7

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

"ANALISIS Y DISEÑO DE UN SISTEMA DE COMPRESION DE AUDIO EN TIEMPO REAL"

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN TELECOMUNICACIONES

P R E S E N T A N

VICTOR JOSE DE JESUS CORRAL LOPEZ

OLIVER CORTES PEREZ

ROBERTO RODRIGUEZ LABASTIDA

DIRECTOR: DR. BOHUMIL PSENICKA

CODIRECTOR: DR. CARLOS RIVERA RIVERA

MEXICO, D. F. 2002

Agradecemos,

A nuestra Facultad,
que nos ha formado como ingenieros
al servicio de nuestro País.

Al Dr. Bohumil Psenicka,
por su paciencia en la
elaboración de este trabajo.

Al Dr. Carlos Rivera,
por su apoyo y ayuda inestimables.

A todos en el Departamento de Telecomunicaciones,
por su apoyo durante la carrera.

A todos los profesores que nos han brindado
su experiencia y sabiduría.

A nuestros compañeros,
por ayudarnos a ser mejores cada día.

A nuestra Alma Mater, la UNAM,
por el orgullo de ser Universitarios

Y a nuestros Padres,
por todo lo que nos han dado.

Dedico este trabajo a todas las personas que han hecho posible que hoy esté aquí, y han despertado una nueva conciencia en mi interior. A mi familia, maestros, amigos, pero muy en especial a mis Padres.

Víctor José de Jesús Corral López

A mis padres, mis hermanos, mis abuelos y tíos

Porque a ellos debo mi formación como persona útil a la sociedad y por su constante preocupación y apoyo que me impulsaron a lo largo de mi vida, haciendo posible llegar a esta meta.

Oliver Cortés Pérez

Este trabajo lo dedico a todos los que me han ayudado a lo largo de mi vida.

A mis amigos
A mis tíos
A mis abuelos
A Sergio
Y a mis Papás

Roberto Rodríguez Labastida

Introducción

Las señales de audio han sido la primera forma de comunicación que ha existido desde el nacimiento de la humanidad, tenemos de esta forma sonidos que representan acciones, y que aun a estas fechas nos dan la capacidad de comunicación. La capacidad que tenemos de oír es lo que propicia nuevas formas de comunicación. El hombre en su afán de transmitir algo más que palabras, creo el arte de la música, con lo cual puede transmitir sus emociones a través de instrumentos, que simulan sonidos de la naturaleza, es por esto que la capacidad de transmitir audio es importante en nuestra sociedad.

Los avances en el campo de las Telecomunicaciones, se basan en el procesamiento de señales, y en el desarrollo de equipo de comunicaciones, siendo estos los dos campos más grandes de investigación. La teoría de información es vital en la primera de estas áreas. El principal problema existente hoy en día es el limitado recurso del ancho de banda, de esta manera el procesar una señal y poder enviar la misma información en un ancho de banda pequeño o con una tasa de transmisión menor es muy importante.

Los avances en la ciencia que se lograrán dentro de pocos años, cambios como la televisión digital terrestre y la radiodifusión digital tendrán modificaciones significativas. Por este motivo es importante desarrollar sistemas que consuman lo menos posible de recursos, en este caso el ancho de banda, que cada vez se va reduciendo debido a las nuevas aplicaciones que se tendrán. Por esto un sistema de compresión de audio es importante para poder ser usado en estas futuras aplicaciones. La introducción masiva de Internet, hará que la transmisión pueda ser vía broadcast o bien vía IP, esta última en la cual ha habido desarrollos muy interesantes, y la radiodifusión digital por este medio será algo que influirá en el conocimiento de nuevas culturas, sin que por esto se desplace la radiodifusión tradicional, más bien se complementaran.

Es importante recordar que toda la tecnología es importada de otros países y que en este país hay gente decidida, comprometida y con la capacidad necesaria para enfocarse a este aspecto. Un proyecto que pueda servir de base a aplicaciones de radiodifusión digital u otras aplicaciones es importante desarrollarlo y que mejor que sea desarrollado por jóvenes mexicanos

Finalmente cabe destacar que la Universidad nos ha dado todas las herramientas para que podamos desarrollar este aspecto y esperamos que nuestro trabajo sea inicio de otros con respecto a este tema.

La compresión es un concepto sencillo de entender y se puede aclarar su significado con solo decir que se reduce la cantidad de datos necesarios para transmitir la misma información, podemos suponer este caso: Una persona desea que alguien que está hablando baje su volumen de voz o incluso que deje de hacer sonidos, esta persona podría decir esta frase: "Podría dejar de hablar por favor", lo cual representa una cantidad de 27 letras con cuatro espacios, o bien podría hacer el sonido de "sssshhhh", lo cual representa 7 letras, esto quiere decir que tuvo el mismo efecto con muchas menos letras, es decir la información estaba contenida en ese sonido pero no tuvo que transmitir un mensaje tan largo. En el

audio no es tan sencillo como esto, este fue un ejemplo y en las siguientes páginas podremos detallar más con respecto al audio.

Objetivos

Analizar diferentes sistemas de compresión de audio que existen actualmente, para conocer sus ventajas y desventajas de cada uno de ellos. Diseñar un sistema de compresión utilizando las mejores características.

Realizar un documento que sirva como referencia bibliográfica para cualquier proyecto relacionado con la compresión de audio, así como la transición a la radiodifusión digital y motivar la investigación en esta área, que a su vez podrá ser inicio investigación sobre estas nuevas tecnologías.

Justificación

La justificación de este proyecto es la falta de investigación en esta área en la universidad lo cual también provoca que la implementación de esta tecnología, en nuestro país sea demasiado costosa, pues son necesarios estudios de grabación dentro de las instalaciones del campus universitario a fin de poder hacer pruebas y poder tener un modelo acústico que sirva para nuestras necesidades.

Para las grandes empresas de radiodifusión, es necesario tener proyectos en esta área sobre todo en estos momentos en los que debemos migrar a tecnologías digitales, tanto para la televisión como para la radio, esto podría provocar elevados costos y que muchas pequeñas estaciones de radio no puedan costear la migración y desaparecer en la transición.

Nuestro trabajo esta orientado a la introducción de estos conocimientos y deberá representar sólo el inicio de la investigación sobre la compresión de audio, para que en un futuro la implementación de este tipo de sistemas para la radiodifusión sea posible en México, reduciendo costos para las estaciones de radio mexicanas y creando nuevas fuentes de empleo.

Alcances y límites de acción

El proyecto tiene como principal objetivo el diseño de un sistema de compresión de audio, como un primer acercamiento a esta área de investigación el sistema de compresión deberá ser más didáctico que eficiente, para que los investigadores y futuras tesis puedan continuar nuestro trabajo, de esta manera tengan una muy buena referencia que les permitan seguir avanzando.

Este sistema no será mejor que los que ya existen pero intentara mostrar la base común de todos los sistemas de compresión actuales. Una de nuestras principales limitaciones es que la mayoría de estos sistemas tiene como parte importante al análisis estadístico de la capacidad perceptual de las personas, el cual debe realizarse en instalaciones especiales que tendrían un costo elevado, locaciones de reverberancia, o salas aisladas de ruido, etc.

Otra de las limitantes es el poco trabajo realizado en la Universidad con respecto a las señales de audio, de esta manera nuestra investigación resulta de gran valor.

Es importante señalar que la implementación en un DSP, se podrá realizar después, pues los algoritmos que se lograron diseñar ya están programados para su fácil interpretación y traducción a lenguaje ensamblador.

1 Antecedentes Teóricos

1.1 Procesamiento Digital de Señales

1.1.1 Teorema del muestreo

La captura de la información acústica se realiza mediante transductores analógicos y requiere la realización de un *muestreo digital* o *conversión analógica digital* (A/D), antes de poder procesar la información por medios informáticos.

El muestreo digital es un proceso de muestreo, a intervalos de tiempo regulares, consistente en la obtención del valor que toma la señal original en un momento dado. El parámetro fundamental del muestreo digital es el *intervalo de muestreo* Δ seg. , o su equivalente *frecuencia de muestreo* 1/Δ Hz. Lógicamente, cuanto menor sea Δ, mayor número de valores obtendremos de la señal, y viceversa.

El resultado de dicho muestreo es la obtención de una serie discreta ordenada $\{x_r\} = \{x_0, x_1, x_3, ..., x_r, ... \}$, en la que el índice r indica la posición de orden temporal del valor x_r. Así, el valor de la señal original, en el tiempo $t = \Delta r$, x(t), se representa por x_r. A la señal continua de origen la llamamos *serie temporal* continua, mientras que a la serie obtenida por el muestreo la llamamos *serie temporal discreta*.

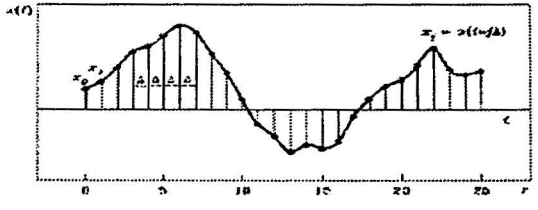


Figura 1.1Muestreo de una señal

Error de cuantificación

La precisión en el valor de la muestra depende del rango de valores enteros que maneje el conversor A/D utilizado. Este rango viene dado por 2^k, donde k es el número de bits en que se exprese el valor de la muestra. Como que la muestra sólo puede tomar uno de los 2^k valores posibles, cuanto mayor sea k, mayor será la aproximación del valor muestral al valor de la señal original.

Para k mucho mayor que uno, la relación entre la señal y el ruido debido a la cuantificación viene dada aproximadamente por

$$\frac{S}{N} = k \bullet 6 \qquad [\text{dB}] \qquad 1.1$$

Así, en nuestra investigación usamos un convertidor A/D con una resolución de 16 bits, por lo que, para un valor de k = 16 bits, tenemos aproximadamente una relación S/R 96 dB, lo que representa un valor más que suficiente para nuestros propósitos.

Teorema del muestreo (Frecuencia de Nyquist)

Teóricamente, si la *frecuencia de muestreo* $f_\Delta = 1/\Delta$ es superior a *dos* veces la máxima frecuencia presente en la señal continua, es posible reconstruirla con toda exactitud a partir de las muestras obtenidas. Decimos teóricamente porque en la práctica, el error de cuantificación y la longitud finita de los registros impiden esta reconstrucción exacta.

Por lo tanto, para que el muestreo sea correcto, deberemos escoger la frecuencia de muestreo de tal forma que

$$f_\Delta \geq 2 f_{max} \qquad 1.2$$

Normalmente se suele filtrar la señal de entrada para eliminar las frecuencias que no pueden ser detectadas mediante el proceso del muestreo, debido a la limitación de los aparatos. Además es conveniente que la frecuencia de muestreo sea *muy* superior al doble de la frecuencia máxima de la señal puesto que ningún filtro pasa bajos puede eliminar completamente las frecuencias superiores a la frecuencia de corte. En la práctica se suele usar una frecuencia de muestreo de entre 5 a 10 veces la frecuencia de corte del filtro pasa bajos

1.1.2 Caracterización de señales

El procesamiento digital de señales trata de la representación de señales por secuencias de números y el posterior proceso de tales secuencias.

Según el rango de variabilidad de la variable independiente, la señal puede ser:
1) Continua en el tiempo f(t), $\tau \in [a,b]$
2) Discreta en el tiempo: $f(t) \in \{t_o, t_1, \ldots, t_n\}$

Según el rango de variabilidad de la amplitud, la señal puede ser:
1) Continua en amplitud
2) Discreta en amplitud

Las señales digitales son discretas en tiempo y en amplitud.

Descripción de señales en el dominio temporal

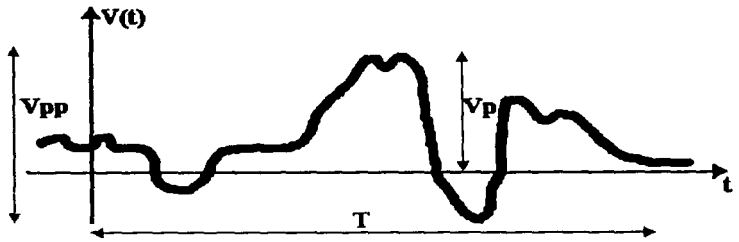


Figura 1.2 Señal analógica en el tiempo

Valor Medio (en un intervalo T):

$$x_t(t) = \frac{1}{T}\int_{\tau}^{\tau+T} x(t)dt$$

Valor Medio temporal:

$$x_t(t) = \lim_{T\to\infty}\frac{1}{T}\langle x_T(t)\rangle = \lim_{T\to\infty}\frac{1}{T}\int_{\tau}^{\tau+T} x(t)dt$$

Valor Medio Cuadrático:

$$P = |x(t)|^2 = \lim_{T\to\infty}\frac{1}{T}\int_{\tau}^{\tau+T}|x(t)|^2 dt$$

Varianza:

$$x_{eff}{}^2 = \left[x(t) - \langle x(t)\rangle\right]^2 = x^2(t) - \left[x(t)\right]^2$$

Señales discretas elementales

Las señales discretas se caracterizan por estar definidas solamente para un conjunto numerable de valores de la variable independiente.

Se representan matemáticamente por secuencias numéricas. En la práctica suelen provenir de un muestreo periódico de una señal analógica

$$x[n] = x_a(nT) \qquad 1.3$$

siendo T el periodo de muestreo

Las señales digitales se obtienen a partir de la cuantización de las señales discretas resultantes del muestreo de las señales analógicas.

Secuencias discretas elementales

Impulso unitario discreto $\delta(n)=1$(Si n=0) , $\delta(n)=0$ (Si $n\neq 0$)

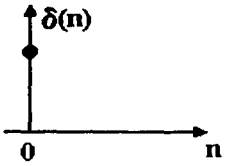


Figura 1.3 Impulso unitario

Escalón unitario discreto: u(n)=1 (Si n>=0) , u(n)=0 (Si n<0)

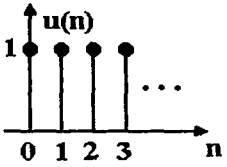


Figura 1.4 Escalón unitario

Identidades:

1. (n)=x(0) δ (n)
2. δ (n)=u(n)-u(n-1)
3. $u(n) = \sum_{m=-\infty}^{\infty} \delta(m)$ q
4. $u(n) = \sum_{k=0}^{\infty} \delta(n-k)$

Secuencia compleja exponencial:

$$x(n) = e^{j\varpi n} = \cos(\varpi n) + j\text{sen}(\varpi n)$$

El conjunto de todos los valores distintos que esta secuencia discreta puede adoptar se encuentra en el intervalo $[-\pi, \pi]$

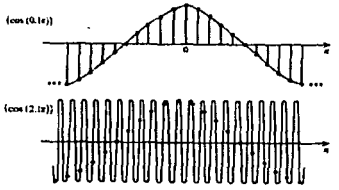


Figura 1.5 Señal muestreada

Las secuencias exponenciales complejas (y sinusoidales) no son necesariamente periódicas (con periodo $T=2\pi/\varpi$), sino que la condición de periodicidad es:

$$\varpi N=2\pi k, \text{ siendo } k \text{ un entero} \qquad 1.4$$

Hay N frecuencias distinguibles para las cuales las secuencias correspondientes son periódicas con periodo N. Este conjunto de frecuencias es:

$$\varpi_k=2\pi k/N, \text{ siendo } k=0,1,2...N-1 \qquad 1.5$$

Clasificación de señales discretas

Señales de Energía: Son señales que tienen energía finita, por lo que son limitadas en tiempo. Las señales con energía infinita pueden ser o no limitadas en el tiempo.

Se define la energía como :

$$E=\sum|x(n)|^2 \qquad 1.6$$

Las señales discretas pueden clasificarse del siguiente modo:

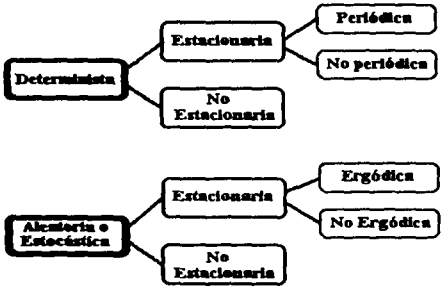


Figura 1.6 Tipos de Señales

Operaciones elementales

- Suma de secuencias: $y(n)=x_1(n)+x_2(n)$
- Multiplicación de secuencias: $y(n)=x_1(n)x_2(n)$
- Adición escalar: $y(n)=x(n)+\alpha$
- Multiplicación por una constante: $y(n)=\alpha\, x(n)$
- Desplazamiento temporal: $n-k \rightarrow y(n-k)$
- Inversión: $-n \rightarrow y(-n)$

Propiedades de simetría

- Secuencia par: $x(-n)=x(n)$
- Secuencia Impar: $x(-n)=-x(n)$
- Toda secuencia arbitraria puede expresarse como la suma de dos componentes, una de las cuales es par y la otra impar: $x(n)=x_e(n)+x_o(n)$

Sistemas lineales e invariantes en el tiempo

Un Sistema es un modelo matemático ó abstracción de un proceso físico que relaciona entradas y salidas según alguna regla preestablecida.

Figura 1.7 Bloque de un sistema lineal y(n)=T[x(n)]

- En general: $y(n) = T\,[x(-\infty), x(n-1), x(n), x(n+1),..., x(\infty)]$
- Sistema Causal: $y(n) = T\,[x(-\infty), x(n-1), x(n)]$
- Sistema causal de memoria finita: $y(n) = T\,[x(n-N),..., x(n-1), x(n)]$
- Sistema invariante en el tiempo: $y(n-m) = T\,[x(n-m)]$
- Sistemas Lineales: Son aquellos verifican el principio de superposición.
- Homogeneidad: Un cambio en la amplitud de la señal de entrada, provoca el mismo cambio de amplitud en la señal de salida
- Aditividad : La respuesta a la suma de dos señales es la suma de las respuestas a cada una de las señales.

  Si: $y_1(n)= T\,[x_1(n)]$, $y_2(n) = T\,[x_2(n)]$

  y se verifica:

  $T\,[ax_1(n) + bx_2(n)] = a\,T[(x_1n)] + bT[x_2(n)]$

  $T\,[ax_1(n) + bx_2(n)] = ay_1(n)+by_2(n)$

Sistemas Invertibles: Si distintas entradas dan lugar a distintas salidas

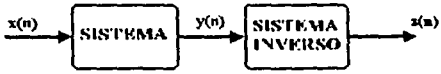
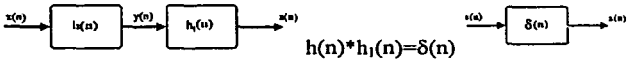


Figura 1.8 Propiedad de Inversión

En el caso de sistemas LIT:



Interacción Señal-Sistema

En general:

$$y[n] = T[x(n)]$$

por otro lado:

$$x(n) = \sum_{k=-\infty}^{k=\infty} x(k)\delta(n-k)$$

Por linealidad:

$$y(n) = T\left[\sum_k x(k)\delta(n-k)\right] = \sum_k x(k)T[\delta(n-k)]$$

Si llamamos:

$$h(n) = T[\delta(n)] \quad (\textit{Respuesta al Impulso del Sistema})$$

Por Invarianza:

$$h(n-k) = T[\delta(n-k)]$$

Luego

$$y(n) = \sum_{k=-\infty}^{k=\infty} x(k)h(n-k) = x(n) * h(n) : \rightarrow \textit{Suma de Convolución}$$

Realizando el cambio: $n-k=j \Rightarrow k=n-j$

$$y(n) = \sum_{k=-\infty}^{k=\infty} x(n-j)h(j) = x(n) * h(n)$$

1.1.2.1 Señales de Audio

Habiendo recibido una educación sonora puramente Analógica hasta mediados los '70, aparecen las primeras aplicaciones del Criterio de Nyquist en Banda Base (hasta ese momento sólo se había usado para telecomunicaciones) en forma de grabadores y reproductores digitales de Audio.

Desde el momento en que se descubrió que una señal muestreada al *doble o más* de su frecuencia máxima, podía ser reconstruida con pérdidas "despreciables", matemáticamente hablando, (Nyquist), parecía cambiar el rumbo de la tecnología del Audio para siempre.

Siendo la Señal Original una forma de onda compleja y *continua*, que representa las *fluctuaciones* de la presión relativa (en las inmediaciones de una fuente sonora) en función del tiempo, se dice que dicha señal es digitalizada cuando es **discretizada**, o sea deja de ser de naturaleza continua para pasar a tener valores perfectamente definidos y *cuantificados*.

El proceso de digitalización de la información consta de tres etapas: Una sección de Entrada, que recibe la señal Analógica y la convierte en Digital, un canal de transmisión (con o sin etapa de almacenamiento) y una etapa de Reconversión de la señal de Digital a Analógica.

El sistema está basado en la *modulación* de la señal de Audio con un tren de impulsos (de frecuencia: $f_{muestreo}$) obteniéndose así una *convolución* en el *dominio de las frecuencias*, esto es *multiplicar* ambas señales en el *dominio del tiempo* para "levantar", en el dominio de las frecuencias, todo el espectro de Audio, almacenándose o no de este modo, para luego de procesos y transferencias (digitales) que el usuario decida, se reconstruya la señal Analógica a través de un convertidor D/A (circuitos correladores).

Es evidente que debido a considerar que un número *finito* de "muestras" (44100 por segundo) de una señal continua equivale a una cantidad *infinita* de instantes de la señal continua, convierte al proceso en un mecanismo no lineal.

A cada "muestra" se le asigna un número correspondiente a la amplitud de cada impulso luego de la modulación. Con 16 bits de codificación se logran $2^{16} = 65536$ niveles de codificación para un *rango dinámico* de 96 dB de la señal de Audio. El cuantificar produce un *error* llamado "*ruido de cuantificación*", pues contamina la señal original con ruido. Este problema es "corregido" sólo cuando se le agrega un ruido llamado "**dither**", con el objeto de facilitar la *correlación* entre señales de entrada y salida. Esto establece en los sistemas digitales un nivel de ruido de fondo diferente de cero.

Para una *frecuencia de* muestreo (*sampleo*) de 44100 Hz la frec. máx. de Audio permitida es de 22050 Hz. Esto se logra acotando el espectro con un filtro pasa bajos. Aquí nos enfrentamos con varios problemas: la abrupta pendiente a manejar y las no linealidades de su respuesta en fase antes de llegar a la frec. de corte. Si llegara a pasar alguna

componente del espectro por sobre la frec. máx. de Audio permitida se generaría un tipo de distorsión llamada "Aliasing" al demodular la señal -en el proceso D/A- que introduciría componentes inexistentes en el material original, entre otros.

A partir de aquí tenemos algunas bases para concluir que una vez entrado en el dominio digital, no es conveniente salir de él, porque lo único que se logra es aumentar el ruido de cuantización en cada paso A/D, y errores en la reconstrucción de la señal analógica al pasar por los circuitos correladores.

Debemos ver al sonido digital como "transparente" y "agresivo" simultáneamente, por su naturaleza lineal en cuanto a su transferencia (2^{16} niveles equidistantes de cuantificación), pero no lineal en cuanto a su método de conversión (cantidad finita de "muestras").

El re-Dithering es el proceso por el cual se enmascaran las armónicas de distorsión provocadas por el truncamiento de las palabras (digitales) de más de 16 bits para llevarse al tamaño estándar de 16 bits mediante la aplicación de ruido apropiadamente ecualizado.

Al interconectar equipos en el dominio Digital, existe una distorsión severa provocada por los canales de transmisión de la información digital (los cables), no apropiados para tal fin, llamada I.S.I. (Inter-Symbol Interference) y Jitter. La primera es generada por un ancho de banda de canal no suficiente, resultando en un redondeo de los pulsos eléctricos. La segunda es debida a la variación en la velocidad de transmisión de la información ante distintas ráfagas digitales perdiéndose la precisión de los instantes de transición de los niveles lógicos. Como consecuencia de esto hay pérdidas de Reloj y falsas lecturas.

En esencia el audio digital es un proceso tecnológico donde una señal analógica (como la producida cuando ondas sonoras en el aire excitan un micrófono) es primeramente convertida en una secuencia continua de números o dígitos conocido como "Código Binario".

Una vez en formato digital la señal es extremadamente inmune a la degradación causada por ruidos del sistema o defectos en el medio, de almacenamiento o de transmisión, a diferencia de los sistemas analógicos precedentes. La señal de audio digitalizada es fácilmente grabada en una variedad de medios ópticos o magnéticos, en los cuales puede ser almacenada indefinidamente con la calidad original sin pérdidas.

La señal digitalizada es luego reconvertida a una señal analógica mediante la reversión del proceso de digitalización para poder ser escuchada por el oído humano, que es análogo.

En los sistemas de grabación y reproducción digital, cada una de estas funciones es ejecutada por separado. En sistemas de procesamiento de señales de audio digital (donde no se ejecutan funciones de grabación - reproducción), ambos procesos de conversión: Analógico a Digital y Digital a Analógico son hechos simultáneamente.

Para éstos sistemas es posible usar una variedad de técnicas, pero la más común se conoce como "modulación codificada de pulsos lineales" o su abreviatura en inglés: PCM (linear pulse code modulation).

La técnica del audio digital que al principio estaba confinada a la grabación y reproducción de música y de otras señales de audio, no ha hecho más que reemplazar la tecnología analógica precedente, hasta tal punto que los discos de vinil prácticamente dejaron de fabricarse desde principios del decenio de los años 90 en los Estados Unidos y en algunos países Europeos. Es posible ya usar técnicas de transmisión digital en ondas de radio.

Esta última década del siglo hemos visto a la tecnología del audio digital reemplazar a la analógica en la mayoría de las funciones, tanto del campo profesional como del consumidor

1.1.3 Filtros

Limpieza de señales contaminadas con ruido de frecuencias conocidas, o en las que la información de interés se encuentra únicamente en ciertas frecuencias:

Filtro Paso Bajo.- Deja pasar frecuencias bajas, y atenúa las altas.
Filtro Paso Alto.- Deja pasar frecuencias altas, y atenúa las bajas.
Filtro Paso Banda.- Deja pasar las frecuencias que se encuentren dentro de un intervalo.
Filtro Supresor de Banda .- No deja pasar las frecuencias que se encuentren dentro de un intervalo.

Filtros recuperadores de señal:
Interpolador lineal.- Dada una secuencia de impulsos equiespaciados (deltas de *Dirac*), devuelve una señal resultante de interpolar linealmente los impulsos.
Regenerador total de señales muestreadas.- Dada una secuencia de impulsos equiespaciados (deltas de *Dirac*) obtenidos por el muestreo periódico de una señal continua, devuelve la señal exacta original (siempre que se cumpla el criterio de *Nyquist* y no se produzca *aliasing*).
Regenerador total de señales muestreadas retenidas.- Filtro recuperador de una señal afectada por un retensor de orden cero *(Zero Order Holder - ZOH)*.
Filtros basados en Funciones de Transferencia –
Filtros de convolución. Normalmente usados para el control en lazo cerrado de una planta lineal e invariante en el tiempo.

1.1.3.1 Filtro FIR

El filtro FIR es un filtro de los llamados *no recursivos*. Podría ser expresado como un filtro de convolución en el que existe un valor *n1* a partir del cual la F.D.T. *h[n]* vale 0.

En este tipo de filtros, la salida es calculada a partir del valor actual de la entrada *g[h]*, y de otros valores pasados. El siguiente ejemplo obtiene la salida del filtro multiplicando el valor actual de entrada por a:

$$f[n] = a\, g[n]$$

En el siguiente ejemplo, se añade un *eco* del valor que había hace 10 cuantos de tiempo con una amplificación de b:

$$f[n] = a\, g[n] + b\, g[n-10]$$

Clasificación

- En función de la forma del módulo de la respuesta en frecuencias
- En función del procedimiento de realización
- En función de la longitud de la respuesta al impulso
- En función de la característica de fase.

Análisis

Proceso por el cual dado un filtro digital Respuesta en Frecuencias

Síntesis o diseño de filtros digitales

En el diseño de filtros selectivos en frecuencia. Las características deseadas del filtro se especifican en el dominio de la frecuencia en función de la respuesta del filtro en magnitud y fase. El proceso del diseño del filtro consiste bien en:

1. La selección de los coeficientes de la ecuación en diferencias, ó
2. La determinación de la respuesta al impulso h(n).

de forma que se cumpla algún criterio sobre las características en el dominio del tiempo o de la frecuencia.

Etapas del diseño

- Especificación de las propiedades.
- Aproximación
- Realización

En general, un filtro digital es un sistema DLI realizado utilizando aritmética de precisión finita.

Diseño de filtros digitales FIR

Aunque los filtros IIR presentan características atractivas, tienen algunos inconvenientes como por ejemplo el no poder aprovechar las ventajas de la FFT en la implementación, ya que para esto es necesario un número de puntos finito. Otra desventaja

es que los IIR alcanzan una magnífica respuesta en amplitud pero la fase de la respuesta no es lineal en todas las frecuencias (en le mejor de los casos en la banda de paso).

Algunas ventajas de los filtros FIR son:

- Facilidad de diseño para filtros de fase lineal.
- Realización eficiente.
- Factible implementación utilizando la FFT.
- Los filtros FIR no recursivos, son siempre estables.
- El ruido de redondeo puede hacerse fácilmente pequeño con realizaciones no recursivas.

Algunas desventajas son:

- Se requiere un número de puntos N alto para aproximar filtros de transición brusca.
- La fase puede no ser entera.
- Filtros FIR simétricos y antisimétricos

Un filtro FIR de longitud M se describe por la ecuación (1.7) en diferencias:

$$y(n) = b_0 x(n) + b_1 x(n-1) + \ldots + b_{M-1} x(n-M+1)$$
$$y(n) = \sum_{k=0}^{M-1} b_k x(n-k) \qquad 1.7$$

ó bien por la convolución (1.8):

$$y(n) = \sum_{k=0}^{M-1} h(k) x(n-k) \qquad 1.8$$

a partir de ambas expresiones, se deduce que: b_k=h(k), k=0,1,2,...,M-1

- El filtro también se puede caracterizar por su función de transferencia:

$$H(z) = \sum_{k=0}^{M-1} h(k) z^{-k}$$

que es un polinomio de grado M-1 en la variable z^{-1}

- Un Filtro FIR tiene fase lineal si su respuesta al impulso satisface la condición:

$$h(n) = \pm h(M-1-n) \qquad \text{Donde n=0,1,2,...,M-1}$$

- Teniendo en cuenta estas condiciones de simetría y antisimetría:

$$H(z) = h(0) + h(1)z^{-1} + h(2)z^{-2} + ... + h(M-2)z^{-(M-2)} + h(M-1)z^{-(M-1)}$$

$$H(z) = z^{\frac{-(M-1)}{2}} \left\{ h\left(\frac{M-1}{2}\right) + \sum_{n=0}^{(M-3)/2} h(n)\left[z^{(M-1-2k)/2} \pm z^{-(M-1-2k)/2}\right] \right\} \quad M \text{ impar}$$

$$H(z) = z^{\frac{-(M-1)}{2}} \sum_{n=0}^{(M/2)-1} h(n)\left[z^{(M-1-2k)/2} \pm z^{-(M-1-2k)/2}\right] \quad M \text{ par}$$

Ahora, si sustituimos z^{-1} por z en la expresión de H(z) y multiplicamos ambos lados de la ecuación resultante por $z^{-(M-1)}$, obtenemos:

$$z^{-(M-1)}H(z-1) = \pm H(z)$$

- Las características de respuesta en frecuencia de filtros FIR de fase lineal se obtienen evaluando H(z) en el círculo unidad.
- Cuando h(n)=h(M-1-n), H(w) se puede expresar como:

$$H(\omega) = H_r(\omega)e^{-j\omega(M-1)/2}$$

donde $H_t(\omega)$ es una función real de ω y se puede expresar como:

$$H_r(\omega) = h\left(\frac{M-1}{2}\right) + 2\sum_{n=0}^{(M-3)/2} h(n)\cos\left[\omega\left(\frac{M-1}{2} - n\right)\right] \quad M \text{ impar}$$

$$H_r(\omega) = 2\sum_{n=0}^{(M-2)-1} h(n)\cos\left[\omega\left(\frac{M-1}{2} - n\right)\right] \quad M \text{ par}$$

La característica de fase del filtro para M impar y par es:

$$\Theta(\omega) = \begin{cases} -\omega\left[\frac{M-1}{2}\right], & H_r(\omega) > 0 \\ -\omega\left[\frac{M-1}{2}\right] + \pi, & H_r(\omega) < 0 \end{cases}$$

- Cuando h(n)=-h(M-1-n), la respuesta al impulso es antisimétrica.
- Para M impar es h((M-1)/2)=0.
- En este caso:

$$H(\omega) = H_r(\omega)e^{j\left[-\omega(M-1)/2+\pi/2\right]}$$

donde:

$$H_r(\omega) = 2\sum_{n=0}^{(M-3)/2} h(n)sen\omega\left(\frac{M-1}{2} - n\right) \quad M \text{ impar}$$

$$H_r(\omega) = 2\sum_{n=0}^{(M-2)-1} h(n)sen\omega\left(\frac{M-1}{2} - n\right) \quad M \text{ par}$$

- La característica de fase del filtro para M par y M impar es:

$$\Theta(\omega) = \begin{Bmatrix} \frac{\pi}{2} - \omega\left[\frac{M-1}{2}\right], \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots H_r(\omega) > 0 \\ \frac{3\pi}{2} - \omega\left[\frac{M-1}{2}\right], \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots H_r(\omega) < 0 \end{Bmatrix}$$

- El problema de diseño de filtros FIR es simplemente el de determinar M coeficientes h(n), n=0,1,...,M-1, a partir de una especificación de la respuesta en frecuencias deseada $H_d(w)$ del filtro FIR. A continuación veremos métodos de diseño basados en la especificación de $H_d(w)$.

1.1.3.2 Diseño de filtros FIR de fase lineal usando ventanas

- Especificación de $H_d(w)$ y determinación (mediante la Transformada de Fourier) de $h_d(n)$, ecuaciones 1.9 y 1.10:

$$H_d(\omega) = \sum_{n=0}^{\infty} h_d(n)e^{-j\omega n} \qquad 1.9$$

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(\omega)e^{j\omega n}d\omega \qquad 1.10$$

- En general, $h_d(n)$ es infinita, por lo que para producir un filtro FIR de longitud M, debe ser truncada en un punto n=M-1. Lo que equivale a multiplicar por una ventana rectangular w(n): $w(n) = \begin{cases} 1, \ldots\ldots\ldots\ldots\ldots n = 0,1,..M-1 \\ 0, \ldots\ldots\ldots\ldots\ldots \text{en otro caso} \end{cases}$

- La respuesta al impulso del filtro FIR será:

$$h(n) = h_d(n)w(n) = \begin{cases} h_d(n), \ldots\ldots\ldots\ldots\ldots n = 0,1,2,\ldots,M-1 \\ 0, \ldots\ldots\ldots\ldots\ldots \text{en otro caso} \end{cases}$$

- Consideremos el efecto de la función ventana en la respuesta en frecuencias deseada $H_d(w)$, y recordemos que multiplicar por una función ventana equivale a una convolución en frecuencias ecuación 1.11 de los espectros, esto es:

$$H(\omega) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(\nu)W(\omega-\nu)d\nu \qquad 1.11$$

La transformada de Fourier de la ventana rectangular es:

$$W(\omega) = \sum_{n=0}^{M-1} e^{-j\omega n} = \frac{1-e^{-j\omega n}}{1-e^{-j\omega}} = e^{-j\omega(M-1)/2}\,\frac{sen(\omega M/2)}{sen(\omega/2)}$$

La función ventana tiene una respuesta en magnitud:

$$|W(\omega)| = \frac{|sen(\omega M/2)|}{|sen(\omega/2)|}$$

Y una fase lineal a tramos:

$$\Theta(\omega) = \begin{cases} -\omega\left[\dfrac{M-1}{2}\right], & sen(\omega M/2) \geq 0 \\ -\omega\left[\dfrac{M-1}{2}\right]+\pi, & sen(\omega M/2) < 0 \end{cases}$$

La convolución de $H_d(w)$ con $W(w)$ tiene el efecto de suavizar $H_d(w)$

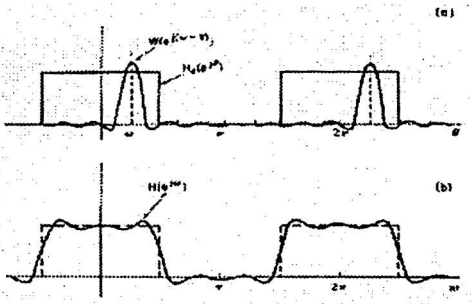


Figura 1.9 (a) Proceso de convolución implicado por el truncamiento de la respuesta al impulso deseada (b) Aproximación típica resultado del ventaneo de la respuesta al impulso deseada

- En la elección de la ventana rectangular, hay que llegar a una solución de compromiso entre dos requisitos antagónicos como son:
- Elegir M de forma que $\omega(e^{j\omega})$ sea lo más estrecho posible.
- Elegir M de forma que la duración de $\omega(n)$ se lo más corta posible.
- Otra solución alternativa consiste en usar ventanas menos abruptas en sus características en el dominio temporal.
- Todas estas funciones ventanas tienen lóbulos laterales más bajos comparados con la ventana rectangular, sin embrago para un mismo valor de M el ancho del lóbulo principal es también más amplio, por lo que la región de transición del filtro será más amplia. Para reducir este ancho, podemos simplemente incrementar la longitud de la ventana. La siguiente tabla resume estas características para los distintos tipos de ventanas:

Tipo de Ventana	Ancho de Transición del lóbulo principal	Pico de lóbulos laterales (dB)
Rectangular	4π /M	-13
Bartlett	8π /M	-27
Hanning	8π /M	-32
Hamming	8π /M	-43
Blackman	12π /M	-58

Tabla 1.1 Ancho espectral de las diferentes ventanas

$$H_d(\omega) = \begin{cases} 1e^{-j\omega(M-1)/2}, & 0 \le |\omega| \le \omega_c \\ 0, & \text{en otro caso} \end{cases}$$

La técnica de ventana se describe mejor con un ejemplo específico. Supongamos que queremos diseñar un filtro FIR de fase lineal paso bajo y simétrico con una respuesta en frecuencias deseada, ecuación 1.12:

$$h_d(n) = \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} e^{j\omega\left(n-\frac{M-1}{2}\right)}d\omega = \frac{sen\omega_c\left(n-\frac{M-1}{2}\right)}{\pi\left(n-\frac{M-1}{2}\right)} \quad n \neq \frac{M-1}{2} \qquad 1.12$$

El retardo (M-1)/2 es para forzar la longitud M. La respuesta al impulso es:

Observar que $h_d(n)$ es no causal y de duración infinita.

$$h(n) = \frac{sen\omega_c\left(n - \frac{M-1}{2}\right)}{\pi\left(n - \frac{M-1}{2}\right)} \quad \ldots\ldots\ldots\ldots 0 \le n \le M-1, \ldots\ldots\ldots n \ne \frac{M-1}{2}$$

Si se selecciona M impar el valor de h(n) en n=(M-1)/2 es:

$$h\left(\frac{M-1}{2}\right) = \frac{w_c}{\pi}$$

1.1.3.3 Diseño de filtros FIR de fase lineal por Muestreo en Frecuencia

Especificamos la respuesta en frecuencias deseada $H_d(w)$ en un conjunto de frecuencias equiespaciadas, ecuación 1.13:

$$w_k = \frac{2\pi}{M}(k+a) \qquad donde \begin{cases} k = 0,1,\ldots(M-1)/2 \\ k = 0,1,\ldots,M/2-1 \\ a = 0 \ldots \acute{o} \ldots 1/2 \end{cases} \qquad 1.13$$

y calculamos la respuesta al impulso h(n) del filtro FIR a partir de estas especificaciones. Para reducir los lóbulos laterales deseable optimizar la especificación de frecuencia en la banda de transición del filtro.

