500;
  0.00019377454248; 0.00015379895629; 0.00012207031250; 0.00009688727124;
  0.00007689947814; 0.00006103515625; 0.00004844363562; 0.00003844973907;
  0.00003051757813; 0.00002422181781; 0.00001922486954; 0.00001525878906;
  0.00001211090890; 0.00000961243477; 0.00000762939453; 0.00000605545445;
  0.00000480621738; 0.00000381469727; 0.00000302772723; 0.00000240310869;
  0.00000190734863; 0.00000151386361; 0.00000120155435; 1E-20
];

N = length(Table_scf); % 63 filas

for i = 1:32,
  si_min = max(abs(S(:, i)));
  if (si_min > Table_scf(1))
    % Esto sucede usualmente cuando la señal de entrada tiene valores
    % mas grandes que uno.
    disp('Advertencia: no se puede encontrar factor de escala.');
    scf(i) = Table_scf(1);
  else
    j = 0;
    while (j < N & si_min > Table_scf(N - j))
      j = j + 1;
    end
    % genera 32 valores scf(i) para cada sub-banda
        scf(i) = Table_scf(N - j);
  end
end
```

g. Analisis_FFT.m

```
function [X, Delta]  = Analisis_FFT(Input, n)
%X = Analisis_FFT(Input, n)
%
%  Calcula el espectro auditivo usando la Transforma Rápida de Fourier.
%  El espectro X se expresa en dB. El tamaño de la transformada es 512 y
%  es centrada sobre las 384 muestras (12 muestras por sub-banda) usadas por el
%  analisis de sub-banda. la primera de las 384 muestras es indexada en n:
```

```
%X = Nivel_presion_sonido(X, scf)
%
%   El nivel de presión del sonido Lsb es calculado para cada sub-banda.
%   X es la densidad espectral de potencia normalizada y scf son los 32 factores
%   de escala (uno por banda).
%
%   Ver también  Factores_escala

%--------------------------------------------------------------------------
% Carga los valores FFT_SIZE=512, N_SUBBAND=32
Common;


% Verificar paámetros de entrada
if (length(X) ~= FFT_SIZE)
  error('Tamaño inesperado de densidad espectral de potencia.');
end

if (length(scf) ~= N_SUBBAND)
  error('Numero inesperado de factores de escala');
end

% Recoge los 32 niveles de presión de sonido de la linea espectral con maxima amplitud
% en el rango de frecuencia correspondiente a la sub-banda i y calcula el
% nivel de presión del sonido Lsb.
Xmin = min(X);
n = FFT_SIZE / 2 / N_SUBBAND; % Tamaño de cada sub-banda(8)

for i = 1:N_SUBBAND,
  local_max = Xmin;
  for j = 1:n,
    local_max = max(X((i - 1) * n + j), local_max);
  end
  Lsb(i) = max(local_max, 20 * log10(scf(i) * 32768) - 10);
end

% Lsb (i) es calculado para cada una de las 32 sub-bandas los -10 db en la expresión
% corrigen la diferencia entre el valor pico y el RMS
```

i. Hallar_componentes tonales.m

```
function [Flags, Tonal_list, Non_tonal_list] = ...
  Hallar_componentes_tonales(X, TH, Map, CB) % asumir fs = 44100
%[Flags, Tonal_list, Non_tonal_list] = Hallar_componentes_tonales(X, TH, Map, CB)
%
%   Identifica y lista las componentes tonal y no-tonal de la señal de audio
%   Se asume en esta implementación que la frecuencia de muestreo
%   fs es 44100 Hz.
%
%   Ver también Decimacion
```

```
%  ..................................................
%    |     | 384 muestras   |      |
%    n-64   n              n+383   n+447
%
%  Una ventana de Hanning se aplica antes de calcular la FFT.
%
%  Finalmente, se realiza una normalización del nivel de presión del sonido
%  de modo que el valor máximo del espectro sea de 96dB;
%  el número de dB añadidos es almacenado en la salida Delta
%
%  Se debe tener cuidado que la entrada no sea cero en todas las muestras.
%  De otra manera W será -INF para todas las muestras.

%--------------------------------------------------------------------------------
Common;

Delta = [];
X = [];

N = length(Input);

% Verifica parámetros de entrada
if (n + FFT_SHIFT - FFT_OVERLAP > N | n < 1)
  error('Entrada demasiado pequeña: No existen suficientes muestras para calcular la FFT.');
end

% Preparar las muestras usadas por la FFT
% Añadir ceros de relleno si las muestras estan perdidas
s = Input(max(1, n - FFT_OVERLAP):min(N, n + FFT_SIZE - FFT_OVERLAP - 1));
s = s(:);

if (n - FFT_OVERLAP < 1)
  s = [zeros(FFT_OVERLAP - n + 1, 1); s];
end
if (N < n - FFT_OVERLAP + FFT_SIZE - 1)
  s = [s; zeros(n - FFT_OVERLAP + FFT_SIZE - 1 - N, 1)];
end

% Preparar la ventana de Hanning
h = sqrt(8/3) * hanning(512, 'periodic');

% Densidad espectral de potencia. X matriz de 512 filas 1 columna
X = max(20 * log10(abs(fft(s .* h)) / FFT_SIZE), MIN_POWER);

% Normalización al nivel de presión de sonido en 96 dB
Delta = 96 - max(X);
X = X +  Delta;
```

h. Nivel_presion_sonido.m

```
function Lsb = Nivel_presion_sonido(X, scf)
```

```
%-------------------------------------------------------------------------
% Carga los valores FFT_SIZE=512, INDEX=1, SPL=2, NOT_EXAMINED=0,
IRRELEVANT=3, MIN_POWER=-200,TONAL=1,NON_TONAL=2
% DRAW=1
Common;

% Verifica los parámetros de entrada. Longitud de X es 512
if (length(X) ~= FFT_SIZE)
  error('Tamaño inesperado de la densidad espectral de potencia.');
end

if (DRAW),
  t = 1:length(X);
end

% Listar los flags para todas las lineas de frecuencia (1 a FFT_SIZE / 2)
Flags = zeros(FFT_SIZE / 2, 1) + NOT_EXAMINED;

% Etiqueta la maxima local
% Una linea espectral es etiquetada como máxima si: X(k) > X(k-1) y X(k) >= X(k+1)
% y si el indice k se encuentra dentro del rango k > 2 & k <= 250
local_max_list = [];
counter = 1;
for k = 2:FFT_SIZE / 2 -  1, % No tenga cuidado de los límites ( k de <2-255>)
  if (X(k) > X(k-1) & X(k) >= X(k+1) & k > 2 & k <= 250)
    local_max_list(counter, INDEX) = k;
    local_max_list(counter, SPL) = X(k);
    counter = counter + 1;
  end
end

%Graficando los valores maximos locales
if (DRAW),
  disp('Local maxima.');
  plot(t, X(t), local_max_list(:, INDEX), local_max_list(:, SPL), 'ko');
  xlabel('Indice de Frecuencia'); ylabel('dB'); title('Local máxima.');
  axis([0 256 0 100]); pause;
end

% Listar las componentes tonales y calcular el nivel de presión del sonido
% Un maximo local se coloca en la lista de componentes tonales si X(k) - X(k + j) >= 7dB
% donde j se escoge de acuerdo a:
% Capa I:
%   j=-2,+2                              para  2<k<63
%   j=-3,-2,+2,+3                        para  63<=k<127
%   j=-6,...,-2,+2,...,+6                para  127<=k<=250

Tonal_list = [];
counter = 1;
if not(isempty(local_max_list))
    for i = 1:length(local_max_list(:, 1)),
```

```
        k = local_max_list(i, INDEX);
        is_tonal = 1;