Ahora, explotaremos una propiedad básica de simetría de la función de respuesta en frecuencia muestreada para simplificar los cálculos. Sea la respuesta en frecuencia deseada del filtro FIR:

$$H(\omega) = \sum_{n=0}^{M-1} h(n) e^{-j\omega n}$$

Supongamos que especificamos la respuesta en frecuencias del filtro en las frecuencias anteriores. Entonces, obtenemos:

$$H(k+a) \equiv H\left(\frac{2\pi}{M}(k+a)\right)$$

$$H(k+a) \equiv \sum_{n=0}^{M-1} h(n) e^{-j2\pi(k+a)n/m} \ldots\ldots\ldots\ldots k = 0,1,\ldots,M-1$$

Expresando h(n) en función de $H(k+a)$, obtenemos:

$$h(n) = \frac{1}{M}\sum_{k=0}^{M-1} H(k+a)e^{-j2\pi(k+a)n/M} \quad \ldots\ldots\ldots\ldots n = 0,1,\ldots,M-1$$

Esta expresión nos permite calcular los valores de h(n) a partir de la especificación de las muestras en frecuencia.

Observar que cuando $a=0$, ambas expresiones se reducen a la DFT e IDFT respectivamente. $H(k+a) = H*(M-k-a)$ Al ser h(n) real:

Esta condición de simetría, junto con las condiciones de simetría para h(n) ayudan a reducir ala mitad las especificaciones en frecuencias. Así, las ecuaciones lineales para determinar h(n) a partir de $H(k+a)$ se simplifican considerablemente.

1.1.3.4 Filtro polifásicos en cuadratura[1]

Estos son filtro que donde se utiliza la interpolación y la decimación, la siguiente figura muestra un ejemplo de banco de filtros polifásicos en cuadratura, es simple pues el banco de análisis (Ho(z) y H1(Z)) dividen la señal de entrada x(n) en dos bandas de frecuencias ([0,fs/4] y [fs/4,fs/2])

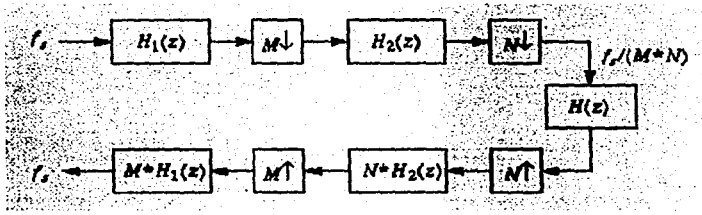


Figura 1.10 Banco de filtros polifásico en cuadratura para separar en dos bandas

La salida del banco de filtros esta dada por la ecuación 1.14:

$$X'(z) = \tfrac{1}{2}[Ho(z)Fo(z) + H1(z)F1(z)]X(z) + \tfrac{1}{2}[Ho(-z)Fo(z) + H1(-z)F1(z)]X(-z) \quad 1.14$$

Donde el segundo término corresponde al término de aliasing

[1] PAPER Almost linear-Phase Ploypluse IIR Lowpass/Highpass Filter Approach, Artur Krukowski and Izzet Kale, University of Westminster, school of electronic and manufacturing Engineering

El diseño de un filtro de análisis, usando un filtros de media banda E(z), el cual tiene dos ceros en el círculo unitario. Debido a estos ceros se puede dividir en una fase mínima H0(z) y un término de máxima fase H1(z).

1.1.4 Transformadas [2]

1.1.4.1 Transformada Zeta

Dada una secuencia x(n), se define su Transformada Z, ecuación 1.15, como:

$$X(z) = Z[x(z)] = \sum_{n=-\infty}^{n=\infty} x(n) z^{-n} \qquad \text{(Transformada bilateral)} \qquad 1.15$$

siendo z una variable compleja: z=x+jy
En el caso de sistemas y señales causales:

$$X(z) = \sum_{n=0}^{n=\infty} x(n) z^{-n} \qquad \text{(Transformada unilateral)} \qquad 1.16$$

Sustituyendo z por su expresión en forma polar, podemos interpretar X(z) en términos de la Transformada de Fourier:

$$X(z) = X(re^{j\omega}) = \sum_{n=-\infty}^{n=\infty} x(n)(re^{j\omega})^{-n} = \sum_{n=-\infty}^{n=\infty} x(n) r^{-n} e^{-j\omega n}$$

Luego, la Transformada Z puede interpretarse como la transformada de Fourier multiplicada por una secuencia exponencial.

A partir de la definición es fácil observar que la Transformada de Fourier de una secuencia coincide con la transformada Z de la misma, evaluada sobre el círculo unidad.
Los principales motivos para introducir esta generalización son que:

- La Transformada de Fourier no converge para todas las secuencias.
- Facilita la resolución de problemas analíticos.
- Permite la utilización de la Teoría de variable compleja en problemas de señales y sistemas discretos.
- Análogamente a la Transformada de Fourier, la transformada Z convierte una convolución en el domino temporal en una multiplicación en el dominio Z.
- Su utilidad principal consiste en el análisis y síntesis de filtros digitales.
- La configuración de las singularidades determina el tipo de filtro digital, bien recursivo o no recursivo, y puede usarse para interpretar su comportamiento frecuencial.

La cuestión de la estabilidad puede enfocarse en términos de la localización de los polos en el plano Z (Dentro del circulo unidad)

Convergencia de la transformada Z

[2] Para más información sobre el uso de transformadas ver en Proakis, D.G. Manolakis, Digital signal processing, principles, algorithms and applications, Prentice Hall Inc., 1996

La Transformada Z no converge para todas las secuencias, ni para todos los valores de z. Para una determinada secuencia, el conjunto de valores de z para los cuales la Transformada Z converge, se denomina *región de convergencia.*

Para que la Transformada Z de una secuencia sea convergente es necesario que la serie sea absolutamente sumable, es decir:

$$\sum_{n=-\infty}^{\infty}\left|x(n)z^{-n}\right| < M \Rightarrow \sum_{n=-\infty}^{\infty}\left|x(n)r^{-n}e^{-j\omega n}\right| = \sum_{n=-\infty}^{\infty}|x(n)|r^{-n} < M$$

Luego, la convergencia o no de la TZ viene determinada por los coeficientes **x(n).**

Ya que la TZ es función de una variable compleja, es conveniente describirla e interpretarla usando el plano compleja.

Un grupo importante de TZ está constituido por aquellas funciones **X(z)** que son racionales, es decir un cociente de polinomios en z. En ellas, las raíces del numerador(valores de z tales que X(z)=0), se denominan ceros de X(z). Análogamente, a las raíces del denominador (valores de z que hacen que $X(z) \rightarrow \infty$) se les denomina polos de X(z). No puede haber polos en la Región de convergencia. Los polos están en el límite de la región de convergencia.

1.1.4.2 La Transformada Z Inversa

Una de las aplicaciones más importantes de la TZ es en el análisis de sistemas discretos LIT. Este análisis suele requerir calcular la TZ inversa.
Los tres métodos básicos para recuperar la secuencia original a partir de su Transformada Z son:

- Inspección directa
- Expansión en fracciones parciales o en series de potencias
- Integral de inversión compleja

El método de *inspección directa* se trata simplemente de familiarizarse con la TZ e identificar ciertos pares.

Si la TZ es una función racional, la expresión en forma de serie de potencias puede obtenerse fácilmente mediante división de polinomios. Podremos observar como precisamente los coeficientes asociados a cada uno de los términos z^{-n} de la serie son los valores de la secuencia , ya que por definición la TZ es:

$$X(z) = \sum_{n=-\infty}^{n=\infty} x(n)z^{-n}$$

Un procedimiento más general consiste en realizar una *Descomposición en Fracciones Simples* e identificar las transformadas simples de los términos así obtenidos.

Si $F(z) = P(z)/Q(z)$ siendo N orden de $Q(z)$ y M el orden de $P(z)$

Si $M<N$ y solo existen polos de primer orden:

$$A_k = [(z - p_k)X(z)]_{z=p_k}$$

$$X(z) = \frac{P(z)}{Q(z)} = \sum_{k=1}^{N} \frac{A_k}{z - z_k}$$

Si $M \geq N$ y solo existen polos simples:

$$X(z) = B_{M-N} z^{M-N} + B_{M-N-1} z^{M-N-1} + \ldots + B_1 z^1 + B_0 + \sum_{k=1}^{N} \frac{A_k}{z - z_k}$$

siendo los B_i los coeficientes obtenidos mediante división hasta que el resto sea de un orden igual al del denominador menos 1. Con este resto se procede a descomponer en fracciones simples y el resultado se añade al de la división.

1.1.4.3 Transformada de Fourier

Para señales x(t) de cuadrado integrable: $\int_{-\infty}^{\infty} |x(t)|^2 dt < \infty$

El dominio de la frecuencia se define por la Transformada de Fourier:

$$X(w) = F[x(t)] = \int_{-\infty}^{\infty} x(t) e^{-jwt} dt \qquad 1.17$$

y se define la Transformada inversa de Fourier:

$$x(t) = F^{-1}[X(w)] = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(w) e^{jwt} dw \qquad 1.18$$

La Transformada de Fourier es un caso límite de las series de Fourier:

$$X(w) = \lim_{T_o \to \infty} (T_o C_n)$$

Algunas Propiedades:

Sea el par $x(t) \Leftrightarrow X(w)$

Linealidad $\alpha x_1(t) + \beta x_2(t) \Leftrightarrow \alpha X_1(e^{jw}) + \beta X_2(e^{jw})$

Dualidad: $X(t) \Leftrightarrow 2\pi x(-w)$

Retraso temporal: $x(t - t_0) \Leftrightarrow X(w) e^{-jwt_0}$

Modulación en frecuencias: $x(t) e^{jw_0 t} \Leftrightarrow 2\pi X(w - w_0)$

1.1.4.4 Transformada Discreta de Fourier

- En general: Si $\sum_{n=-\infty}^{\infty}|x(n)| < M$, se define, ecuación 1.19:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \qquad 1.19$$

- Representación de una secuencia a partir de exponenciales complejas:

$$x(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n}d\omega$$

- La respuesta de un sistema discreto lineal e invariante, a una entrada x(n) es la superposición de las respuestas a cada una de las exponenciales complejas contenidas en la representación de x(n).

$$y(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(e^{j\omega})X(e^{j\omega})e^{j\omega n}d\omega$$

PROPIEDADES

1) Linealidad

Sean: $\begin{cases} x_1(n) \Leftrightarrow X_1(e^{jw}) \\ x_2(n) \Leftrightarrow X_2(e^{jw}) \end{cases}$, y sean α y β constantes

Entonces: $\alpha x_1(n) + \beta x_2 \Leftrightarrow \alpha X_1(e^{jw}) + \beta X_2(e^{jw})$

2) Desplazamiento temporal: $x(n-n_0) \Leftrightarrow X(e^{jw})e^{-jwn_0}$

3) Inversión: $x(-n) \Leftrightarrow X(e^{-jw})$

4) Desplazamiento en frecuencia: $e^{jw_0 n}x(n) \Leftrightarrow X(e^{j(w-w_0)})$

Modulación: $x(n)\cos(w_0 n) \Leftrightarrow 1/2\{X(e^{j(w-w_0)}) + X(e^{j(w+w_0)})\}$

5) Diferenciación: $nx(n) \Leftrightarrow j\frac{dX(e^{jw})}{dw}$

6) Teorema de Parseval: $E = \sum_{n=-\infty}^{\infty} |x(n)|^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} |X(e^{jw})|^2 dw$

7) Teorema de Convolución: $y(n) = x(n) * h(n) \Leftrightarrow X(e^{jw})H(e^{jw})$

8) Teorema de las Ventanas :

$$y(n) = x(n)w(n) \Leftrightarrow Y(e^{jw}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\vartheta})W(e^{j(w-\vartheta)})d\vartheta$$

10) Transformada de Fourier de secuencias con simetría:

- La Transformada de Fourier de una secuencia par es una función real.
- La Transformada de Fourier de una secuencia impar es una función imaginaria

Representación de secuencias de duración finita: la DFT

1) Los resultados anteriores sugieren dos puntos de vista orientados a la representación de Fourier de secuencias de duración finita:

2) Representar una secuencia de duración finita N por una secuencia periódica de periodo N y considerar su representación como un periodo del DSF de la secuencia periódica.

Representar una secuencia de duración finita N

Representación de secuencias periódicas: SDF

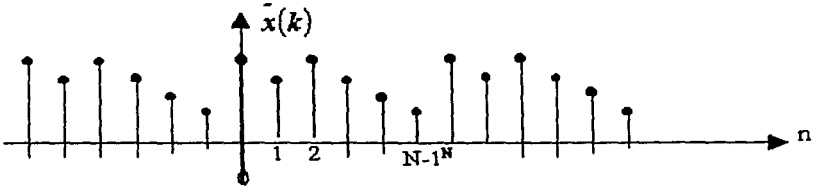


Figura 1.11 Representación de una señal periódica,

La señal $x(n)$, al ser periódica, admite ser desarrollada en Series de Fourier

Paralelismo continuo-discreto del desarrollo en series

$x(t)$	$\longrightarrow$	$x(n)$
Periodo....T	$\longrightarrow$	*Periodo.....N*
$\Omega_0 = \frac{2n}{T}$	$\longrightarrow$	$\omega_0 = \frac{2n}{N}$
Orden.....k	$\longrightarrow$	*orden.....k*
instante.....t	$\longrightarrow$	*elemento.....n*
$e^{j\frac{2n}{T}kt}$	$\longrightarrow$	$e^{j\frac{2n}{N}kn}$

Tabla 1.2 Relación entre tiempo y secuencia de la transformada de Fourier

Exponencial $e^{j\frac{2n}{N}kn}$ — Periódicas respecto a k $\Rightarrow$ el desarrollo sólo tendrá N exponenciales complejas distintas

Representación en DFS de una secuencia periódica.

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{-j\frac{2\pi}{N}nk} \qquad X(k) = \sum_{k=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}nk}$$

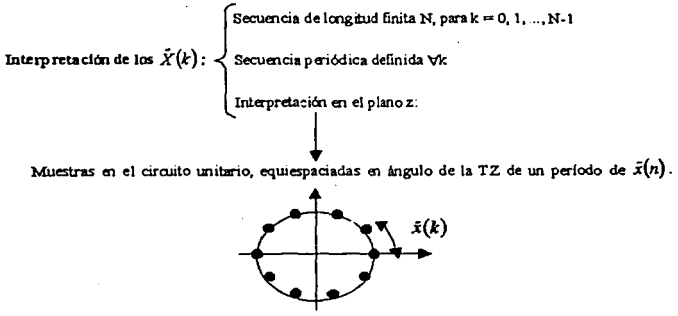


Figura 1.12 Transformada de Fourier en el dominio Z

Convolución periódica

$$Sean\begin{cases} x_1(n) \Leftrightarrow X_1(k) \\ x_2(n) \Leftrightarrow X_2(k) \end{cases}$$

El $x_3(n)$ tal que su DSF es $X_1(k)X_2(k)$

$$x_3(n) = \sum_{m=0}^{N-1} x_1(m)x_2(n-m)$$

$$DUALIDAD\begin{cases} x_3(n) = x_1(n)x_2(n) \Leftrightarrow X_3(k) \\ X_3(k) = \frac{1}{N}\sum_{l=0}^{N-1} X_1(l)X_2(k-l) \end{cases}$$

Transformada discreta de Fourier (Directa ecuación 1.20, Inversa ecuación 1.21)

DFT:

$$X(k) = \begin{cases} \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} x(n)W_N^{kn} \Leftarrow 0 \le k \le N-1 \\ 0 \Leftarrow resto \end{cases} \qquad 1.20$$

IDFT:

$$x(n) = \begin{cases} \frac{1}{N}\sum_{k=0}^{N-1} X(k)W_N^{-kn} \Leftarrow 0 \le n \le N-1 \\ 0 \Leftarrow resto \end{cases} \qquad 1.21$$

Convolución lineal usando la DFT.

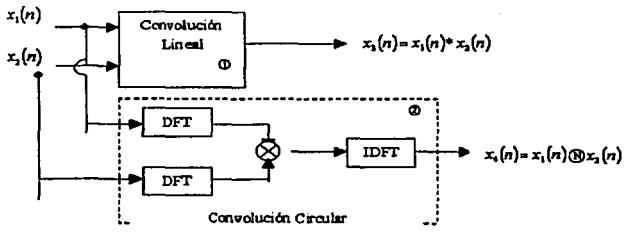


Figura 1.13 Diagrama de bloques de la convolución lineal usando DFT

Convolución de dos secuencias finitas de igual número de puntos.

$$Sean \begin{cases} Long[x_1(n)] = N \\ Long[x_2(n)] = N \end{cases}$$

Convolución lineal $$x_3(n) = \sum_{M=0}^{N-1} x_1(m)x_2(n-m)$$

$$Long[x_3(n)] = 2N-1$$

$$\Rightarrow \begin{cases} X_1(k) = \sum_{n=0}^{2N-2} x_1(n)W_{2N-1}^{nk} \\ X_2(k) = \sum_{n=0}^{2N-2} x_2(n)W_{N-1}^{nk} \end{cases}$$

$$x_4 = \frac{1}{2N-1}\left[\sum_{n=0}^{2N-2}[X_1(k)X_2(k)]W_{2N-1}^{nk}\right]R_{2N-1}(n) \qquad x_3 = x_2$$

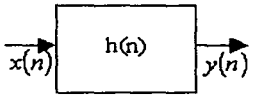
b) Convolución de dos secuencias finitas de distinto número de puntos.

En general si $\begin{cases} \log[x_1(n)] = N_1 \\ \log[x_2(n)] = N_2 \end{cases}$

$$long[x(n)_1 * x_2(n)] = N_1 + N_2 - 1$$

$\Rightarrow$ DFT'S sobre la base de $N \geq N_1 + N_2 - 1$ puntos

c) Convolución de una secuencia finita con otra de un número indefinido de puntos



$$Long[h(n)] = M$$

$$x(n) \rightarrow Longitud\ indefinida$$

$$y(n) = x(n) * h(n)$$

Figura 1.14 Convolución de dos secuencias con número de elementos indefinidos.

Solo se realiza la convolución como tal pues el tratar de realizarla por medio de la DFT es un poco más complejo y se gasta más procesamiento.

1.1.4.5 Transformada Rápida de Fourier

La transformada rápida de Fourier es simplemente un algoritmo rápido para la evaluación numérica de integrales de Fourier desarrollado en los laboratorios de IBM, y su importancia radica en la rapidez de cálculo conseguida, en aplicaciones como : ecualización y filtrado en equipos de audio / vídeo en tiempo real, comunicaciones, etc.

Evidentemente hacemos uso del mismo en el programa para obtener rápidamente el espectro de la señal a partir de la señal temporal de entrada, aunque se podría haber hecho a partir de la integral discreta de Fourier, siendo en este caso necesario mucho más tiempo de cálculo.

La diferencia de velocidad de cálculo entre la tradicional transformada discreta y la FFT aumenta según aumenta el número de muestras a analizar, según se puede apreciar en la gráfica, ya que mientras una aumenta el número de operaciones necesarias para la resolución de forma exponencial, la otra lo hace de forma prácticamente lineal.

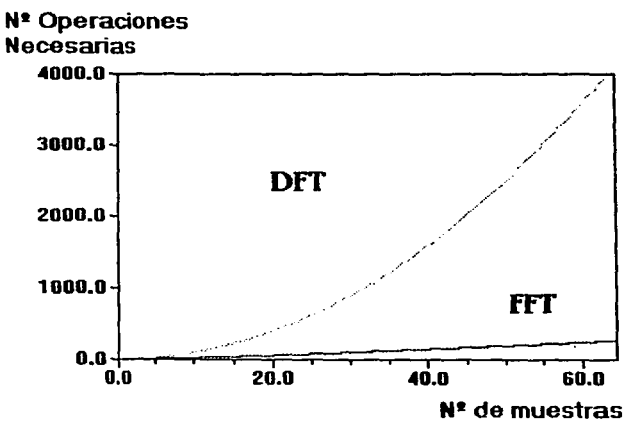


Figura 1.15 Comparación en el número de operaciones aplicando FFT y DFT

El algoritmo FFT lo único que busca es resolver de la manera más eficiente posible la siguiente expresión, ecuación 1.22:

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-jk\Omega n} \qquad 1.22$$

donde como sabemos $\Omega=2\pi/N$. La evaluación directa de esta sumatoria implica N^2 multiplicaciones. Haciendo una serie de reordenaciones, conseguiremos con la FFT reducirlo a *N*Log2(N)* operaciones.

Primero se deben separar las muestras pares y las impares:

$$X(k)=\frac{1}{N}\left[\sum_{n=0}^{N/2-1}x(2n)e^{-2jk\Omega n}+\sum_{n=0}^{N/2-1}x(2n+1)e^{-2(n+1)jk\Omega}\right]$$

A continuación sacamos fuera de la sumatoria impar la exponencial $e^{-jk\Omega}$

$$X(k)=\frac{1}{N}\left[\sum_{n=0}^{N/2-1}x(2n)e^{-2jk\Omega n}+e^{-jk\Omega}*\sum_{n=0}^{N/2-1}x(2n+1)e^{-2jk\Omega n}\right]$$

Si paramos a observar esta expresión, podemos ver que si ponemos

Y=FFT(x[0], x[2], x[4], ..., x[N-2])

y

Z=FFT(x[1], x[3], x[5], ..., x[N-1])

entonces

$$x(k)=\frac{1}{2}\left[Y(k)+e^{-jk\Omega}Z(k)\right]\text{.....}si..0\le k\le\frac{N}{2}$$

$$x(k)=\frac{1}{2}\left[Y\left(k-\frac{N}{2}\right)-e^{-jk\Omega}Z\left(k-\frac{N}{2}\right)\right]\text{.....}si..\frac{N}{2}\le k<N$$

El problema ha sido reducido al cálculo de dos FFTs de tamaño N/2 y realizar N multiplicaciones complejas. Es conveniente observar que el bit menos significativo de k determina siempre si k es par o impar. Repitiendo este proceso reiteradamente, conseguimos extraer la transformada de x.

Total de operaciones	**Para cada X(k)**		**Todos los X(k)**	
	Productos	*Sumas*	*Productos*	*Sumas*
Operaciones complejas	N	N-1	N^2	N(N-1)
Operaciones reales	4N	4N-2	$4N^2$	N(4N-2)

Tabla 1.3 Comparación de operaciones efectuadas con la DFT y la FFT

1.1.4.6 Transformada Coseno

Si f es continua a pedazos en [0,π], la transformada de Fourier finita en cosenos Fc, ecuación 1.23 de f está definida como:

$$Fc(n) = \int_0^{\pi} f(x)\cos(nx)dx \qquad 1.23$$

Para n=0,1,2,.... Esta transformada se denota como Cn{f(x)}.

La motivación de esta transformada viene del desarrollo de Fourier en cosenos de f en [0,π]. Insertando las formulas integrales de los coeficientes, la serie es:

$$\frac{1}{n}\int_0^{\pi} f(\xi)d\xi + \sum_{n=1}^{\infty}\left[\frac{2}{\pi}\int_0^{\pi} f(\xi)\cos(n\xi)d\xi\right]\cos(nx)$$

En términos de la transformada de Fourier finita en cosenos, la serie es:

$$\frac{2}{\pi}\left[\frac{1}{2}Fc(0) + \sum_{n=1}^{\infty} Fc(n)\cos(nx)\right]$$

Esta serie es una fórmula de inversión para la transformada de Fourier finita en cosenos. Dada la transformada Fc, la fórmula de inversión genera un desarrollo de Fourier en cosenos en [0, π] de una función f cuya transformada de Fourier finita en cosenos sea Fc.

El siguiente teorema nos da la fórmula de operación fundamental para la transformada de Fourier finita en cosenos.

Teorema: Sean f y f' continuas en [0, π] y supongamos que f'' es continua a pedazos en [0,π].

Entonces

$$Cn\{f''(x) = -n^2 Fc(n) - f'(0) + (-1)^n f'(\pi)\}$$

Para n=1,2,3....

La demostración es tan solo una integral que resuelve por partes dos veces[3]

Cabe aclarar que esta transformada al igual que la de Fourier cumple con dos características del álgebra lineal que son:

- Ortogonalidad
- Normalizada

1.1.4.7 Transformadas Wavelets[4]

Las técnicas de análisis Wavelet emplean regiones de tamaño variable, para el análisis de las señales deja usar durante largo tiempo intervalos donde se necesita mucha información que precisa poca frecuencia y pequeñas regiones donde la información necesita altas frecuencias

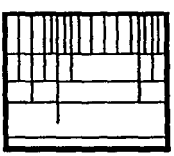


Figura 1.16 Esquema de análisis Wavelet

El análisis Wavelet es capaz de mostrar aspectos de la señal que otras técnicas no logran encontrar

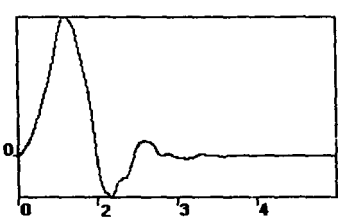


Figura 1.17 Señal de la cual se saco el análisis Wavelet

La transformada Wavelet consiste en comparar la señal con ciertas funciones Wavelet, las cuales se obtienen a partir de las Wavelet madre. La comparación permite obtener unos coeficientes que son susceptibles de interpretación y posterior manipulación.

[3] La demostración en la serie de senos esta disponible en el libro de Matemáticas Avanzadas de la bibliografía
[4] El uso de Wavelets es para el manejo de imágenes. Si desea más información consulte libros sobre procesamiento de imagénes.

En cualquier caso, un requisito básico es la posibilidad de invertir la transformada, recuperando la señal a partir de esos coeficientes Wavelet calculado.

El cálculo de la transformada Wavelet para todas las posibles escalas supone una gran cantidad de información. Escoger sólo aquellas escalas y posiciones que resulten interesantes para ciertos estudios es una tarea difícil. Si se escogen aquellas escalas y posiciones basadas en potencias de dos, los resultados serán más eficaces. Este análisis se denomina DWT.

Para muchas señales la información más importante se encuentra en las frecuencias bajas, mientras que en las altas frecuencias se encuentran los detalles o matices de la señal. Por ejemplo, en el caso de la voz humana, si eliminamos los componentes con altas frecuencias, la voz suena diferente pero se sigue entendiendo su mensaje. En cambio, si lo que se elimina son las componentes de bajas frecuencias, el mensaje se vuelve irreconocible. Por eso el análisis Wavelet permite descomponer la señal en aproximaciones y detalles, a éste proceso se le conoce con el nombre de análisis. Este filtrado nos proporciona el doble de datos de los que son necesarios, este problema se soluciona con la operación de *downsampling*

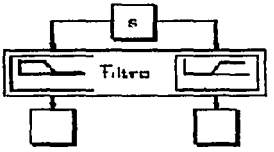


Figura 1.18 Filtros de Análisis

El proceso de reconstrucción, también denominado síntesis, se encarga de la obtención de la señal a partir de los detalles y aproximaciones. Éste proceso se lleva a cabo con la transformada Wavelet discreta inversa

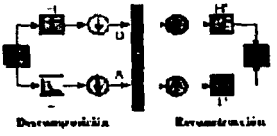


Figura 1.19 Método EZW

El método de compresión EZW fue propuesto por Shapiro en 1993. Este método explota las propiedades aportadas por la DWT para obtener resultados satisfactorios en la compresión: un gran porcentaje de coeficientes Wavelets próximos a cero y la agrupación de la energía de la señal (comúnmente se utilizan imágenes).

El EZW es sensible al grupo de bits transmitidos por orden de significancia, lo que le permite una compresión progresiva (*embedded coding o código embebido*) de la imagen. Cuantos más bits se añadan al resultado de la compresión, más detalles se estarán transmitiendo.

La implementación del EZW se realiza mediante el algoritmo de incrustación definido por Shapiro. El algoritmo de incrustación cuantifica los coeficientes Wavelet de la imagen en pasos dados por potencias de dos, reduciéndose progresivamente en sucesivas iteraciones. La primera operación a realizar consiste en calcular el umbral de partida *n* (primer paso)

A continuación, se construye un mapa de significancias (*zerotrees*) basado en la búsqueda de los coeficientes mayores o iguales al umbral determinado en el paso anterior

El siguiente proceso se denomina refinamiento. Consiste en la transmisión de los bits de todos los coeficientes detectados en los pasos previos. Este proceso se repite decrementando el umbral hasta alcanzar el umbral cero

La estructura *zerotree* agrupa los coeficientes de cuatro en cuatro: cada coeficiente tiene cuatro hijos, cada uno los cuales tiene sus propios cuatro hijos y así sucesivamente. Por lo general, los hijos tienen unas magnitudes menores que las de sus padres. El EZW se aprovecha de esta organización basada en el hecho de que los coeficientes Wavelet se decrementan a medida que aumenta la escala. Así se puede garantizar que los coeficientes de un *quadtree* son más pequeños que el umbral de estudio, si su padre es más pequeño que el umbral antes mencionado

Funcionamiento del algoritmo SPIHT

Tradicionalmente el principal impedimento para obtener un alto nivel de compresión en imágenes se encuentra en la codificación de la información. Actualmente existen métodos que obtienen un rendimiento óptimo, pero a costa de algoritmos de una complejidad computacional elevada. Por contra, el algoritmo SPIHT de Said y Pearlman obtiene resultados similares con una complejidad baja. El tipo de codificación que realiza se basa en la clasificación por orden de bits significativos

El SPIHT ofrece una nueva y mejor implementación del EZW basada en la utilización de conjuntos de datos organizados en árboles jerárquicos, es decir, el SPIHT tiene en cuenta la significancia de la descendencia del coeficiente que codifica.

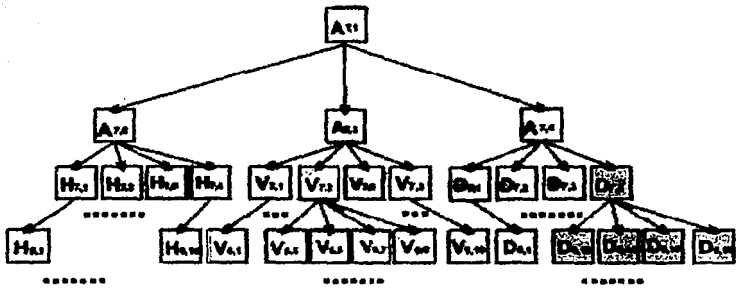


Figura 1.20 Utilización de árboles jerárquicos en el método de compresión

1.1.5 Cuantización

Es el proceso de convertir números con precisión infinita en números de un conjunto finito.

- Si las amplitudes de las muestras están en el rango (x_{min},x_{max}) y se utilizan L niveles, entonces el tamaño del intervalo de cuantización es:

$$\Delta = \frac{(x_{max} - x_{min})}{L}$$

y los niveles de amplitud permitidos son:

$$q_i = X_{min} + \frac{\Delta}{2} + i\Delta_{i=0,\ldots L-1}$$

- En la figura se muestran dos tipos de características I/O:

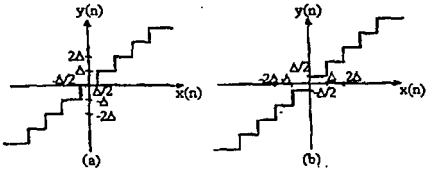


Figura 1.21 (a) Cuantizador que pasa por cero (Midtread type)
(b) Cuantizador que no pasa por cero (Midriser type)

- La Cuantización da lugar a una pérdida de precisión, de forma que la secuencia de salida del cuantizador puede expresarse como:

$$x(n) = x(n) + e_q(n)$$

- La amplitud de cada muestra de $e_q(n)$ está en el rango:

$$\frac{-\Delta}{2} \leq e_q \leq \frac{\Delta}{2}$$

- Potencia ó valor cuadrático medio de la señal de error (de cuantización):

$$E\{e_q(n)\} = \frac{\Delta^2}{12}$$

- Para una señal de longitud finita, una estimación de la potencia del error es:

$$P_q = \frac{\sum_{n=0}^{N-1} e_q^2(n)}{N}$$

- Se define la relación señal-ruido como la relación entre la potencia de la señal y la potencia del ruido:

$$SNR = \frac{E\{x^2(n)\}}{E\{e_q^2(n)\}}$$

- Para una señal de longitud finita:

$$SNR \approx \frac{\sum_{n=0}^{N-1} x^2(n)}{\sum_{n=0}^{N-1} e_q^2(n)}$$

- Frecuentemente la relación señal ruido se expresa también en una escala en decibelios:

$$SNR_{dB} = 10\log_{10}(SNR) \qquad 1.24$$

Modificación de la frecuencia de muestreo

- La frecuencia de muestreo de una señal discreta temporal puede modificarse mediante dos métodos diferentes:

1. Convertir la señal discreta en una señal continua y remuestrearla.
2. Procesar directamente la señal discreta.

Decimador: Sistema que reduce la frecuencia de muestreo en una relación de enteros

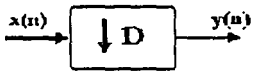


Figura 1.22 Diagrama de bloque de un decimador

- Un *decimador* tiene una característica I/O: $x_D(n)=x(nD)$
- Retener una de cada D muestras de la señal de entrada.

Puede demostrarse que:

$$X_D(w) = \frac{1}{D}\sum_{k=0}^{D-1} X(w+\frac{2\pi k}{D})$$

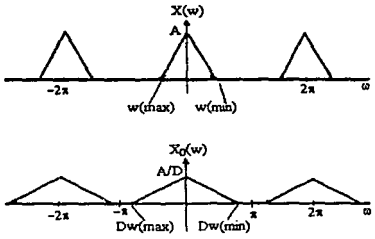


Figura 1.23 Efecto del decimador a la señal muestreada

- Para prevenir el aliasing, x(n) debe estar limitada en banda de forma que:

$$X(w) = 0 \ldots\ldots\ldots para. |W| \leq \frac{\pi}{D}$$

- En la práctica:

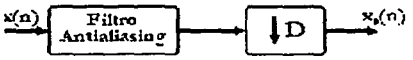


Figura 1.24 Bloque de un decimador real, necesita un filtro antialiasing

Interpolador

- Sistema que incrementa la frecuencia de muestreo en una relación de enteros.

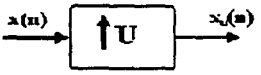


Figura 1.25 Bloque de in Interpolador (upsampler)

- Un *interpolador* se define por:

$$x_U(n) = x(n/U) \text{ para } n = kU \text{ y 0 para cualquier otro n.}$$

En este caso, la señal de salida se obtiene insertando U-1 muestras de amplitud 0 entre cada dos muestras consecutivas de x(n).

- En la práctica:

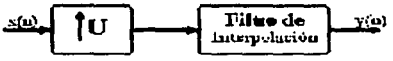


Figura 1.26 Bloque de interpolador real que, al igual que el decimador necesita un filtro también

- El espectro en frecuencias de la señal sobremuestreada es:

$$X_U(w) = X(wU)$$

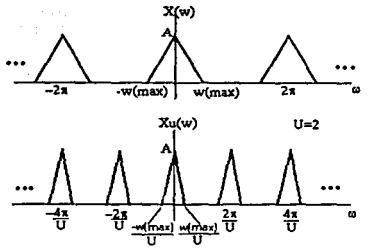


Figura 1.27 Señal sobremuestreada, efecto del interpolador a la señal

1.2 Teoría de la información

- **Compresión sin pérdidas (*lossless)*.** Permite reconstruir el archivo original sin pérdida de información. Si se trata de una imagen la reproduce tal cual era. Utiliza el algoritmo DPCM (*Differential Pulse Code Modulation*).
- **Compresión con pérdidas (*lossly)*.** Reconstruye el archivo original de forma aproximada, lo que es válido en aquellos casos en que la información perdida es irrelevante. Sacrifica algunos elementos de información en tanto no se altera perceptiblemente el impacto total del contenido. Se consigue niveles de compresión mucho más altos, del orden de 100:1.

1.2.1 Código de Huffman

Considérese un mensaje cuya cantidad de símbolos tiende al infinito; entonces:

N_{total} = total de letras del mensaje.
N_A = total de letras A del mensaje.
N_B = total de letras B del mensaje.
N_C = total de letras C del mensaje.
- - - - - - - - - - - -
- - - - - - - - - - - -
N_Z = total de letras Z del mensaje.

$$I_{TOTAL} = N_A \log_2 \frac{1}{P(A)} + N_B \log_2 \frac{1}{P(B)} + \ldots + N_Z \log_2 \frac{1}{P(Z)}$$

Así mismo:

$$promedio\ de\ información = \frac{I_{TOTAL}}{N_{TOTAL}}$$

El promedio de información es una cantidad muy importante y se le ha dado un nombre especial: ENTROPÍA. Su símbolo es una H muy particular: ℵ. Entonces:

$$\aleph = \frac{I_{TOTAL}}{N_{TOTAL}} = \frac{N_A}{N_{TOTAL}} \log_2 \frac{1}{P(A)} + \ldots + \frac{N_Z}{N_{TOTAL}} \log_2 \frac{1}{P(Z)}$$

Es evidente que: $\frac{N_A}{N_{TOTAL}} = P(A)$

Y así para todos los demás símbolos; por lo que el promedio de información de un alfabeto es:

$$\aleph = P(A)\log_2 \frac{1}{P(A)} + P(B)\log_2 \frac{1}{P(B)} + \ldots + P(Z)\log_2 \frac{1}{P(Z)} \qquad \left[\frac{bits}{símbolo}\right]$$

Expresado en forma compacta:

$$= \sum_{\delta=A}^{Z} P(\delta)\log_2 \frac{1}{P(\delta)} \qquad \left[\frac{bits}{símbolo}\right] \qquad 1.25$$

Método de Shannon-Fanon para obtener códigos óptimos.

Este procedimiento se puede aplicar a alfabetos con cualquier cantidad de símbolos; para explicarlo, se hará uso de un alfabeto de pocas letras, de modo que no haya muchos pasos repetitivos y el método quede claro.

Sea un alfabeto de 8 letras, cuyas probabilidades son:

P(A) = 12.5%
P(B) = 6.25%
P(D) = 25%
P(E) = 6.25%
P(F) = 12.5%
P(G) = 6.25%
P(H) = 25%
P(I) = 6.25%

Después de verificar que la suma de las probabilidades es de 100%, es necesario acomodar la lista de letras en orden decreciente de probabilidades, como se ve a continuación:

P(H) = 25.0%
P(D) = 25.0%
P(A) = 12.5%
P(F) = 12.5%

P(B) = 6.25%
P(E) = 6.25%
P(G) = 6.25%
P(I) = 6.25%

Nótese que hay varias formas (96) de hacer esta ordenación; se puede usar cualquiera de ellas y el promedio de bits/letra no cambiará. El siguiente paso es dividir el conjunto en dos

H	25	0
D	25	0
A	12.5	1
F	12.5	1
B	6.25	1
E	6.25	1
G	6.25	1
I	6.25	1

subconjuntos que tengan más o menos las mismas probabilidades. En este caso la división es exacta entre la D y la A. En este momento, le ponemos ceros al primer subconjunto y unos al segundo subconjunto, como se ve a continuación:

H	25	0	0
D	25	0	1
A	12.5	1	0
F	12.5	1	0
B	6.25	1	1
E	6.25	1	1
G	6.25	1	1
I	6.25	1	1

Ahora hay dos subconjuntos que a su vez deben ser divididos en mitades, como se ve a continuación, asignando ceros a la primera mitad y unos a la segunda mitad.
Ahora hay cuatro subconjuntos: {H}, {D}, {A,F}, {B,E,G,I}. Los dos primeros ya no pueden ser subdivididos, pero cada uno de los dos últimos sí:

H	25	0	0	
D	25	0	1	
A	12.5	1	0	0
F	12.5	1	0	1
B	6.25	1	1	0
E	6.25	1	1	0
G	6.25	1	1	1
I	6.25	1	1	1

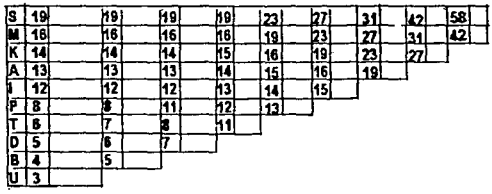
Una última subdivisión de los subconjuntos {B,E} y {G,I} se muestra en el siguiente arreglo:

H	25	0	0		
D	25	0	1		
A	12.5	1	0	0	
F	12.5	1	0	1	
B	6.25	1	1	0	0
E	6.25	1	1	0	1
G	6.25	1	1	1	0
I	6.25	1	1	1	1

Ahora, cada símbolo del alfabeto propuesto tiene asignado un conjunto de bits y el código optimo ha sido obtenido. Para verificar esto último calcularemos el promedio y la entropía

Método de Huffman para obtener códigos óptimos.

Formemos el siguiente arreglo: se trata de las letras del ejercicio anterior y la primera columna de números contiene sus probabilidades.

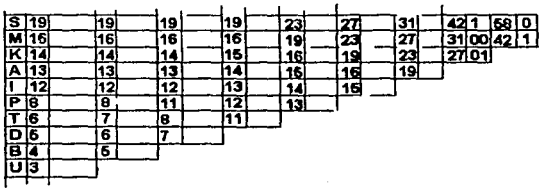


Para formar la segunda columna, transferimos los números de la primera, excepto los dos últimos, que al sumarlos dan 7 y ponemos el 7 entre el 8 y el 6.

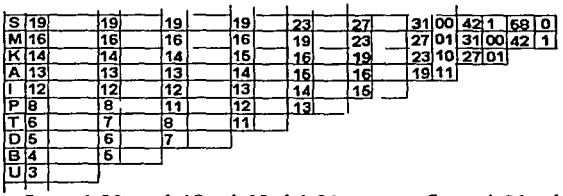
Para formar la tercera columna, transferimos los números de la segunda, excepto los dos últimos, que al sumarlos dan 11 que lo colocamos entre el 12 y el 8.

Con la misma regla se forman las otras columnas: la suma de los dos últimos números de una columna se coloca en la siguiente columna en el lugar más bajo posible, conservando la variación decreciente.

Para llenar las columnas vacías, veamos el siguiente arreglo:



Inicialmente se le regala un 0 al 58 y un 1 al 42. El 1 del 42 se transfiere al 42 de la columna de la izquierda y el 0 del 58 se transfiere al 31 y al 27 de la columna de la izquierda. Finalmente, se le regala un 0 al 31 y un 1 al 27.
Para seguir adelante, veamos el siguiente arreglo:



El 1 del 42 se transfiere al 23 y al 19, el 00 del 31 se transfiere al 31, el 01 del 27 se transfiere al 27. Finalmente, se le regala un 0 al 23 y un 1 al 19.

Con estas reglas se llenan las demás columnas y la de la extrema izquierda contiene el código binario optimizado, como se ve enseguida.

S	19	11	19	11	19	11	19	11	23	10	27	01	31	00	42	1	58	0
M	16	000	16	000	16	000	16	000	19	11	23	10	27	01	31	00	42	1
K	14	010	14	010	14	010	15	001	16	000	19	11	23	10	27	01		
A	13	011	13	011	13	011	14	010	15	001	16	000	19	11				
I	12	100	12	100	12	100	13	011	14	010	15	001						
P	8	0010	8	0010	11	101	12	100	13	011								
T	6	1010	7	0011	8	0010	11	101										
D	5	1011	6	1010	7	0011												
B	4	00110	5	1011														
U	3	00111																

Como se puede ver, aunque las combinaciones de bits son diferentes, cada letra del código de Huffman tiene la misma cantidad de bits que la correspondiente del código de Shannon-Fanon.

1.3 Historia del MPEG y el audio digital

En el año 1987, Leonardo Chiariglione, quien había pertenecido al grupo CCIR (*International Radio Consultative Committee*) encargado de la estandarización de la Televisión de Alta Definición, *High Definition Television* (HDTV); y desilusionado del fracaso en adoptar un único estándar internacional (Japón, Europa y Estados Unidos pugnaban por su propio formato), asistía al encuentro del JPEG (*Joint Pictures Experts Group*, Grupo Unido de Expertos en Imágenes). Este grupo había sido formado por la Organización de Estándares Internacionales ISO y por la Comisión Electrotécnica Internacional IEC para formular un estándar que permitiera disminuir la cantidad de espacio de almacenamiento requerido para las imágenes fijas.

En este encuentro, Chiariglione quedó profundamente impresionado por lo que se podía lograr entre un grupo de expertos cuando no se manejaban los intereses de ninguna industria. Así que se aproximó al Director del JPEG, Hiroshi Yoshuda, y sugirió la creación de un grupo sucesor, que se encargara de estandarizar la codificación digital de las imágenes en movimiento.

De esta manera, en 1988 y con Yoshuda como representante ante la ISO, la organización ISO/IEC crea el Comité Técnico Unido sobre Tecnologías de la Información, Subcomité 29, Grupo de Trabajo 11 (ISO/IEC JTC1/SC29/WG11), más conocido como MPEG (Moving Picture Experts Group), bajo la dirección de Chiariglione, encargándole el desarrollo de estándares para la representación codificada de imágenes en movimiento, la información del audio asociado, y su combinación para la grabación y lectura en un medio de almacenamiento digital.

El primer objetivo del grupo, en ese momento constituido por 12 personas, fue la posibilidad de leer discos compactos con imágenes en movimiento a una tasa cercana a 1.5 Mbps. Ya para el año 1993 se concluía el primer estándar definitivo, conocido como MPEG-1 (numerado como ISO/IEC 11172). Al año siguiente se daba a conocer el estándar MPEG-2 (ISO/IEC 13818) que involucraba compresión de datos (flujos de bits) originalmente a 270 Mbps (sin compresión) hasta lograr una tasa de bits entre 2 y 9 Mbps (después de la compresión). MPEG-2 presentaba compatibilidad con el estándar anterior y fue pensado para imágenes de televisión entrelazadas. El estándar MPEG-3, cuya intención era estandarizar la Televisión de Alta Definición HDTV, fue posteriormente incluido en el MPEG-2.

Los objetivos actuales del grupo son los nuevos estándares MPEG-4 y MPEG-7. MPEG-4 (*Coding of Audio-Visual Objects*, Codificación de Objetos Audiovisuales; con numeración ISO/IEC 14496) fue aprobado de manera formal en los primeros meses del año 2000, aunque posteriormente se han estado incluyendo mejoras adicionales. MPEG-4 permite la manipulación de imágenes por el usuario final a través de Internet y se habla de una posible relación con la realidad virtual.

También se está trabajando en el llamado MPEG-7 (ISO/IEC 15938), cuyo nombre formal es *Multimedia Content Description Interface*, Interface de Descripción del Contenido Multimedia, que intenta abarcar todos los aspectos que involucra la multimedia. Éste se encuentra en su fase final con miras a ser presentado para aprobación definitiva en julio del año 2001.

A finales de 1999, MPEG (actualmente constituido por más de 300 expertos) empezó a trabajar en el nuevo estándar MPEG-21 (*Multimedia Framework*, Estructura Multimedia), cuyo objetivo primordial es proporcionar estándares que estarán fundamentados principalmente en el punto de vista de los usuarios, y no tanto de la industria

2 Psicoacústica

2.1 Introducción

Antes de profundizar sobre matemáticas y los problemas de los sistemas de audio es necesario conocer las particularidades de la percepción humana del sonido. Para lograrlo se debe conocer el mecanismo fisiológico, neurofisiológico y cognoscitivo de la escucha y en consecuencia conocer las limitaciones y cualidades de la percepción del sonido.

También es importante el campo de estudio de la psicoacústica. La psicoacústica es un campo de la ciencia que estudia como percibimos el sonido y como extraemos la información más importante de las señales acústicas. Las principales investigaciones de psicoacústica están dirigidas hacia temas como escucha direccional, pitch, timbre y la percepción del volumen (loudness), análisis de una escena auditiva (la separación de las fuentes de sonido y los parámetros de las señales de audio), además de las funciones del oído, codificación neuronal del sonido, el mecanismo de interacción entre múltiples fuentes de sonido, los caminos neuronales del oído hasta la corteza de la audición, etc., como se puede ver la psicoacústica tiene un campo de estudio muy amplio y también podemos ver porque es tan importante en la compresión de audio, en la siguiente sección se muestran algunos fenómenos que se han encontrado por medio de la psicoacústica.

Oído izquierdo y derecho.

El oído es un órgano altamente desarrollado, por ejemplo, en comparación el ojo sólo puede recibir frecuencias en una octava, pero hay que recordar que el oído solamente es útil con el poder interpretativo del cerebro. Estos juicios mentales forman las bases para que podamos experimentar desde un simple sonido hasta la música. El oído derecho y el izquierdo no difieren fisiológicamente en su capacidad para detectar sonido, pero si hay diferencias en las funciones de sus respectivos hemisferios del cerebro derecho e izquierdo, ya que dividen sus funciones cerebrales. Las principales conexiones de los oídos al cerebro son cruzadas, el oído derecho esta conectado con el hemisferio izquierdo y el oído izquierdo esta conectado con el hemisferio derecho del cerebro. El hemisferio derecho es perceptualmente superior para las palabras. Por el otro lado, principalmente el lóbulo temporal derecho es el que se encarga de procesar información no verbal, por lo tanto percibimos mejor la melodía con el oído izquierdo.

Respuesta logarítmica del oído.

La respuesta en frecuencia del oído es logarítmica, esto se puede demostrar por medio de la percepción de los intervalos musicales. Por ejemplo, el intervalo entre 100 y 200Hz se percibe como una octava, como se percibe el intervalo entre 1000 y 2000 Hz. Sin embargo linealmente el segundo intervalo es mucho más grande que el primero y aun así el oído los percibe como el mismo intervalo. Por esta razón se usa una notación musical logarítmica. Cada cuatro y medio espacios o líneas en el pentagrama representan una octava, lo que podría ser algunas decenas de hertz más alto o algunos miles, dependiendo de la clave y líneas usadas.

Pitch.

Frecuencia es una medida física mientras que el pitch no lo es, es una medida subjetiva y compleja que se basa en la frecuencia como otras cantidades físicas como la longitud de onda y la intensidad. Por ejemplo, si se reproduce una onda senoidal a 200Hz a un volumen bajo y luego a un volumen más fuerte la mayoría de las personas dirán que el sonido tiene una pitch más bajo. De hecho se necesita in aumento de 10% en la frecuencia para mantener una evaluación subjetiva de pitch constante a bajas frecuencias. Por otro lado, en la región más sensible, de 1 a 5 kHz, casi no hay cambio del pitch con sonidos más fuertes. También, con tonos musicales el efecto es mucho menor.

Beat frecuencies.

Beat frecuencies ocurre cuando dos tonos aproximadamente iguales son reproducidos simultáneamente. Cuando la diferencia entre las frecuencias de los tonos es por si misma una frecuencia audible. El efecto es especialmente audible cuando las frecuencias son altas, los tonos son muy fuertes, y separados no más de una quinta. Aunque hay personas que dicen escuchar la suma de los tonos. También puede suceder que se escuche un tono intermedio, especialmente debajo de los 200 Hz donde la habilidad para discriminar entre tonos disminuye. Por ejemplo, si simultáneamente se reproducen tonos de 65 y 98 Hz no se escuchara como un perfecto quinto, pero si como un tono de 82 Hz. Por otro lado, cuando tonos debajo de los 500 Hz se escuchan uno tras otro, el oído puede diferenciar entre pitches separados solo 2 Hz.

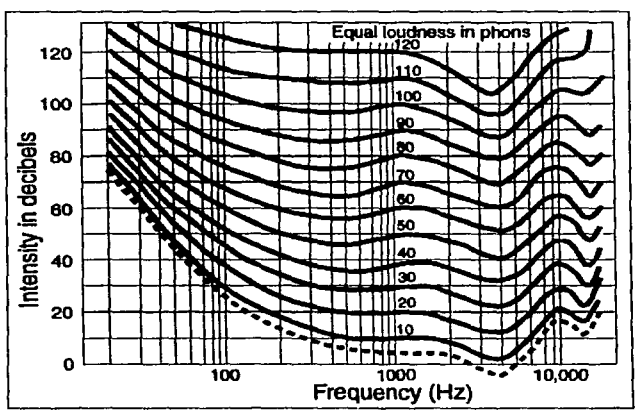


Figura 2.1 Curvas de Robison-Dadson (equal loudness).

Respuesta en frecuencia del oído

El oído tiene un rango dinámico muy amplio, el umbral de dolor es de 120 dB SPL y tiene una intensidad de 1,000,000,000,000 veces más grande que el umbral de escucha. La sensibilidad depende de la frecuencia, la máxima sensibilidad se encuentra entre 1 a 5 kHz, con una relativa insensibilidad a las altas y bajas frecuencias esto lo podemos observar

por medio de los contornos de igual intensidad (equal loudness) como las curvas de Robinson-Dadson (figura 2.1) cada contorno describe el nivel que debe tener el tono para que se perciba con la misma fuerza en diferentes frecuencias estas curvas fueron realizadas por medio de pruebas. El contorno más bajo describe el nivel mínimo audible. Es decir el nivel mínimo de presión de sonido a través de la banda audible de frecuencias, que una persona con oído normal puede percibir. Por ejemplo, un tono de 30 Hz apenas audible debe ser 60dB más fuerte que una tono apenas audible de 4 kHz. La respuesta varia con la fuerza del sonido. Los sonidos más fuertes tienen la respuesta más plana.

Ubicación de la fuentes de sonido.

Todas las características del sonido, excepto una, pueden percibirse con un oído; la localización sólo puede realizarse con 2 oídos. Cuando el sonido se origina desde un lado, el cerebro usa claves como diferencias en intensidad, la complejidad de la forma de onda y tiempo de retraso para determinar la dirección de origen del sonido. Por ejemplo cuando un sonido igual es reproducido en las dos bocinas, el cerebro en lugar de ubicar un sonido en la derecha y un sonido en la izquierda, el cerebro lo ubica en el centro entre las dos bocinas. Debido a que los dos oídos reciben la misma información. De esta manera el estereo no es más que dos canales monoaurales. El resto es simple ilusión.

Entropía perceptual.

Este tema es de mucha importancia para este trabajo ya que de ella depende lo eficiente que pueda ser el algoritmo de compresión. El oído percibe solo una porción de la información en una señal de audio; esta puede ser llamada entropía perceptual. Una señal con entropía pequeña puede ser reducida eficientemente, las señales con entropía grande no pueden ser reducidas. Por esta razón, un codificador tiene una tasa de datos variable en la salida, cuando la información es pobre la tasa es baja y cuando es más información la tasa aumenta, es decir que aunque la frecuencia de muestreo sea constante la entropía de la señal de audio varía, provocando que la tasa de datos varíe. Usando psicoacústica se puede remover información irrelevante. La señal original no puede ser reconstruida exactamente. Un sistema de compresión de datos de audio, reduce la entropía, por medio del modelado de la entropía perceptual. Sólo la información irrelevante se elimina, por lo tanto esa reducción de datos puede ser inaudible. Un codificador perceptual debe usar modelos psicoacústicos para identificar información irrelevante contenida en una señal de audio.

Tradicionalmente, un diseñador de un sistema de audio, tiene que usar parámetro objetivos como sus metas de diseño (respuesta plana en frecuencia, mínimo ruido, etc.). Los diseñadores de codificadores preceptuales reconocen que el receptor final es un sistema de audición humana. Y usan el desempeño del oído humano como su criterio de diseño. Después de todo, cualquier experiencia musical es puramente subjetiva.

2.2 *Fisiología del oído humano.*

El oido realiza la transformación de energía acústica a mecánica y finalmente se envían impulsos eléctricos al cerebro[5], en donde se percibe la información contenida en el sonido. Una vista simplificada del diseño del oído humano se muestra a continuación, El oído externo colecta el sonido y su intrincada forma permite una cierta direccionalidad. El

[5] www.helsinki.fi/~ssyreeni/dsound/dsound-c-04

canal auditivo resuena aproximadamente a 3 kHz, previendo una sensibilidad extra en un rango crítico para la inteligibilidad del habla. El tímpano traduce la energía acústica en energía mecánica; este alcanza su máxima excursión a los 120 SPL y después empieza a distorsionar la forma de onda. Los 3 huesos en el oído medio coloquialmente conocidos como , martillo, yunque y estribo (los 3 huesos más pequeños del cuerpo), permiten un acoplamiento entre las impedancias del aire que es fácilmente comprimible y el fluido no comprimible que se encuentra en el oído interno llamado endolinfa, además ayudan a regular el nivel de la señal para evitar daños si la señal es muy grande. Los canales vestibulares no afectan la audición, pero son parte del sistema de detección de movimiento proporcionando sensación de balance. La membrana basilar enrollada detecta la amplitud y frecuencia del sonido, estas vibraciones son convertidas en impulsos eléctricos y enviados al cerebro como información neuronal a lo largo de un grupo de fibras nerviosas. El cerebro decodifica al período del estímulo y apunta a la máxima estimulación a lo largo de la membrana para determinar la frecuencia; y la actividad de estimulación en las regiones circundantes es ignorada .

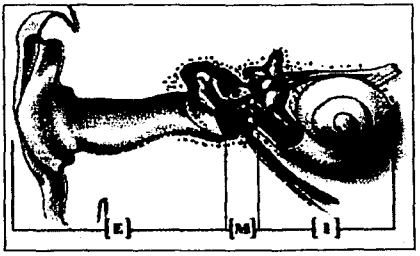


Figura 2.2. Imagen simplificada del oído. [E] Oído externo. [M] Oído Medio. [I] Oído Interno

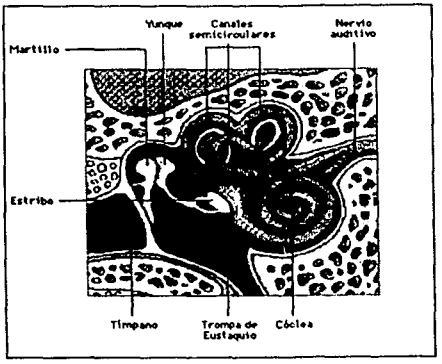


Figura 2.3 Esquema del oído interno y Medio

El ancho de la membrana basilar varía a lo largo de la coclea. Cerca de la ventana, la membrana es bastante angosta mientras que al final la membrana es más ancha. La densidad y dureza varían de manera similar (cerca de la ventana hay una cantidad considerable y al final hay muchos menos). También las células de cabello y la estereocilia cambian de manera similar, al final los cabellos son más flexibles y largos a diferencia de los que se encuentran cerca de la ventana, que son más cortos y duros. Todas estas características afectan la manera en que se comporta el órgano de Corti, es decir, las frecuencias más altas se censan al principio de la membrana y las frecuencias bajas al final de la membrana. Esto tiene como consecuencia que el oído puede realizar la separación en frecuencias del sonido. Además los nervios que llevan la información al cerebro también están ordenados por frecuencia, este patrón se repite en la estructura cerebral. A esto se le llama *organización tonotopica.*

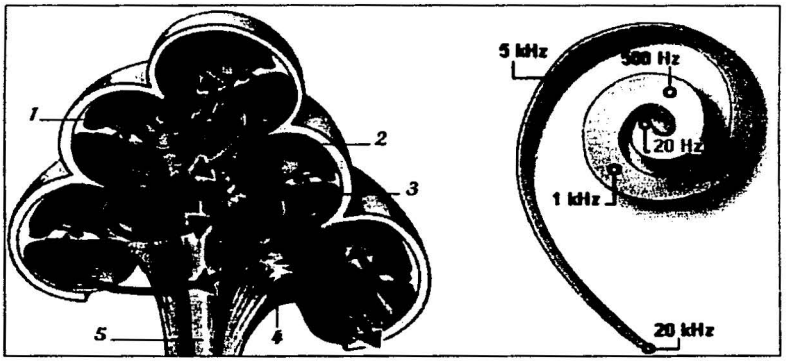


Figura 2.4(1) Conducto coclear. (2) scala vestibuli. (3) scala typani.
(4) ganglio espiral. (5)Nervios auditivos.

Esta descomposición en frecuencias además de ser algo muy importante para el funcionamiento de un codificador perceptual también es muy conveniente si tomamos en cuenta que las neuronas no pueden funcionar a tasa menores a 1000Hz, por lo que una codificación directa del sonido no funcionaria (también hay que recordar que las neuronas funcionan como pulsos binarios).

La membrana basilar contiene alrededor de 30,000 células de cabellos acomodados en múltiples columnas a lo largo de la membrana basilar de aproximadamente 32 mm de largo, este es el órgano de Corti. Las células detectan vibraciones locales de la membrana basilar y transportan la información de audio al cerebro por medio de impulsos eléctricos. La discriminación de frecuencias dicta que a bajas frecuencias se pueden distinguir diferencias de solo unos hertz. Sin embargo, a frecuencias altas los tonos deben diferir por miles de hertz para distinguirlos. En cualquier caso, los células de cabellos responden a la estimulación más fuerte en su región local, esto es llamado una banda crítica, un concepto introducido por Harvey Fletcher. Experimentos muestran que estas bandas críticas son

mucho más angostas a frecuencias bajas que a altas frecuencias; hay 3 o 4 bandas críticas debajo de los 5 kHz; el oído recibe más información de bajas frecuencias y menos de las altas frecuencias. En frecuencias entre 20 y 400 Hz las bandas críticas son de aproximadamente 100 Hz y aproximadamente de 1 octava para frecuencias entre 1 y 7 kHz.

Eberhard Zwicker modeló al oído con 24 bandas críticas arbitrarias para frecuencias debajo de los 15 kHz; y una más para frecuencias de 15 a 20 kHz fisiológicamente cada banda ocupa una longitud de 1.3 mm en la membrana basilar, con 1300 células de cabellos primario. La banda crítica para un tono senoidal de 1 kHz es de aproximadamente 160 Hz de ancho; por lo tanto una ruido o señal de error que tiene 160 Hz de ancho y centrada a 1 kHz, es solo audible solo si es más grande que un tono senoidal de 1 kHz. Como otro ejemplo, la fuerza (loudness) de una banda de ruido a una constante SPL, permanece constante cuando el ancho de la banda aumenta; sin embargo cuando se excede el ancho de la banda crítica la fuerza (loudness) aumenta. Las bandas críticas son similares a una analizador de espectro con frecuencias centrales variables. Las bandas críticas no son fijas, son variables en frecuencia y cualquier tono audible creara una banda crítica centrada en esa frecuencia. El concepto de banda crítica es un fenómeno empírico. Una banda crítica es en donde subjetivamente la respuesta cambia abruptamente.

Las bandas críticas se han usado para explicar la consonancia y disonancia. Los tonos con una diferencia de frecuencia mayor a una banda crítica son generalmente más consonantes; los tonos con un intervalo menor que una banda críticas tiene a ser disonantes y si es un intervalo de alrededor de 0.2 veces una banda crítica se vuelven aun más disonantes. Además la disonancia tiende a aumentar a bajas frecuencias: por ejemplo los músicos tienden a evitar terceras a bajas frecuencias .

El bark (en honor al físico alemán Georg 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:

$$1\ bark(Hz) \cong f/100 \qquad para\ \ f < 500\ Hz$$

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

f = frecuencia.

En la siguiente grafica se muestra como la forma de un umbral de enmascaramiento expresada en Barks tiene la misma forma sin importar la frecuencia de la señal enmascaradora.

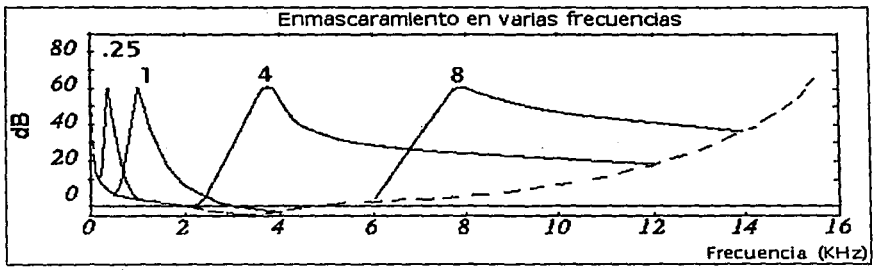


Figura 2.5 Enmascaramiento en varias frecuencias [kHz].

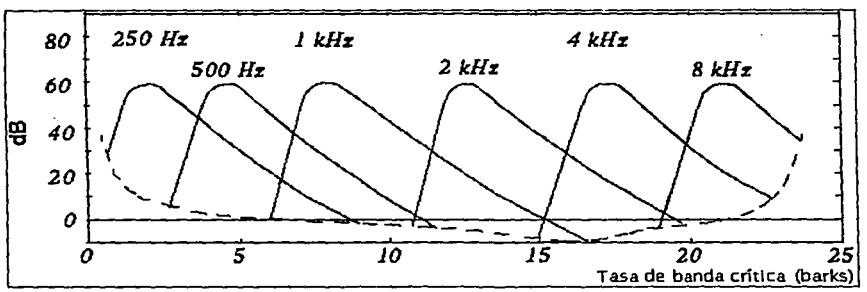


Figura 2.6 Enmascaramiento en varias frecuencias [barks].

La teoría pitch place explica la acción de la membrana basilar. Transportada por el fluido circundante un sonido viaja una cierta longitud de la membrana y se detiene en un lugar particular a lo largo de la membrana, donde ocurre la vibración más grande de la membrana, en donde cada lugar corresponde a diferentes frecuencias. Específicamente las frecuencias altas son censadas en la membrana, cerca del oído medio, mientras que la bajas frecuencias son censadas al final de la membrana. Una onda excitada por un sonido de alta frecuencia no alcanza el final de la membrana basilar. Sin embargo, a bajas frecuencias el sonido pasara por todos los lugares de altas frecuencias y alcanzara el final de la membrana. Porque las células de cabello tienden a vibrar a la frecuencia de mayor estimulación. Ellas transformaran la frecuencia en una banda crítica, ignorando las estimulaciones menores alrededor de una estimulación mayor. Esta curva de excitación es descrita por la función coclear expandida. Esto explica, por ejemplo, porque mediciones con bandas anchas no describen el fenómeno del umbral el cual está basado en condiciones de frecuencia local. Hay cerca de 620 grados de frecuencias diferenciables distribuidas a lo largo de la membrana basilar, por lo tanto una resolución de 1/25 Bark es razonable. En conclusión las bandas críticas son importantes para la codificación perceptual porque esto muestra que el oído discrimina entre energía en la banda, y energía fuera de la banda en particular esto promueve el enmascaramiento.

2.3 Umbral de audición y enmascaramiento:

Existen dos fenómenos fundamentales que gobiernan la audición humana, son el mínimo umbral de audición y la amplitud de enmascaramiento, como se muestra en la figura 2.7 . La curva de umbral de audición describe el nivel mínimo en el cual el oído puede detectar un tono a una frecuencia dada. El umbral esta referido a 0 dB a 1 kHz. El oído es más sensitivo en el rango de 1 a 5 Hz, donde podemos oír varias señales debajo de los 0 dB de referencia. Generalmente, 2 tonos de igual potencia y diferente frecuencia no sonarán con igual fuerza. De manera similar la habilidad de escuchar el ruido y la distorsión varia de acuerdo con la frecuencia. La sensibilidad disminuye a altas y bajas frecuencias. Por ejemplo, un tono de 20 Hz tiene que ser 70 dB más fuerte que un tono de 1 kHz para apenas ser audible. La fuerza percibida puede ser expresada en sones; un son describe la fuerza de un tono a 1 kHz y 40 dB SPL. Una fuerza de 2 sones corresponde a 50 dB SPL; de manera similar, cada vez que se duplica en sones es un aumento de 10 dB SPL. Por ejemplo 64 sones corresponden a 100 dB SPL. Un codificador perceptual compara la señal de entrada con el umbral mínimo, y descarta señales que caen debajo del umbral, porque estas señales no podrán ser escuchadas.

Enmascaramiento por amplitud sucede cuando un tono cambia la curva de umbral y la eleva en una región de frecuencias que circundan el tono. El umbral de enmascaramiento describe el nivel donde es apenas audible. Cuando los tonos son reproducidos simultáneamente, el enmascaramiento ocurre cuando los tonos más fuertes completamente desvanecen los tonos más débiles. Por ejemplo un tono a 500 Hz puede enmascarar a un tono más débil de 600 Hz. En otras palabras, la presencia física de un sonido ciertamente no asegura que sea escuchada y por el contrario puede asegurar que otro no se escuche. El sonido fuerte es llamado enmascarador y el sonido débil se llama enmascarado. La teoría de enmascaramiento argumenta que un tono débil es apenas detectable cuando la energía es igual a parte de la señal de enmascaramiento en una banda crítica. Generalmente, dependiendo de la amplitud relativa los tonos son enmascarados por tonos más fuertes en un frecuencia similar.

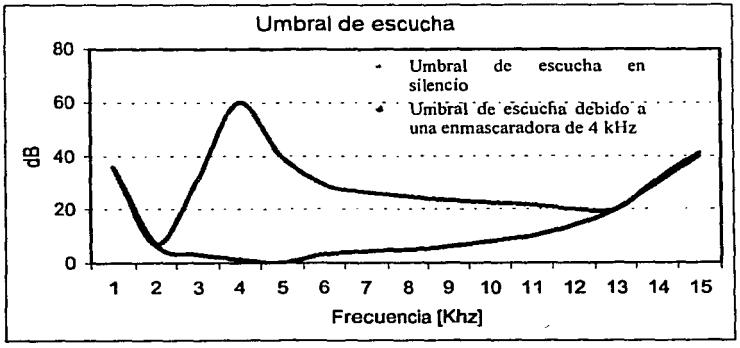


Figura 2.7 Umbral de silencio y Umbral debido a una enmascaradora.

Los umbrales de enmascaramiento en ocasiones son expresadas como un nivel de excitación, esto se obtiene sumando de 2 a 6 dB al nivel de presión acústica en donde el tono es apenas audible. Bajas frecuencias pueden interferir con la percepción de las altas frecuencias. El enmascaramiento puede sobrepasar las bandas críticas adyacentes cuando una señal es fuerte, o contiene armónicas; por ejemplo una señal compleja de 1 kHz puede enmascarar a una señal de 2 kHz. Bajas amplitudes provocan pequeños enmascaramientos. Bandas angostas como tonos senoidales también provocan pequeños enmascaramiento. Del mismo modo tonos más complejos proveen enmascaramientos más grandes con curvas de enmascaramiento con una banda muy grande de enmascaramiento.

Muchas curvas de enmascaramiento se han derivado de estudios en los cuales se han usado como estímulos enmascaradores tanto tonos simples como bandas angostas de ruido. Generalmente un tono simple como enmascarador produce variaciones en la curva de enmascaramiento, cerca del tono debido a beat interferencia entre el enmascarador y el tono enmascarado. Las bandas de ruido angostas no producen este efecto. Además los tonos enmascaradores parecen extenderse más fácilmente en altas frecuencias que los enmascaradores de ruido.

Muchos estudios examinan enmascaramiento de un tono por otro tono o por ruido; en codificación perceptual, es al contrario, se preocupan más por el ruido de cuantización que es enmascarado por un tono de música. Como se puede notar las bandas de ruido provocan curvas de enmascaramiento que difieren de los tonos senoidales; generalmente, música es más parecida a tonos en sus propiedades de enmascaramiento. Enmascaradores de tonos senoidales se usan en modelos de enmascaramiento porque estos proveen el peor caso. Tonos complejos provocan enmascaramientos más grandes. En general un tono es inaudible cuando esta 4 dB debajo y a 1/3 de octava de un ruido de enmascaramiento en una banda crítica. Por el contrario, cuando una banda de ruido esta a 1/3 de octava de un tono puro, el tono debe ser 24 dB más suave que el tono, es decir que debe ser 20 dB más fuerte para enmascara un ruido. Finalmente para afinar un criterio de enmascaramiento se necesita escuchar con cuidado. Además, los codificadores presumen un umbral de silencio que debajo de las curvas estándar, por que no se puede saber a que nivel se escuchara finalmente el contenido. Relativamente pocos estudios han usado a la música como estímulo. Sin embargo, en general se acepta que las curvas de tonos y ruido son modelos válidos para la codificación de la música.

Las curvas de enmascaramiento simultaneas son asimétricas. La inclinación de las curvas modificadas son menos empinadas en el lado de la altas frecuencias. Debido a esto es relativamente fácil para un tono bajo el enmascarar a un tono alto, pero el caso inverso es más difícil. Específicamente la inclinación inferior es de 27 dB /bark; la superior es de -20 a -5 dependiendo de la amplitud del enmascarador. Los enmascaradores de bajo nivel influencian bandas relativamente angostas de frecuencias. Sin embargo conforme el nivel del enmascarador aumenta las curvas de umbral se vuelven más anchas, y en particular la inclinación superior disminuye; y la inclinación inferior casi no se afecta.

2.4 Enmascaramiento temporal

El enmascaramiento temporal sucede cuando los tonos son reproducidos cerca en el tiempo pero no simultáneamente. Un tono débil puede ser enmascarado por un ruido que ocurre después, a esto se le llama preenmascaramiento. También una señal puede ser enmascarada por un ruido que termina antes que la señal empiece, a esto se le llama posenmascaramiento. Es decir, un tono fuerte que sucede antes o después que un tono débil puede enmascararlo . Al igual que el enmascaramiento simultáneo aumenta cuando las diferencias en la frecuencias disminuyen y también aumentan cuando las diferencias en tiempo disminuyen. Un tono de 80 dB, podría crear un posenmascaramiento de 40 dB por 20ms y uno de 0 dB por 200 ms, en preenmascaramiento puede provocar una mascara de 60 dB por 1 ms, y de 0 dB por 25 ms. El enmascaramiento temporal disminuye cuando la duración de enmascarador disminuye, además un tono es posenmascarado por otro tono que suena antes cuando están cerca en frecuencia o cuando el tono enmascarador es más bajo en frecuencia. Lógicamente el enmascaramiento es menos fuerte cuando es temporal que cuando ocurren al mismo tiempo.

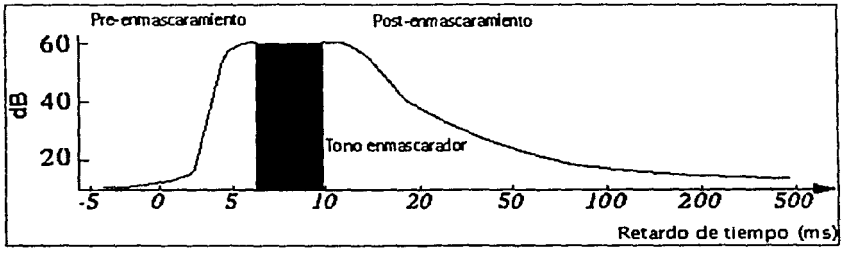


Figura 2.8 Umbral de enmascaramiento en el tiempo debido a un sonido fuerte de 60 dB.

El enmascaramiento temporal sugiere que el cerebro integra los sonido en un periodo de tiempo, y procesa la información en ráfagas en la corteza de la audición. Pero también podría ser que el cerebro solo procesa los sonidos fuertes más rápido que los sonidos débiles. De cualquier manera el enmascaramiento temporal es muy importante en la codificación en el dominio de la frecuencia. Estos codificadores tienen una limitada resolución temporal porque ellos operan en bloques o muestras, debido a esto provocan un error en el tiempo. Idealmente, los banco de filtros deben proveer una resolución temporal de 2 o 4 ms. actuando juntos, el enmascaramiento temporal y de amplitud, el contorno de la mascara puede ser descrito en un plano en el dominio de la frecuencia y tiempo, y los sonido que caigan debajo del contorno serán enmascarados, la función de los codificadores preceptúales, deben identificar los contorno que cambiaran con las condiciones de la señal y codificar la señal apropiadamente.

En la siguiente gráfica se muestran ambos fenómenos el enmascaramiento temporal y el enmascaramiento en frecuencia.

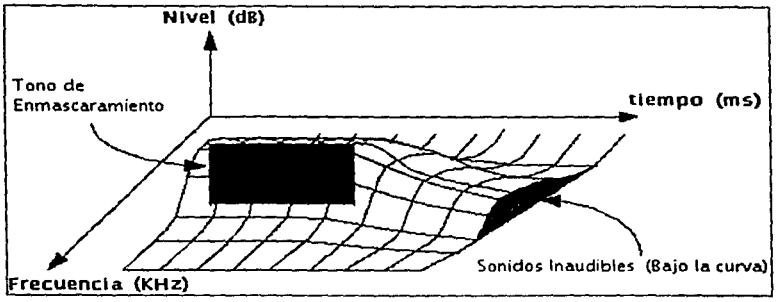


Figura 2.9 Gráfica del umbral de enmascaramiento en el tiempo y la frecuencia.