        % Capa I
        % Examinar las frecuencias vecinas
        if (2 < k & k < 63)
          J = [-2 2];
        elseif (63 <= k & k < 127)
          J = [-3 -2 2 3];
        elseif (127 <= k & k < 250)
          J = [-6:-2, 2:6];
        else
          is_tonal = 0;
        end

        for j = J,
          is_tonal = is_tonal & (X(k) - X(k + j) >= 7);
        end

        % Si X(k) es en realidad una componente tonal entonces los siguientes datos son
listados
        %    - número de indice k de la línea espectral
        %    - nivel de presión del sonido: Xtm(k)=X(k-1)+X(k)+X(k+1)
        %    - fijar flag tonal
        if is_tonal
          Tonal_list(counter, INDEX) = k;
          Tonal_list(counter, SPL) = 10 * log10(10^(X(k - 1) / 10) + ...
            10^(X(k) / 10) + 10^(X(k + 1) / 10));
          Flags(k) = TONAL;
          for j = [J -1 1],
            Flags(k + j) = IRRELEVANT;
          end
          counter = counter + 1;
        end
    end
        if (DRAW),
          disp('Componentes tonales');
          plot(t, X(t), Tonal_list(:, INDEX), Tonal_list(:, SPL), 'ro');
          xlabel('Indice de Frecuencia'); ylabel('dB'); title('Componentes tonales');
          axis([0 256 0 100]); pause;
        end
end
% Lista las componentes no tonales y calcula la potencia
% Todas las líneas espectrales que no han sido examinadas durante el proceso
% de busqueda son sumadas para formar la componente no tonal.
Non_tonal_list = [];
counter = 1;
for i = 1:length(CB(:, 1)) - 1,,
  % Para cada banda crítica, calcula la potencia
  % en las componentes no-tonales
  power  = MIN_POWER; % Suma Parcial
```

```
  weight = 0; % Usada para calcular la media geométrica de la banda crítica
  for k = TH(CB(i), INDEX):TH(CB(i + 1), INDEX) - 1, % En cada banda crítica
    if (Flags(k) == NOT_EXAMINED),
      power    = 10 * log10(10^(power / 10) + 10^(X(k) / 10));
      weight   = weight + 10^(X(k) / 10) * (TH(Map(k), BARK) - TH(CB(i), BARK)');
      Flags(k) = IRRELEVANT;
    end
  end

  % El numero de índice para la componente no tonal es el índice mas cercano
  % a la media geométrica de la banda crítica
  if (power <= MIN_POWER)
    index = round(mean(TH(CB(i), INDEX), TH(CB(i + 1), INDEX)));
  else
    index = TH(CB(i), INDEX) + round(weight / 10^(power / 10) * ...
      (TH(CB(i + 1), INDEX) - TH(CB(i), INDEX)));
  end
  if (index < 1)
    index = 1;
  end
  if (index > length(Flags))
    index = length(Flags);
  end
  if (Flags(index) == TONAL)
    index = index + 1; % Dos componentes tonales no pueden ser consecutivas
  end

  % Para cada sub-banda
  %   - indice de la componente no-tonal
  %   - nivel de presión de sonido de este componente
  Non_tonal_list(i, INDEX) = index;
  Non_tonal_list(i, SPL) = power;
  Flags(index) = NON_TONAL;
end

if (DRAW),
  disp('Componentes Tonales y No-tonales');
  plot(t, X(t), Tonal_list(:, INDEX), Tonal_list(:, SPL), 'ro', ...
    Non_tonal_list(:, INDEX), Non_tonal_list(:, SPL), 'go');
  xlabel('Indice de Frecuencia'); ylabel('dB');
  title('Componentes Tonales y No-tonales');
  axis([0 256 0 100]); pause;
end
```

j. Decimacion.m

```
function [DFlags, DTonal_list, DNon_tonal_list] = ...
  Decimacion(X, Tonal_list, Non_tonal_list, Flags, TH, Map)
%[DFlags, DTonal_list, DNon_tonal_list] = ...
%   Decimacion(X, Tonal_list, Non_tonal_list, Flags, TH, Map)
%
```

```
%   las Componentes las cuales estan por debajo del umbral auditivo o son menores
%   que la mitad del ancho de banda crítica de un componente vecino  son eliminadas.
%   DFlags, DTonal_list and DNon_tonal_list
%   contienen la lista de banderas, componentes tonales y no-tonales después de
%   la decimación.
%
%   Ver también Hallar_componentes_tonales

%-------------------------------------------------------------------------------
% Carga los valores INDEX=1, SPL=2, ATH=2, IRRELEVANT=3
Common;

if (DRAW),
  t = 1:length(X);
end

DFlags = Flags; % Flags después de la decimación


% las componentes Tonales o no-tonales son considerados si son menores que
% el umbral absoluto de escucha encontrado en TH(:, ATH).

% Caso No tonal y su gráfica
% Paso 5a
%
DNon_tonal_list = [];
if not(isempty(Non_tonal_list))
     for i = 1:length(Non_tonal_list(:, 1)),
       k = Non_tonal_list(i, INDEX);
       %if (k > length(Map))
       %   DFlags(k) = IRRELEVANT;
       %else
       if (Non_tonal_list(i, SPL) < TH(Map(k), ATH))
            DFlags(k) = IRRELEVANT;
         else
           DNon_tonal_list = [DNon_tonal_list; Non_tonal_list(i, :)];
       end
       %end
     end
     if (DRAW),
       disp('Componentes No-tonales v. umbral absoluto.');
       plot(t, X(t), Non_tonal_list(:, INDEX), Non_tonal_list(:, SPL), 'go', ...
         DNon_tonal_list(:, INDEX), DNon_tonal_list(:, SPL), 'yo', ...
         TH(:, INDEX), TH(:, ATH));
       xlabel('Indice de Frecuencia'); ylabel('dB');
       title('Componentes No-tonales v. umbral absoluto.');
       axis([0 256 -20 100]); pause;
     end
end
```

```
% Caso Tonal (primera parte) y su gráfica
% Paso 5a
DTonal_list = [];
if not(isempty(Tonal_list))
    for i = 1:length(Tonal_list(:, 1)),
      k = Tonal_list(i, INDEX);
      %if (k > length(Map))
      %   DFlags(k) = IRRELEVANT;
      %else
      if (Tonal_list(i, SPL) < TH(Map(k), ATH))
      DFlags(k) = IRRELEVANT;
      else
      DTonal_list = [DTonal_list; Tonal_list(i, :)];
      end
      %end
    end
    if (DRAW),
      disp('Componentes tonales v. umbral absoluto.');
      plot(t, X(t), Tonal_list(:, INDEX), Tonal_list(:, SPL), 'ro', ...
        DTonal_list(:, INDEX), DTonal_list(:, SPL), 'mo', TH(:, INDEX), ...
        TH(:, ATH)); axis([0 256 0 100]);
      xlabel('Indice de Frecuencia'); ylabel('dB');
      title('Componentes tonales v. umbral absoluto.');
      axis([0 256 -20 100]); pause;
    end
end

% Eliminar  las componentes tonales que son menores que la mitad (0.5 Bark) del
% ancho de la banda crítico de una componente vecina
% (segunda parte del caso tonal)
% Paso 5b
if not(isempty(DTonal_list))
   i = 1;
   while (i < length(DTonal_list(:, 1)))
      k      = DTonal_list(i, INDEX);
      k_next = DTonal_list(i + 1, INDEX);
      if (TH(Map(k_next), BARK) - TH(Map(k), BARK) < 0.5)
         if (DTonal_list(i, SPL) < DTonal_list(i + 1, SPL))
            DTonal_list = DTonal_list([1:i - 1, ...
                  i + 1:length(DTonal_list(:, 1))], :);
            DFlags(k) = IRRELEVANT;
         else
            DTonal_list = DTonal_list([1:i, ...
                  i + 2:length(DTonal_list(:, 1))], :);
            DFlags(k_next) = IRRELEVANT;
         end
      else
         i = i + 1;
      end
   end
   if (DRAW)
```

```
      disp('Componentes tonales demasiado cercanas para eliminarse entre si.');
      plot(t, X(t), Tonal_list(:, INDEX), Tonal_list(:, SPL), 'ro', ...
         DNon_tonal_list(:, INDEX), DNon_tonal_list(:, SPL), 'go', ...
         TH(:, INDEX), TH(:, ATH)); axis([0 256 0 100]);
      xlabel('Indice de Frecuencia'); ylabel('dB');
      title('Componentes tonales demasiado cercanas para eliminarse entre si.');
      axis([0 256 -20 100]); pause;
   end
end
```

k. Umbrales enmasc individual.m

```
function  [LTt, LTn] = Umbral_enmascaramiento_individual(X, Tonal_list, ...
   Non_tonal_list, TH, Map)
%[LTt, LTn] = Umbral_enmascaramiento_individual(X, Tonal_list, ...
%   Non_tonal_list, TH, Map)
%
%   Calcula el efecto de enmascaramiento de las componentes tonales y no tonales
%   en las frecuencia espectrales vecinas. la potencia del enmascarador
%   es sumada con el indice de enmascaramiento y la función de enmascaramiento.
%   LTt: umbral de enmascaramiento para las componentes tonales
%   LTn: umbral de enmascaramiento para las componentes no-tonales
%--------------------------------------------------------------------------------
% Carga los valores INDEX=1, SPL=2, BARK=2, MIN_POWER=-200, DRAW=1
Common;