La reproducción de múltiples canales representan más oportunidades para la reducción de datos. El sistema Joint estereo toma ventaja de la redundancia entre los canales para aumentar su eficiencia . La tasas de datos de una señal estereo es el doble de una mono, pero muchos programas no son en realidad dos monoaurales. En lugar de eso los canales comparten algunos niveles e información en fase, para crear imágenes fantasma. Los códigos de Joint estereo codifican la información sólo una vez. Debido a esto un canal Joint estereo de 256 kbps se podría desempeñar mejor que 2 de 128 kbps.

Usando diversas y dinámicas claves psicoacústicas y análisis de señales, los componentes inaudibles de una señal pueden ser removidos con una degradación aceptable. Por ejemplo, un sonido fuerte en un canal puede enmascarar a otros suaves en otros canales. Cerca de los 2 kHz, la localización se realiza principalmente por amplitud porque el oído no puede seguir formas de onda rápidas individualmente. Solo sigue la señal envolvente, no su fase. Debido a esto la forma de onda por si misma se convierte menos crítica. Además el oído tiene una habilidad limitada para localizar sonidos similares en frecuencia. Para crear un sonido ambiental, las altas frecuencias en cada canal pueden ser divididas en bandas y combinadas banda por banda en un canal compuesto. Las bandas de un canal común pueden ser reproducidos en todas las bocinas o localizadas en alguna bocina específica según la envolvente. A esto se le pueden agregar más criterios de enmascaramiento para canales compuestos. Muchos sistemas de audio usan 5.1 canales, 3 frontales, 2 canales independientes traseros y un subwoofer, en general los bits requeridos para codificar una señal de varios canales es proporcional a la raíz cuadrada del número de canales, por ejemplo para un sistemas de 5.1 canales requeriría 2.26 veces el número de bits necesario para codificar un solo canal.

3 Diferentes sistemas de compresión

3.1 Introducción

Toda compresión de datos de audio se basa en la comprensión del mecanismo auditivo, por lo que constituye una forma de codificación perceptual. El oído es sólo capaz de extraer una cierta proporción de la información contenida en un determinado sonido. A esto se le puede denominar entropía perceptual, siendo redundante el sonido adicional. Un sistema ideal debe eliminar toda redundancia, dejando únicamente la entropía.

Existen muchos tipos diferentes de compresión de audio y cada uno permite un factor de compresión diferente. Algunas aplicaciones como DCC (Digital Compact Cassette) y DAB (Digital Audio Broadcasting) requieren un valor de 0.25. Para el Minidisco, es de 0.2. La transmisión de audio por Internet requiere más compresión todavía, que sólo puede realizarse empleando técnicas sofisticadas, a continuación se presentan unas de estas.

La codificación subbanda imita el mecanismo de análisis en frecuencia del oído humano y divide el espectro de audio en un gran número de bandas de frecuencia diferentes con el fin de poder explotar el hecho de que la mayoría de las bandas contienen señales cuyo nivel es inferior al de la señal más alta. Las señales en estas bandas pueden ser entonces cuantificadas independientemente. El error de cuantificación que resulta es confinado a los límites de frecuencia de la banda y así este puede arreglarse para ser enmascarado por el material del programa. Las técnicas usadas en las capas 1 y 2 de MPEG audio son basadas en la codificación de la subbanda como son aquéllas usadas en el DCC.

En la codificación por transformación la forma de onda de audio en el dominio del tiempo es convertida a una representación en el dominio de la frecuencia como una transformada de Fourier, Discreta del Coseno o Wavelet. La codificación por transformación toma ventaja del hecho de que la amplitud o cubierta de una señal de audio cambia relativamente despacio y así los coeficientes de la transformada pueden transmitirse relativamente con poca frecuencia. Claramente tal aproximación falla en presencia de transitorios y se requieren en la práctica sistemas adaptables. Los transitorios causan que los coeficientes puedan ser frecuentemente actualizados mientras que en las partes estacionarias de la señal como las notas sostenidas la tasa de actualización puede reducirse. La codificación por Transformada Discreta del Coseno se usa en la capa 3 (layer 3) de MPEG audio y en el sistema de compresión del Minidisco.

3.1.1 Codificación por Sub-Bandas

En la compresión de los datos, una subbanda aprovecha el hecho de que los sonidos reales no tienen una energía espectral uniforme. La longitud de la palabra del audio PCM está basada en el rango dinámico requerido. Cuando una señal con un espectro no uniforme es transmitida por PCM, todo el rango dinámico es ocupado únicamente por la componente espectral más alta, y todas las demás componentes son codificadas con excesivo headroom (área entre el nivel normal de funcionamiento y el nivel de recorte). En su forma más simple, la codificación de la subbanda funciona dividiendo la señal de audio en un número

de bandas de frecuencia y comprimiendo y expandiendo cada banda de acuerdo con su propio nivel. Las bandas en las que hay poca energía dan como resultado amplitudes pequeñas que pueden transmitirse con una longitud de palabra corta. Por tanto, cada banda se traduce en muestras de longitud variable, pero la suma de todas las longitudes de palabra de la muestra es inferior a la de la PCM, pudiendo obtenerse así una ganancia de codificación.

El número de subbandas que pueden utilizarse depende de qué otra técnica de compresión se vaya a combinar con la codificación de la subbanda. Si se tiene la intención de utilizar la compresión basada en el enmascaramiento auditivo, es preferible que las subbandas sean más estrechas que las bandas críticas del oído y, por tanto, se requerirá un gran número; así, por ejemplo en MPEG y PASC (Precision Adaptative Subband Coding) se emplean 32 sub-bandas. **La Figura 3.1** muestra la condición crítica en la que el tono del enmascaramiento se encuentra en el límite superior de la subbanda. Se observará que cuanto más estrecha es la subbanda, mayor es el ruido de recuantificación que puede enmascararse. No obstante, la utilización de un número excesivo de subbandas acentúa la complejidad y el retardo de codificación, y también arriesga el pre-eco en los transitorios que exceden el enmascaramiento temporal.

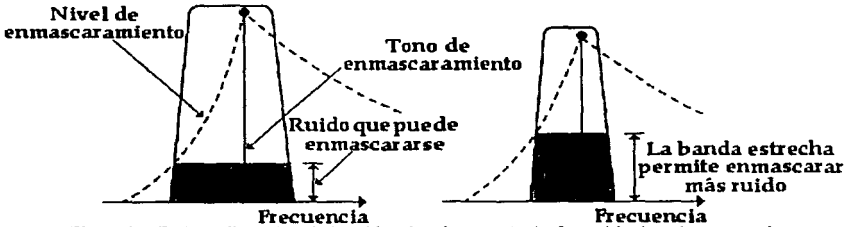


Figura 3.1 En la codificación de la subbanda, el caso más desfavorable tiene lugar cuando el tono de enmascaramiento se encuentra en el límite superior de la subbanda

Las Figuras 3.2 y 3.3 muestran los diagramas de bloques de un codificador y un decodificador de subbandas, respectivamente. En la entrada, el rango de frecuencias es dividido en subbandas mediante un banco de filtros tal como un filtro especular en cuadratura. Los datos descompuestos de la subbanda se organizan en bloques de tamaño fijo, antes del proceso de reducción. Aunque todas las subbandas pueden utilizar bloques de la misma longitud, algunos codificadores pueden utilizar bloques que se hacen más largos a medida que disminuye la frecuencia de la subbanda. Los bloques de las subbandas también se denominan bins de frecuencia.

La ganancia de codificación se obtiene cuando la forma de onda de cada banda pasa a través de un recuantificador. La recuantificación se consigue multiplicando los valores de las muestras por una constante y redondeando el resultado por exceso o por defecto de acuerdo con la longitud de palabra requerida. Por ejemplo, si en una subbanda determinada la forma de onda es de 36 dB o menos a fondo de escala, al menos habrá 6 bits en cada muestra que sólo reproducen el bit de signo. Multiplicando por 64, entrarán en uso los bits

de orden superior de la muestra, permitiendo que se pierdan bits en el extremo más bajo al redondear a una longitud de palabra menor. Cuanto menor es la longitud de palabra, mayor es la ganancia de codificación, pero más toscos resultan los escalones de cuantificación y, por tanto, el nivel de error de cuantificación.

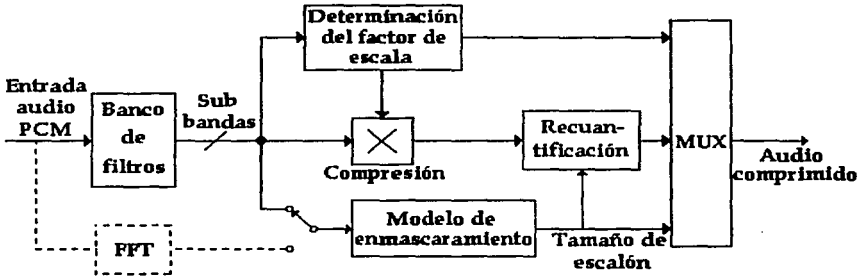


Figura 3.2 Diagrama de bloques de un codificador de subbandas

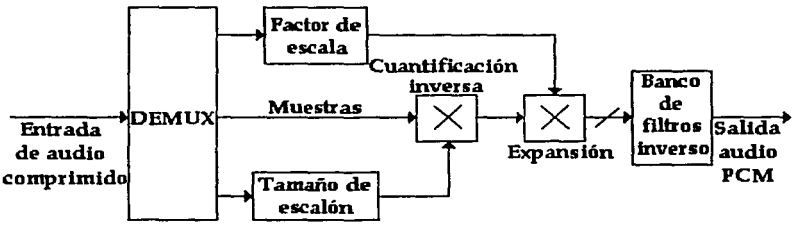


Figura 3.3 Diagrama de bloques de un decodificador de subbandas

3.1.2 Codificación por Transformación

El análisis de Fourier permite representar cualquier forma de onda mediante un conjunto de componentes armónicamente relacionados de amplitud y fase adecuadas. La transformada de una forma de onda de audio típica varía de manera relativamente lenta. La lenta señal sonora procedente del tubo de un órgano o de la cuerda de un violín, o el lento decrecimiento de la mayoría de los sonidos musicales, permite la reducción de la frecuencia a la que la transformada es muestreada, obteniéndose una ganancia de codificación. Es posible obtener una ganancia de codificación adicional si las componentes que experimentarán el enmascaramiento se cuantifican de manera más rudimentaria.

Las transformadas prácticas requieren bloques de muestras en lugar de cadenas interminables. La solución está en cortar la forma de onda en cortos segmentos solapados y, seguidamente, transformar cada uno de ellos individualmente tal como indica la Figura 3.4. De este modo, cada muestra de entrada aparece en sólo dos transformadas, pero con una ponderación variable dependiendo de su posición en el eje temporal.

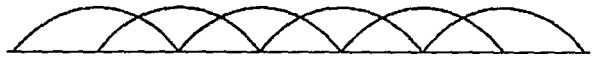


Figura 3.4 La codificación por transformación se realiza en la práctica en bloques cortos

La DFT (Discrete Frequency Transform o Transformada de Frecuencia Discreta) requiere gran número de cálculos, debido al requisito de tener que utilizar una aritmética compleja para obtener la fase de las componentes, así como la amplitud. Una alternativa consiste en emplear la Transformada Discreta del Coseno (DCT). Esta presenta una ventaja cuando se utiliza con ventanas solapadas. En la Transformada Discreta del Coseno Modificada (MDCT), se usan ventanas con un solapamiento del 50%. De este modo, se obtiene el doble de coeficientes necesarios, que se submuestrean por un factor de dos para obtener una transformada muestreada críticamente, lo cual tiene como resultado un efecto un aliasing potencial en el dominio de la frecuencia. Sin embargo, variando levemente la transformada, los productos de aliasing en la segunda mitad de una determinada ventana son iguales en tamaño, pero de polaridad opuesta a los productos de aliasing de la primera mitad de la siguiente ventana, por lo que se eliminarán en su reconstrucción. Éste es el principio de la eliminación del aliasing en el dominio temporal (TDAC, Time Domain Aliasing Cancellation).

La recuantificación realizada en el codificador eleva el ruido de cuantificación en el bin de la frecuencia, pero lo hace durante todo el tiempo que dura el bloque. La Figura 3.5 muestra que, si se produce un transitorio hacia el extremo final de un bloque, el decodificador reproduce la forma de onda correctamente, pero el ruido de cuantificación comenzará al principio del bloque y puede dar lugar a un pre-eco en el que el ruido se oye antes que el transitorio.

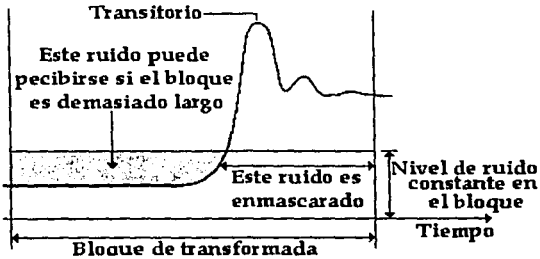


Figura 3.5 Transitorio en el final de un bloque de una transformada

La solución es utilizar una ventana de tiempo variable de acuerdo con el contenido del transitorio de la forma de onda de audio. Cuando se producen transitorios musicales, se necesitan bloques cortos, por lo que la resolución de la frecuencia y, por tanto, la ganancia de codificación será baja. En otras ocasiones, los bloques pueden hacerse más grandes,

mejorando así la resolución de la frecuencia de la transformada y obteniéndose una mayor ganancia de codificación.

3.2 Codificación MPEG de Audio

El estándar MPEG Audio contempla tres niveles diferentes de codificación-decodificación de la señal de audio, de los cuales sólo el primero está totalmente terminado. Los otros dos son aplicables, y de hecho se utilizan habitualmente, pero siguen abiertos a ampliaciones. Estos tres niveles son:

MPEG-1: "Codificación de imágenes en movimiento y audio asociado para medios de almacenamiento digital hasta 1.5 Mbit/s"

MPEG-2: "Codificación genérica de imágenes en movimiento e información de audio asociada"

MPEG-3: la planificación original contemplaba su aplicación a sistemas HDTV; finalmente fue incluido dentro de MPEG-2.

MPEG-4: "Codificación de objetos audiovisuales"

A su vez, MPEG describe tres capas de codificación de audio denominados capa-1, capa-2 y capa-3. Del primero al tercero aumentan tanto la complejidad del codificador como la calidad del sonido. Los tres son compatibles jerárquicamente, esto es, el decodificador capa-i es capaz de interpretar información producida por un codificador capa-i y todos los niveles por debajo del i. Así, un decodificador capa-3 acepta los tres niveles de codificación, mientras la capa-2 sólo acepta el 1 y el 2.

MPEG define, para cada capa, el formato del bitstream y el decodificador (que puede ser implementado de diferentes maneras). Con vistas a admitir futuras mejoras no se define el codificador, pero en un apartado informativo se da un ejemplo de codificador para cada uno de las capas. Hay que decir que tanto MPEG-1 como MPEG-2 emplean estas tres capas, pero este último añade nuevas características.

3.2.1 MPEG-1[6]

Las normas MPEG-1 de audio definen tres capas (layers) de codificación, que se distinguen por su tasa de compresión para una calidad de audio percibida dada.

La especificación MPEG-1 de audio prevé cuatro modos principales de transmisión:

Estereo: los canales izquierdo y derecho se codifican de manera completamente independiente

Joint_stereo: aprovechamiento de la redundancia entre los canales izquierdo y derecho a fin de reducir el flujo (con dos codificaciones posibles; intensity_stereo o MS_stereo)

Dual_channel: los dos canales son independientes (sonido bilingüe, por ejemplo)

Mono: un solo canal de sonido

[6] Para más información revizar la norma ISO/IEC 11172 - Coding Of Moving Picture And Associated Audio For Digital Storage Media At Up To About 1,5 Mbit/s - Part 3: Audio, ISO/IEC.

Este es el sistema que describe la norma ISO en lo referente al sistema MPEG-1:

Codificación: el codificador procesa la señal de audio digital y produce el bitstream empaquetado para su almacenamiento y/o transmisión. El algoritmo de codificación no está determinado, y puede utilizar enmascaramiento, cuantización variable y escalado. Sin embargo, debe ajustarse a las especificaciones del decodificador.

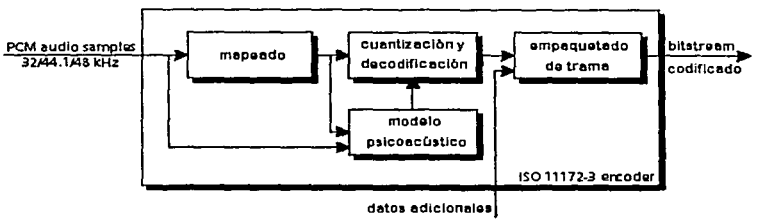


Figura 3.6 Codificador según la norma ISO 11172-3

Las muestras se introducen en el codificador y a continuación el mapeador crea una representación filtrada y submuestreada de la señal de entrada. Las muestras mapeadas se denominan tanto muestras de subbanda (capas 1 y 2) como muestras de subbanda transformadas (capa 3). El modelo psicoacústico crea una serie de datos (dependiendo de la implementación del codificador) que sirven para controlar la cuantización y codificación. Este último bloque crea a su vez su propia serie de datos, de nuevo dependiendo de la implementación. Por último, el bloque de empaquetamiento de trama se encarga de agrupar como corresponde todos los datos, pudiendo añadir algunos más, llamados datos adicionales, como por ejemplo CRC o información del usuario.

Capa 1. También llamada "pre-MUSICAM" utiliza el algoritmo PASC, desarrollado por PHILIPS para su casete de audio digital (DCC). Utiliza una velocidad fija entre las 14 posibles (de 32 a 448 Kbits/s); la calidad Hi-Fi necesita 192 Kbits/s por canal de audio (384 Kbits/s en estéreo). Su principal ventaja es la relativa sencillez para implementar el codificador y el decodificador.

- El mapeado tiempo-frecuencia se realiza con un banco de filtros polifase con 32 subbandas. Los filtros polifase consisten en un conjunto de filtros con el mismo

ancho de banda con interrelaciones de fase especiales que ofrecen una implementación eficiente del filtro subbanda. Se denomina filtro subbanda al que cubre todo el rango de frecuencias deseado. En general, los filtros polifase combinan una baja complejidad de computación con un diseño flexible y múltiples opciones de implementación.

- El modelo psicoacústico utiliza una FFT (Fast Fourier Transform) de 512 puntos para obtener información espectral detallada de la señal. El resultado de la aplicación de la FFT se utiliza para determinar los enmascaramientos en la señal, cada uno de los cuales produce un nivel de enmascaramiento, según la frecuencia, intensidad y tono. Para cada subbanda, los niveles individuales se combinan y forman uno global, que se compara con el máximo nivel de señal en la banda, produciendo el SMR que se introduce en el cuantizador.
- El bloque de cuantización y codificación examina las muestras de cada subbanda, encuentra el máximo valor absoluto y lo cuantiza con 6 bits. Este valor es el factor de escala de la subbanda. A continuación se determina la asignación de bits para cada subbanda minimizando el NMR (noise-to-mask ratio) total. Es posible que algunas subbandas con un gran enmascaramiento terminen con cero bits, es decir, no se codificará ninguna muestra. Por último las muestras de subbanda se cuantizan linealmente según el número de bits asignados a dicha subbanda concreta.

El trabajo del empaquetador de trama es sencillo. La trama, según la definición ISO, es la menor parte del bitstream decodificable por sí misma. Cada trama empieza con una cabecera para sincronización y diferenciación, así como 16 bits opcionales de CRC para detección y corrección de errores. Se emplean, para cada subbanda, 4 bits para describir la asignación de bits y otros 6 para el factor de escala. El resto de bits en la trama se utilizan para la información de las muestras.

Capa 2. Su algoritmo se conoce bajo el nombre de MUSICAM, es el estándar adoptado para la radio (DAB) y televisión (DVB) digitales europeas. Permite obtener una calidad equivalente con un flujo menor (reducción del 30% al 50%) que el de la capa 1, a costa de un incremento moderado de la complejidad tanto del codificador como del decodificador.

El flujo, constante, puede escogerse entre 32 y 192 Kbits/s por canal, la calidad subjetiva Hi-Fi se obtiene a partir de 128 Kbits/s por canal, es decir, 256 Kbits/s en estéreo.

- El mapeado de tiempo-frecuencia es idéntico a la de la capa I.
- El modelo psicoacústico es similar, salvo que utiliza una FFT de 1024 puntos para obtener mayor resolución espectral. En los demás aspectos, es idéntico.
- El bloque de cuantización y codificación también es similar, generando factores de escala de 6 bits para cada subbanda. Sin embargo, las tramas de la capa II son tres veces más largas que las de la capa I, de forma que se concede a cada subbanda tres factores de escala, y el codificador utiliza uno, dos o los tres, según la diferencia que haya entre ellos. La asignación de bits es similar a la de la capa I.
- El creador del formato de trama: la definición ISO de trama es la misma que en el punto anterior. Utiliza la misma cabecera y

estructura de CRC que la capa I. El número de bits que utilizan para describir la asignación de bits varía con las subbandas: 4 bits para las inferiores, 3 para las medias y dos para las superiores, adecuándose a las bandas críticas. Los factores de escala se codifican junto a un número de dos bits que indica si se utilizan uno, dos o los tres. Las muestras de subbanda se cuantizan y a continuación se asocian en grupos de tres, llamados gránulos. Cada uno se codifica con una palabra clave, lo que permite interceptar mucha más información redundante que en la capa I. Cada trama contiene, pues, 1152 muestras PCM. A 48 kHz. Cada trama lleva 24 ms de sonido.

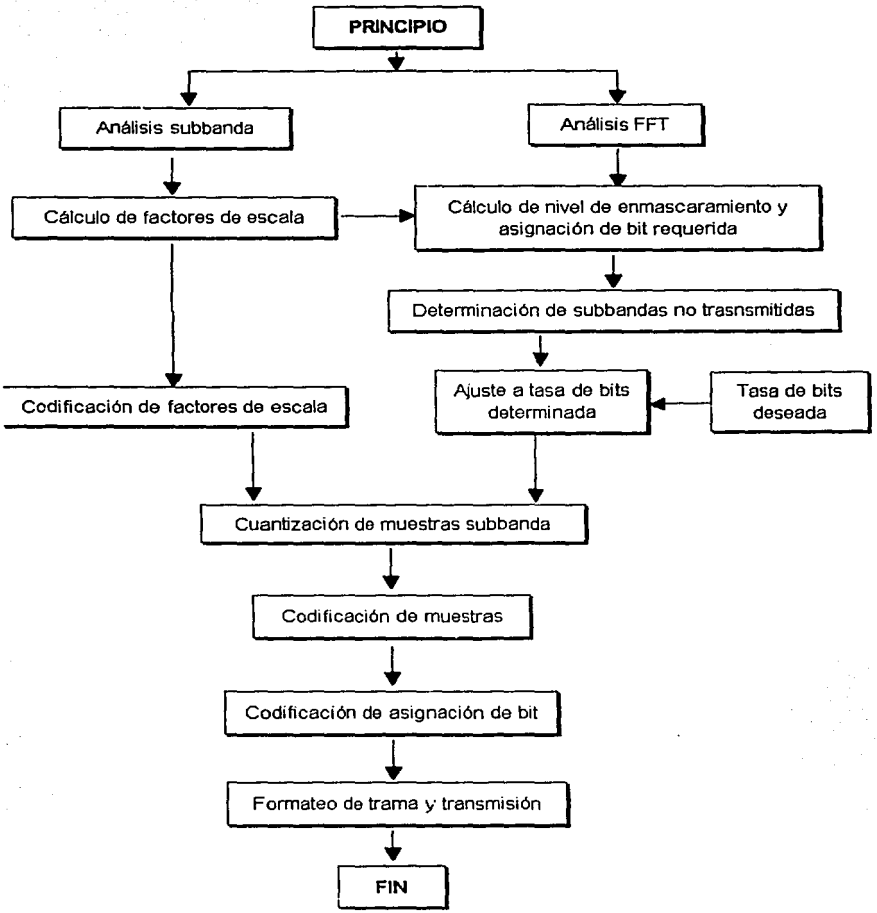


Figura 3.7 Diagrama de flujos del codificador para capa-1 y capa-2 según ISO 11172-3

Capa 3. Es de desarrollo más reciente y utiliza un modelo psicoacústico diferente (llamado modelo 2), una codificación Huffman y un análisis de la señal basado en la DCT en vez de en la codificación en subbandas de las capas 2 y 3. Están permitidos los dos tipos de codificación Joint_stereo.

Permite un flujo variable y una tasa de compresión aproximadamente dos veces más elevada que la capa 2, a costa de una complejidad claramente mayor del codificador y del decodificador, así como de un tiempo de codificación/decodificación más largo. La calidad Hi-Fi se obtiene a partir de los 64 Kbits/s por canal (128 Kbits/s en estéreo). Está destinada principalmente a aplicaciones de redes de baja velocidad (por ejemplo INTERNET).

- La capa III es sustancialmente más complicada que los dos anteriores e incluye una serie de mejoras cuyo análisis resultaría desbordante, de manera que no entraremos en tantos detalles. Su diagrama de flujos es conceptualmente semejante al visto para las otras dos capas, salvo que se realizan múltiples iteraciones para procesar los datos con el mayor nivel de calidad en un cierto tiempo, lo cual complica su diseño hasta el punto de que los diagramas ISO ocupan decenas de páginas.
- El mapeado de tiempo-frecuencia añade un nuevo banco de filtros, el DCT (Discrete Cosine Transform), que con el polifase forman el denominado filtro híbrido. Proporciona una resolución en frecuencia variable, 6x32 o 18x32 subbandas, ajustándose mucho mejor a las bandas críticas de las diferentes frecuencias.
- El modelo psicoacústico es una modificación del empleado en la capa II, y utiliza un método denominado predicción polinómica. Incluye los efectos del enmascaramiento temporal.
- El bloque de cuantización y codificación también emplea algoritmos muy sofisticados que permiten tramas de longitud variable. La gran diferencia con las otros dos capas es que la variable controlada es el ruido, a través de bucles iterativos que lo reducen al mínimo posible en cada paso.
- El creador de formato de trama: la definición de trama para este capa según ISO varía respecto de la de los niveles anteriores: "mínima parte del bitstream decodificable mediante el uso de información principal adquirida previamente". Las tramas contienen información de 1152 muestras y empiezan con la misma cabecera de sincronización y diferenciación, pero la información perteneciente a una misma trama no se encuentra generalmente entre dos cabeceras. La longitud de la trama puede variarse en caso de necesidad. Además de tratar con esta información, la capa III incluye codificación Huffman de longitud variable, un método de codificación entrópica que sin pérdida de información elimina redundancia. Los métodos de longitud variable se caracterizan, en general, por asignar palabras cortas a los eventos más frecuentes, dejando las largas para los menos frecuentes.

Los diagramas de bloques de un codificador y un decodificador MPEG de audio.

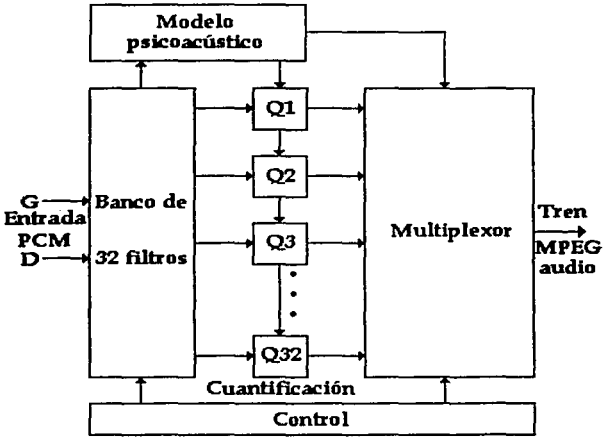


Figura 3.8 Diagrama de bloques de un codificador MPEG de audio.

3.2.1.1 La decodificación:

El decodificador debe procesar el bitstream para reconstruir la señal de audio digital. La especificación de este elemento sí esta totalmente definida y debe seguirse en todos sus puntos. La figura ilustra el esquema del decodificador.

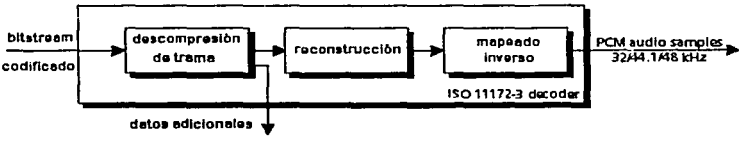


Figura 3.9 Decodificador según la norma ISO 11172-3

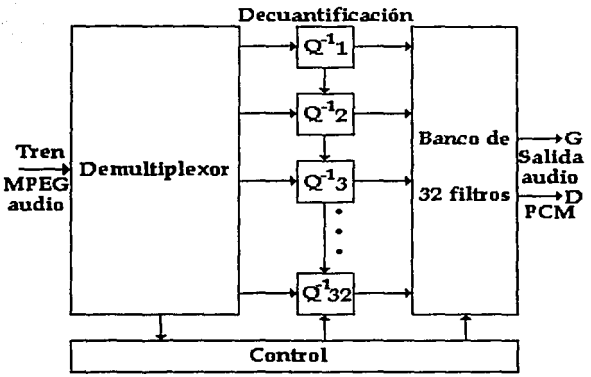


Figura 3.10 Decodificador.

Los datos del bitstream son desempaquetados para recuperar las diversas partes de la información. El bloque de reconstrucción recompone la versión cuantizada de la serie de muestras mapeadas. El mapeador inverso transforma estas muestras de nuevo a PCM.

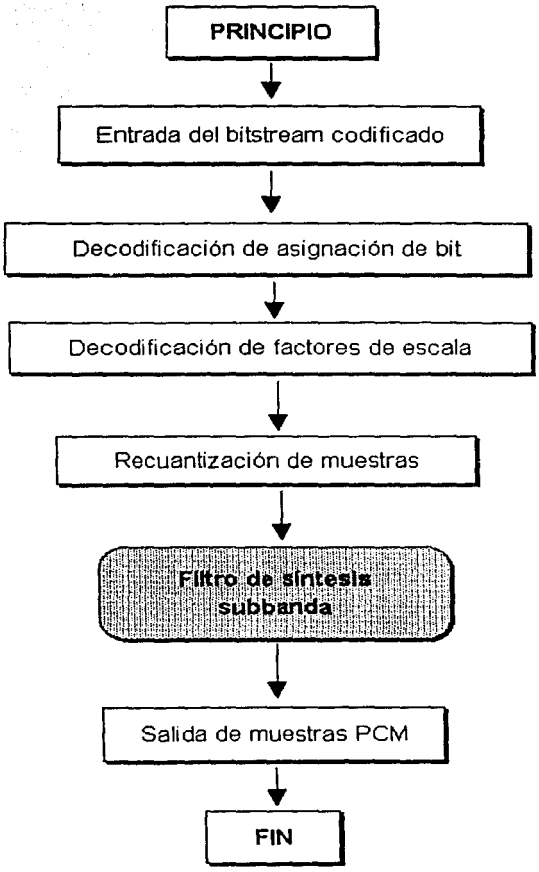


Figura 3.11 Diagrama de flujos del decodificador para capa-1 y capa-2 según ISO 11172-3

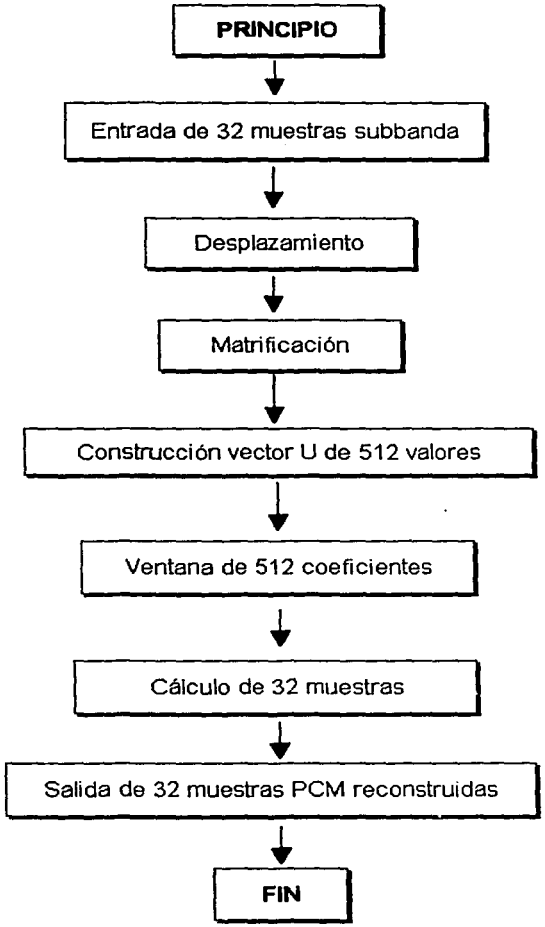


Figura 3.12 Detalle del filtro de la figura 3.11

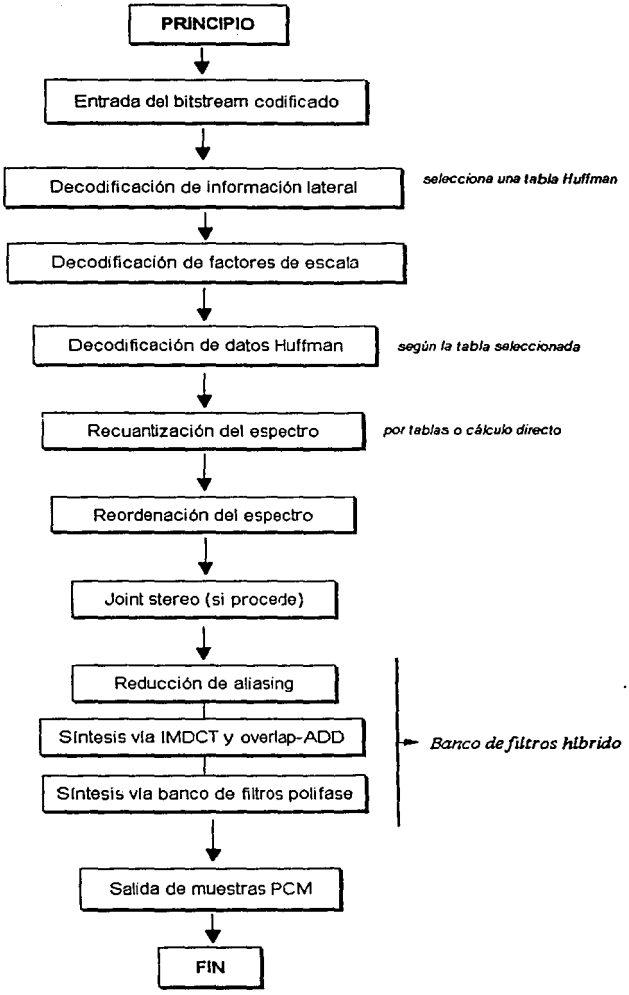


Figura 3.13 Diagrama de flujos del decodificador para capa-3 según ISO 11172-3.

3.2.1.2 Formato de la Trama MPEG de Audio

La trama constituye la unidad de acceso elemental para una frecuencia de audio MPEG. Una trama (capa 1, 2 o 3), se descompone en 4 partes:

- Cabecera de 32 bits (header)
- Paridad sobre 16 bits (CRC)
- Datos de audio (AUDIO), longitud variable
- Datos auxiliares (AD, ancillary data)

Capa 1. La trama MPEG de audio de la capa 1 se compone de 384 muestras PCM de audio de entrada. Cuando el número de muestreos PCM es independiente de la frecuencia de muestreo, la duración de la trama es inversamente proporcional a la frecuencia de muestreo. Esta es de:

- 12 ms a 32 kHz
- 8.7 ms a 44.1 kHz
- 8 ms a 48 kHz

En la Figura 3.14 se puede ver una estructura de una trama de la capa 1.

Cabecera	CRC	Audio			AD
Cabecera 32 bits	Paridad 16 bits	Asignación bits/MT-SB	Factores de escala	Muestreos de Sub-banda (MT-SB) (12 x 32 muestreos de Sub-banda)	Datos auxiliares

Figura 3.14 Representación de la estructura de una trama MPEG de audio, capa 1.

Observaciones:

- La cabecera transporta la sincronización y la información de sistema, detalladas en el cuadro 3.1.
- La utilización de paridad (CRC) es optativa.
- El campo de asignación de los bits/MT-SB (32 enteros codificados sobre 4 bits) define la resolución de codificación (de 0 a 15 bits) de los muestreos de cada una de las 32 subbandas.
- El campo factor de escala (32 enteros codificados sobre 6 bits) indica para cada subbanda el factor multiplicador de los muestreos de esta forma cuantificados.

Capa 2. La trama se compone en este caso de 12 gránulos de 3 x 32 = 96 muestreos de audio PCM, es decir:

- 36 ms a 32 kHz
- 26.1 ms a 44.1 kHz
- 24 ms a 48 kHz

La estructura de la parte de audio difiere de la capa 1 debido a una asignación de bits más compleja, motivada por la mayor cantidad de opciones de codificación. En la Figura 3.15 se puede observar una estructura de una trama de la capa 2.

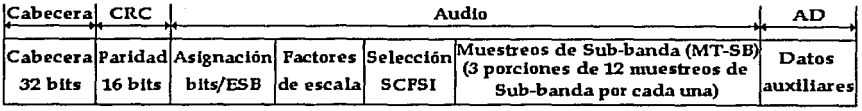


Figura 3.15 Representación de la estructura de una trama MPEG de audio, capa 2

Observaciones:

- La cabecera transporta la sincronización y la información del sistema. (Ver cuadro 13).
- La utilización de la paridad (CRC) es optativa.
- El campo asignación de los bits/MT-SB (32 enteros codificados sobre 2 o 4 bits, según la subbanda) define la resolución de codificación de los muestreos de cada una de las subbandas y si están o no agrupadas en 3.
- El campo SCFSI (Scale Factor Selection Information) (32 enteros codificados sobre 2 bits) indica si el factor de escala de subbanda se aplica a toda la trama o si hay 2 o 3 factores de escala.
- El campo factor de escala indica el factor multiplicador de los muestreos de esta forma cuantificados para la porción de trama definida por SCFSI.

Campo	Comentario	N°de bits
Syncword	tren 1111 1111 1111 (FFF hex)	12
ID	siempre a "1" para MPEG-1 de audio	1
layer	11=1, 10=11, 01=111, 00 reservado (capa)	2
protection_bit	0 si se añade redundancia, 1 si no	1
bitrate_index	15 valores (0000=flujo libre, 1111=prohibido)	4
Sampling_frequency	00=44.1 kHz, 01=48, 10=32, 11=reservado	2
padding_bit	1=ajuste (necesario para Fmuestreo=44.1 kHz)	1
private_bit	no especificado, uso libre	1
mode	00=stereo, 01=joint, 10=dual, 11=mono	2
mode_extension	margen de las subbandas en intensity_stereo	2
copyright	1=copyright, 0=libre	1
original/copy	1=original, 0=copia	1
Emphasis	00=no, 01=50/75□ s, 10=reservado, 11=J17	2

Tabla 3.1 Campos de la cabecera de una trama MPEG-1 de audio

3.2.2 MPEG-4 (Audio estructurado)[7]

Como se menciona anteriormente, la compresión se logra al eliminar la redundancia en las señales de audio. Entre más redundancias se conozcan y exploten, una mayor compresión tendremos.