% Umbral de enmascaramiento individual para componentes tonales y no-tonales
% son fijadas a -infinito puesto que la función de enmascaramiento tiene
% atenuación infinita fuera de -3 y +8 barks, esto significa que la componente
% no tiene efecto de enmascaramiento en frecuencias fuera de esos rangos
if isempty(Tonal_list)
   LTt = [];
else
   LTt = zeros(length(Tonal_list(:, 1)), length(TH(:, 1))) + MIN_POWER;
end
LTn = zeros(length(Non_tonal_list(:, 1)), length(TH(:, 1))) + MIN_POWER;

% Solamente un subconjunto de muestras son consideradas para el cálculo de
% Umbral de enmascaramiento global. El número de estas muestras depende
% de la tasa de muestreo y de la capa de codificación. Toda la información
% necesaria está en TH la que contiene las frecuencias, tasa de banda crítica
% y umbral absoluto.
for i = 1:length(TH(:, 1))
   zi = TH(i, BARK);  % Tasa de banda crítica de la frecuencia considerada

   if not(isempty(Tonal_list))
         % Para cada componente tonal
         % Umbral de enmascaramiento individual:
         % LTtm = Xtm[z(j)] + avtm[z(j)] + vf[z(j),z(i)] dB
         for k = 1:length(Tonal_list(:, 1)),
```

```
      j  = Tonal_list(k, INDEX);
      zj = TH(Map(j), BARK); % Tasa de banda crítica del enmascarador
      dz = zi - zj;          % Distancia en Bark al enmascarador

      if (dz >= -3 & dz < 8)

        % Indice de enmascaramiento para enmascaradores tonales (dB)
        avtm = -1.525 - 0.275 * zj - 4.5;

        % Función de enmascaramiento depende de la distancia en Bark al
       % enmascarador dz=z(i)-z(j)
        if (-3 <= dz & dz < -1)
          vf = 17 * (dz + 1) - (0.4 * X(j) + 6);
        elseif (-1 <= dz & dz < 0)
          vf = (0.4 * X(j) + 6) * dz;
        elseif (0 <= dz & dz < 1)
          vf = -17 * dz;
        elseif (1 <= dz & dz < 8)
          vf = - (dz - 1) * (17 - 0.15 * X(j)) - 17;
        end

        LTt(k, i) = Tonal_list(k, SPL) + avtm + vf;
      end
  end
end


% Para cada componente no-tonal
% Umbral de enmascaramiento individual:
% LTnm = Xnm[z(j)] + avnm[z(j)] + vf[z(j),z(i)] dB

  for k = 1:length(Non_tonal_list(:, 1)),
  j  = Non_tonal_list(k, INDEX);
  zj = TH(Map(j), BARK); % Tasa de banda crítica del enmascarador
  dz = zi - zj;          % Distancia en Bark al enmascarador

  if (dz >= -3 & dz < 8)

    % Indice de enmascaramiento para componentes no-tonales (dB)
    avnm = -1.525 - 0.175 * zj - 0.5;

    % Función de enmascaramiento
    if (-3 <= dz & dz < -1)
      vf = 17 * (dz + 1) - (0.4 * X(j) + 6);
    elseif (-1 <= dz & dz < 0)
      vf = (0.4 * X(j) + 6) * dz;
    elseif (0 <= dz & dz < 1)
      vf = -17 * dz;
    elseif (1 <= dz & dz < 8)
      vf = - (dz - 1) * (17 - 0.15 * X(j)) - 17;
    end
```

```
      LTn(k, i) = No|n_tonal_list(k, SPL) + avnm + vf;
    end
  end
end

% Adicionar los Umbrales de enmascaramiento individual en la gráfica actual
if (DRAW)
  if not(isempty(Tonal_list))
        hold on;
        for j = 1:length(Tonal_list(:, 1))
        plot(TH(:, INDEX), LTt(j, :), 'r:');
        end
        disp('Umbral de enmascaramiento para componentes tonales.');
    pause;
  end
        for j = 1:length(Non_tonal_list(:, 1))
          plot(TH(:, INDEX), LTn(j, :), 'g:');
        end
  hold off;
  disp('Umbral de enmascaramiento para componentes no-tonales.');
  pause;
end
```

l. Umbral_enmasc_global.m

```
function  LTg = Umbral_enmasc_global(LTq, LTt, LTn)
%LTg = Umbral_enmasc_global(LTq, LTt, LTn)
%
%  Calcula el umbral de enmascaramiento global para el subconjunto de lineas de frecuencia
%  definidas en la tabla. Es la suma en la escala de amplitud
%  normal cuadrada del espectro del umbral de enmascaramiento individual
%  y del umbral absoluto.
%  LTq : umbral de enmascaramiento absoluto
%  LTt : umbral de enmascaramiento individual para componentes tonales
%  LTn : umbral de enmascaramiento individual para componentes no-tonales
%
%  Ver también Umbrales_enmasc_individual, Tabla_umbral_absoluto

%-------------------------------------------------------------------------------
Common;

N = length(LTq(:, 1));
if not(isempty(LTt))
  m = length(LTt(:, 1));
end
n = length(LTn(:, 1));

% El enmascaramiento global es calculado por el subconjunto de frecuencias
% definidas en [1, Tabla 1.b]. Ellos son la suma de las energías
```

```
% correspondiente a los umbrales de enmascaramiento individual (LTt, LTn)
% y al umbral en reposo (LTq).
for i = 1:N

  % Umbral en reposo
  temp = 10^(LTq(i) / 10);;

  % Contribución de las componentes tonales
  if not(isempty(LTt))
     for j = 1:m,
         temp = temp + 10^(LTt(j, i) / 10);
    end
  end

  % Contribución de las componentes de ruido
  for j = 1:n,
    temp = temp + 10^(LTn(j, i) / 10);
  end

  LTg(i) = 10 * log10(temp);
end
```

m. Umbral enmasc minimo.m

```
function  LTmin = Umbral_enmasc_minimo(LTg, Map)
%LTmin = Umbral_enmasc_minimo(LTg, Map)
%
%   Encuentra el valor mínimo del umbral de enmascaramiento global para cada sub-banda.
%
%
%   Ver también Umbral_enmasc_global

%--------------- ------------------------------------------------------------
Common;

Subband_size = FFT_SIZE / 2 / N_SUBBAND;

for n = 1:N_SUBBAND, % Para cada sub-banda

  LTmin(n) = LTg(Map((n - 1) * Subband_size + 1));

  for j = 2:Subband_size, % Prueba todas las muestras en esta sub-banda
    if (LTg(Map((n - 1) * Subband_size + j)) < LTmin(n))
      LTmin(n) = LTg(Map((n - 1) * Subband_size + j));
    end
  end
end
```

n. Common.m

```
%  Este conjunto de variables 'globales' son usadas en otras funciones.
%  Ellos corresponden a la capa de Audio MPEG Capa I con modelo psicoacústico
%
%

%-------------------------------------------------------------------------

FFT_SHIFT   = 384;
FFT_SIZE    = 512;
FFT_OVERLAP = (FFT_SIZE - FFT_SHIFT) / 2;

N_SUBBAND = 32;

% Flags para análisis tonal
NOT_EXAMINED = 0;
TONAL        = 1;
NON_TONAL    = 2;
IRRELEVANT   = 3;

MIN_POWER = -200;

% Fijar a uno para ver los gráficos y explicación textual
DRAW = 1;

% Indices son usados en tablas como la tabla de umbral
% o en la lista de componentes tonales y no-tonales.
INDEX = 1;
BARK  = 2;
SPL   = 2;
ATH   = 3;
```

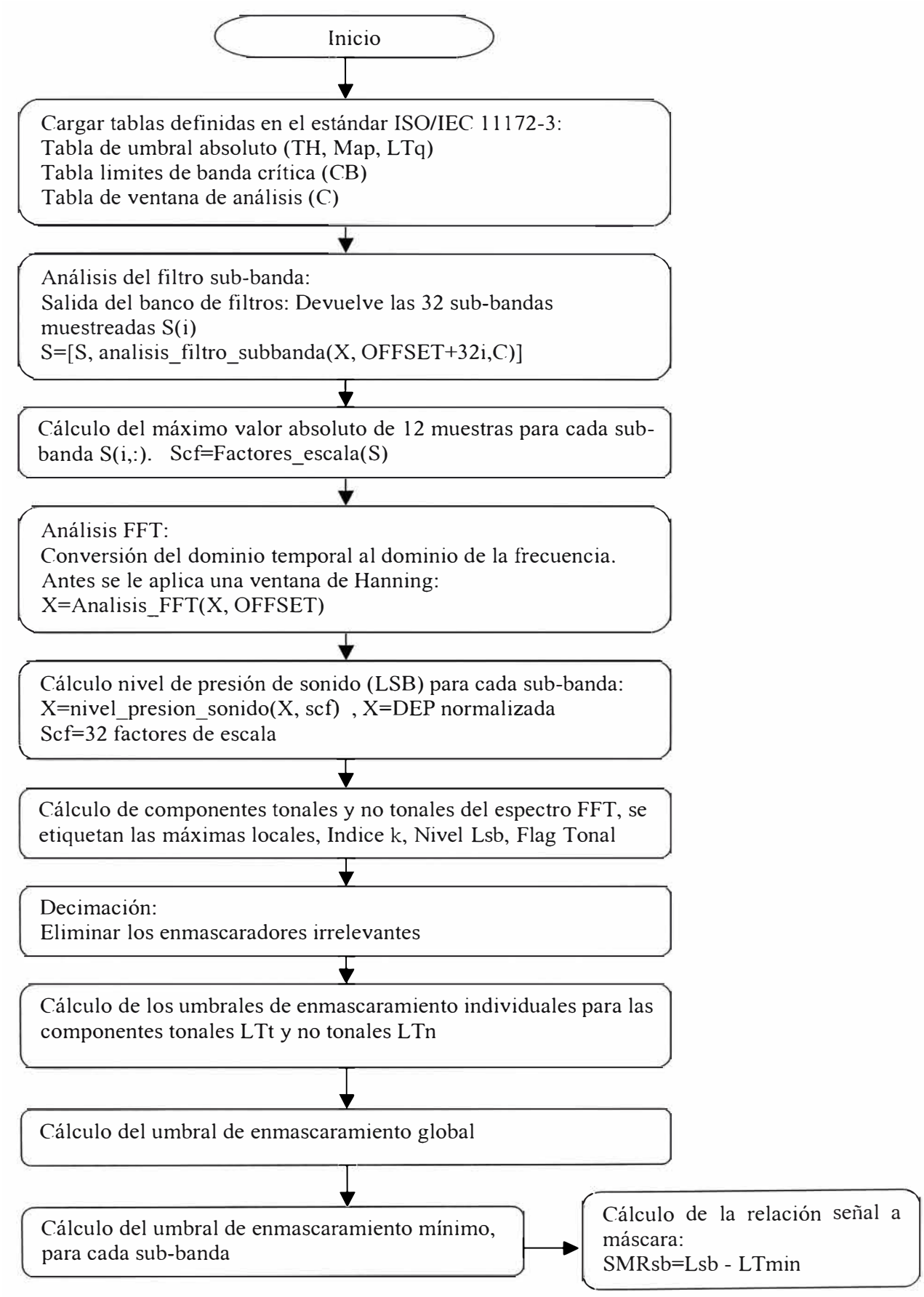

Figura 5.1 Diagrama de flujo del Programa Modelo Psicoacústico

5.2 Resultados de la Simulación

5.2.1 Valores Locales Máximos

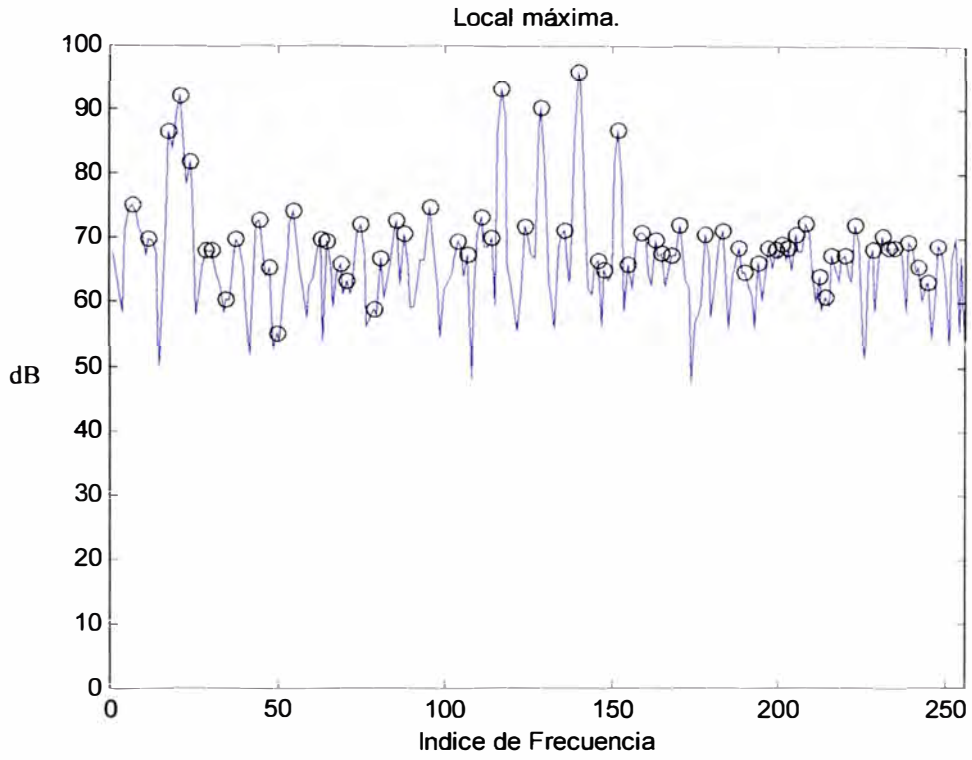

Figura 5.2 Local máxima vs. índice de frecuencia k

Una línea espectral *X(k)* es etiquetada como máximo local si:

$$X(k) > X(k-1) \quad y \quad X(k) >= X(k+1)$$

En la figura 5.2, la línea en azul representa el espectro auditivo DEP normalizado, sobre la que se pueden apreciar los máximos locales de la señal de audio (encerrados en círculos) en cada línea espectral versus el índice de frecuencia (valor *k*), el que para la capa I debe encontrarse dentro del rango $2< k <= 250$. Determinados los máximos locales, podremos calcular las componentes tonales y no tonales.

5.2.2 Componentes Tonales

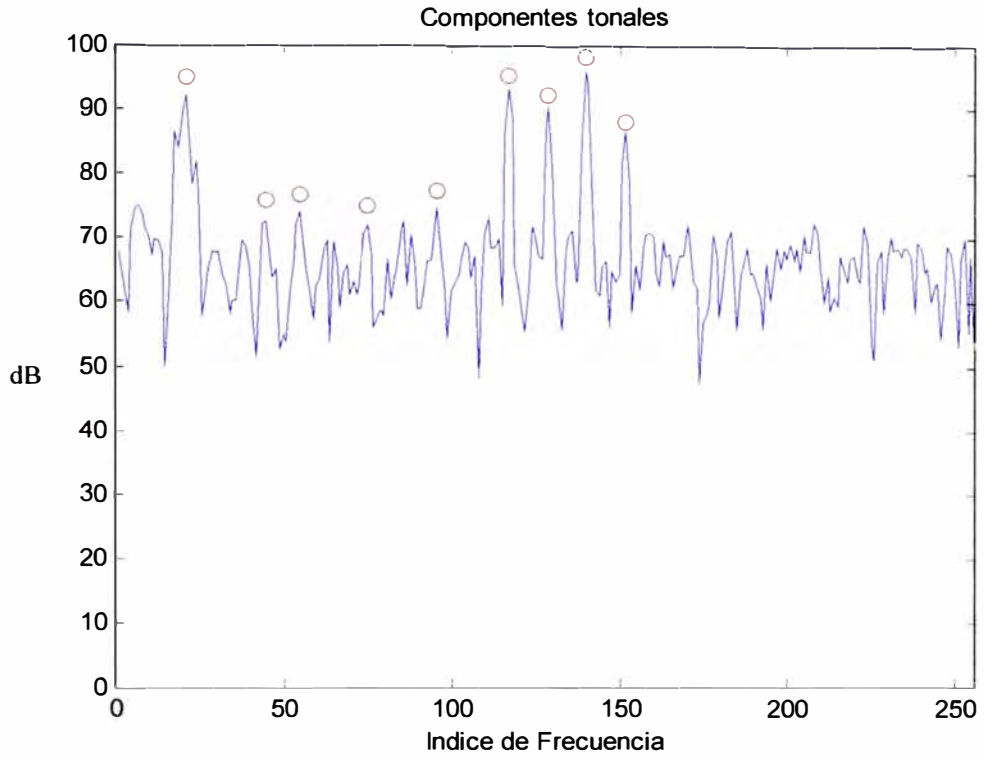

Figura 5.3 Componentes tonales vs. índice de frecuencia k

Un máximo local se coloca en la lista de componentes tonales si:

$$X(k) - X(k+j) >= 7 \text{ dB}$$

Con esta restricción no todos los valores locales máximos hallados en el paso anterior se consideran componentes tonales. En la figura 5.3 se aprecian las componentes tonales (en círculos rojos) escogidos entre los valores máximos de cada línea espectral $X(k)$ versus el índice de frecuencia (k). El nivel de presión de sonido $Xtm(k)=X(k-1)+X(k)+X(k+1)$, en dB es calculado para obtener las componentes tonales máximas.

5.2.3 Componentes Tonales y No Tonales

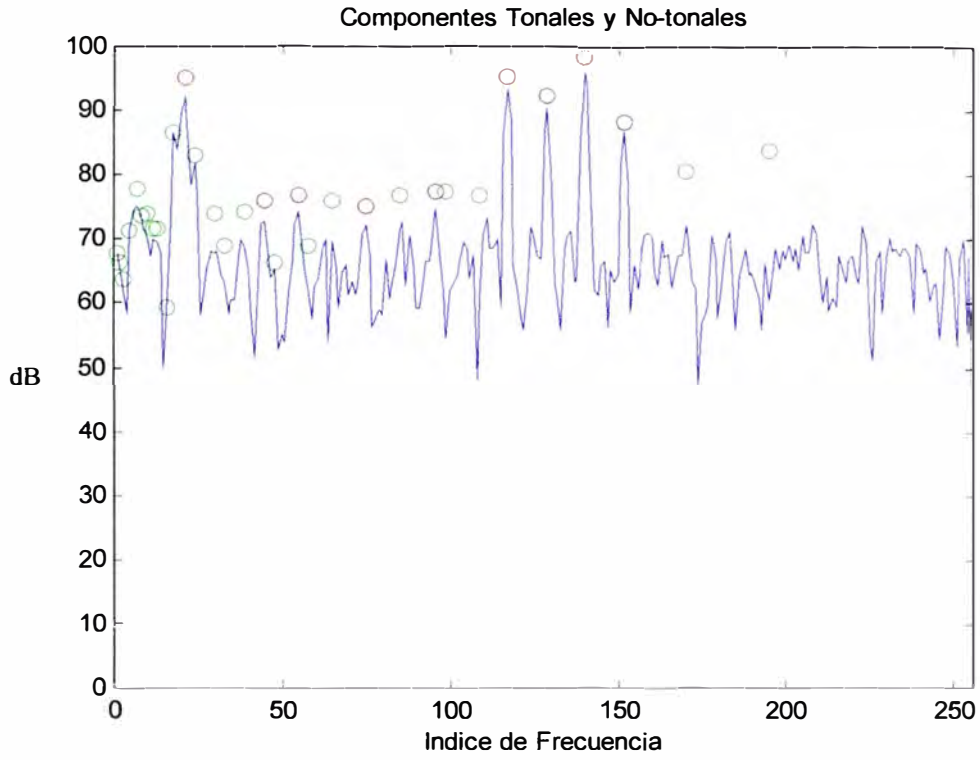

Figura 5.4 Componentes tonales y no tonales vs. índice de frecuencia k

Todas las líneas espectrales que no han sido examinadas durante el proceso de búsqueda de las componentes tonales son sumadas para formar la componente no tonal. La figura 5.4 muestra las componentes tonales (círculos rojos) y las no tonales (círculos verdes).

Para cada banda crítica, se calcula la potencia en las componentes no-tonales. El numero de índice para la componente no tonal es el índice mas cercano a la media geométrica de la banda crítica.

5.2.4 Componentes Tonales – No Tonales y Umbral Absoluto

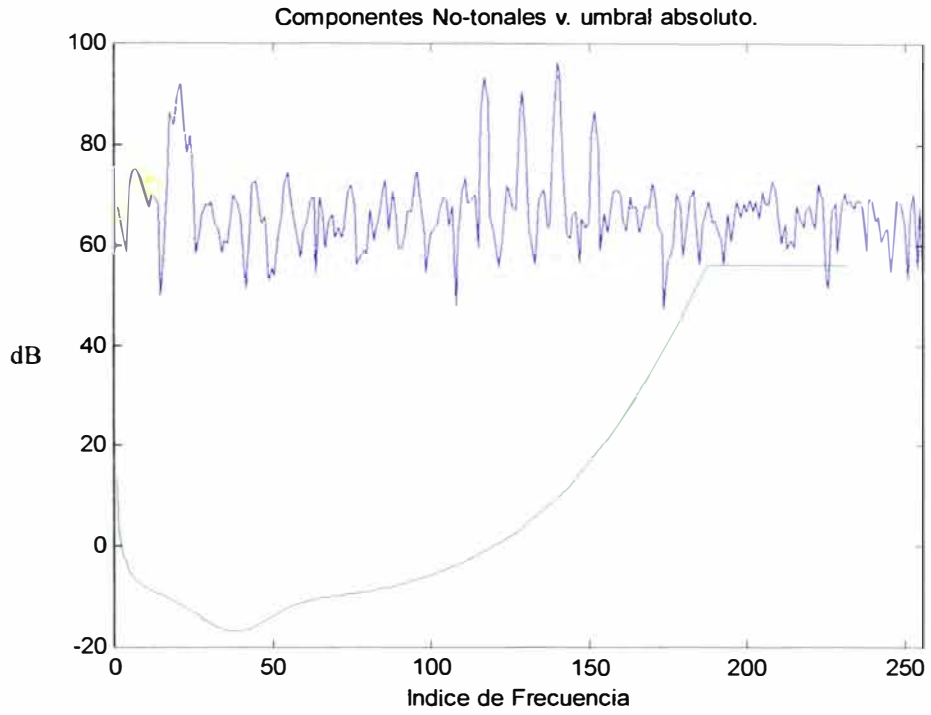

Figura 5.5 Componentes no tonales, umbral absoluto vs. índice de frecuencia k

La figura 5.5 muestra las componentes no tonales (círculos amarillos) y el umbral absoluto (línea verde). En el gráfico solo se consideran las componentes no tonales sobre el umbral absoluto, las otras componentes que están por debajo del umbral auditivo o son menores que la mitad del ancho de banda crítico de un componente vecino son eliminadas, puesto que son inaudibles, para esto se considera la siguiente relación:

$$Xnm(k) >= LTq(k)$$

Donde: Xnm = Componente no tonal.

$LTq(k)$ = Umbral absoluto

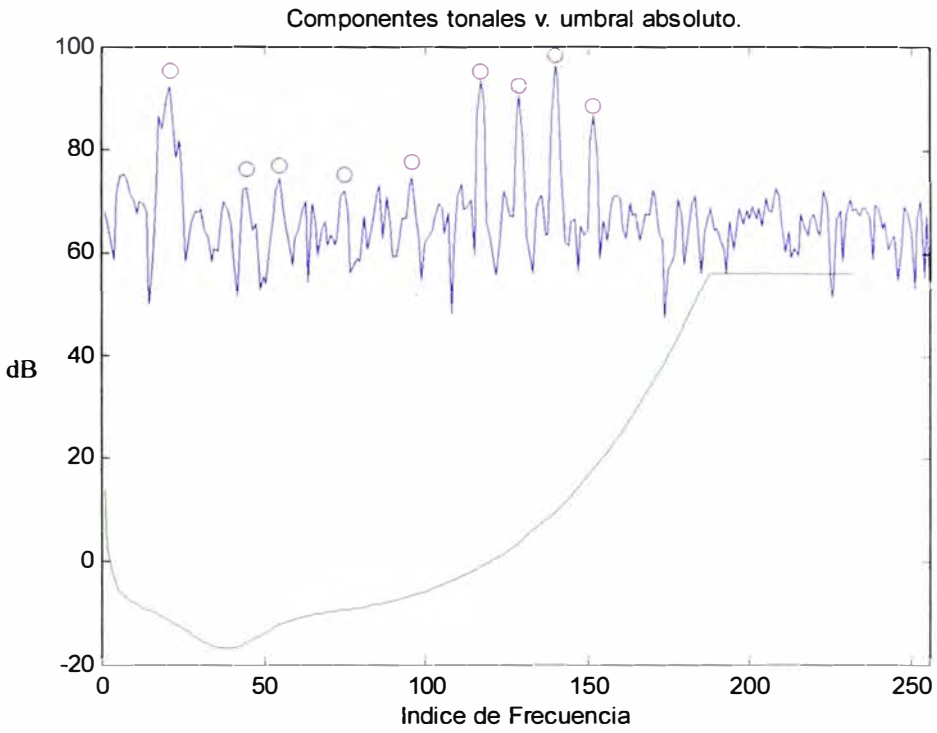