Los sistemas de audio estructurado comprimen el sonido explotando lo que se llama redundancia estructural y añadiéndole un sistema basado en redundancia de proceso.

La redundancia estructural es el resultado de la forma como es creado el sonido por el hombre. Los mismos sonidos, o sonidos que son muy similares, ocurren una y otra vez. Por ejemplo, un concierto de piano consiste en muchas notas. Cada vez que se toca una tecla, un sonido muy similar es creado por el mecanismo del piano. En una primera aproximación se puede tocar que cada vez que se toca la tecla el sonido es el mismo, en una aproximación más profunda se podría ver como el mismo sonido con diferencia sólo en la amplitud, dependiendo de la fuerza con la que se toco la tecla; en otra aproximación todavía más profunda notaríamos que el sonido cambia por muchos factores. En un sistema PCM en la representación del concierto, cada nota tendría un valor completamente independiente y una secuencia de datos diferente. Esto también es válido para un sistema perceptual.

En un sistema estructurado, se puede explotar y remover la redundancia estructural del sonido asumiendo que cada ocurrencia de una nota es la misma, excepto por una diferencia descrita por un algoritmo de unos cuantos parámetros. En la etapa del modelo de transmisión (o almacenamiento) sólo entra el sonido básico (una muestra de la nota u otro algoritmo) y el algoritmo que describe la diferencia. Luego, la transmisión del sonido, sólo se codifica la nota deseada, la ocurrencia en el tiempo y los parámetros de control del algoritmo diferenciador.

La codificación estructurada de audio difiere de los sistemas tradicionales en que el modelo del sonido no está fijo en el protocolo, sino dinámicamente descrito como una parte de la cadena de transmisión, donde puede cambiar de señal a señal.

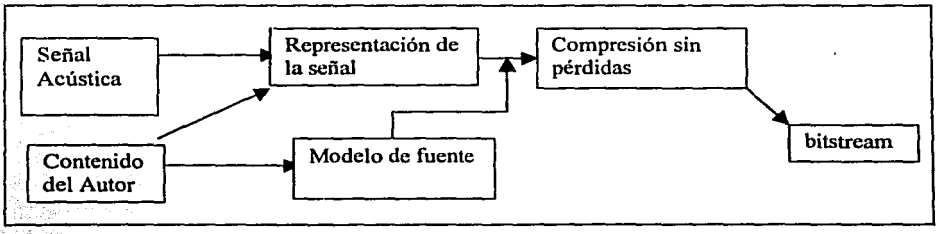


Figura 3.16 Diagrama del sistema de compresión de MPEG 4.

[7] Para más información revizar la norma **ISO/IEC JTC1/SC29/WG11**

3.3 ATRAC

Para poder proporcionar la misma cantidad de música, 74 min, en un mini-disc de 2.5 pulgadas que en un CD estándar de 4.27 pulgadas, se ha desarrollado una nueva tecnología de audio digital de compresión llamada ATRAC (Adaptive Transform Acoustic Coding). Esta tecnología comprime la información hasta un quinto posibilitando la grabación de 74 min y dejando espacio restante para la grabación de títulos y demás.

En una codificación lineal de 16 bits, que se utiliza actualmente en los formatos CD y Dat, con una frecuencia de muestreo de 44.1 kHz, la señal analógica se muestrea aproximadamente una vez cada 0.02 milisegundos. Cada muestra se cuantifica a una resolución de 16 bits. Por tanto, con el CD o Dat cuando la señal analógica se convierte en un dato digital en tiempo real, se utilizan 16 bits de datos cada 0.02 milisegundos sin contar con la amplitud de la señal y ni siquiera con la presencia de la señal. El ATRAC se inicia con los mismos datos digitales de 16 bits, aunque analiza segmentos de datos en cuanto a su contenido en forma de onda. Basándose en este análisis, el ATRAC extrae y codifica sólo aquellos componentes de frecuencia que resultan en realidad audibles para el oído humano. Este método de codificación es mucho más eficaz que la técnica de codificación lineal utilizada para los CD y Dat, aunque la calidad de sonido es comparable. Durante la secuencia de acontecimientos se utilizan los siguientes principios psicoacústicos fundamentales:

- Umbral de audibilidad: al disminuir el nivel de sonido, existe un nivel por debajo del cual el oído humano no puede detectar su presencia. Este umbral varía con la frecuencia. El umbral de audibilidad es mínimo para sonidos con una frecuencia de aproximadamente 4 kHz; es decir, los sonidos que se acercan a esta frecuencia son más difíciles de detectar por el oído humano. Analizando los componentes de la frecuencia de una señal de audio, es posible identificar aquellos componentes que permanecen por debajo del umbral de audibilidad. Estos componentes se pueden extraer de la señal original sin afectar (teóricamente) a la calidad del sonido percibido.
- Enmascaramiento auditivo: si se producen dos sonidos, uno alto y el otro suave, simultáneamente y quedan muy cerca uno de otro en frecuencia, resulta difícil o imposible escuchar el sonido suave (efecto enmascaramiento). Por consiguiente, cuando una señal de audio tiene un componente de alto nivel y otro de bajo nivel en frecuencias cercanas, este último se puede eliminar sin afectar (teóricamente) a la calidad del sonido percibido. Es más, con una amplitud de señal general creciente, resulta posible eliminar un gran número de componentes sin efectos audibles (teóricamente).
- Este sistema es muy parecido al MPEG1 capa 3 ya que es un modelo perceptual. En este modelo la señal es dividida en 4 bandas por un filtro de espejo en cuadratura. Luego se aplica un control de ganancia a cada banda y el resultado se pasa al dominio de la frecuencia con una transformada de coseno modificada. Luego se separa este en componentes tonales y no tonales y se genera la cadena de bits.
- Una memoria a prueba de vibraciones: los sistemas convencionales de lectura óptica pueden perder la pista con facilidad cuando reciben golpes o vibraciones. En los CD audio digitales, esto provoca saltos o interrupciones y, típicamente los fabricantes de

CD procuran minimizar estos incidentes utilizando suspensiones mecánicas y servosistemas. El sistema mini-disc ha resuelto también este problema. Mientras el lector de MD puede leer información a una media de 1.4 Mbit por segundo, el decodificador ATRAC requiere una media de datos de tan sólo 0.3 Mbit por segundo para la reproducción en tiempo real. Esta diferencia en la velocidad de procesamiento hace posible el uso de un buffer de lectura avanzada, situado entre el lector y el decodificador. Si se utiliza un chip de memoria de 1 Mbit para el buffer, éste puede almacenar hasta 3 segundos de información digital. En caso de que el lector se desvíe de su posición, la memoria del buffer continúa suministrando la información correcta al decodificador ATRAC.

Como las señales entran en la memoria del buffer más rápido de lo que tardan en salir éste termina llenándose. En ese momento, el lector del MD deja de leer momentáneamente para reemprender la lectura cuando el chip tenga la memoria vacía. Utilizando un concepto llamado reposicionamiento de sector, el lector del MD tiene la capacidad de reanudar la lectura desde el punto correcto después de su desplazamiento. Cuando se graban señales en un MD la información direccionada se señala cada 1.3 milisegundo, y cuando el lector se sale de sitio el reproductor reconoce la interrupción, identifica la dirección equivocada, e inmediatamente devuelve el lector a su posición correcta. Este mecanismo es el que utilizan algunos reproductores de CD actuales.

4 Procesador Digital de Señales

4.1 Introducción

La plataforma de dispositivos TMS320C6000 ('C6000) consiste de un grupo de procesadores para señales digitales (DSPs) que usan un avanzado sistema de palabras largas de instrucción (VLIW), para lograr un alto performance, gracias al aumento de paralelismo a nivel de instrucción. El adelantado sistema VelociTI para palabras largas de instrucción (VLIW), utiliza una arquitectura con distintas unidades de ejecución múltiple, que operan en paralelo para ejecutar múltiples instrucciones, durante un solo ciclo del reloj.

El paralelismo es la llave para alcanzar el alto nivel de performance, ya que se incrementa la capacidad de procesamiento dentro de los nuevos DSPs, superando a los DSPs tradicionales.

4.1.1 Vista Global de la Familia TMS320

La familia TMS320 consiste de procesadores para señales digitales (DSPs), que trabajan con formato de punto fijo, punto flotante y multiprocesamiento. Los DSPs TMS320 se diseñan específicamente para el procesamiento de señales en tiempo real.

4.1.2 Historia de los DSPs TMS320

En 1982, Texas Instruments presentó los primeros DSPs de punto fijo, con la familia TMS320 y en particular con el DSP TMS32010. Antes de finalizar ese año, la revista Productos Electrónicos otorgó el título a los TMS32010 del Producto del Año.

Hoy la familia TMS320 consiste de las siguientes generaciones: 'C1x, 'C2x, 'C27x, 'C5x y 'C54x, 'C55x, DSPs de punto fijo; 'C3x y 'C4x, DSPs de punto flotante; y 'C8x DSPs de multiprocesamiento. Hay una nueva generación de DSPs ahora, la plataforma TMS320C6000, con performance y características que los hacen muy flexibles. El compromiso de Texas Instruments es tener para todo el mundo las mejores soluciones en DSP.

4.1.3 Vista Global de la Plataforma de los DSPs TMS320C6000

Con una velocidad de 2000 millones de instrucciones por segundo (MIPS) y un compilador de C eficaz, los DSPs TMS320C6000 dan una ilimitada cantidad de posibilidades a la arquitectura del sistema, lo que la hace diferente de otros productos similares. Gran performance, facilidad de uso y accesibilidad económica hacen de la plataforma TMS320C6000 la solución ideal para aplicaciones de multicanales y multifunciones, como las siguientes:

- Poleo de Módems.
- Ciclos internos en estaciones inalámbricas.
- Servidores de acceso remoto (RAS).
- Sistemas digitales de ciclos de suscripción (DSL).
- Módems de cable.
- Sistemas de telefonía multicanal.

La plataforma TMS320C6000 también es la solución ideal para implementarse en nuevas aplicaciones, por ejemplo:

- Seguridad personalizada en la casa con reconocimiento del rostro y las huellas dactilares.
- Control adelantado de cruceros en navegación (GPS) y anulación de accidentes.
- Diagnóstico médico remoto.
- Estaciones en forma de haz.
- Graficación en 3-D para Realidad Virtual.
- Reconocimiento de palabras.
- Audio.
- Radar.
- Modelado atmosférico
- Análisis de elementos finitos.
- Imágenes (por ejemplo, reconocimiento de huellas digitales, ultrasonido y MRI).

4.1.4 Opciones y Características de los Dispositivos TMS320C6000

Los dispositivos 'C6000 ejecutan ocho instrucciones de 32 bits por ciclo. El dispositivo central de la CPU consiste de 32 registros de propósito general con 32 bits de longitud y ocho unidades funcionales:

- Dos multiplicadores.
- Seis unidades lógicas aritméticas (ALUs).
- La generación 'C6000 tiene un juego completo de herramientas de desarrollo perfeccionadas, incluso un compilador de C eficaz, un optimizador de lenguaje ensamblador para la simplificación del ensamblado, programación y cargado del lenguaje, y un Debugger Windows para obtener una excelente visibilidad del código fuente durante la ejecución característica. En resumen, las características de los dispositivos 'C6000 incluyen:
- CPU VLIW avanzado con ocho unidades funcionales, incluyendo dos multiplicadores y seis unidades aritmética.
- Ejecuta ocho instrucciones por ciclo, lo que es diez veces el performance de otros DSPs.
- Permite al diseñador desarrollar, de forma eficaz, código RISC ligero para lograr un rápido desarrollo.
- Paquete de instrucciones
- Da la equivalencia del tamaño de código para las ocho instrucciones que son ejecutadas consecutivamente; o sea, en paralelo.
- Reduce el tamaño del código, minimiza los ciclos fetch del programa y disminuye el consumo de energía.
- Ejecución condicional de todas las instrucciones.
- Reduce bifurcación costosa.
- Aumenta el paralelismo para lograr un performance más alto.
- Ejecución eficaz del código en unidades funcionales independientes.
- Mayor eficiencia del compilador en C con la colección de referencias del DSP.

- Primer optimizador de lenguaje ensamblador para un desarrollo más rápido y una mejora del paralelismo.
- 8 / 16 / 32 bits de datos apoyan y mantienen apoyo de eficaz de memoria en una gran variedad de aplicaciones 40 bit en las opciones aritméticas le agregan precisión extra a los codificadores de voz y a otras aplicaciones intensivas computables.
- Soporte de las funciones aritméticas para evitar saturación y normalización.
- Manipulación de campos y extracción de instrucciones, inicialización, limpieza y conteo de bits, en apoyo a las funciones comunes encontradas durante el control y a la manipulación de los datos en las aplicaciones.
- Hardware apoyado en las normas de IEEE para instrucciones de precisión doble y simple (sólo en el caso del DSP 'C6701).
- Conexión compatible con DSPs de punto fijo y punto flotante DSPs.

4.1.5 Vista Global de la Memoria del TMS320C6000

La configuración de memoria interna varía entre los diferentes dispositivos 'C6000. Todos los dispositivos incluyen:

- Memoria interna para datos y para programas.
- Periféricos internos.

La memoria externa se accede a través de la interface de memoria externa (EMIF) TMS320C6201/C6202/C6701: los 'C6201, 'C6202 y 'C6701 tienen los datos separados de los programas. La memoria interna del programa puede trazarse en el espacio de direcciones de la CPU u operarse como memoria cache. Un camino de 256 bits se proporciona desde la CPU para permitir un flujo continuo de ocho instrucciones de 32 bits cada una, hasta alcanzar el máximo performance.

La memoria de datos se accede a través del controlador de memoria de datos, que realiza las funciones siguientes:

- La CPU y el acceso directo a memoria (DMA) controlan el acceso a la memoria interna de datos y realizan el arbitraje necesario.
- Los datos de la CPU acceden al EMIF.
- La CPU accede a través de los periféricos internos.

La memoria interna de datos esta dividida en bancos de 16 bits de ancho. El controlador de memoria de datos realiza el arbitraje entre la CPU y el control DMA independientemente para cada banco, permitiendo a ambos, CPU y DMA, el acceso a localidades diferentes de memoria de forma simultanea, para evitar conflictos. El controlador de memoria de datos permite la configuración endianness. La conexión LENDIAN activa la selección de la configuración endianness en el dispositivo.

TMS320C6211/C6711: La arquitectura 'C6211/C6711 está basada en memoria cache, con un primer nivel donde se separa el programa de los datos. Éstos espacios de cache no están incluidos en el mapa de memoria y siempre están habilitados. El nivel uno de memoria cache es accesible sólo desde la CPU.

El controlador del nivel uno de memoria cache para programa (L1P) une la CPU y L1P. Un camino de 256 bits de ancho se proporciona desde la CPU para permitir el flujo continuo de las 8 instrucciones de 32 bits en el máximo performance. El controlador de la memoria cache de datos (L1D) proporciona la interface entre CPU y L1D. La memoria L1D permite accesos simultáneos a ambos lados de la CPU. En el siguiente nivel de

memoria, una vez franqueada la memoria cache de datos (L1D) y de programas (L1P), el controlador de los requerimientos de memoria se denomina L2. El controlador L2 facilita:

- La CPU y el acceso directo a memoria reforzada (EDMA) controlan los accesos a la memoria interna y el arbitraje necesario para el performance.
- El acceso de datos a la CPU se hace con el EMIF.
- La CPU accede a los periféricos internos.
- Envía demandas a EMIF para la emisión de datos L2.

La memoria SRAM interna de los dispositivos 'C6211/C6711 unifica programas con datos en el espacio de memoria. El espacio de memoria L2 puede configurarse como una sólo mapa de memoria para SRAM y para toda la memoria cache o como una combinación de ambos.

4.1.6 Vista Global de los Periféricos del TMS320C6000

Los periféricos accesibles al usuario se configuran a través de un conjunto de registros de control del mapa de memoria registros del mando. El controlador del bus de periféricos realiza el control para el acceso a los periféricos dentro del chip. La configuración lógica del Boot está conectada a través de señales externas solamente y la lógica negada sólo es accesible directamente desde la CPU. Los siguientes son algunos de los periféricos:

- Controlador DMA: El controlador DMA transfiere los datos entre rangos de direcciones en el mapa de memoria sin la intervención de la CPU. El controlador DMA tiene cuatro canales programables y un quinto canal auxiliar.
- Controlador EDMA: El controlador EDMA realiza las mismas funciones que el Controlador DMA. El EDMA tiene dieciséis canales programables, así como un espacio en memoria RAM para retener configuraciones múltiples en las transferencias futuras.

- HPI: El HPI es un puerto paralelo por el cual el procesador puede tener acceso directo al espacio de memoria de la CPU. El host facilita el acceso debido a que manipula directamente la interface. El host y la CPU pueden intercambiar información a través de memoria interna o externa. Además, el host tiene acceso directo al mapa trazado para los periféricos.
- Bús de expansión: El bús de expansión es un reemplazo para HPI, así como una expansión de EMIF. La expansión proporciona dos áreas distintas de funcionalidad, (el puerto del host y puertos de I/O) cuando pueden coexistir en un sistema. El puerto del host para la expansión del bus puede operar en cualquier modo esclavo asíncrono, similar al HPI o en modo maestro esclavo síncrono. Esto permite al dispositivo unir a una gran variedad de protocolos de bus para el host. Las estructuras FIFOs síncronas y los dispositivos de I/O periféricos asíncronos pueden unirse al bus de expansión.
- EMIF: El EMIF soporta pegar pequeñas interfaces para varios dispositivos externos e incluye:
  - Ráfagas síncronas SRAM (SBSRAM).
  - DRAM síncrono (SDRAM).
  - Dispositivos asíncronos, incluyendo SRAM, ROM y FIFOs.
  - Un dispositivo externo de memoria compartida.

Configuración del Boot: Los TMS320C62x y TMS320C67x proporcionan una variedad de configuraciones del boot que determinan las acciones que el DSP realiza después del reset para preparar para la iniciación. Esto incluye cargar el código en un espacio de memoria ROM externa en el EMIF y cargar el código a través del HPI, bus de expansión, para el host externo.

McBSP: El puerto serie multicanal con buffers (McBSP) está basado en la interface estándar de puerto serial presente en las plataformas de los dispositivos TMS320C2000 y 'C5000. Además, el puerto puede cargar automáticamente las muestras seriales de los buffers a la memoria con la ayuda del controlador DMA/EDMA. También tiene la capacidad de multicanal compatible con las normas de redes de computadoras T1, E1, SCSA y MVIP. Como sus antecesores, proporciona:

- Comunicación full-duplex.
- Dos registros buffers que permiten un flujo continuo de datos.
- Tramas independientes y reloj para recepción y transmisión de datos.
- Interface directa a codecs según las normas industriales, interface análoga dentro del chip (AICs) y otras conexiones seriales para los dispositivos A/D y D/A.

Además, el McBSP tiene las siguientes capacidades:

- Interface directa a:
  - Tramas T1/E1.
  - Dispositivos dóciles ST-BUS.
  - Dispositivos dóciles IOM-2.
  - Dispositivos dóciles AC97.
  - Dispositivos dóciles IIS.
  - Dispositivos SPI.
  - Transmisión de Multicanal y recepción de 128 canales.
- Una amplia selección del tamaño de datos, que incluyen 8, 12, 16, 20, 24 y 32 bits.
- Ley-μ y ley-A de compansión.
- 8bits datos para transferir con primero LSB o MSB.
- Polaridad programable para tramas de sincronización y relojes de datos.
- Reloj interno altamente programable y generación de tramas.
- Cronómetro: Los dispositivos 'C6000 tienen dos cronómetros de 32 bits y de propósito general que son usado para:
  - Medir el tiempo de los eventos.
  - Contar eventos.
  - Generar pulsos.
  - Interrumpir la CPU.
  - Enviar sincronización de eventos al controlador DMA/EDMA.

Selector de Interrupciones: El conjunto de periféricos 'C6000 produce de 14 a 16 fuentes de interrupción. La CPU tiene 12 interrupciones disponibles. El selector de interrupciones permite escoger entre las 12 interrupciones, según las necesidades del sistema. El selector

de interrupciones también permite cambiar la polaridad de las entradas de interrupción externa.

Bajo consumo de energía: La lógica de bajo consumo de energía permite reducir tiempos para limitar el consumo de energía. La mayor parte de la energía del dispositivo se disipa operando lógica CMOS mientras el circuito cambia de un estado lógico a otro. Para prevenir algunos o todos los cambios lógicos del chip, se pueden realizar importantes ahorros de energía manejando diversos datos fuera del contexto operacional. Arquitectura del TMS320C6000.

Los DSP's 'C62x utilizan punto fijo para el procesamiento de las señales digitales y representan el primer DSP que usa arquitectura VelociTI. VelociTI es alto rendimiento y arquitectura avanzada para palabras largas de instrucciones (VLIW), lo que viene a ser una opción excelente para multicanales, multifunciones y manejo del performance de las aplicaciones. Los 'C67x son DSP's de punto flotante con los mismos rasgos generales. Es el segundo DSP que usa arquitectura VelociTI. Los DSP's 'C64x son de punto fijo con los mismos rasgos. Es el tercer DSP que usa arquitectura VelociTI. Los DSPs 'C6000 están basados en las CPU's 'C6000 que tienen las siguientes características:

- Unidad fetch de programa.
- Unidad distribuidora de instrucción.
- Unidad decodificadora de instrucción.
- Dos caminos de datos, cada uno con cuatro unidades funcionales.
- Treinta y dos registros de 32-bits ('C62x y 'C67x).
- Sesenta y cuatro registros de 32-bits ('C64x).
- Control de registros.
- Control de lógica.
- Prueba, emulación e interrupción lógica.

Dentro de la documentación tiene un ejemplo de Audio mostrado en el apéndice destinado al procesador digital de señales.

4.2 Funciones

A continuación citaremos algunas funciones que sirven para el procesamiento.

4.2.1 FFT

bitrev_cplx Complex Bit-Reverse

Declaración *void bitrev_cplx (int *x, short *index, int nx)*

Argumentos
x[nh]: Apuntador al vector de entrada compleja de tamaño *nx*
nx: Número de elementos en el vector *x*. *nx* debe ser una potencia de 2
index[]: Arreglo de tamaño ≈ sqrt(nx) creado por la rutina ***bitrev_cplx*** para permitir la rápida implementación del bit reversal

radix2 Complex Forward FFT (radix 2)

Declaración *void radix2(int nx, short x[ ],short w[ ])*

Argumentos
nx: Número de elementos complejos en el vector *x* Debe ser una potencia de 2 tal que 4<=nx<=65536
*x[2*nx]:* Apuntador a la secuencia de entrada y salida. tamaño de elementos 2*nx
w[nx]: Apuntador a el vector de coeficientes FFT de tamaño nx

r4fft Complex Forward FFT (radix 4)

Declaración *void r4fft(int nx, short x[ ],short w[ ])*

Argumentos
nx: Número de elementos complejos en el vector *x* Debe ser una potencia de 4 tal que 4<=nx<=65536
*x[2*nx]:* Apuntador a la secuencia de entrada y salida. tamaño de elementos 2*nx
w[nx]: Apuntador a el vector de coeficientes FFT de tamaño nx

4.2.2 Filtrado y Convolución

fir_cplx Filtro FIR complejo (radix 2)

Declaración *void fircplx(short *x, short *h, short, *r, int nh, int nx)*

Argumentos
*x[2*nx]:* Apuntador a el arreglo de entrada de tamaño 2*nx
*h[2*nh]:* Apuntador a el arreglo de coeficientes de tamaño 2*nh
*r[2*nr]:* Apuntador a el arreglo de salida de tamaño 2*nr
nh: Número de coeficientes en el vector h, debe ser un múltiplo de 2
nx: Número de muestras para calcular

fir_gen Filtro FIR (Propósito general)

Declaración *void fir_gen(short *x, short *h, short, *r, int nh, int nr)*

Argumentos
x[nr + nh - 1]: Apuntador a el arreglo de entrada de tamaño nr + nh - 1
h[nh]: Apuntador a el arreglo de coeficientes de tamaño nh
r[nr]: Apuntador a el arreglo de salida de tamaño nr
nh: Número de coeficientes nh>=5
nx: Número de muestras para calcular nr>=1

fir_r4 Filtro FIR (radix 4)

Declaración *void fir_r4(short *x, short *h, short, *r, int nh, int nr)*

Argumentos

x[nr + nh - 1]: Apuntador a el arreglo de entrada de tamaño nr + nh - 1
h[nh]: Apuntador a el arreglo de coeficientes de tamaño nh
r[nr]: Apuntador a el arreglo de salida de tamaño nr
nh: Número de coeficientes nh>=8, multiplo de 4
nx: Número de muestras para calcular nr>=2, pares

fir_r8 Filtro FIR (radix 8)

Declaración *void fir_r8(short *x, short *h, short, *r, int nh, int nr)*

Argumentos

x[nr + nh - 1]: Apuntador a el arreglo de entrada de tamaño nr + nh – 1
h[nh]: Apuntador a el arreglo de coeficientes de tamaño nh
r[nr]: Apuntador a el arreglo de salida de tamaño nr
nh: Número de coeficientes nh>=8, multiplo de 8
nx: Número de muestras para calcular nr>=2, pares

fir_sym Filtro FIR simétrico (radix 8)

Declaración *void fir_sym(short *x, short *h, short, *r, int nh, int nr, int s)*

Argumentos

x[nr + 2nh]: Apuntador a el arreglo de entrada de tamaño nr + 2nh
*h[2*nh+1]:* Apuntador a el arreglo de coeficientes de tamaño 2nh+1
r[nr]: Apuntador a el arreglo de salida de tamaño nr
nh: Número de coeficientes nh>=8, multiplo de 8
nx: Número de muestras para calcular nr>=2, iguales
s: Número de dígitos insignificantes para truncar

Iir IIR con 5 coeficientes per Biquad

Declaración *void iir(short *r1, short *x, short, *r2, short h2, short h1, int nr)*

Argumentos

r1[nr]: arreglo de salida (Usado)
x[nr+4]: Arreglo de entrada
r2[nr]: Arreglo de salida (almacenado)
h1: Coeficientes del filtro
h2: Coeficientes del filtro
nr: Número de muestras de salida

4.3 Implementación en el Microprocesador

En el Microprocesador empleado se tiene una característica particular, la cual nos hace imposible la implementación y es el convertidor Analógico Digital, el Microprocesador tiene un DAC de 8 kHz y nuestra implementación requiere un convertidor de 44.1 kHz, lo

que nos lleva a plasmarlo en forma de un diagrama de bloques. El cual se presenta a continuación:

Requerimientos

- TMS320C6711 DSK
- PCM3003 Audio Daughter Card
- Algoritmo
- Code Composer Studio

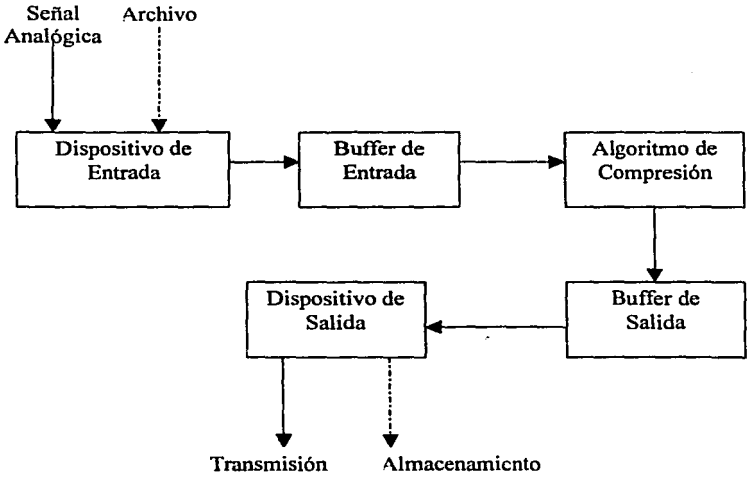


Figura 4.1 Diagrama de Bloques de la implementación

En la implementación el dispositivo de entrada es la tarjeta hija (daughter board) PCM3003 y el buffer de entrada se agrega al algoritmo lo cual no es más que una variable donde se van a estar guardando los valores que tome del buffer del convertidor Analógico Digital, similar a la salidael buffer será otra variable

Además se utiliza el puerto serial de la tarjeta que utiliza el DAC, el cual tiene su registro y con el podemos tomar los datos del buffer siempre y cuando exista una interrupción que nos indique que hay datos en él.

La forma en que se conecta la tarjeta PCM3003 es de la siguiente forma:

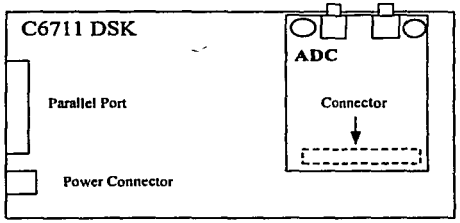


Figura 4.2 Conexión entre DSK y PCM 3003

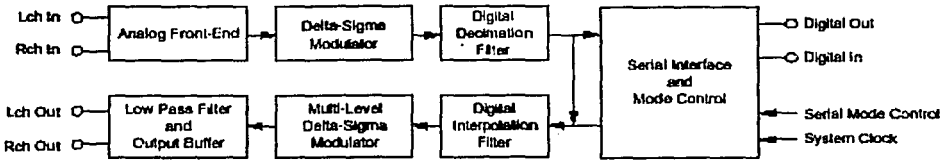


Figura 4.3 Convertidor Analógico Digital de la tarjeta PCM 3003

5 Sistemas de compresión propuestos.

5.1 Sistema 1

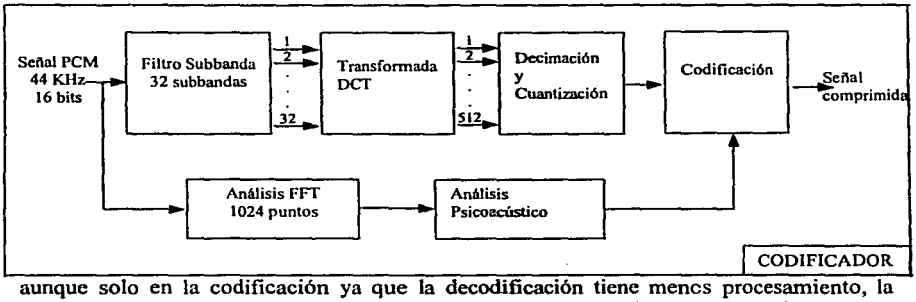
Este sistema es muy parecido al MPEG 1 layer 3 aunque en la parte de cuantización es mucho más sencilla aunque también hay que señalar que se basan en el mismo principio, y precisamente la diferencia en su desempeño también se debe a esa diferencia ya que no usa un sistema tan complicado para la repartición del ruido y tampoco tiene un lazo de control que asegure un tasa de salida de bits.

El sistema no es muy adecuado para un sistema en tiempo real ya que es bastante complejo y el tiempo de procesamiento necesario para completar el proceso es grande, la mayor parte del procesamiento se debe a la transformada FFT y el análisis psicoacústico. Aunque el algoritmo se ha modificado para mejorar su tiempo de ejecución consideramos que aún se puede mejorar, además en un equipo relativamente poderoso es posible que funcione en tiempo real. No obstante, esta desventaja, el algoritmo mantiene una muy buena calidad y eso podría justificar su alto consumo de procesamiento.

En general en este sistema se realizan dos procedimientos simultanemente por un lado se realiza el análisis psicoacustico del cual se obtendrán la minima relación señal a ruido y con esta información cuantizar con la menor cantidad de bits. Por otro lado se realiza la separación en subbandas y después la transformada DCT a cada una de las 32 subbandas, y después son cuantizadas con la resolución necesaria deacuado con el análisis psicoacustico.

5.1.1 Codificador

Como ya se mencionó anteriormente el sistema tiene un procesamiento complejo



aunque solo en la codificación ya que la decodificación tiene menos procesamiento, la razón es que para realizar el análisis psicoacústico es necesario hacer muchos ciclos en los que es inevitable recorrer toda la matriz que representa un bloque, además es forzoso realizar la FFT, proceso de la codificación, por ultimo se ordena la información y se almacena.

Figura 5.1 Diagrama a bloques del codificador del sistema de compresión.

5.1.1.1 Análisis psicoacústico

Para nuestro algoritmo, debido a la dificultad de crear un modelo psicoacústico completo, decidimos usar el modelo I, del estándar 11172-3. El modelo psicoacústico II que se usa en la Capa III tiene mejoras adicionales que se adaptan mejor a las propiedades del oído humano, en comparación con el modelo empleado en las otras dos capas (modelo I). Sin embargo el modelo II tiene una complicación mucho más elevada además de ser poco clara, debido a esto decidimos usar el modelo psicoacústico que, aunque menos eficiente, es más sencillo de implementar. La función del modelo psicoacústico es la de calcular la relación de la señal al enmascarador (SMR) para que con esos datos se realice la repartición de bits.

Todo el proceso de análisis psicoacústico se realiza por medio de pasos que se describen a continuación:

1) Primero el modelo convierte el audio al dominio de la frecuencia por medio de la FFT de 1024 puntos para conseguir una buena resolución de frecuencia y poder calcular correctamente los umbrales de enmascaramiento. Antes de la FFT, se aplica una ventana de Hanning convencional para evitar las discontinuidades en los extremos de la señal.

2) Una vez que la señal se encuentra en el dominio de la frecuencia el siguiente paso es encontrar los máximos locales y separar las componentes tonales de las no tonales.

3) De estos componentes tonales se eliminan los que no tengan la amplitud y separación en barks mínima para ser relevantes.

4) Después se calcula el efecto de enmascaramiento de cada componente sobre las líneas adyacentes.

5) En el siguiente paso se suman las contribuciones de cada componente en cada una de las líneas de frecuencia y se calcula el umbral de enmascaramiento global.

6) Posteriormente se escoge el umbral más bajo en cada una de las 32 subbandas

7) Por ultimo se calcula la relación entre la señal y el enmascarador.

En la siguiente sección se describirá con más detalle cada una de las partes del proceso de análisis psicoacústico

5.1.1.1.1 Análisis FFT

El primer proceso que se debe llevar a cabo es la alineación en el tiempo ya que cuando se realiza la evaluación psicoacústica, los datos de audio que son enviados al modelo deben corresponder al mismo instante de tiempo que los datos que serán cuantificados. El modelo psicoacústico debe tener en cuenta el retardo de los datos al pasar

por el banco de filtros y aplicar un desplazamiento adicional, de tal manera que los datos relevantes queden centrados en la ventana del análisis psicoacústico. El retardo a través del banco de filtros es 256 muestras y el desplazamiento necesario para centrar las 384 muestras, dentro de la FFT de 512 puntos, es:

$$(512 - 384) / 2 = 64 \text{ puntos} \qquad (5.1)$$

El desplazamiento requerido es, entonces, de 320 puntos para alinear los datos del modelo I con la salida del banco de filtros polifásico.

5.1.1.1.1 Representación espectral

El modelo psicoacústico realiza una conversión del tiempo a la frecuencia totalmente independiente del mapeo realizado por el banco de filtros porque necesita una mejor resolución en frecuencia para calcular con gran precisión los umbrales de enmascaramiento. Ambos modelos usan una transformada de Fourier para realizar el mapeo.

En este caso el modelo psicoacústico usa una FFT de 1024 puntos. El análisis psicoacústico se debe realizar para 1152 muestras, así que la FFT de 1024 puntos no proporciona cobertura total. Idealmente, la FFT debería cubrir todas las 1152 muestras; aunque 1024 puntos es un compromiso razonable ya que las muestras que se omiten, no tienen mayor impacto en el análisis psicoacústico.

Para realizar esta transformada en el programa de compresión se uso una implementación realizada por Don Cross, debido a esto no fue necesario el realizar un programa que realizara la transformada FFT

5.1.1.1.2 Identificación de componentes tonales y no tonales

Ambos modelos identifican y separan las componentes tonales y las componentes de ruido en la señal de audio. Esto se debe a que cada componente presenta un tipo de enmascaramiento diferente.

El modelo I identifica las componentes tonales basado en los picos locales del espectro de potencias. Después de procesar todas las componentes tonales, el modelo concentra los valores espectrales restantes en una única componente no-tonal por banda crítica. El índice de frecuencia de cada una de estas componentes no-tonales es el valor más cercano a la media geométrica de la banda crítica a la cual pertenece cada componente no-tonal.

El primer paso es la ubicación de todos los máximos locales en el espectro de potencia y después se identifican cuales son componentes tonales y no tonales a las componentes tonales se les identifica al revisar los niveles que tienen las líneas de frecuencia adyacentes, el número de bandas adyacentes que se deben analizar lo establece el mismo modelo psicoacústico.

Para las componentes tonales, el modelo concentra los valores espectrales restantes en una única componente no-tonal por banda crítica. El índice de frecuencia de cada una de estas componentes no-tonales es el valor más cercano a la media geométrica de la banda crítica a la cual pertenece cada componente no-tonal.

5.1.1.1.3 Reducción

El siguiente proceso es la reducción que también se le conoce como "DECIMATION OF MASKERS" (disminución en la cantidad de componentes enmascarantes), su función es eliminar las componentes que por su nivel de potencia o separación no deben ser analizados.

Primero se analiza el caso no tonal en el cual lo que se revisa es que las componentes no estén por debajo del mínimo umbral de audición si están por debajo de este nivel son eliminadas.

Para el caso tonal se eliminan las componentes que se encuentren a menos de medio Bark de distancia entre ellas y se elimina a la menor potencia conservando a la que tenga un nivel de potencia más alto.

5.1.1.1.4 Umbral de enmascaramiento individual

La primer parte de este proceso es determinar la Función de dispersión que es la capacidad enmascarante de una componente determinada distribuirse por toda la banda crítica que la rodea. Ambos modelos determinan el umbral de enmascaramiento de ruido para ambos tipos de componentes; para lograr esto, el modelo I compara con un enmascaramiento determinado empíricamente, mientras que el modelo II aplica una función de dispersión.

Para poder calcular el umbral de enmascaramiento global (que es el siguiente paso), el modelo I debe calcular primero los umbrales de enmascaramiento que cada componente tonal o no-tonal genera sobre la señal de audio (llamados "UMBRALES DE ENMASCARAMIENTO INDIVIDUALES").