Figura 5.6 Componentes tonales, umbral absoluto vs. índice de frecuencia k

La figura 5.6 muestra las componentes tonales relevantes (círculos rosados), es decir las componentes tonales sobre el umbral absoluto (línea verde). Este caso sigue el mismo criterio del tipo no tonal, y se considera la siguiente relación:

$$Xtm(k) >= LTq(k)$$

Donde: Xtm = Componente tonal.

$LTq(k)$ = Umbral absoluto

5.2.5 Decimación de las Componentes Tonales y No Tonales

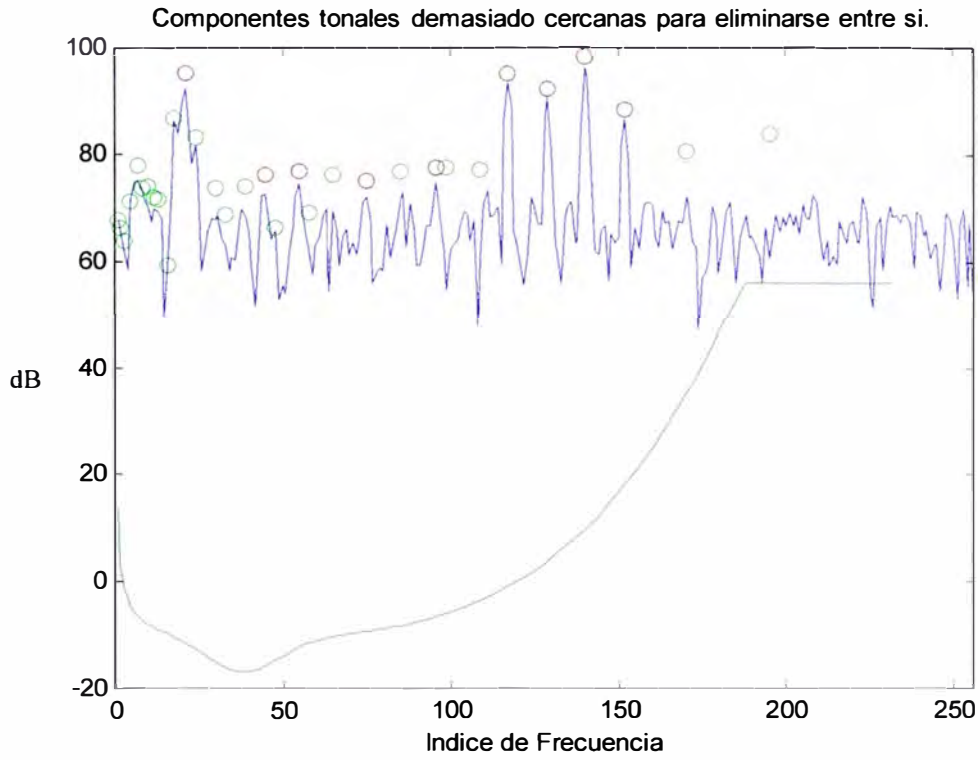

Figura 5.7 Decimación de las componentes tonales y no tonales

En la figura 6.7, se han eliminado las componentes tonales que son menores que la mitad del ancho de la banda crítico (0.5 Bark) de una componente vecina, mediante el proceso denominado "decimación de enmascaradores" que es un procedimiento usado para reducir el número de enmascaradores (componentes tonales y no tonales que enmascaran el sonido), es decir se trata de mantener las componentes con mas alta energía de la lista de componentes tonales.

5.2.6 Umbrales de Enmascaramiento Individuales Tonales

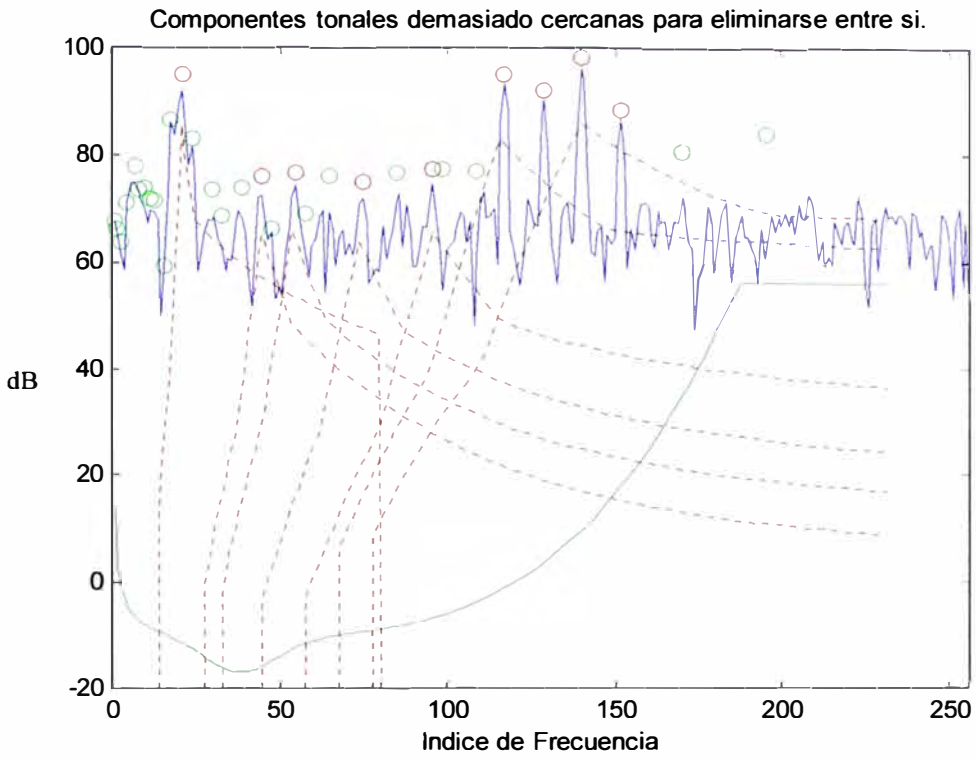

Figura 5.8 Umbrales de enmascaramiento individuales tonales

En la figura 5.8 se muestra las componentes tonales (círculos rojos), las componentes no tonales (círculos verdes), el umbral de enmascaramiento absoluto (línea verde) y los umbrales de enmascaramiento individual (líneas punteadas rojas) para cada componente tonal. Se aprecia que estos umbrales son mas anchos a medida que aumenta el valor del índice de frecuencia, comprobándose que el enmascaramiento es mas alto a altas frecuencias. El umbral de enmascaramiento individual para las componentes tonales está dado por la siguiente expresión:

$$LTtm[z(j),z(i)] = Xtm[z(j)]+avtm[z(j)]+vf[z(j),z(i)] \text{ dB}$$

Si *dz* < -3 Bark, ó *dz* >= 8 Bark, el enmascaramiento no es considerado , por lo que no se grafica (*LTtm es* fijado a –8dB fuera de este rango).

5.2.7 Umbrales de Enmascaramiento Individuales No Tonales

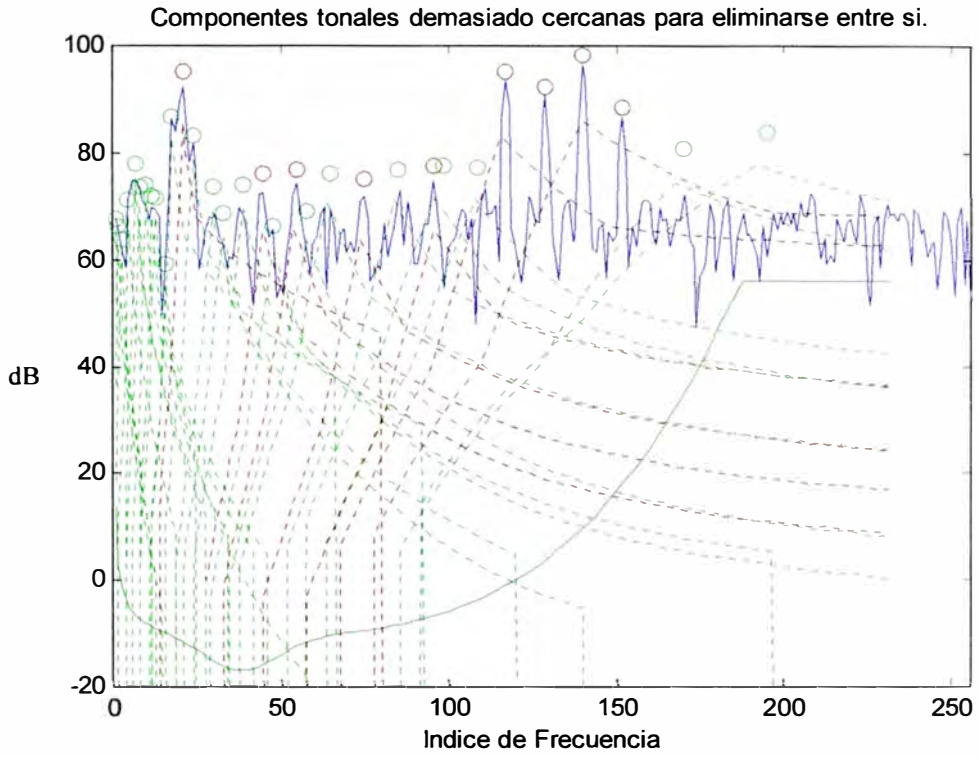

Figura 5.9 Umbrales de enmascaramiento individuales no tonales

En la figura 5.9 se muestra las componentes tonales (círculos rojos), las componentes no tonales (círculos verdes), el umbral de enmascaramiento absoluto (línea verde), los umbrales de enmascaramiento individual para cada componente tonal (líneas punteadas rojas) y para las componentes no tonales (líneas punteadas verdes), como en el caso del enmascaramiento individual se observa que son mas

anchos a medida que aumenta el valor del índice de frecuencia; en las frecuencias mas bajas el ancho de los umbrales es mucho mas angosto.

El umbral de enmascaramiento individual para las componentes no tonales está dado por la siguiente expresión:

$LTnm[z(j),z(i)] = Xnm[z(j)]+avnm[z(j)]+vf[z(j),z(i)]$ dB

Si dz < -3 Bark, ó dz >= 8 Bark, el enmascaramiento no es considerado , por lo que no se grafica (*LTnm* es fijado a –8dB fuera de este rango).

5.2.8 Umbral de Enmascaramiento Global

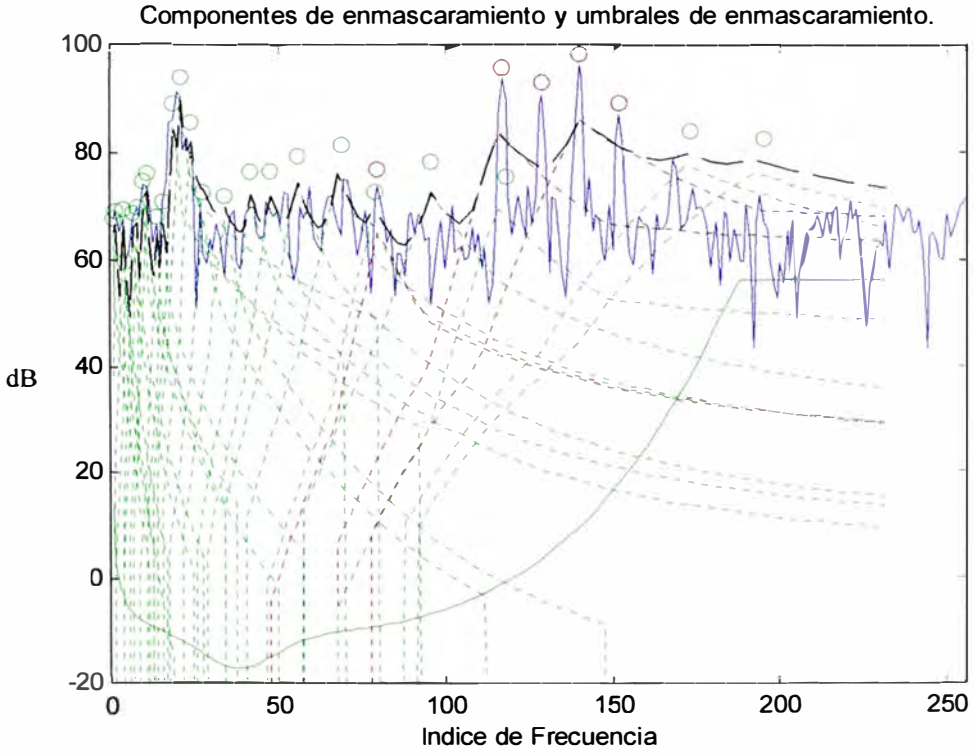

Figura 5.10 Umbral de enmascaramiento global

En la figura 5.10 se aprecia además el umbral de enmascaramiento global (líneas punteadas negras), el que resulta de la combinación del umbral absoluto con los umbrales individuales calculados anteriormente para determinar el umbral de enmascaramiento global sobre toda la banda de audio. Se deriva de los valores mas altos y bajos de los umbrales de enmascaramiento individuales de cada uno de los enmascaradores tonales y no tonales, y en adición del umbral en reposo $LTq(i)$.

5.2.9 Umbral de Enmascaramiento Mínimo

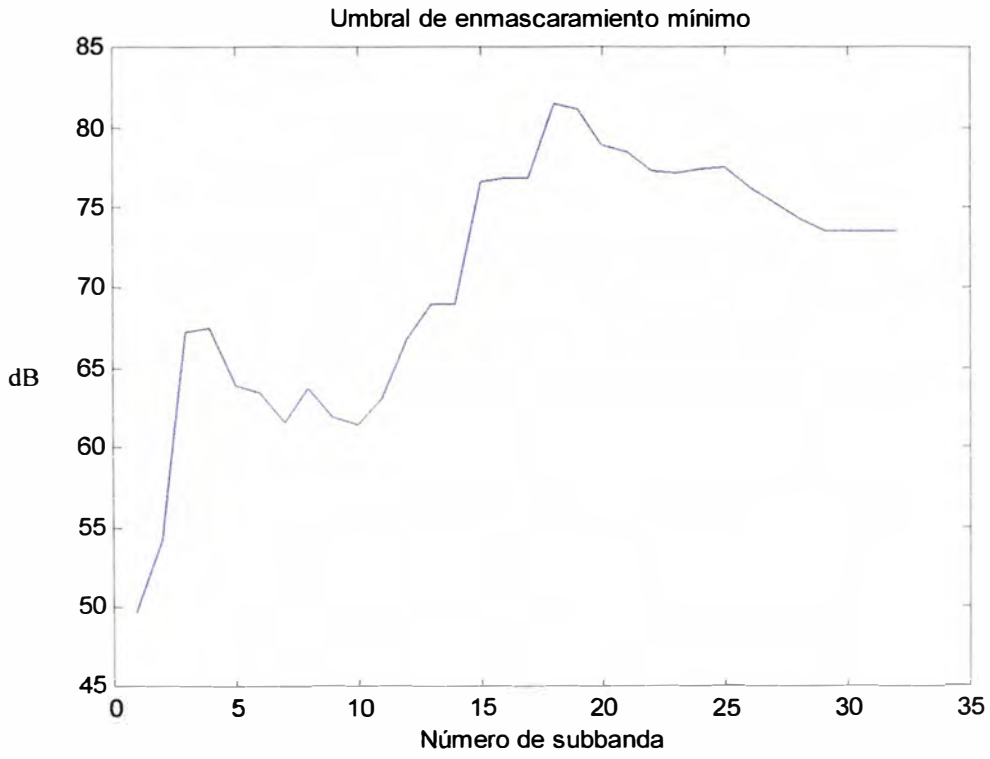

Figura 5.11 Umbral de enmascaramiento mínimo

El mínimo nivel de enmascaramiento *LTmin(n)* en la sub-banda *n* se determina por la siguiente expresión:

$$LTmin(n) = MIN[LTg(i)] \text{ dB}$$

Para encontrar el umbral de enmascaramiento mínimo en cada sub-banda, simplemente se extrae el mínimo valor del espectro global incluido entre las dos frecuencias límites de cada sub-banda, o sea, el valor extraído del umbral global debe ser el valor mínimo de enmascaramiento en la sub-banda. Este método se comporta bien para las sub-bandas más bajas donde la sub-banda es estrecha con respecto a las bandas críticas, pero se vuelve inadecuado para las sub-bandas altas porque una banda crítica en esta frecuencia se distribuye sobre varias sub-bandas. Con este resultado se calculará la relación de señal a máscara SMR, la que se usará en el bloque MDCT y en el bloque de cuantización no uniforme para decidir como asignar el número total de bits de código disponible para la cuantización de señales de sub-banda.

CONCLUSIONES

En cuanto al modelo psicoacústico, no todos los sonidos tienen la mısma relevancia. Esta propiedad es usada por los mecanismos de compresión de audio con el objeto de disminuir la cantidad de datos necesarios para representar un sonido, basándose en un modelo psicoacústico que simula el comportamiento del oído humano, así se toma ventaja de las limitaciones del oído (así como del cerebro) para responder a todas las componentes en una onda de audio compleja y, de esta manera, lograr que los mecanismos de compresión calculen lo que se oirá de un sonido particular, descartando el material indetectable o codificándolo con menos precisión, idealmente sin cambiar la calidad del sonido percibido.

Las bandas críticas son mucho mas estrechas en las bajas frecuencias que en las altas; el 75% de las bandas críticas están por debajo de los 5 Khz, lo que implica que el oído recibe mas información en las bajas que en las altas frecuencias. Estas bandas tienen un ancho de aproximadamente 100 Hz para las frecuencias de 20 a 400 Hz, este ancho aumenta de manera logarítmica a medida que aumenta la frecuencia.