El modelo I calcula el efecto de enmascaramiento que cada componente enmascarante (tonal o no-tonal) tiene sobre las líneas de frecuencia adyacentes a ella. Este análisis sólo es necesario hacerlo para las líneas de frecuencia que se encuentran entre -3 y +8 barks a partir de la componente enmascarante.

En otras palabra, el análisis abarca todas las líneas de frecuencia que se encuentren tres bandas críticas a la izquierda (hacia las bajas frecuencias), y ocho bandas críticas a la derecha (hacia las altas frecuencias) de la componente enmascarante. Esto se debe a que el efecto de enmascaramiento de la componente tonal o no-tonal que está siendo analizada (por más intensidad que ésta tenga) es demasiado tenue por fuera de este rango.

Para ambos el proceso es el mismo, primero se calcula el índice de enmascaramiento que para el caso tonal es:

$$avtm = -1.525-0.275*zj-4.5 \qquad (5.2)$$

y para el caso no tonal es:

$$avnm = -1.525-0.175*zj-0.5 \qquad (5.3)$$

donde zj es la tasa de banda crítica de la componente enmascarante.

después la función de enmascaramiento que para ambos casos es:

$$vf = \begin{cases} -3 \le dz < -1, & vf = 17*(dz+1)-(0.4*F(j)+6) \\ -1 \le dz < 0, & vf = (0.4*F(j)+6)*dz \\ 0 \le dz < 1, & vf = -17*dz \\ 1 \le dz < 8, & vf = -(dz-1)*(17-0.15*F(j))-17 \end{cases} \qquad (5.4)$$

donde $F(j)$ es el valor de la densidad espectral de potencia de la señal de entrada y dz es la distancia en Barks entre el enmascarante y la línea que se analiza.

Y por ultimo se calcula el umbral de enmascaramiento para cada componente y su influencia en cada línea adyacente.

Para el caso tona y no tonal el calculo es el mismo

Caso tonal:

$$UET(k,i) = F(j)+avtm+vf \qquad (5.5)$$

Caso no tonal:

$$UENT(k,i) = F(j)+avtm+vf; \qquad (5.6)$$

5.1.1.1.5 Umbral de enmascaramiento global.

El cálculo del umbral es de la siguiente manera; al umbral absoluto se le suman las contribuciones de cada uno de los componentes tonales y no tonales en cada subbanda.

Ambos modelos psicoacústicos incluyen un umbral de enmascaramiento absoluto, el cual ha sido determinado empíricamente, este es el mínimo umbral auditivo en un ambiente silencioso. Se debe recordar que éste es la intensidad del sonido más débil que se puede escuchar cuando no hay más sonidos presentes.

5.1.1.1.6 Umbral de enmascaramiento mínimo.

Ambos modelos psicoacústicos seleccionan el mínimo umbral de enmascaramiento en cada subbanda.

Con el modelo I, para encontrar el umbral de enmascaramiento mínimo en cada subbanda, simplemente se extrae el mínimo valor del espectro global incluido entre las dos frecuencias límites de cada subbanda, o sea, el valor extraído del umbral global debe ser el valor mínimo de enmascaramiento en la subbanda. Este método se comporta bien para las subbandas más bajas donde la subbanda es estrecha con respecto a las bandas críticas, pero se vuelve inadecuado para las subbandas altas porque una banda crítica en esta frecuencia

se distribuye sobre varias subbandas. Esta imprecisión se incrementa todavía más, debido a que el modelo I concentra todas las componentes no-tonales, dentro de cada banda crítica, en un único valor para una sola frecuencia.

5.1.1.1.7 Factores de escala

Esta en realidad es una parte del análisis psicoacústico para la capa II, su función es encontrar los factores de escala que serán usados para encontrar lo niveles de presión acústica, el proceso es el siguiente:

Determina el máximo valor absoluto de 3 grupos de 12 muestras subbanda para cada una de las 32 subbandas y lo almacena en MAXS. Busca en la tabla el valor más cercano por exceso al valor MAXS

Aunque en realidad se encuentran 3 valores de factores de escala sólo se selecciona el mayor de los tres, este valor se pasa a la siguiente parte del análisis que es encontrar el nivel de presión sonora.

5.1.1.1.8 Nivel de presión sonora

La función de esta parte el proceso es determinar el nivel de presión sonora para cada subbanda el proceso es el siguiente primero se encuentra el valor de potencia más alto en cada subbanda en el vector de densidad espectral de potencia que se obtuvo en el análisis FFT, después se selecciona el valor más grande entre el factor de escala y el que se obtuvo del análisis FFT será el nivel de presión acústica.

5.1.1.2 Filtro de análisis

Su función en el proceso completo es la de dividir la señal de audio en 32 bandas de frecuencias tratando de simular el fenómeno de las bandas criticas en el oido, sin embargo no es muy adecuado, debido a que tienen un ancho de banda fijo; a diferencia del oído en el que el ancho de las bandas criticas varía con la frecuencia, para ejemplificar esto basta con decir que el oído tiene una limitada selectividad en frecuencia que varía en exactitud desde menos de 100 Hz para las frecuencias más bajas y hasta un poco más de 4 kHz para las frecuencias más altas. En consecuencia el ancho de banda que proporcionan los filtros es demasiado amplio para las frecuencias bajas y demasiado estrecho para las frecuencias altas; así que el número de bits del cuantizador no se puede optimizar para la sensitividad al ruido dentro de cada banda crítica. Entonces, lo mejor es que al espectro audible se le hagan particiones en bandas críticas (por medio de la transformada DCT) que reflejen la selectividad en frecuencia del oído humano.

El filtro es relativamente simple, pero da una buena resolución en el tiempo con una aceptable resolución en frecuencia. El banco de filtros polifásico presenta pérdidas; inclusive sin cuantización no hay posibilidad de recuperar exactamente la señal de entrada. Afortunadamente, el oído humano no es capaz de percibir el error introducido por el banco de filtros. También existe solapamiento en frecuencia entre bandas adyacentes del filtro;

por lo tanto, una señal en una frecuencia particular puede afectar las dos salidas adyacentes en el banco de filtros. El proceso es el siguiente:

Se inicia haciendo el siguiente corrimiento.

$$X[i]= X[i\text{-}32] \qquad i=511\ldots31. \qquad (5.7)$$

donde $X[i]$ es un vector que almacena los últimos 512 valores y es inicializado con ceros después se introducen los nuevos 32 valores

$$X[i] =S_{pcm}[i], \qquad i=0\ldots31. \qquad (5.8)$$

Después se multiplica por los coeficientes de la ventana de análisis:

$$Z[i]=C[i]*X[i], \qquad i=0\ldots511. \qquad (5.9)$$

Ahora se realiza el calculo parcial

$$Y[i]=\sum_{j=0}^{7} Z[i+64*j], \qquad i=0\ldots31 \qquad (5.10)$$

y por ultimo se calculan las 32 muestras

$$S[i]=\sum_{k=0}^{63} M[i,k]*Y[k] \qquad i=0\ldots31 \qquad (5.11)$$

en esta figura se muestra un esquema que muestra el funcionamiento del filtro de análisis.

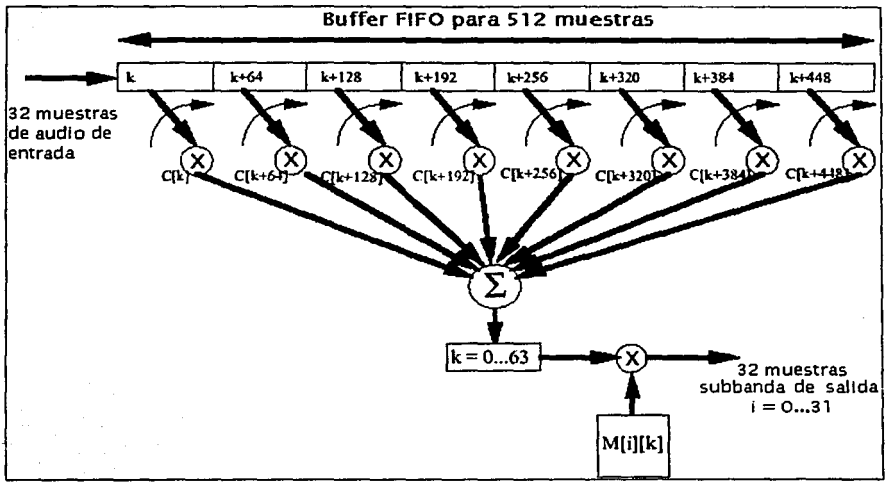


Figura 5.2 Diagrama del filtro subbanda

5.1.1.3 Transformada DCT

Como ya se ha mencionado anteriormente para poder realizar una compresión eficiente es necesario que la información tenga una entropía baja[8], como ya sabemos el audio tiene una entropía muy elevada en el dominio del tiempo pero si se mapea en otro dominio como la frecuencia su entropía es más baja, así que para lograr una buena compresión es necesario el encontrar una transformada que tenga propiedades de concentración de entropía cercanas a las optimas, para el caso del audio, la transformada DCT aunque no es la óptima aunque tiene muy buenas características para la compresión de audio, además de que se puede reconstruir una secuencia muy aproximada con pocos coeficientes lo que es muy útil para la compresión.

Le ecuación de la transformada DCT[9] es el siguiente

$$U[k] = \varpi[k] \sum_{m=1}^{M} u[m] \cos \frac{\pi(2m-1)(k-1)}{2M} \qquad k = 1,\ldots,M \qquad (5.12)$$

[8] www.helsinki.fi/~ssyreeni/dsound/dsound-t-01
[9] A Oppenheim y R. Schafer. *Discrete Time Signal Procesing*

donde

$$\varpi(k)=\begin{cases}\frac{1}{\sqrt{M}}, & k=1\\ \sqrt{\frac{2}{M}}, & 2\le k\le M\end{cases} \qquad (5.13)$$

y el vector u,1 vector con las muestras en el dominio del tiempo.

La implementación que se uso en el programa no implica más la misma definición de la transformada con pequeñas modificaciones para mejorar el rendimiento.

5.1.1.4 Decimación y Cuantización

El proceso de cuantización el proceso en el que se le asignan la cantidad de bits a cada banda, en el MPEG 1 layer 3 este proceso aunque bastante más complejo y eficiente, se basa en el mismo principio que nuestro algoritmo. El procedimiento que proponemos se basa en que un componente ya sea tonal o no tonal genera un enmascaramiento en las líneas de frecuencia adyacentes a la enmascarante, entre las señales que estas componentes puede enmascar esta el ruido de cuantización además de sonidos que se encuentren debajo de este umbral de enmascaramiento.

Basándonos en este principio se realiza el siguiente proceso:
Primero con el valor de la relación entre la señal y el enmascarante calculamos el número de bit necesarios para cada banda.

$$Nbits = \text{ceil}(SMR(sb)/(20*\log(2))) \qquad (5.14)$$

El siguiente paso evita que se agregue más ruido del que se permite, para explicar este proceso es necesario recordar que la ecuación en la que se basa el cálculo de bits es el ruido de cuantización pero este valor sólo es valido cuando el valor de la señal es el máximo, es decir esa relación es válida sólo en el mejor de los casos, pero si la señal no tiene el valor máximo la relación señal a ruido será menor agregando más ruido a la señal de que ese había calculado.

Para resolver en parte este problema, se encuentra el valor máximo y mínimo de cada banda y se ajusta para que el rango dinámico en la cuantización no exceda esos valores, esto no soluciona completamente el problema pero si ayuda a que el ruido que se agregue no sea demasiado, también hay que señalar que además de los datos de la señal también hay que enviar datos adicionales que son necesarios para recuperar correctamente los datos aunque estos datos no representan una cantidad importante. Una vez que se adecuó el rango dinámico, se realiza la decimación y la cuantización. Para este caso no se realiza codificación, a diferencia del MPEG1 capa 3, que usa una codificación de Huffman

la razón que para poder realizar una tabla de Huffman seria necesario el realizar muchas pruebas para tener suficientes datos estadísticos y lograr que la codificación realmente mejore el rendimiento del sistema.

5.1.1.5 Formato de la trama

Primero hay que señalar que este formato sólo está pensado para el almacenamiento por lo que no se incluye ningún sistema de protección contra errores. El tamaño de la trama es variable y depende de la cantidad de información que se necesite transportar, el tamaño máximo de la trama es 640 bits y el mínimo de 32 bits. Primero se envían el valor necesario para la decuantización e interpolación que son dos (sf y offs), y a continuación los datos.

La trama es la siguiente:

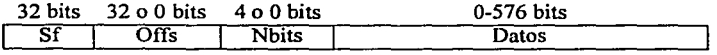


Figura 5.3 Trama del Sistema de compresión

donde

Sf Este valor nos permite el realizar la interpolación.

Offs Este valor también se necesitan en la interpolación, con este valor se aprovecha al máximo, el rango dinámico que admite el número de bits.

Nbits. Indica la longitud en bits de los datos que se leerán en esa subbanda

Datos Contiene los datos que servirán para reconstruir los datos

El orden de las tramas es fijo y progresivo es decir la primera trama correspondería a la primera subbanda el siguiente a la subbanda 2 y así sucesivamente hasta la banda 32 y a partir de ahí se vuelve a repetir, para cada bloque.

5.1.2 Decodificador

Como ya lo mencionamos anteriormente la decodificación es mucho más sencilla que la codificación y mucho más rápida, inclusive se puede decir que la decodificación si se puede realizar en tiempo real ya que no realiza los procesos necesarios para el análisis psicoacústico, la parte que más procesamiento utiliza es la del filtro de síntesis.

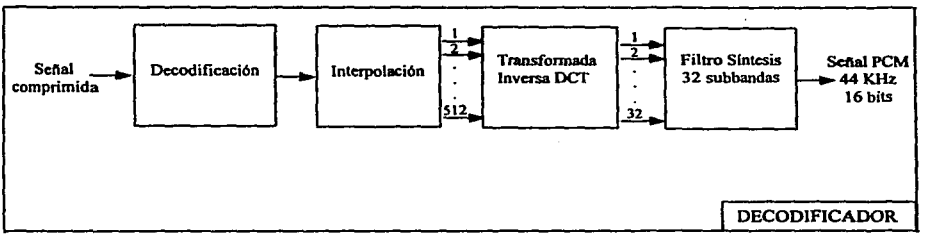


Figura 5.4 Diagrama a bloques del decodificador del sistema de compresión.

5.1.2.1 Decuantización e Interpolación

Para la decuantización e interpolación sólo son necesarios los dos coeficientes que se encuentran al principio de la trama, el resultado de multiplicar el coeficiente *sf* por los datos de la trama se suman al coeficiente *offs* y tendremos el valor que se usará en la siguiente parte del proceso que la transformada IDCT.

Hay que señalar que para realizar este proceso es necesario llevar un control muy estricto de los datos que se están leyendo ya que el orden es muy importante si se pierde el orden se perderá toda la información ya que no se usa ningún carácter de control ni nada parecido (excepto al final del archivo).

5.1.2.2 Transformada IDCT

Una vez que se han recupero los coeficientes de la transformada DCT el siguiente paso es realizar la transformada inversa coseno.

La ecuación de la transformada inversa es la siguiente:

$$u[m]=\sum_{k=1}^{M}\varpi[k]U[k]\cos\frac{\pi(2m-1)(k-1)}{2M} \qquad m=1,\ldots,M \qquad (5.15)$$

donde

$$\varpi(k)=\begin{cases}\dfrac{1}{\sqrt{M}}, & k=1\\[2ex] \sqrt{\dfrac{2}{M}}, & 2\le k\le M\end{cases} \qquad (5.16)$$

La implementación que se usó en el sistema sólo implica el programar la definición de la transformada con pequeñas modificaciones para mejorar su desempeño.

5.1.2.3 Filtro de síntesis

Este proceso de síntesis esta basado en el MPEG-2 audio (ISO/IEC standard 13818-3), la complejidad del decodificador es mucho menor que la del codificador y la mayor parte del procesamiento se dedica a este filtro. El proceso es el siguiente:

Primero inicia con un vector de 32 muestras

$$S_{sb}=[i] \qquad i=0...31 \qquad (5.17)$$

después, el vector *V[i]* de 1024 elementos es recorrido 64 posiciones de acuerdo al siguiente proceso:

$$V[i]=V[i-64] \qquad i=1023...64 \qquad (5.18)$$

Y el siguiente proceso es esta operación

$$V[i]=\sum_{k=0}^{31} S_{sb}[k]\cos(\pi(2k+1)(16+i)/64), \qquad i=0...63 \quad (5.19)$$

Entonces se crea un vector *U[i]* de 512 elementos, y se realiza el siguiente proceso haciendo k=0...7 y a l=0...31;

$$U[64*k+l]=V[128*k+l],$$
$$U[64*k+32+l]=V[128*k+96+l], \qquad (5.20)$$

Alternativamente, también se puede escribir de la siguiente manera :

$$U[u+32*v]=V[u+32*v+((v+1)>>1)<<6)] \qquad (5.21);$$

donde >> y << son operadores de el corrimiento binario hacia la derecha y hacia la izquierda[10] (y los huecos se rellenan con ceros).

Finalmente, se calculan las nuevas 32 muestras PCM de salida de la siguiente manera:

$$S_{pcm}[j]=\sum_{i=0}^{15} W[j+32*i], \qquad j=0...31,\ W[i]=U[i]*D[i] \qquad (5.22)$$

donde D[i] son los coeficientes de la ventana de síntesis, la ecuación 6 también puede ser escrita

$$S_{pcm}[j]=\sum_{i=0}^{15} U[j+32*i]*D[j+32*i], \qquad (5.23)$$

[10] Código fuente ISO `dist10` que se puede encontrar en _ `ftp://ftp.tnt.uni-hannover.de/pub/MPEG/audio/`

como se puede observar en esta descripción el proceso de decodificación del archivo es mucho más sencillo que la codificación y el proceso que más procesamiento requiere es el filtro de síntesis.

La salida del filtro de síntesis es la señal reconstruida *PCM*, de aquí la señal puede ser enviada a un dispositivo de salida como un convertidor D/A o escribirse en un archivo.

5.2 Sistema 2

Este sistema es muy parecido al anterior, pero debido a que no usa el modelo psicoacústico de MPEG el sistema es más rápido y funciona adecuadamente, en este sistema la parte que procesamiento utiliza es la del filtro de síntesis, también a diferencia del anterior utiliza una codificación de Huffman que mejora un poco su rendimiento.

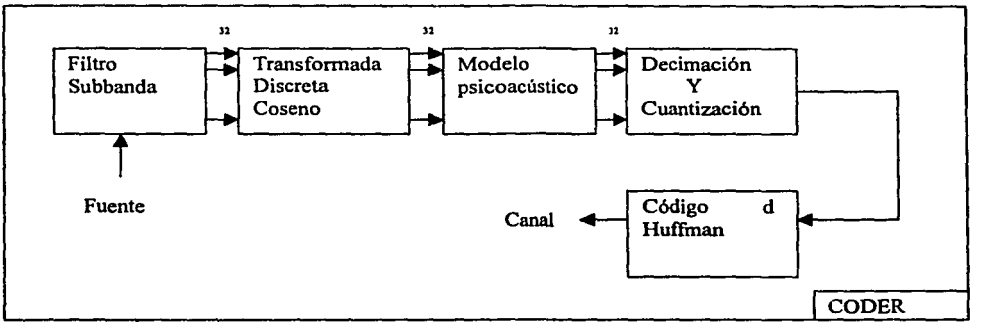


Figura 5.5 Diagrama de bloques

implementación es la misma que se usó en el primer sistema.

5.2.2.2 Transformada DCT

La transformada DCT es la misma que en el programa anterior de cualquier manera aquí se muestra la ecuación de la transformada:

$$U[k] = \varpi[k]\sum_{m=1}^{M} u[m]\cos\frac{\pi(2m-1)(k-1)}{2M} \qquad k = 1,\ldots,M \qquad (5.12)$$

donde

$$\varpi(k) = \begin{cases} \frac{1}{\sqrt{M}}, & k = 1 \\ \sqrt{\frac{2}{M}}, & 2 \le k \le M \end{cases} \qquad (5.13)$$

y el vector u el vector con las muestras en el dominio del tiempo y U la señal en el dominio el coseno.

5.2.2.3 Análisis Psicoacústico

Este modelo psicoacústico se basa en las diferencias en la audición de frecuencias en el oído humano. En el modelo se eliminan los niveles más bajos de cada banda de frecuencia de cada bloque de información.

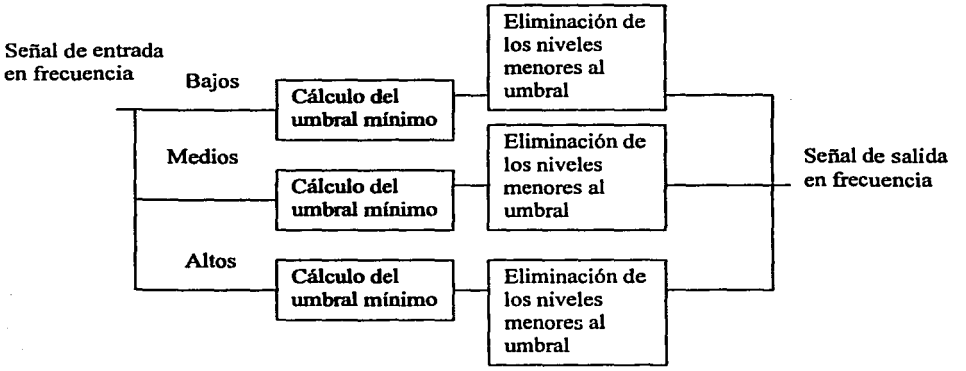


Figura 5.6 Diagrama a bloques del análisis psicoacústico del sistema 2.

El primer paso es el de dividir las bandas de frecuencia en 3 tipos: Bajas, Medias o Altas, esto con el fin de darles diferentes umbrales de mínimos a cada tipo de banda, dependiendo de la respuesta del oído en esas frecuencias.

Luego se estima el umbral mínimo calculando la media de cada banda y modificándolo con un coeficiente obtenido empíricamente mediante diferentes pruebas con diversas personas y canciones, con el fin de obtener el mejor modelo perceptual, logrando una buena relación de compresión sin sacrificar mucho la calidad.

- Para bandas bajas (1 a 6): Se eliminan los valores menores a 0.5 veces el promedio de esa banda en el bloque de información.
- Para bandas bajas (7 a 13): Se eliminan los valores menores a 1.25 veces el promedio de esa banda en el bloque de información.

- Para bandas bajas (14 a 32): Se eliminan los valores menores a 2 veces el promedio de esa banda en el bloque de información.

5.2.2.4 Decimación y Cuantización

La decimación y cuantización es prácticamente la misma aunque difieren en un aspecto, el código anterior tiene como salida números de longitud variable, en este caso la salida tiene número de longitud fija, y se hace una cuantización a 6 bits.

5.2.2.5 Codificación de Huffman

Para la obtención del código de Huffman utilizado se realizaron varias pruebas y se observó el comportamiento de los valores de las muestras en cada una de ellas, con esto se obtuvo una probabilidad para cada valor posible.

Una vez obtenidas las probabilidades para cada prueba las juntamos para sacar una probabilidad final para cada valor. Se obtuvo la entropía mínima posible para este caso y se obtuvo el valor de 3.93bits/palabra.

Finalmente se realizó la tabla de Huffman para con los datos previos y se logró un promedio de 4 bits/palabra, lo cual implica una reducción de 2 bits más con respecto a la obtenida en la decimación. Es importante notar que la palabra con más probabilidad (más del 50%) sólo esta codificada con un bit y la palabra con menor probabilidad con 10.

Ver el Apéndice para consultar la tabla de correspondencias.

5.2.3 Decodificador

La complejidad computacional del decodificador es la misma que la del decodificador anterior y el proceso es el mismo a continuación se muestra un diagrama en el que se muestra su funcionamiento.

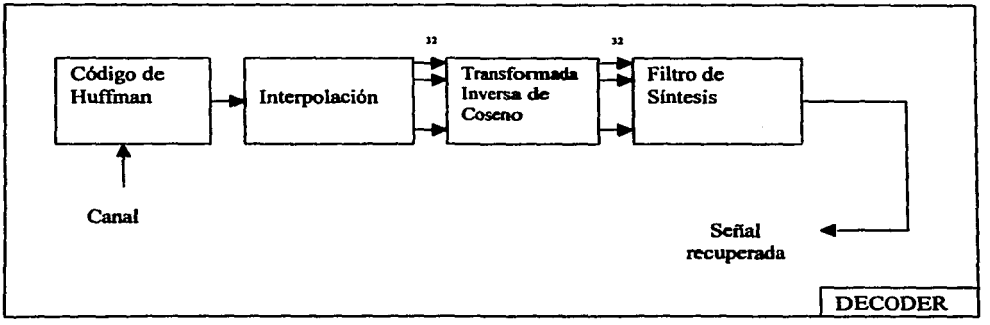


Figura 5.7 Diagrama a bloques del decodificador del sistema 2.

5.2.3.1 Decodificación Huffman

La decodificación de Huffman es muy sencilla, el método consiste en ir leyendo y comparando con la tabla de correspondencias en la cual las palabras estarán ordenadas de tal manera que las palabras con mayor probabilidad aparezcan más rápidamente, esto evidentemente tiene un impacto en el desempeño del programa, una vez que se ha encontrado la palabra en la tabla su valor correspondiente, se sustituye en una matriz que contiene la información del bloque que se esta analizando y de esta manera se recuperan los datos, una vez que se ha recuperado un bloque completo se continua con el proceso.

5.2.3.2 Interpolación y Decuantización

El método es igual al del primer algoritmo con la única diferencia que la longitud de las palabras es fija, esto no afecta el número de operaciones necesarias ya que es prácticamente la misma, también hay una diferencia en la cuantizacion ya que en este caso se hace una cuantización lineal y en la anterior una cuantización no lineal sino logarítmica.

5.2.3.3 Transformada IDCT

Ya que se recuperaron los datos que resultaron la trnasformada coseno discreta a la información se le aplica la transformada IDCT para recuperar los datos que se usarán para el filtro de síntesis, la ecuación de la transformada IDCT es:

$$u[m]=\sum_{k=1}^{M}\varpi[k]U[k]\cos\frac{\pi(2m-1)(k-1)}{2M} \qquad m=1,\ldots,M \qquad (5.15)$$

donde

$$\varpi(k)=\begin{cases}\dfrac{1}{\sqrt{M}}, & k=1\\[2ex] \sqrt{\dfrac{2}{M}}, & 2\le k\le M\end{cases} \qquad (5.16)$$

Donde $U[k]$ es la señal en el dominio del coseno y $u[m]$ es la señal en el dominio del tiempo.

5.2.3.4 Filtro de Síntesis

El filtro de síntesis usa el mismo algoritmo e implementación, este proceso es el que requiere un mayor procesamiento en el decodificador, al igual que en el caso anterior la salida del filtro de síntesis es ya la señal reconstruida y se puede enviar a un archivo o aún dispositivo de salida como un convertidor D/A.

6 Resultados

6.1 Sistema 1

Este sistema no se probó en una gran cantidad de personas, sin embargo se tiene una muy buena calidad, aunque una relación de compresión pequeña. Los resultados se obtuvieron de la siguiente forma: se hicieron las pruebas con diez personas; seis canciones comprimidas tienen una gran calidad, donde la mayor compresión posible fue de 3 a 1.

	Porcentaje	Número de incidencias
Errores	51.4 %	36
Aciertos	20.0 %	14
No supo	28.6%	20

Tabla 6.1 Tabla de resultados del sistema de compresión 1.

Aunque varias personas pudieron diferenciar que canción era la original y cual la canción comprimida, no podían explicar la diferencia, la razón de este fenómeno puede ser que al eliminar parte de la información la energía total del sonido disminuye, esto provoca que aunque la diferencia se nota no se puede describir, este problema se puede solucionar al sustituir la energía que se eliminó, con la información que se conservó, para lograr que esto funcione adecuadamente es necesario realizar muchas pruebas, por esta razón no corregimos este error.

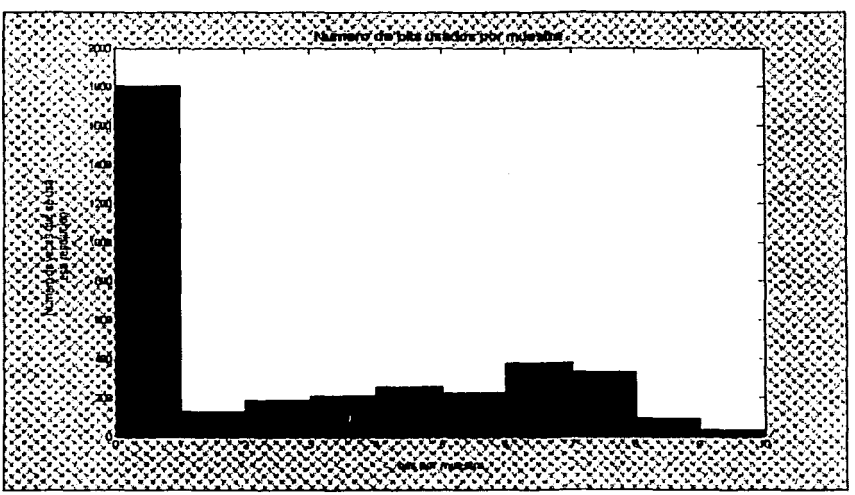


Figura 6.1 Bits por muestra – número de veces que se uso esa cantidad de bits

En esta gráfica se muestran el número de bits por muestra y el número de veces que se utilizan esa cantidad de bits por muestra.

Con esta gráfica es muy fácil ver como se realiza la compresión ya que si no tuviera compresión solo se presentaría una barra con un tamaño enorme en 16 bits; otro aspecto importante es que el número máximo de bits que se usaron fue de 10 para este caso, esto también no muestra lo bien que funciona el sistema de compresión, el número total de bits que se uso en la canción comprimida fue de 10 829 y el número de bits que se usarían sin compresión es 58 368.

En la siguiente imagen se muestra una imagen se muestra una gráfica que muestra la intensidad (color), tiempo(escala horizontal) y frecuencia (escala vertical):

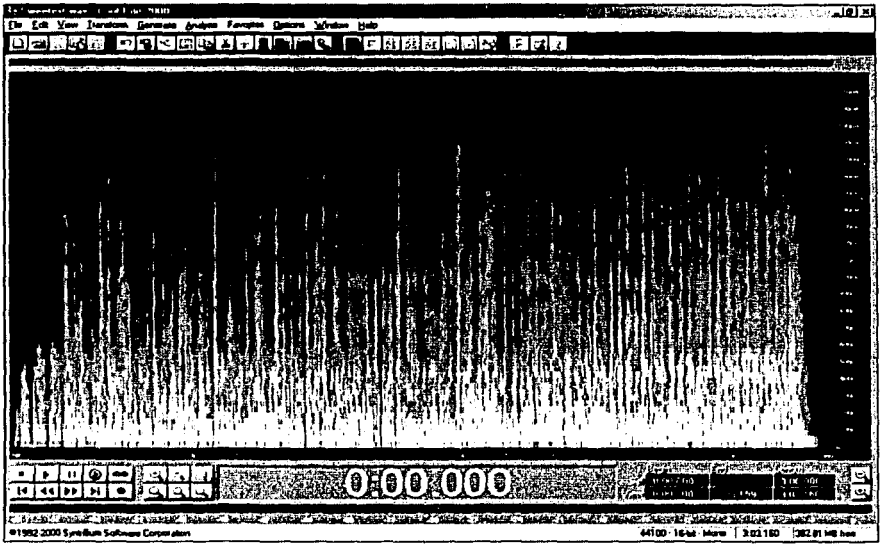


Figura 6.2 Espectro de una canción original.

En las gráficas es evidente que se pierde información sin embargo esta diferencia no puede ser percibida por las personas.

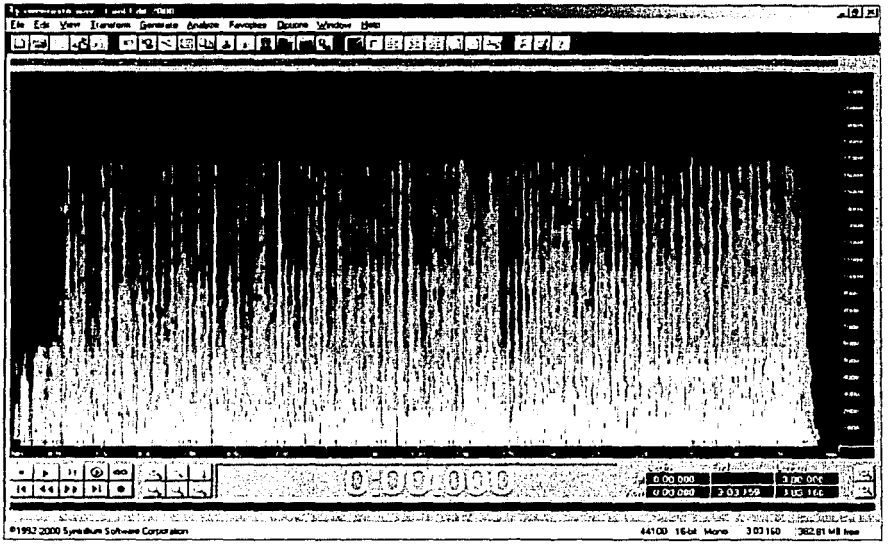


Figura 6.3 Espectro de una canción comprimida.

En cuanto al tiempo necesarios para el procesamiento es relativamente grande ya que necesita 3 veces el tiempo de la canción (en una computadora AMD K6-III a 400 MHz) para comprimir y descomprimir, pero esto se puede mejorar, de hecho, estos tiempo de procesamiento es el resultado de mejoras en el algoritmo con lo que se ha logrado un aumento en el rendimiento del 40% en la rapidez.

6.2 Sistema 2

A continuación presentamos los diagramas de espectro de las cuatro canciones tanto la canción original como la comprimida.

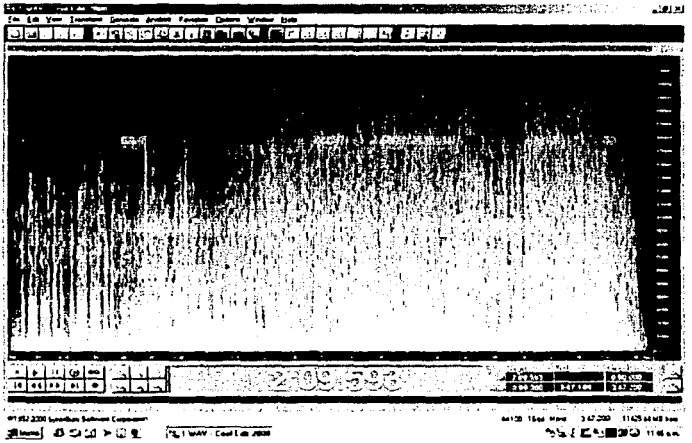


Figura 6.4 Canción 1 (Original)

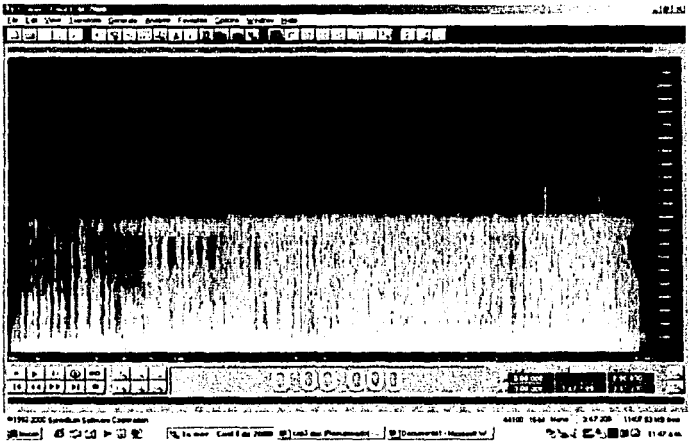


Figura 6.5 Canción 1 (Reconstruida)

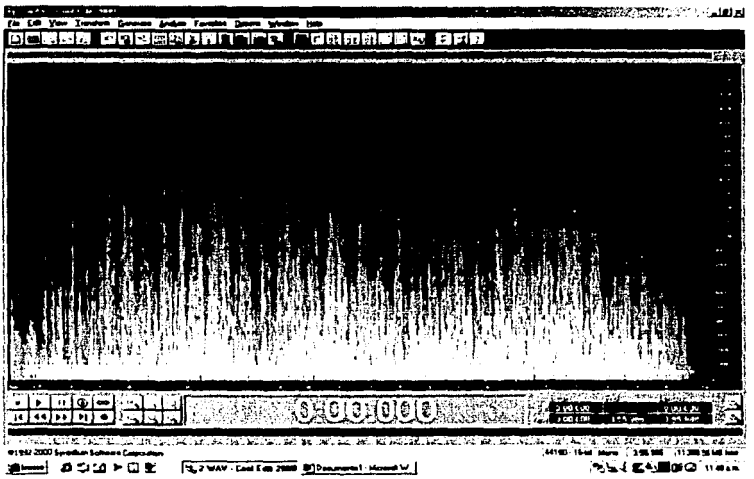


Figura 6.6 Canción 2 (Original)

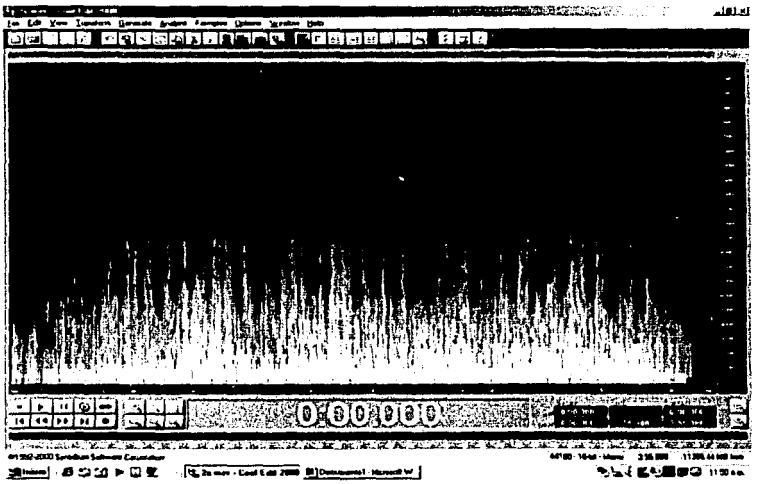


Figura 6.7 Canción 2 (Reconstruida)

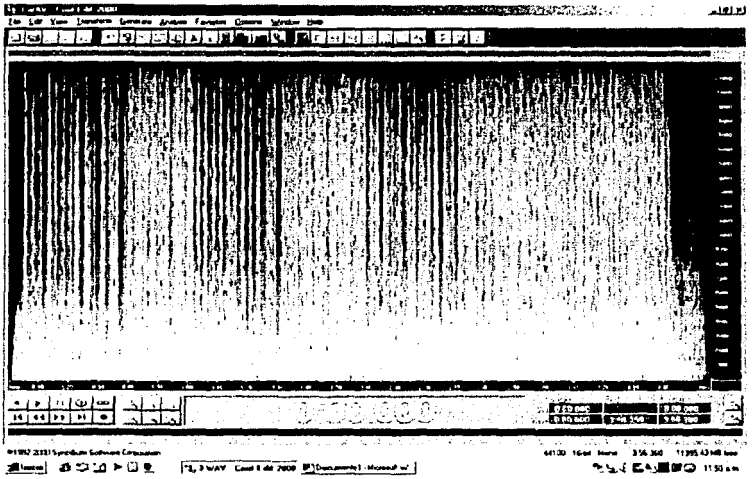


Figura 6.8 Canción 3 (Original)

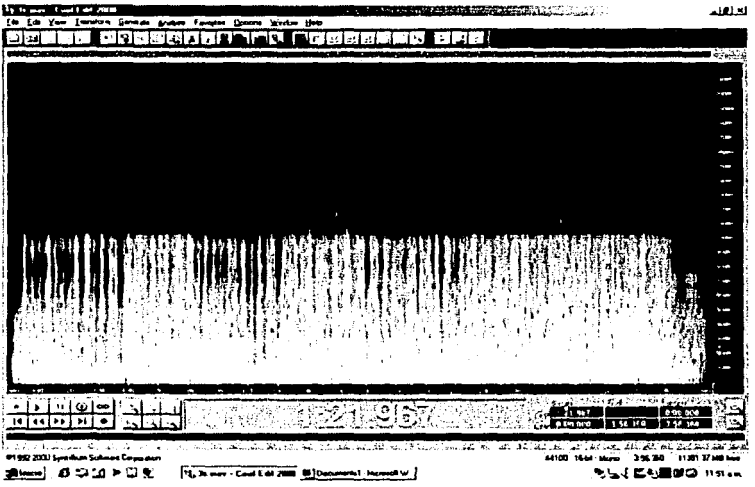


Figura 6.9 Canción 3 (Reconstruida)

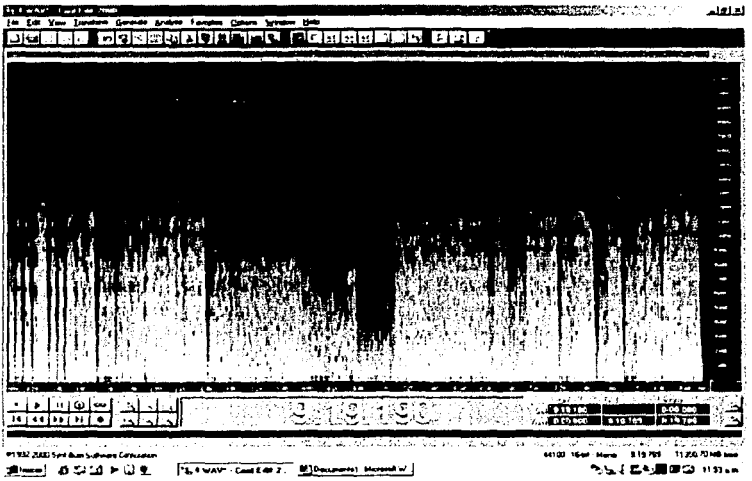


Figura 6.10 Canción 4 (Original)

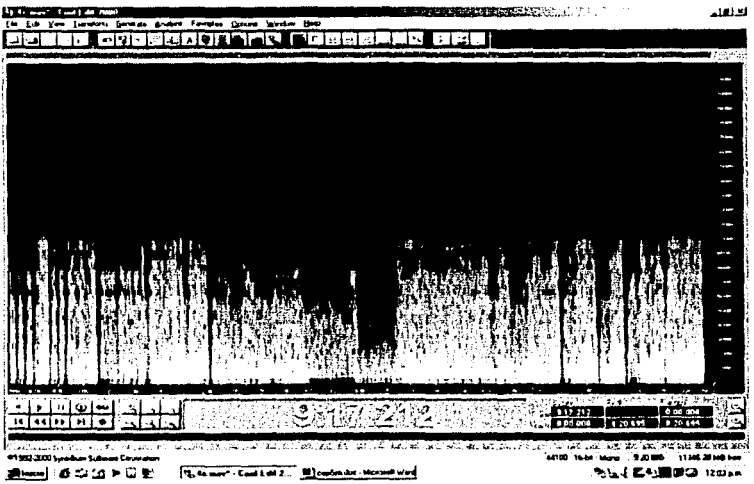


Figura 6.11 Canción 4 (Reconstruida)

En la siguiente tabla se muestra el resultado de cuatro canciones comprimidas, con sus diferentes tiempos y tasas de compresión, es importante aclarar que usar el "joint stereo" significa una ganancia en la tasa de compresión de 1.4 veces mayor de los resultados que se muestran.

Canción	Tasa de Compresión	Tiempo de Compresión y descompresión	Tamaño de la canción
1	4.82	8.36	4:40
2	5.3	8.15	3:52
3	4.72	8.10	3:45
4	6.22	19.34	9:01

Tabla 6.2 Tabla de resultados del sistema de compresión 2.

Las canciones utilizadas representan diferentes géneros musicales, para notar las diferencias que existe en su nivel de percepción acústica.

6.3 Pruebas y Resultados Perceptuales

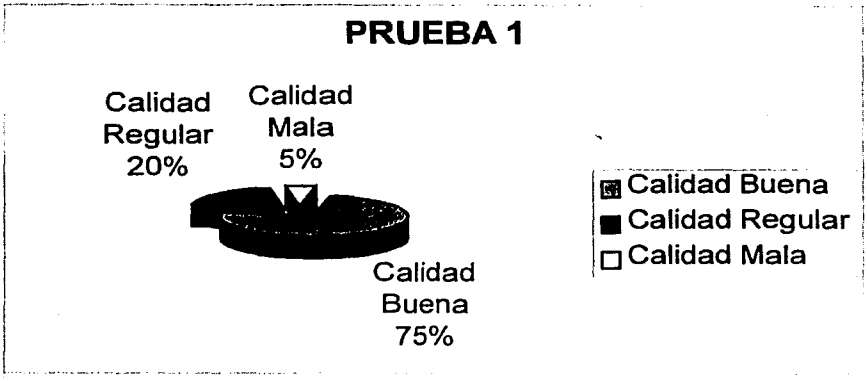
Las pruebas se realizaron en diferentes grupos y con diferentes melodías, esto es para representar diferentes géneros musicales y también diferentes rangos de frecuencias, por esto las canciones escogidas fueron:

- Una que representa al rango de voz
- Una que representa bajos y agudos
- Un solo instrumento en todo su rango
- Conjunto de instrumentos buscando todo el rango auditivo

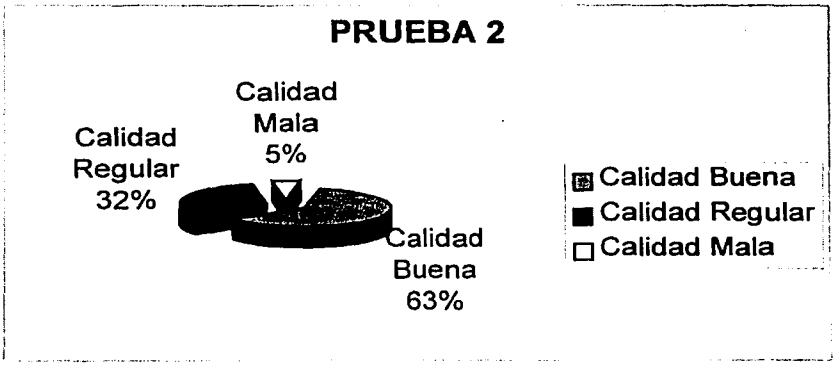
Las pruebas fueron realizadas a diferentes grupos de personas, y se comprobó el daño auditivo que sufren las personas conforme al tiempo, es decir que un niño puede encontrar más errores en las melodías que una persona adulta, y aun así los resultados fueron satisfactorios.

Se evaluaron más de 200 personas y los resultados se pueden ver en la siguiente figura

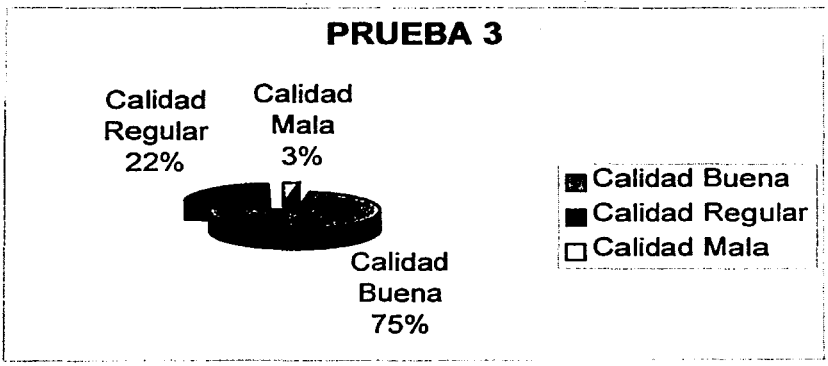
Para la primera canción:



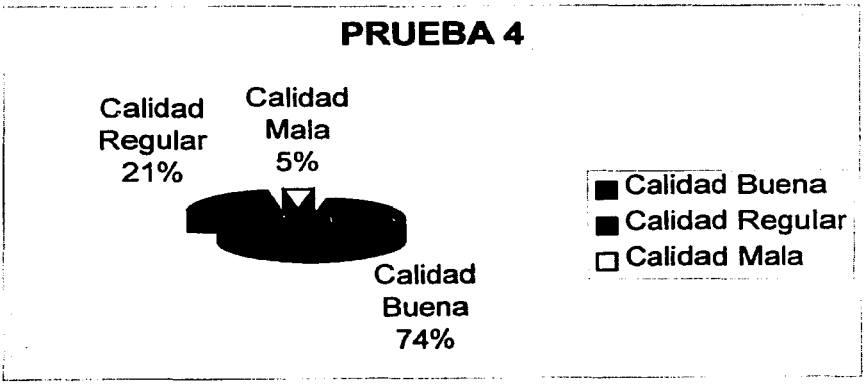
Segunda Canción:



Tercera Canción:

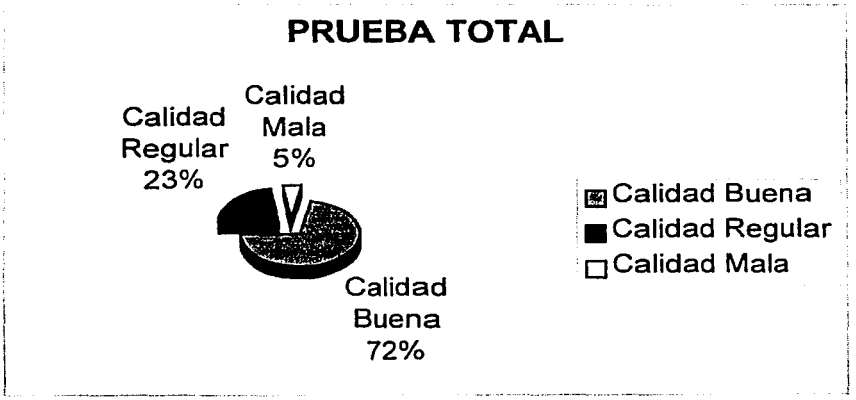


Cuarta Canción:



La prueba total que definiría si el algoritmo nuevo sería aceptado por la población, ya que este sería transparente al usuario y podría ser encapsulado en IP para su transmisión digital.

PRUEBA TOTAL:



7 Conclusiones

El presente trabajo refleja el esfuerzo por la investigación de quienes lo elaboraron y como primera conclusión tenemos un trabajo de alto valor bibliográfico, esto es debido a que puede ser considerado como una fuente de conceptos con relación al tema de compresión de audio y sus aplicaciones.

Aplicaciones como radiodifusión digital, transmisión vía IP o para educación a distancia son posibles aplicaciones del algoritmo creado. Estas aplicaciones serán de gran utilidad en pocos años pues se dará un cambio en las tecnologías actuales de comunicación

Es necesario decir también que las pruebas realizadas, fueron satisfactorias, y podemos señalar que los dos sistemas que proponemos uno tiene una calidad muy buena aunque su relación de compresión no es tan grande, y el segundo sistema nos da una relación mucho mayor, un tiempo de procesamiento menor aunque se sacrifica un poco la calidad obtenida, y aún así se considera aceptable en las pruebas preceptúales que aplicamos.

Con respecto al trabajo en el Procesador Digital de Señales (DSP, por sus siglas en inglés), nuestro sistema no se puede implementar en él, debido a que el convertidor Analógico Digital sólo puede tomar muestras a 8kHz, esto implica que se filtra la señal a 4kHz, lo cual es inútil en el audio, ya que la tasa de muestreo requerida es de 44.1kHz, para que la máxima frecuencia aceptada sea de 22.05kHz, como se menciona en el primer capítulo sobre el teorema de muestreo de Nyquist.

También mencionamos en el trabajo el uso del Procesador Digital de Señales, el cual tiene características importantes en la implementación de proyectos en él, es un procesador muy potente, que es de mucha ayuda.

Este procesador, es importante mencionar, tiene cualidades que sabiéndolo manejar es una herramienta útil en el campo del procesamiento, pues no gasta recursos como lo haría un Microcontrolador o un Microprocesador de uso general.

En este momento existen chips que realizan esta función solamente, por lo que no es recomendable usar un DSP de uso general.

Dentro de nuestro trabajo, como se ve en los apéndices, están dos programas que pueden ser simples de entender y al mismo tiempo interpretar, para implementarlo en el Procesador; esto sería posible, si se tuviera una tarjeta que sirva de convertidor Analógico Digital que sea para audio.

Aún así trabajando con el procesador, aprendimos una serie de fundamentos básicos para trabajar con él.

Con respecto al estándar MPEG capa 3, se utilizaron partes del mismo, como fueron los filtros subbanda, pero al llegar a la codificación por transformadas no los utilizamos, debido a que esto no representa algún conocimiento y sólo sería la copia del estándar. Todo esto se justifica con el hecho de que el estándar maneja muchas tablas creadas por el equipo de expertos, y al no tener el estándar como tal sino las diferentes partes obtenida de diversos artículos e investigaciones, resultaron para nosotros incomprensibles ya que los datos nunca supimos de donde se obtienen, y preferimos hacer uso de los conocimientos dados en la universidad para crear nuestro código, y esto se lleva desde a utilizar archivos WAV, así como desarrollar nuestro propio código perceptual, sin recurrir a tablas que no sabemos de donde salen los valores.

El uso de la Transforma Coseno Discreta, y no la Coseno Discreta Modificada ya que notamos que el uso de la Transformada Coseno Modificada sirve solamente para el estándar, y como habíamos dicho el copiar el estándar no lo consideramos como ningún mérito de investigación.

Con respecto a los filtros, utilizamos los referentes al estándar por que conocemos su funcionamiento, así como los datos de las tablas sacadas del estándar son comprensibles ya que son sólo factores de escala usados para impedir el aliasing.

Cabe aclarar que todas aquellas personas que siguen un estándar y lo llevan a cabo tiene un valor muy meritorio, pues el seguir todas las especificaciones no es una tarea fácil, ya que se necesita entender que se está haciendo, para que los resultados sean los correctos. Simplemente nosotros queríamos realizar algo que fuera hecho por nosotros, tomando en cuenta que la investigación en México es la base para que el país siga creciendo.

El tiempo necesario para el procesamiento es lo suficientemente satisfactorio para la implementación en tiempo real y además tiene una calidad aceptable en audio. Por lo que este sistema puede ser implementado en un DSP y aunque necesita herramientas de procesamiento muy veloces, creemos que la tecnología ha llegado al punto en que este no sería un problema grande.

El aspecto de la calidad es importante mencionarlo por separado, esto es por que el audio depende de las personas, de esta forma se puede entender que el concepto de calidad del audio digital es subjetiva, aunque existen parámetros para cuantificar la distorsión estos no implican que si la gente acepta el sistema se tengan resultados mejores en estos parámetros. Las pruebas que se muestran en el capítulo de resultados, son muy importantes, por que nos señalan que tan bueno es nuestro sistema, y se puede comparar con otros que estén en el mercado y los resultados serían aceptables.

Una conclusión en especial importante para nosotros es que este trabajo pude servir de base para el cambio digital que se está llevando a cabo, logrando de esta manera poder implementar nuevas tecnologías que servirán para la migración hacia la radiodifusión digital, es entonces de alto valor si se le da seguimiento tanto por parte nuestra como de la Universidad, de esta manera se puede tener un papel importante en la migración de tecnologías, ya que están hechas en México y no serían importadas.

Asimismo deseamos que este trabajo pueda servir como impulso de otros proyectos con respecto al tema y deseamos que las nuevas generaciones que tengan intención de seguir con la compresión de audio para su posible transmisión digital.

Finalmente queremos concluir nuestro trabajo, teniendo en cuenta que todos los objetivos, tuvimos una buena calidad, el manejo del estándar y de los programas utilizados nos dieron conocimientos buenos con respecto del tema, así como de herramientas necesarias para realizarlo y queremos que se dé seguimiento a este trabajo, ya sea por nuestra parte, para que dejemos de importar tecnologías y este país avance en la investigación.

Apéndice A

Código Fuente

Sistema 1

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <psico.h>

#define pi 3.14159265358979


FILE *readheader(char filename[13],char *headerc, unsigned long *tam)
{

FILE *stream;

stream=fopen(filename,"r+b");

if(fread(headerc,sizeof(char),40,stream)==0)
      return NULL;
else
{
      fread(tam,sizeof(long),1,stream);
      return stream;
}
}


int round(double num)
{
if(num>0)
      if((num-floor(num))<0.5)
            return (int)floor(num);
      else
            return (int)ceil(num);
else
      if(num<0)
            if((ceil(num)-num)<0.5)
                  return (int)ceil(num);
            else
                  return (int)floor(num);
      else
            return 0;
}


FILE *writeheader(char filename[13], char *headerc,unsigned long *tam)
{
FILE *stream;
```

```
stream=fopen(filename,"w+b");

fwrite(headerc,sizeof(char),40,stream);
fwrite(tam,sizeof(long),1,stream);

return stream;
}


int wavread(FILE *stream, short *samples)
{
int numread, i;

numread=fread(samples,sizeof(short),1152,stream);

if (numread==0 & numread<1153)
{
      for(i=numread; i<1152; i++)
      samples[i]=0;
}

return numread;
}


void Filtro_Subbanda(short *muestras, double *C, double *FIFO, double *S, double fcos[64], double
MatS[32][64])
{
int i,j,k;
double Z[512], Y[64], sum;


for(i=511; i>=32; i--)
      FIFO[i]=FIFO[i-32];

for(i=31; i>=0; i--)
      FIFO[i]=((double)muestras[(31-i)])/32768;


for(i=0; i<64; i++)
{
      Y[i]=0;
      for(j=i; j<512; j+=64)
            Y[i]+=C[j]*FIFO[j];
}


/* for( i=-16; i<48; i++)
      fcos[i+16]=i*pi/64;*/

for(i=0; i<32; i++)
{
      S[i]=0;
      for(j=0; j<64; j++)
      {
```

```
    /* sum=0;
        for(k=j; k<512; k+=64)
            sum+=C[k]*FIFO[k]; */
//          S[i]+=cos((2*(i+1)-1)*fcos[j])*Y[j];
        S[i]+=MatS[i][j]*Y[j];
    }
}

}

void Sintesis_Subbanda(double *Ssb, double *D, double *V, short *Spcmi, double Mat[64][32])
{
int i,j,k,l;
double U[512], Spcm[32];
double sum;

for(i=1023; i>=64;i--)
    V[i]=V[i-64];


for(i=0; i<64;i++)
{
    sum=0;
    for(k=0;k<32;k++)
        sum+=Ssb[k]*Mat[i][k];
    V[i]=(double)sum;
}



for(j=0; j<32; j++)
{
    sum=0;
    for(i=0; i<16; i++)
        sum+=V[j+(i<<5)+(((i+1)>>1)<<6)]*D[j+(i<<5)];
    if(fabs(sum)>1)
        Spcmi[j]=sum/fabs(sum)*32767;
    else
        Spcmi[j]=sum*32767;
}



}

void load_tab(char filename [13],double *TAB)
{
FILE *stream;

stream=fopen(filename,"r+b");

fread(TAB,sizeof(double),512,stream);

fclose(stream);
```

```
}

void DCT(double *u, double *U)
{

int m,k,M=36;

U[0]=0;
for(m=0;m<M;m++)
      U[0]+=(1.666666666666667e-001)*u[m];

for(k=1;k<M;k++)
{
      U[k]=0;
      for(m=0;m<M;m++)
            U[k]+=(2.357022603955158e-001)*u[m]*cos(pi*(2*m+1)*k/(2*M));
}
}


void IDCT(double *U, double *u)
{

int m,k,M=36;

for(m=0;m<M;m++)
{

      u[m]=(1.666666666666667e-001)*U[0];
      for(k=1;k<M;k++)
            u[m]+=(2.357022603955158e-001)*U[k]*cos(pi*(2*m+1)*k/(2*M));
}
}


int main()
{
char filenamein[13], filenameout[13], *headerc;
FILE *streamin,*streamout;
double *DCTtmp,*Sb,S[36][32],*FIFO, *SMR;
double *Stmp,*Stmp2,*V,DCTr[36][32],DCTrec[36][32], DCTc[36][32];
short *samplesin, *samplesout;
short *bsamplesin, *bsamplesout;
double *C,*D,*F,*BS,*BSR;
int j,i,k,nm,sb,ind;
unsigned long tam[1],a=0;
double x,buf=0;

double min[32],max[32],Tsf[32],Toff[32],TNbits[32];
double sf,offs,rd2;

int DCTint[36][32],Nbits, Nbitssc=0,Nbitscc=0;

int Tsigno[36][32]={0};

CLF *LT,*LNT,*LTR,*LNTR;
```

```
double **UA, *MAP, *UES, *UEG, *FDE, *UEM, *NPS, Mat[64][32], MatS[32][64];

int temp;

double *U;
double *u;
double **LBC;
double fcos[64];
double win[1024];


SMR=(double *)malloc(32*sizeof(double));
Sb=(double *)malloc(32*sizeof(double));
DCTtmp=(double *)malloc(36*sizeof(double));
u=(double *)malloc(36*sizeof(double));

sample_in=(short *)malloc(1152*sizeof(short));
sample_out=(short *)malloc(1152*sizeof(short));

Stmp2=(double *)malloc(36*sizeof(double));
Stmp=(double *)malloc(32*sizeof(double));
FIFO=(double *)malloc(512*sizeof(double));
V=(double *)malloc(1024*sizeof(double));
C=(double *)malloc(512*sizeof(double));
D=(double *)malloc(512*sizeof(double));
F=(double *)malloc(1024*sizeof(double));
BS=malloc(512*sizeof(double));
BS=(double *)malloc(512*sizeof(double));
BSR=(double *)malloc(512*sizeof(double));
UEG=(double *)malloc(130*sizeof(double));
FDE=(double *)malloc(130*sizeof(double));
UEM=(double *)malloc(130*sizeof(double));
NPS=(double *)malloc(32*sizeof(double));

LNT=malloc(26*sizeof(CLF));
LT=malloc(512*sizeof(CLF));
LNT=malloc(512*sizeof(CLF));

LTR=malloc(512*sizeof(CLF));
LNTR=malloc(512*sizeof(CLF));

MAP=malloc(512*sizeof(double));

UES=malloc(130*sizeof(double));

UA=malloc(130*sizeof(double *));


for(i=0;i<130;i++)
      UA[i]=malloc(3*sizeof(double));


for(i=0; i<64;i++)
      for(k=0;k<32;k++)
            Mat[i][k]=cos(pi*(2*k+1)*(16+i)/64);
```

```
for(i=0; i<512; i++)
{
      FIFO[i]=0;
      V[i]=0;
}
for(i=512;i<1024;i++)
      V[i]=0;


headerc=(char *)malloc(40*sizeof(char));

printf("nombre del archivo de entrada: ");
gets(filenamein);

printf("nombre del archivo de salida: ");
gets(filenameout);


streamin=readheader(filenamein, headerc, tam);

streamout=writeheader(filenameout, headerc, tam);



if(*tam==0)
{
      printf("No se pudo leer el archivo");
      return 0;
}

Umbral_Absoluto(UA, MAP, UES);



LBC=Limites_banda_critica();


load_tab("C.dat",C);
load_tab("D.dat",D);

for( i=-16; i<48; i++)
      fcos[i+16]=i*pi/64;

for(i=0; i<32; i++)
      for(j=0; j<64; j++)
            MatS[i][j]=cos((2*(i+1)-1)*fcos[j]);

for(i=0;i<1024;i++)
      win[i]=(0.5*(1-cos(2*pi*(i+1)/(1024+1))))*sqrt(2.666666666666666666);

NPS=(double *)malloc(32*sizeof(double));

bsamplesin=samplesin;

bsamplesout=samplesout;
```

```
while(wavread(streamin,samplesin) != 0)
{
    nm=0;
    samplesin=bsamplesin;

    Analisis_FFT(F,samplesin,win);

    Componentes_tonales(F, UA, MAP, LBC, BS, LT, LNT);

    Reduccion(BS, MAP, UA, LT, LNT, BSR, LTR, LNTR);

    Umbrales_enmasc_global(F, UES, LTR, LNTR, UA,MAP,UEG);

    Umbral_enmasc_minimo(UEG,MAP,UEM);

    for(i=0; i<1152; i+=32)
    {
        Filtro_Subbanda(samplesin, C, FIFO , Sb, fcos,MatS);
        for(j=0; j<32; j++)
            S[nm][j]=Sb[j];
        nm++;
        samplesin+=32;
    }

    /*****************************************/

    for(sb=0;sb<32;sb++)
    {
        for(ind=0;ind<36;ind++)
            Stmp2[ind]=S[ind][sb];

        DCT(Stmp2, DCTtmp);

        min[sb]=log10(fabs(DCTtmp[0])+.1);
        max[sb]=log10(fabs(DCTtmp[0])+.1);

        for(ind=0;ind<36;ind++)
        {
            DCTr[ind][sb]=log10(fabs(DCTtmp[ind])+.1);
            DCTc[ind][sb]=DCTtmp[ind];

            if(DCTtmp[ind]>0)
                Tsigno[ind][sb]=1;
            else
                Tsigno[ind][sb]=-1;

            if(DCTr[ind][sb]>max[sb])
                max[sb]=DCTr[ind][sb];
```

```
            if(DCTr[ind][sb]<min[sb])
                min[sb]=DCTr[ind][sb];
        }

    }

    Factores_escala(S, FDE);


    Nivel_presion_sonora(F, FDE, NPS);


    for(i=0;i<32;i++)
        SMR[i]=(NPS[i]-UEM[i])/6;


    for(sb=0;sb<32;sb++)
    {
        Nbits=ceil(SMR[sb]/(20*0.3010));
        rd2=(max[sb]-min[sb])/2;
        offs=rd2+min[sb];
        if(Nbits>0)
        {
            Nbitscc+=(Nbits+1)*36;
            sf=pow(2,Nbits-1)/rd2;
        }
        else
        {
            //Nbitscc+=64;
            sf=0;
            Nbits=0;
        }
        TNbits[sb]=Nbits;
        Tsf[sb]=sf;
        Toff[sb]=offs;
        for(ind=0;ind<36;ind++)
            DCTint[ind][sb]=(int)((DCTr[ind][sb]-offs)*sf);

        Nbitscc+=68;

    }

    for(sb=0;sb<32;sb++)
    {
        sf=Tsf[sb];
        offs=Toff[sb];
        if(sf==0)
            for(ind=0;ind<36;ind++)
                DCTrec[ind][sb]=0;
        else
            for(ind=0;ind<36;ind++)
                DCTrec[ind][sb]=Tsigno[ind][sb]*(pow(10,(double)DCTint[ind][sb]/sf+offs)-.1);

    }
```

```
    for(sb=0;sb<32;sb++)
    {
        for(ind=0;ind<36;ind++)
            DCTtmp[ind]=DCTrec[ind][sb];


        IDCT(DCTtmp,Stmp2);
        for(ind=0;ind<36;ind++)
            S[ind][sb]=Stmp2[ind];
    }

    /*************************************************/
    samplesin=bsamplesin;

    samplesout=bsamplesout;

    for(i=0; i<36; i++)
    {
        for(j=0; j<32; j++)
            Stmp[j]=S[i][j];
        Sintesis_Subbanda(Stmp,D,V,samplesout,Mat);
        samplesout+=32;

    }

    samplesout=bsamplesout;

    fwrite(samplesout,sizeof(short),1152,streamout);

    a+=1152;
    x=tam[0]/2;
    x=a/x*100;

    Nbitssc+=1152*16;

    if(ceil(buf)<x)
    {
        printf("%f\n",x);
        buf=x;
    }
}


fclose(streamin);
fclose(streamout);

printf("\n%f %i %i\n",(double)Nbitssc/(double)Nbitscc,Nbitssc,Nbitscc);
getch();


return 0;
}
```

Función del Analisis Psicuacústico

```
#include <math.h>
#include <stdlib.h>

#define    BLKSIZE    1024
#define    LOGBLKSIZE 10
#define    BLKSIZE_S  256
#define    LOGBLKSIZE_S 8

#define pi 3.14159265358979


struct clf{
        int nb;
        double np;
};

typedef struct clf CLF;


void Umbral_Absoluto(double **UA, double *MAP, double *UES)
{

        int i,j;
        //UMBRAL_ABSOLUTO Mínimo Umbral Auditivo.
        //
        //  [UA,MAP,UES]=UMBRAL_ABSOLUTO
        //  Proporciona en la matriz UA, los valores de:
        //  - Índices de líneas de frecuencia (columna 1)
        //  - Tasa de Banda Crítica (columna 2)
        //  - Umbral Absoluto (columna 3)
        //  para la capa II con una frecuencia de muestreo de 44.1 kHz. En el vector MAP se
        //  proporciona el mapeo entre las líneas de frecuencia y su índice para la matriz UA.
        //  El vector UES contiene solamente el Umbral Absoluto, también llamado Umbral en
        //  Silencio.

        // Almacena en la matriz TablaUA la tabla denominada "Frecuencias, Tasas de Banda
        // Crítica y Umbral Absoluto" para la Capa II a 44.1 kHz, proporcionada por el
        // estándar ISO 11172-3.
        // ---------------------------------------
        // Frecuencia | Tasa de Banda Crítica | Umbral Absoluto
        //  (Hz)                  (z)                 (dB)

        double  TablaUA[130][3] =
        {
                {43.07,     0.425,                  45.05},
                {86.13,     0.850,                  25.87},
                {129.20,            1.273,                  18.70},
                {172.27,            1.694,                  14.85},
                {215.33,            2.112,                  12.41},
                {258.40,            2.525,                  10.72},
                {301.46,            2.934,                  9.47},
                {344.53,            3.337,                  8.50},
                {387.60,            3.733,                  7.73},
                {430.66,            4.124,                  7.10},
```

```
{473.73,  4.507,  6.56},
{516.80,  4.882,  6.11},
{559.86,  5.249,  5.72},
{602.93,  5.608,  5.37},
{646.00,  5.959,  5.07},
{689.06,  6.301,  4.79},
{732.13,  6.634,  4.55},
{775.20,  6.959,  4.32},
{818.26,  7.274,  4.11},
{861.33,  7.581,  3.92},
{904.39,  7.879,  3.74},
{947.46,  8.169,  3.57},
{947.46,  8.450,  3.40},
{1033.59, 8.723,  3.25},
{1076.66, 8.987,  3.10},
{1119.73, 9.244,  2.95},
{1162.79, 9.493,  2.81},
{1205.86, 9.734,  2.67},
{1248.93, 9.968,  2.53},
{1291.99, 10.195, 2.39},
{1335.06, 10.416, 2.25},
{1378.13, 10.629, 2.11},
{1421.19, 10.836, 1.97},
{1464.26, 11.037, 1.83},
{1507.32, 11.232, 1.68},
{1550.39, 11.421, 1.53},
{1593.46, 11.605, 1.38},
{1636.52, 11.783, 1.23},
{1679.59, 11.957, 1.07},
{1722.66, 12.125, 0.90},
{1765.72, 12.289, 0.74},
{1808.79, 12.448, 0.56},
{1851.86, 12.603, 0.39},
{1894.92, 12.753, 0.21},
{1937.99, 12.900, 0.02},
{1981.05, 13.042, -0.17},
{2024.12, 13.181, -0.36},
{2067.19, 13.317, -0.56},
{2153.32, 13.578, -0.96},
{2239.45, 13.826, -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, 15.284, -4.09},
{2928.52, 15.460, -4.37},
{3014.65, 15.631, -4.60},
{3100.78, 15.796, -4.78},
{3186.91, 15.955, -4.91},
{3273.05, 16.110, -4.97},
{3359.18, 16.260, -4.98},
{3445.31, 16.406, -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,	17.205,	-3.54},
{4048.24,	17.327,	-3.19},
{4134.38,	17.447,	-2.82},
{4306.64,	17.680,	-2.06},
{4478.91,	17.905,	-1.32},
{4651.17,	18.121,	-0.64},
{4823.44,	18.331,	-0.04},
{4995.70,	18.534,	0.47},
{5167.97,	18.731,	0.89},
{5340.23,	18.922,	1.23},
{5512.50,	19.108,	1.51},
{5684.77,	19.289,	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,	20.972,	4.02},
{7751.95,	21.099,	4.32},
{7924.22,	21.222,	4.64},
{8096.48,	21.342,	4.98},
{8268.75,	21.457,	5.35},
{8613.28,	21.677,	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,	22.576,	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,	23.415,	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,	24.125,	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,      24.542,      68.00},
        {19982.81,      24.574,      68.00},
    };

    // Convierte la primera columna de la matriz TablaUA de 130 valores de frecuencia
    // a sus equivalentes 130 Indices de líneas de frecuencia.


    for (i=0;i<130;i++)
    {
        UA[i][0]=round(TablaUA[i][0]/44100*1024)-1;
        UA[i][1]=TablaUA[i][1];
        UA[i][2]=TablaUA[i][2];
    }

    // Almacena en el vector MAP el resultado del mapeo entre 512 líneas de frecuencia
    // y sus 130 índices para la matriz UA. Primero, se mapea el valor inicial.
    MAP[0] = 0;

    // Después, se mapean los valores finales.

    for(i=UA[129][0]; i<512; i++)
        MAP[i]=129;

    // Por último, se mapean todos los demás valores.

    for(i=1; i<129; i++)
        for(j=UA[i][0];j<=UA[i+1][0];j++)
            MAP[j] = i;

    // Como la mínima tasa de bits es 96 Kbps, se disminuyen en 12 dB los valores
    // del Umbral Absoluto, almacenados en la tercera columna de la matriz UA.

    for(i=0; i<130; i++)
    {
        UA[i][2]=UA[i][2]-12;
        // Almacena en el vector UES la tercera columna de la matriz UA que contiene
        // los valores de Umbral en Silencio o Umbral Absoluto para la Capa II.
        UES[i]=UA[i][2];
    }


}


double **Limites_banda_critica(void)
{
    //function LBC = Limites_banda_critica
    //LIMITES_BANDA_CRITICA Límites de las Bandas Críticas.
    //
```

```
// LBC=LIMITES_BANDA_CRITICA
// Proporciona, en la matriz LBC, los valores de la tabla "LÍMITES DE LAS BANDAS
// CRÍTICAS" tal y como está dada en el estándar ISO 11172-3; para la Capa II,
// y una frecuencia de muestreo de 44.1 kHz.
//
// Ver también UMBRAL_ABSOLUTO

// En la matriz LBC se almacena la tabla "Límites de las Bandas Críticas" para la
// Capa II a 44.1 kHz, proporcionada por el estándar ISO 11172-3. Las frecuencias
// corresponden al límite superior de cada banda crítica.

// Índice  |      Frecuencia     | Tasa de Banda Crítica
//                                 (Hz)                    (z)

int i;
double TLBC[27][3] = {
        {0,                     43.066,                 0.425},
        {1,                     86.133,                 0.850},
        {2,                     129.199,                1.273},
        {4,                     215.332,                2.112},
        {6,                     301.465,                2.934},
        {9,                     430.664,                4.124},
        {12,                    559.863,                5.249},
        {15,                    689.063,                6.301},
        {18,                    818.262,                7.274},
        {21,                    947.461,                8.169},
        {25,                    1119.727,               9.244},
        {29,                    1291.992,               10.195},
        {34,                    1507.324,               11.232},
        {39,                    1722.656,               12.125},
        {45,                    1981.055,               13.042},
        {50,                    2325.586,               14.062},
        {55,                    2756.250,               15.100},
        {61,                    3273.047,               16.110},
        {68,                    3875.977,               17.079},
        {73,                    4478.906,               17.904},
        {78,                    5340.234,               18.922},
        {84,                    6373.828,               19.963},
        {91,                    7579.688,               20.971},
        {98,                    9302 344,               22.074},
        {104,                   11369.531,              22.984},
        {116,                   15503.906,              24.013},
        {129,                   19982.813,              24.574}
};
double **LBC;
LBC=malloc(27*sizeof(double *));

for(i=0;i<27;i++)
{
        LBC[i]=malloc(3*sizeof(double));
        LBC[i][0]=TLBC[i][0];
        LBC[i][1]=TLBC[i][1];
        LBC[i][2]=TLBC[i][2];
}
return LBC;
}
```

```
/*==========================================================================
-==

   fourierd.c - Don Cross <dcross@intersrv.com>

   http://www.intersrv.com/~dcross/fft.html

   Contains definitions for doing Fourier transforms
   and inverse Fourier transforms.

   This module performs operations on arrays of 'double'.