El diagrama de bloques del codificador muestra la conversión primero de la señal de audio al dominio espectral, usando la FFT de 1024 puntos, antes de esto se aplica la ventana de Hanning para evitar las discontinuidades en los extremos de la señal. El estándar de la ISO/IEC 11172-3 proporciona los modelos psicoacústicos necesarios. La precisión del modelo o la capa a usar es dependiente de la aplicación y de la tasa de bits que se quiera lograr.

El uso de los filtros es normado por la ISO/IEC 11172-3. Sin embargo para nuestro modelo se puede incluir un filtro pasa-altos a la entrada del codificador con una frecuencia de corte entre 2 y 10 Hz. El uso de tal filtro evita el innecesario requerimiento de una alta tasas de bits y aumenta la calidad total en el sonido. El codificador utiliza los elementos esenciales para el muestreo-retención, filtraje, corrección de errores CRC, codificación Huffman necesarios para una adecuada compresión de datos, obviamente la calidad o capacidad de compresión dependerá del tipo de polinomio elegido para la codificación, y la fidelidad del sonido que el oyente requiera.

Las gráficas obtenidas del modelo de simulación muestran como varían la frecuencias tonales y no tonales respecto al nivel umbral y su relación con las sub-bandas y el índice de frecuencia. Gracias al modelo psicoacústico solo aquellas componentes sobre el umbral absoluto y aquellas no enmascaradas serán consideradas, puesto que las demás son imperceptibles por el oído humano.

APENDICE A

GLOSARIO

AAC	(Advanced Audio Coding) Codificación de Audio Avanzada.
ADPCM	(Adaptive Differential Pulse Code Modulation) Modulación por Código de Pulsos Diferencial Adaptivo
ATRAC	(Adaptive Transform Acoustic Coding) Codificación Acústica Adaptiva Transformada .
DAB	(Digital Audio Broadcasting) Difusión Digital de Audio conocido como radio digital.
DPCM	(Differential Pulse Code Modulation) Modulación Diferencial por Código de Pulsos
DSP	(Digital Signal Processor) Procesador Digital de Señales.
DVD	(Digital Versatile Disk) Disco Digital Versátil
Banco de Filtros	Un conjunto de filtros pasa-banda que cubren la señal de audio completa en un Rango de Frecuencias.
FFT	(Fast Fourier Transform) Transformada Rápida de Fourier. Un algoritmo para el cálculo de la transformada de Fourier de un conjunto de valores discretos. La FFT expresa los datos en términos de sus componentes de frecuencias, también resuelve el problema inverso de reconstrucción de una señal a partir de los datos de frecuencia.

HDTV	(High Definition Televisión) Televisión de Alta Definición.
Hi-Fi	(High Fidelity Audio) Audio de Alta Fidelidad
IC	(Integrated Circuit) Circuito Integrado.
Joint stereo coding	Cualquier método que explota la irrelevancia Estereofónica o la redundancia Estereofónica.
Layer III	MPEG-1 consiste de tres capas I, II y III.
LFE	(Low Frequency Element channel) Elemento de canal de baja frecuencia.
MDCT	Una transformada que tiene la propiedad de cancelar el solapamiento o "aliasing" en el dominio del tiempo.
MP3	Nombre popular del estándar MPEG-1 Capa III estándar.
MPEG	(Motion Picture Experts Group) Grupo Experto de Imágenes en Movimiento.
Multilingüe	Una presentación de un diálogo en más de un idioma..
NBC	Non Backward Compatibility, nombre antiguo de ACC
PCM	(Pulse Code Modulation) Modulación por Código de Pulsos.
PQF	(Polyphase Quadrature Filter) Filtro Polifásico en Cuadratura.
Modelo Psicoacústico	Un modelo matemático del comportamiento de enmascaramiento del sistema auditivo humano.

RAM	(Random Access Memory) Memoria de lectura y escritura.
ROM	(Read Only Memory) Memoria de solo lectura.
Información Secundaria	Información necesaria en el flujo de bits para controlar el decodificador.
SMR	(Signal to mask ratio) Relación de señal a máscara.
S/PDIF	(Sony/Phillips Digital InterFace) Interface Digital Sony/Phillips
SSR	(Scalable Sampling Rate) Tasa de muestreo escalable.
TCP/IP	Conjunto de Protocolos de Internet TCP/IP.
Twin VQ	(Transform-domain Weighted Interleave Vector Quantization). Transformación de Cuantización de Vectores. Proceso de compresión de audio desarrollado por NTT (Nippon Telephone & Telegraph).

BIBLIOGRAFIA

[1] Lai, "Real Time Implementation of MPEG-1 Layer 3 Audio Decoder on a DSP Chip", National Chiao-Tung University - Taiwan, 2001.

[2] Alan V. Oppenheim, Ronald Schafer, "Tratamiento de Señales en Tiempo Discreto", Prentice Hall, 2000.

[3] M. Kumar, M. Zubair, "A High Performance Software Implementation of MPEG Audio Encoder". IBM T.J. Watson Research Center-Yorktown-NY-USA, 2002.
http://www.mp3-tech.org/programmer/docs

[4] Fabien Petitcolas, "MPEG Psychoacoustic Model I for MATLAB", Universidad de Cambridge - Inglaterra, 2002.
http://www.cl.cam.ac.uk/~fapp2/software/mpeg/

[5] F. Baungarte, C. Ferekidis, H. Fuchs. "A Non Linear Psychoacoustic Model Applied to the ISO MPEG Layer 3 Coder", 2002.
http://www.mp3-tech.org/programmer/docs

[6] ISO IEC 11172-3 – "Coding of Moving Picture And Associated Audio For Digital Storage Media at up about 1.5 Mbps- Part 3- Audio".

[7] Davis Pan, "A Tutorial on MPEG/Audio Compression, IEEE Multimedia", Vol. 2, No. 2. 1995.

[8] ISO/IEC 13818-7, "Information Technology – Generic Coding of Moving Pictures and Associated Audio Information, Part 7: Advanced Audio Coding", 1997.

[9] K. Brandenburg, H. Popp, "An Introduction to MPEG – Layer 3", Fraunhofer Institut Für Integrierte Schaultengen (IIS). EBU Technical Review. Junio 2000.

[10] M. Dietz, H. Popp, K. Brandenburg y R. Friedrich, "Audio Compression for Network Transmission", Journal of the AES, Vol. 44, No. 1-2, 1996.

[11] D. Y. Pan, "Digital Audio Compression", Digital Technical Journal Vol. 5 No.2, 1993.

[12] K. Blair Benson, Audio Engineering Handbook, McGraw-Hill Book Company,1988 ISBN 0-07-004777-4.

[13] Thomas Gleerup, "Low-Bitrate Audio Coding: A Comparison", Lecture Course C5126 Audio Engineering, 1997.
http://www.it.dtu.dk/~tmg/c5126/c5126_paper.htm