   Revision history:

1998 September 19 [Don Cross]
   Updated coding standards.
   Improved efficiency of trig calculations.

==========================================================================
=*/

#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#include <fourier.h>
#include <ddcmath.h>

#define CHECKPOINTER(p)  CheckPointer(p,#p)

static void CheckPointer ( void *p, char *name )
{
   if ( p == NULL )
   {
      fprintf ( stderr, "Error in fft_double():  %s == NULL\n", name );
      exit(1);
   }
}


void fft (
   double   RealIn[1024],
   double   *energy )
{

         double   RealOut[1024];
   double   ImagOut[1024];

         double   ImagIn[1024]={0};
         int      InverseTransform=0;
         unsigned NumSamples=1024;
   unsigned NumBits;    /* Number of bits needed to store indices */
   unsigned i, j, k, n;
   unsigned BlockSize, BlockEnd;
```

```
double angle_numerator = 2.0 * DDC_PI;
double tr, ti;     /* temp real, temp imaginary */


CHECKPOINTER ( RealIn );
CHECKPOINTER ( RealOut );
CHECKPOINTER ( ImagOut );

NumBits = NumberOfBitsNeeded ( NumSamples );

/*
**  Do simultaneous data copy and bit-reversal ordering into outputs...
*/

for ( i=0; i < NumSamples; i++ )
{
   j = ReverseBits ( i, NumBits );
   RealOut[j] = RealIn[i];
   ImagOut[j] = (ImagIn == NULL) ? 0.0 : ImagIn[i];
}

/*
**  Do the FFT itself...
*/

BlockEnd = 1;
for ( BlockSize = 2; BlockSize <= NumSamples/2; BlockSize <<= 1 )
{
   double delta_angle = angle_numerator / (double)BlockSize;
   double sm2 = sin ( -2 * delta_angle );
   double sm1 = sin ( -delta_angle );
   double cm2 = cos ( -2 * delta_angle );
   double cm1 = cos ( -delta_angle );
   double w = 2 * cm1;
   double ar[3], ai[3];
   double temp;

   for ( i=0; i < NumSamples/2; i += BlockSize )
   {
      ar[2] = cm2;
      ar[1] = cm1;

      ai[2] = sm2;
      ai[1] = sm1;

      for ( j=i, n=0; n < BlockEnd; j++, n++ )
      {
         ar[0] = w*ar[1] - ar[2];
         ar[2] = ar[1];
         ar[1] = ar[0];

         ai[0] = w*ai[1] - ai[2];
         ai[2] = ai[1];
         ai[1] = ai[0];
```

```
            k = j + BlockEnd;
            tr = ar[0]*RealOut[k] - ai[0]*ImagOut[k];
            ti = ar[0]*ImagOut[k] + ai[0]*RealOut[k];

            RealOut[k] = RealOut[j] - tr;
            ImagOut[k] = ImagOut[j] - ti;

            RealOut[j] += tr;
            ImagOut[j] += ti;
          }
        }

        BlockEnd = BlockSize;
    }

    /*
    **  Need to normalize if inverse transform...
    */


        for(i=0;i<512;i++)
                energy[i]=sqrt(pow(RealOut[i],2)+pow(ImagOut[i],2));
}


void Analisis_FFT(double *F,short *ENT, double win[1024])
{
        double st[1024],h[1024];
        int i;

        for(i=64; i<1088;i++)
                st[i-64]=(double)ENT[i]/32768;



        for(i=0;i<1024;i++)
                st[i]=st[i]*win[i];

        fft(st,F);

        for(i=0; i<512; i++)
        {
                F[i]=20*log10(fabs(F[i])/1024);
                if(F[i]<-200)
                        F[i]=-200;
        }

}


void Componentes_tonales(double *F, double **UA, double *MAP, double **LBC, double *BS, CLF
*LT, CLF *LNT)
{
        int c,k,i,tonal,ind,m,IND, aux, aux2;
```

```
CLF LML[512];
double POT,PESO;

int J[4][23]={    {-2,2,0},
                  {-3,-2,2,3,0},
                  {-6,-5,-4,-3,-2,2,3,4,5,6,0},
                  {-12,-11,-10,-9,-8,-7,-6,-5,-4,-3,-2,2,3,4,5,6,7,8,9,10,11,12,0}};


LML[0].nb=0;
LML[0].np=0;
LT[0].nb=0;
LT[0].np=0;
LNT[0].nb=0;
LNT[0].np=0;

for(i=0;i<512;i++)
        BS[i]=0;

c=1;


for(k=2;k<500;k++)
{

        if(F[k]>F[k-1] && F[k]>=F[k+1])
        {
                LML[c].nb=k;
                LML[c].np=F[k];
                LML[0].nb++;
                c++;
        }
}

c=1;
if(LML[0].nb!=0)
        for(i=1;i<=LML[0].nb;i++)
        {
                k=LML[i].nb;
                tonal=1;
                if(1<k & k<62)
        ind=0;
                else
        if(62<=k & k<126)
                ind=1;
        else
                if(126<=k & k<254)
                        ind=2;
                else
                        if(254<=k & k<=499)
                                ind=3;
                        else
                                tonal=0;

                m=0;
                while(J[ind][m] != 0)
```

```
                tonal=tonal & ((F[k]-F[k+J[ind][m++]])>=7);
                        if (tonal)
                        {
                LT[c].nb=k;
                LT[c].np=10*log10(pow(10,F[k-1]/10)+pow(10,F[k]/10)+pow(10,F[k+1]/10));
                BS[k]=1;
                m=0;
                while(J[ind][m] !=0 )
                        BS[k+J[ind][m++]]=3;
                BS[k+1]=3;
                BS[k-1]=3;
                LT[0].nb++;
                c++;
                        }
                }


        LNT[0].nb=26;
        for (i=0;i<26;i++)
        {
                POT=-200;
                PESO=0;
                aux=UA [(int)LBC[i][0]] [0];

                aux2=UA[(int)LBC[i+1][0]][0]-1;


                for (k=aux; k<=aux2; k++)
                        if(BS[k]==0)
                        {
                POT=10*log10(pow(10,(POT/10))+pow(10,(F[k]/10)));
                PESO+=pow(10,F[k]/10)*(UA[(int)MAP[k]][1]-(i+1));
                BS[k]=3;

                        }
                if(POT<=-200)
                        IND=round(UA[(int)LBC[i][0]][0]+UA[(int)LBC[i+1][0]][0]+1);
                else

        IND=UA[(int)LBC[i][0]][0]+round(PESO/pow(10,POT/10)*(UA[(int)LBC[i+1][0]][0]-
UA[(int)LBC[i][0]][0]));



                if (IND<0)
                        IND=0;
                if (IND>511)
                        IND=511;
                if (BS[IND]==1)
                        IND++;
                LNT[i+1].nb=IND;
                LNT[i+1].np=POT;
                BS[IND]=2;
        }

}
```

```
//REDUCCION Reduce la cantidad de componentes enmascarantes tonales y no-tonales
//  de la señal de audio.
//
//  [BSR,LTR,LNTR]=REDUCCION(LT,LNT,BS,UA,MAP)
//  La matriz LTR lista las componentes tonales y la matriz LNTR lista las
//  componentes no-tonales; ambas después de disminuir la cantidad de
//  componentes enmascarantes. Las componentes que se encuentran por debajo
//  del mínimo umbral auditivo, o que su distancia con respecto a otra componente
//  es menor a medio Bark, son eliminadas. Las matrices LTR y LNTR se componen
//  de dos columnas, la primera proporciona el índice de la línea de frecuencia
//  y la segunda el nivel de presión sonora de esa línea. El vector BSR
//  proporciona las banderas para cada una de las 512 líneas de frecuencia,
//  después de realizar la disminución sobre LT y LNT, tal que:
//  - Componente no examinada = 0.
//  - Componente tonal = 1.
//  - Componente no tonal = 2.
//  - Componente irrelevante = 3.
//
//  UA es la matriz de umbral absoluto y MAP es el vector de mapeo entre las
//  líneas de frecuencia y su índice para la matriz UA; ambos obtenidos con
//  UMBRAL_ABSOLUTO. LBC es la matriz de límites de las bandas críticas
//  obtenida con LIMITES_BANDA_CRITICA. LT y LNT son las matrices de
//  componentes tonales y de componentes no-tonales, y BS es el vector de
//  banderas, todos obtenidos con COMPONENTES_TONALES.
//
//  Ver también UMBRAL_ABSOLUTO, LIMITES_BANDA_CRITICA, COMPONENTES_TONALES.


// a) Caso no-tonal y caso tonal, parte I:
//    Elimina las componentes tonales (filas de la matriz LT) o las componentes
//    no-tonales (filas de la matriz LNT) en las cuales el nivel de presión sonora
//    esté por debajo del mínimo umbral auditivo (tercera columna de la matriz
//    UA); y fija en 3 su valor de bandera en el vector BSR.

// Primero, se analiza el caso no-tonal.


void Reduccion(int *BS, double *MAP, double **UA, CLF *LT, CLF *LNT, int *BSR, CLF *LTR, CLF
*LNTR)
{
        int contLTR=1,contLNTR=1;
        int i,k,sigk,j;


        LTR[0].nb=0;

        LNTR[0].nb=0;

        if (LNT[0].nb!=0)
        {
                for (i=1; i<=26; i++)
                {
```

```
                k = LNT[i].nb;

                if (LNT[i].np < UA[(int)MAP[k]][2])
        BSR[k] = 3;
                else
                {

        LNTR[contLNTR].nb=LNT[i].nb;
        LNTR[contLNTR].np=LNT[i].np;
        contLNTR++;
        LNTR[0].nb++;

                }

        }
}

// Después, se analiza el caso tonal, parte I.
if (LT[0].nb!=0)
{
        for (i=1; i<=LT[0].nb; i++)
        {
                k = LT[i].nb;
                if ((int)LT[i].np < UA[(int)MAP[k]][2])
        BSR[k] = 3;
                else
                {
        LTR[contLTR].nb=LT[i].nb;
        LTR[contLTR].np=LT[i].np;
        contLTR++;
        LTR[0].nb++;
                }
        }
}
// b) Caso tonal, parte II:
//   Elimina dos o más componentes tonales (filas de la matriz LT) cuya distancia
//   respecto a otra componente es menor que medio bark, y fija en 3 su valor de
//   bandera en el vector BSR. Se elimina(n) la(s) componente(s) con menor nivel
//   de presión sonora y se conserva la componente con mayor nivel.

if (LNTR[0].nb!=0)
{
        i = 1;
        while (i <= LTR[0].nb)
        {
                k = LTR[i].nb;
                sigk = LTR[i+1].nb;
                if ( (UA[(int)MAP[sigk]][1] - (double)UA[(int)MAP[k]][1]) < 0.5)
                {
        if (LTR[i].np < LTR[i+1].np)
        {
                // Elimina la componente con índice k.
                for(j=i-1; j<LTR[0].nb; j++)
                {
```

```
                        LTR[j].nb=LTR[j+1].nb;
                        LTR[j].np=LTR[j+1].np;
                    }
                    LTR[0].nb--;
                    BSR[k] = 3;
                }
                else
                {
                    // Elimina la componente con índice k+1.
                    for(j=i+1; j<(LTR[0].nb-1); j++)
                    {
                        LTR[j].nb=LTR[j+1].nb;
                        LTR[j].np=LTR[j+1].np;
                    }
                    LTR[0].nb--;
                    BSR[k] = 3;
                }
                    }
                    i++;
                }
        }

}


void Umbrales_enmasc_global(double *F, double *UES, CLF *LTR, CLF *LNTR, double
**UA,double *MAP,double *UEG)
{
        int k,zi,zj,j,i,dz,avtm,avnm;
        double vf;

        //UMBRALES_ENMASC_INDIVIDUAL Calcula los umbrales de enmascaramiento
individual.
        //
        // [UET,UENT]=UMBRALES_ENMASC_INDIVIDUAL(F,LTR,LNTR,UA,MAP)
        // Calcula el efecto enmascarante de las componentes tonales (matriz UET) y
        // de las componentes no tonales (matriz UENT) sobre las líneas de frecuencia
        // vecinas. Para esto, el nivel de presión sonora de cada componente enmascarante
        // es sumado con su índice de enmascaramiento y con su función de enmascaramiento.
        //
        // F es el vector de densidad espectral de potencia normalizada, obtenido con
        // ANALISIS_FFT. UA es la matriz de umbral absoluto y MAP es el vector de mapeo
        // entre las líneas de frecuencia y su índice para la matriz UA; ambos obtenidos
        // con UMBRAL_ABSOLUTO. La matriz LTR es la lista reducida de las componentes
        // tonales y la matriz LNTR es la lista reducida de las componentes no-tonales,
        // ambas obtenidas con REDUCCION.
        //
        // Ver también ANALISIS_FFT, COMPONENTES_TONALES, UMBRAL ABSOLUTO,
REDUCCION.
        // Los umbrales de enmascaramiento individual para las componentes tonales
        // y no tonales se fijan en -INF, ya que la función de enmascaramiento tiene
        // atenuación infinita más allá de -3 y de +8 Barks, o sea, la componente no
        // tiene efecto enmascarante sobre frecuencias más allá de aquellos rangos.

        // Sólo un subconjunto de las muestras son consideradas para el futuro cálculo
```

```
// del umbral de enmascaramiento global. El número de estas muestras dep}e
// de la tasa de muestreo y de la capa de codificación. Toda la información
// necesaria está en la matriz UA, la cual contiene las frecuencias, las tasas
// de banda crítica y el umbral absoluto.

for(i=0;i<130;i++)
        UEG[i]=pow(10,UES[i]/10);


if (((int)LTR[0].nb != 0) && ((int)LNTR[0].nb != 0))
        for (i=0; i<130; i++)
        {

                zi = UA[i][1];                  // Tasa de banda crítica de la frecuencia
considerada.
                if (LTR[0].nb != 0)
                {
        // Para las componentes tonales.
        for (k=1; k<=LTR[0].nb; k++)
        {
                j = LTR[k].nb;
                zj = UA[(int)MAP[j]][1]; // Tasa de banda crítica de la componente
enmascarante.
                dz = zi-zj; // Distancia en Barks hasta la componente enmascarante.

                // Como la componente tonal tiene atenuación infinita más allá de -3 y
                // de +8 Barks, entonces los cálculos sólo se realizan para el rango
                // dz = {-3...+8}.
                if (dz>=-3 & dz<8)
                {
                        // Índice de Enmascaramiento.
                        avtm = -1.525-0.275*zj-4.5;

                        // Función de Enmascaramiento.
                        if (-3<=dz & dz<-1)
                                vf = 17*(dz+1)-(0.4*F[j]+6);
                        else if (-1<=dz & dz<0)
                                vf = (0.4*F[j]+6)*dz;
                        else if (0<=dz & dz<1)
                                vf = -17*dz;
                        else if (1<=dz & dz<8)
                                vf = -(dz-1)*(17-0.15*F[j])-17;

                        // Umbral de enmascaramiento, componentes tonales.
                        UEG[i]=UEG[i] + pow(10,F[j]+avtm+vf);
                }
        }
                }

                // Para las componentes no tonales.
                if (LNTR[0].nb != 0)
                {
        for (k=1; k<=LTR[0].nb; k++)
        {
                j = LNTR[k].nb;
```

```
                    zj = UA[(int)MAP[j]][1]; // Tasa de banda crítica de la componente
enmascarante.
                    dz = zi-zj; // Distancia en Barks hasta la componente enmascarante.
                    // Como la componente no-tonal tiene atenuación infinita más allá
                    // de -3 y de +8 Barks, entonces los cálculos sólo se realizan para
                    // el rango dz = {-3...+8}.
                    if (dz>=-3 & dz<8)
                    {
                            // Índice de Enmascaramiento.
                            avnm = -1.525-0.175*zj-0.5;

                            // Función de Enmascaramiento.
                            if (-3<=dz & dz<-1)
                                    vf = 17*(dz+1)-(0.4*F[j]+6);
                            else if (-1<=dz & dz<0)
                                    vf = (0.4*F[j]+6)*dz;
                            else if (0<=dz & dz<1)
                                    vf = -17*dz;
                            else if (1<=dz & dz<8)
                                    vf = -(dz-1)*(17-0.15*F[j])-17;

                            // Umbral de enmascaramiento, componente no-tonal.
                            UEG[i]+=pow(10,F[j]+avtm+vf);
                    }
            }
                    }

            }
            for(i=0;i<130;i++)
                    UEG[i]=10*log10(UEG[i]);
    }

void Umbral_enmasc_minimo(double *UEG,double MAP[512],double *UEM)
{
    int n,i;
    double min;
    //function UEM = Umbral_enmasc_minimo(UEG,MAP)
    //UMBRAL_ENMASC_MINIMO Calcula el umbral de enmascaramiento mínimo.
    //
    //  UEM=UMBRAL_ENMASC_MINIMO(UEG,MAP)
    //  Retorna en el vector UEM el valor mínimo del umbral de enmascaramiento
    //  global para cada una de las 32 subbandas.
    //
    //  El vector UEG es el umbral de enmascaramiento global obtenido con
    //  UMBRAL_ENMASC_GLOBAL. El vector MAP es el mapeo entre las líneas de
    //  frecuencia y su índice para la matriz UA en la función UMBRAL_ABSOLUTO.
    //
    //  Ver también UMBRAL_ENMASC_GLOBAL, UMBRAL_ABSOLUTO.
    // Extrae del umbral de enmascaramiento global (vector UEG), su valor mínimo
    // en cada subbanda (vector UEM).
    for (n=0; n<32; n++)
    {
            min=UEG[(int)MAP[n*16]];
            for(i=MAP[n*16]; i<=MAP[n*16+15];i++)
                if(UEG[i]<min)
            min=UEG[i];
```

```
                UEM[n]=min;
        }
}

void Factores_escala(double S[36][32], double *FDE)
{
        //FACTORES_ESCALA Calcula los factores de escala
        //
        //  FDE=FACTORES_ESCALA(S)
        //  Obtiene los factores de escala (matriz FDE) para cada una de las subbandas en
        //  la capa II. S es la matriz de 36x32 muestras subbanda obtenidas con
        //  FILTRO_SUBBANDA.
        //
        //  Se determina el máximo valor absoluto para grupos de 12 muestras subbanda y
        //  se compara con los valores de la tabla de factores de escala; eligiendo su
        //  valor más cercano por exceso como el factor de escala asociado a cada grupo.
        //
        //  Ver también FILTRO_SUBBANDA
        // Tabla de Factores de Escala proporcionada por el estándar ISO 11172-3.

        int a,i,ind,j;

        double MAXS,Xk;

        double TFE[64]={  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};
```

```
        for (a=0; a<36; a+=12)
        {
                for (i=0; i<32; i++)
                {
                        // Determina el máximo valor absoluto de 3 grupos (Indice a) de 12
muestras
                        // subbanda para cada una de las 32 subbandas (Indice i) y lo almacena en
                        // MAXS.

                        MAXS=fabs(S[a][i]);
                        for(ind=a; ind<=a+11; ind++)
                if(fabs(S[ind][i])>MAXS)
                        MAXS=fabs(S[ind][i]);

                        // Realiza un barrido de la tabla de factores de escala hasta encontrar en
                        // ella un valor mayor que el valor de MAXS.
                        j = 0;
                        while (j<64 & MAXS>TFE[63-j])
                j++;

                        // Crea el vector MAXT, que es compuesto de cada factor de escala elegido
de
                        // la tabla (para cada i) en el paso anterior.
                        //MAXT[i] = TFE[63-j];

                        if(FDE[i]<TFE[63-j])
                FDE[i]=TFE[63-j];

                }
                // Almacena en la matriz FDE los valores que va tomando MAXT (para cada a);
                // resultando una matriz de 3x32, o sea, 3 factores de escala elegidos para
                // cada subbanda.
        }
}

void Nivel_presion_sonora(double *F, double *FDE,double *NPS)
{
        int k,i=0,ind;
        double XKl[512],Xk;

        for(k=0; k<512; k+=16)
        {
                Xk=F[k];
                for(ind=k;ind<=k+15;ind++)
                        if(Xk<F[ind])
                        Xk=F[ind];
                XKl[i++]=Xk;
        }
        for(i=0; i<32;i++)
        {
                NPS[i]=XKl[i];
                if( NPS[i]<( 20*log10(FDE[i]*32768)-10))
                        NPS[i]= 20*log10(FDE[i]*32768)-10;
        }
}
```

Sistema 2

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>

#define pi 3.14159265358979
#define BIT 10000


struct {
unsigned int dato;
unsigned int tamano;
int entrada;
}datos[62];


FILE *readheader(char filename[13],char *headerc, unsigned long *tam)
{
        FILE *stream;

        stream=fopen(filename,"r+b");

        if(fread(headerc,sizeof(char),40,stream)==0)
                return NULL;
        else
        {
                fread(tam,sizeof(long),1,stream);
                return stream;
        }
}


int round(double num)
{
        if(num>0)
                if((num-floor(num))<0.5)
                        return (int)floor(num);
                else
                        return (int)ceil(num);
        else
                if(num<0)
                        if((ceil(num)-num)<0.5)
                return (int)ceil(num);
                        else
                return (int)floor(num);
                else
                        return 0;
}
```

```
FILE *writeheader(char filename[13], char *headerc,unsigned long *tam)
{
        FILE *stream;

        stream=fopen(filename,"w+b");

        fwrite(headerc,sizeof(char),40,stream);
        fwrite(tam,sizeof(long),1,stream);

        return stream;
}


int wavread(FILE *stream, short *samples)
{
        int numread, i;

        numread=fread(samples,sizeof(short),1152,stream);

        if (numread==0 & numread<1153)
        {
                for(i=numread; i<1152; i++)
                samples[i]=0;
        }

        return numread;
}


void Filtro_Subbanda(short *muestras, double *C, double *FIFO , double *S)
{
        int i,j;
        double Z[512], Y[64], fcos[64];

        for(i=511; i>=32; i--)
                FIFO[i]=FIFO[i-32];

        for(i=31; i>=0; i--)
                FIFO[i]=((double)muestras[(31-i)])/32800;

        for(i=0; i<512; i++)
                Z[i]=C[i]*FIFO[i];

        for(i=0; i<64; i++)
        {
                Y[i]=0;
                for(j=i; j<512; j+=64)
                        Y[i]+=Z[j];
        }


        for( i=-16; i<48; i++)
                fcos[i+16]=i*pi/64;

        for(i=0; i<32; i++)
```

```
        {
                S[i]=0;
                for(j=0; j<64; j++)
                        S[i]+=cos((2*(i+1)-1)*fcos[j])*Y[j];
        }

        return S;
}

void Sintesis_Subbanda(double *Ssb, double *D, double *V, short *Spcmi)
{
        int i,j,k,l;
        double U[512], Spcm[32];
        double sum,max;

        for(i=1023; i>=64;i--)
                V[i]=V[i-64];


        for(i=0; i<64;i++)
        {
                sum=0;
                for(k=0;k<32;k++)
                        sum+=Ssb[k]*cos(pi*(2*k+1)*(16+i)/64);
                V[i]=(double)sum;
        }

        for(k=0; k<8; k++)
                for(l=0; l<32; l++)
                {
                        U[64*k+l]=V[128*k+l];
                        U[64*k+32+l]=V[128*k+96+l];
                }

        for(i=0, i<512; i++)
                U[i]=U[i]*D[i];



        max=.8;
        for(j=0; j<32;j++)
        {
                Spcm[j]=0;

                for(i=0; i<16;i++)
                {
                        Spcm[j]+=U[j+32*i];
                }
                if(fabs(Spcm[j])>max && fabs(Spcm[j])>.95)
                        max=fabs(Spcm[j]);


        }
```

```
        for(i=0; i<32;i++)
        {
                Spcmi[i]=(short)(Spcm[i]/(max+.2)*32768);
        }

}
void load_tab(char filename [13],double *TAB)
{
        FILE *stream;

        stream=fopen(filename,"r+b");

        fread(TAB,sizeof(double),512,stream);

        fclose(stream);

}
void DCT(double *u, double *U)
{

        int m,k,M=36;


        U[0]=0;
        for(m=0;m<M;m++)
                U[0]+=(1.666666666666667e-001)*u[m];

        for(k=1;k<M;k++)
        {
                U[k]=0;
                for(m=0;m<M;m++)
                        U[k]+=(2.357022603955158e-001)*u[m]*cos(pi*(2*m+1)*k/(2*M));
        }
}


void IDCT(double *U,double *u)
{

        int m,k,M=36;


        for(m=0;m<M;m++)
        {

                u[m]=(1.666666666666667e-001)*U[0];
                for(k=1;k<M;k++)
                        u[m]+=(2.357022603955158e-001)*U[k]*cos(pi*(2*m+1)*k/(2*M));
        }
}


int main()
```

```
{
        char filenamein[13], filenameout[13],filenamecom[13], *headerc;
        FILE *streamin,*streamout,*streamb,*streamsf;
        double *DCTtmp,*Sb,S[36][32],*FIFO;
        double *Stmp,*Stmp2,*V,DCTr[36][32];
        short *samplesin, *samplesout;
        short *bsamplesin, *bsamplesout;
        double *C,*D,prom[32];
        int j,i,nm,sb,ind,flag=0;
        unsigned long tam[1],a=0;
        double x,buf=0;
        unsigned int bitbuff[360],conttam=0;

        float rd2,max,min,sf,offs /**buffsf*/;

        int numblock, DCTdec[36][32], /*numbuffsf*/ flag1;



        int contceros=0,elim=0,elim2=0;

        int temp;
        Sb=(double *)malloc(32*sizeof(double));
        samplesin=(short *)malloc(1152*sizeof(short));
        samplesout=(short *)malloc(1152*sizeof(short));

        DCTtmp=(double *)malloc(36*sizeof(double));

        Stmp2=(double *)malloc(36*sizeof(double));
        Stmp=(double *)malloc(32*sizeof(double));
        FIFO=(double *)malloc(512*sizeof(double));
        V=(double *)malloc(1024*sizeof(double));
        C=(double *)malloc(512*sizeof(double));
        D=(double *)malloc(512*sizeof(double));

        //buffsf=(float *)malloc(1024*sizeof(float));

/*****************************************************************/

            datos[1-1].dato=0;
                datos[1-1].tamano=1;
                datos[1-1].entrada=0;

                datos[2-1].dato=14;
                datos[2-1].tamano=4;
                datos[2-1].entrada=31;

      datos[3-1].dato=15;
                datos[3-1].tamano=4;
                datos[3-1].entrada=-31;

                datos[4-1].dato=36;
                datos[4-1].tamano=4;
                datos[4-1].entrada=-2;
```

```
datos[5-1].dato=37;
            datos[5-1].tamano=6;
            datos[5-1].entrada=1;

            datos[6-1].dato=40;
            datos[6-1].tamano=6;
            datos[6-1].entrada=-3;

datos[7-1].dato=41;
            datos[7-1].tamano=6;
            datos[7-1].entrada=-1;

            datos[8-1].dato=42;
            datos[8-1].tamano=6;
            datos[8-1].entrada=2;

datos[9-1].dato=44;
            datos[9-1].tamano=6;
            datos[9-1].entrada=-5;

            datos[10-1].dato=45;
            datos[10-1].tamano=6;
            datos[10-1].entrada=3;

datos[11-1].dato=48;
            datos[11-1].tamano=6;
            datos[11-1].entrada=-4;

            datos[12-1].dato=49;
            datos[12-1].tamano=6;
            datos[12-1].entrada=-7;

datos[13-1].dato=51;
            datos[13-1].tamano=6;
            datos[13-1].entrada=-6;

            datos[14-1].dato=52;
            datos[14-1].tamano=6;
            datos[14-1].entrada=6;

datos[15-1].dato=54;
            datos[15-1].tamano=6;
            datos[15-1].entrada=5;

            datos[16-1].dato=55;
            datos[16-1].tamano=6;
            datos[16-1].entrada=4;

datos[17-1].dato=65;
            datos[17-1].tamano=7;
            datos[17-1].entrada=7;

            datos[18-1].dato=68;
            datos[18-1].tamano=7;
            datos[18-1].entrada=-30;
```

```
datos[19-1].dato=69;
            datos[19-1].tamano=7;
            datos[19-1].entrada=-9;

            datos[20-1].dato=70;
            datos[20-1].tamano=7;
            datos[20-1].entrada=8;

            datos[21-1].dato=71;
            datos[21-1].tamano=7;
            datos[21-1].entrada=30;

            datos[22-1].dato=77;
            datos[22-1].tamano=7;
            datos[22-1].entrada=-10;

datos[23-1].dato=78;
            datos[23-1].tamano=7;
            datos[23-1].entrada=9;

            datos[24-1].dato=87;
            datos[24-1].tamano=7;
            datos[24-1].entrada=-8;

datos[25-1].dato=93;
            datos[25-1].tamano=7;
            datos[25-1].entrada=-12;

            datos[26-1].dato=100;
            datos[26-1].tamano=7;
            datos[26-1].entrada=10;

datos[27-1].dato=101;
            datos[27-1].tamano=7;
            datos[27-1].entrada=11;

            datos[28-1].dato=129;
            datos[28-1].tamano=8;
            datos[28-1].entrada=12;

datos[29-1].dato=134;
            datos[29-1].tamano=8;
            datos[29-1].entrada=-13;

            datos[30-1].dato=152;
            datos[30-1].tamano=8;
            datos[30-1].entrada=-11;

            datos[31-1].dato=158;
            datos[31-1].tamano=8;
            datos[31-1].entrada=-17;

            datos[32-1].dato=159;
            datos[32-1].tamano=8;
            datos[32-1].entrada=13;
```

```
datos[33-1].dato=173;
            datos[33-1].tamano=8;
            datos[33-1].entrada=19;

            datos[34-1].dato=184;
            datos[34-1].tamano=8;
            datos[34-1].entrada=21;

datos[35-1].dato=188;
            datos[35-1].tamano=8;
            datos[35-1].entrada=-16;

            datos[36-1].dato=189;
            datos[36-1].tamano=8;
            datos[36-1].entrada=-15;

datos[37-1].dato=190;
            datos[37-1].tamano=8;
            datos[37-1].entrada=14;

            datos[38-1].dato=213;
            datos[38-1].tamano=8;
            datos[38-1].entrada=15;

datos[39-1].dato=214;
            datos[39-1].tamano=8;
            datos[39-1].entrada=16;

            datos[40-1].dato=256;
            datos[40-1].tamano=9;
            datos[40-1].entrada=-19;

            datos[41-1].dato=257;
            datos[41-1].tamano=9;
            datos[41-1].entrada=-24;

            datos[42-1].dato=264;
            datos[42-1].tamano=9;
            datos[42-1].entrada=-14;

datos[43-1].dato=265;
            datos[43-1].tamano=9;
            datos[43-1].entrada=20;

            datos[44-1].dato=266;
            datos[44-1].tamano=9;
            datos[44-1].entrada=23;

datos[45-1].dato=267;
            datos[45-1].tamano=9;
            datos[45-1].entrada=28;

            datos[46-1].dato=271;
            datos[46-1].tamano=9;
            datos[46-1].entrada=-25;
```

```
datos[47-1].dato=306;
        datos[47-1].tamano=9;
        datos[47-1].entrada=-18;

        datos[48-1].dato=307;
        datos[48-1].tamano=9;
        datos[48-1].entrada=-23;

datos[49-1].dato=344;
        datos[49-1].tamano=9;
        datos[49-1].entrada=-21;

        datos[50-1].dato=345;
        datos[50-1].tamano=9;
        datos[50-1].entrada=18;

        datos[51-1].dato=370;
        datos[51-1].tamano=9;
        datos[51-1].entrada=22;

        datos[52-1].dato=371;
        datos[52-1].tamano=9;
        datos[52-1].entrada=-26;

datos[53-1].dato=382;
        datos[53-1].tamano=9;
        datos[53-1].entrada=-22;

        datos[54-1].dato=383;
        datos[54-1].tamano=9;
        datos[54-1].entrada=-20;

datos[55-1].dato=424;
        datos[55-1].tamano=9;
        datos[55-1].entrada=25;

        datos[56-1].dato=540;
        datos[56-1].tamano=10;
        datos[56-1].entrada=-27;

datos[57-1].dato=541;
        datos[57-1].tamano=10;
        datos[57-1].entrada=24;

        datos[58-1].dato=850;
        datos[58-1].tamano=10;
        datos[58-1].entrada=17;

datos[59-1].dato=851;
        datos[59-1].tamano=10;
        datos[59-1].entrada=-29;

        datos[60-1].dato=860;
        datos[60-1].tamano=10;
        datos[60-1].entrada=-28;
```

```
                datos[61-1].dato=861;
                datos[61-1].tamano=10;
                datos[61-1].entrada=26;

                datos[62-1].dato=862;
                datos[62-1].tamano=10;
                datos[62-1].entrada=27;

        datos[63-1].dato=863;
                datos[63-1].tamano=10;
                datos[63-1].entrada=29;

/*****************************************************************/

        for(i=0; i<512; i++)
        {
                FIFO[i]=0;
                V[i]=0;
        }
        for(i=512;i<1024;i++)
                V[i]=0;


        headerc=(char *)malloc(40*sizeof(char));

        printf("nombre del archivo de entrada(.wav): ");
        gets(filenamein);

        printf("nombre del archivo de salida(.wav): ");
        gets(filenameout);

    printf("nombre del archivo compreso(.crc): ");
        gets(filenamecom);


        streamin=readheader(filenamein, headerc, tam);

        streamout=writeheader(filenameout, headerc, tam);

        streamb=writeheader(filenamecom, headerc, tam);


        //streamsf=fopen("outsf.crc","w+b");


//      numbuffsf=tam[0]/(2*1152);

//      buffsf=malloc(numbuffsf*sizeof(float));

        if(*tam==0)
        {
                printf("No se pudo leer el archivo");
                return 0;
```

```
        }

        load_tab("C.dat",C);
        load_tab("D.dat",D);

        bsamplesin=samplesin;

        bsamplesout=samplesout;

        numblock=0;
        while(wavread(streamin,samplesin) != 0)
        {

                nm=0;
                samplesin=bsamplesin;
                for(i=0; i<1152; i+=32)
                {
                        Filtro_Subbanda(samplesin, C, FIFO,Sb);
                        for(j=0; j<32; j++)
                S[nm][j]=Sb[j];
                        nm++;
                        samplesin+=32;
                }
/*********************************************/

                for(sb=0;sb<32;sb++)
                {
                        for(ind=0;ind<36;ind++)
                Stmp2[ind]=S[ind][sb];

                        DCT(Stmp2,DCTtmp);

                        prom[sb]=0;

                        for(ind=0;ind<36;ind++) {
                DCTr[ind][sb]=DCTtmp[ind];
                        //      printf("%e\n",DCTtmp[ind]);
                           prom[sb]+=fabs(DCTtmp[ind]/36);

                        }

                }
//   getch();
                min=DCTr[0][0];
                max=DCTr[0][0];
        /*      printf("%e\n",min);
                printf("%e\n",max);
                printf("%e\n",DCTr[0][0]);
                getch();*/
                /*********************************************/
                for (sb=0;sb<32;sb++)
                        for (ind=0;ind<36;ind++)
                        {


                if (sb <1 || (sb >13 && sb <32)){
```

```
                if ((DCTr[ind][sb]< (prom[sb]*0.5)) && (DCTr[ind][sb] > (-1*prom[sb]*0.5))){

                        DCTr[ind][sb]=0;
                        elim++;
                }
                else if (DCTr[ind][sb]< 0.05 && DCTr[ind][sb] > -0.05){

                        DCTr[ind][sb]=0;
                        elim2++;
                }

        }

                  if (sb >1 && sb <6){

                if ((DCTr[ind][sb]< (prom[sb]*2)) && (DCTr[ind][sb] > (-1*prom[sb]*2))){

                        DCTr[ind][sb]=0;
                elim++;
                }
           else if (DCTr[ind][sb]< 0.001 && DCTr[ind][sb] > -0.001){

                        DCTr[ind][sb]=0;

                        elim2++;
                }
        }


                  if (sb >7 && sb <13){

                if ((DCTr[ind][sb]< (prom[sb]*0.8)) && (DCTr[ind][sb] > (-1*prom[sb]*0.8))){

                        DCTr[ind][sb]=0;
                elim++;
                }
           else if (DCTr[ind][sb]< 0.005 && DCTr[ind][sb] > -0.005){

                        DCTr[ind][sb]=0;

                        elim2++;
                }
        }


                  if(min>DCTr[ind][sb])
           min=DCTr[ind][sb];
                  if(max<DCTr[ind][sb])
           max=DCTr[ind][sb];
                  }
/*****************************************/

        rd2=(max-min)/2;
        if(rd2==0)
```

```
                flag1=1;
        else
                flag1=0;
        offs=rd2+min;
        if (flag1==0)
                sf=32/rd2;
        for (sb=0;sb<32;sb++){
                for (ind=0;ind<36;ind++){
                        if (flag1==0)
                        {
                DCTdec[ind][sb]=(int)(sf*(DCTr[ind][sb]-offs));
                        //      printf("%f %f %f ",min,max,rd2);
                if(DCTdec[ind][sb]==32)
                        DCTdec[ind][sb]=31;
                if(DCTdec[ind][sb]==-32)
                        DCTdec[ind][sb]=-31;
                        }
                        else
                        {
                DCTdec[ind][sb]=0;
                sf=0;
                }
        //      printf("%i ",DCTdec[ind][sb]);
                }

        }
//      buffsf[numblock++]=offs;
//      buffsf[numblock++]=sf;


                        //getch();
/*****************************************/
                //printf("elim:%i  j:%i \n",elim,j);


                //getch();



                for(sb=0;sb<32;sb++)
                {
                        for(ind=0;ind<36;ind++)
                        {
                DCTtmp[ind]=DCTr[ind][sb];
                /*****/
                if(DCTr[ind][sb]==0)
                        contceros++;



                        }

                        IDCT(DCTtmp,Stmp2);
                        for(ind=0;ind<36;ind++)
                S[ind][sb]=Stmp2[ind];
                }
```

```
                conttam=0;

                for (sb=0;sb<32;sb++){
                        for(ind=0;ind<36;ind++){
                i=0;
                //printf("%i\n",DCTdec[ind][sb]);
                while (DCTdec[ind][sb]!=datos[i].entrada && i<64){
                        i++;
                 //  printf("%i,%i \n",datos[i].entrada,DCTdec[ind][sb]);
                }

                if (i>62){
                        printf("No encontró el dato:%i,%i\n",DCTdec[ind][sb],i);getch();}

                if (flag==1)

                        bitbuff[360]=bitbuff[360] << datos[i].tamano;


        conttam+=datos[i].tamano;
                flag=1;

                //printf("No e pudo leer el archivo3");

                bitbuff[360]=bitbuff[360]|datos[i].dato;

                //printf("%i,%i \n",sb,ind);
                        }
                }

                conttam=(int)ceil(conttam/16);

                fwrite(bitbuff,sizeof(short),conttam,streamb);




/*************************************************/
                samplesin=bsamplesin;

                samplesout=bsamplesout;

                for(i=0; i<36; i++)
                {
                        for(j=0; j<32; j++)
                Stmp[j]=S[i][j];
                        Sintesis_Subbanda(Stmp,D,V,samplesout);
                        samplesout+=32;
```

```
                }

                samplesout=bsamplesout;

                fwrite(samplesout,sizeof(short),1152,streamout);

                a+=1152;
                x=tam[0]/2;
                x=a/x*100;

                if(ceil(buf)<x)
                {
                        printf("%f\n",x);
                        buf=x;
                }
                //printf("No se pudo leer el archivo5");
        }


        x=(tam[0]*.5)/((tam[0]*.5-(elim+elim2)));

        printf("%f,%i,%i,%i",x,elim,elim2, elim+elim2);

getch();

//      fwrite(buffsf,sizeof(float),numbuffsf,streamsf);
//      fclose(streamsf);

        fclose(streamin);
        fclose(streamout);
        fclose(streamb);

        printf("\nHOLA\n");

        x=(tam[0]*.5)/((tam[0]*.5-(elim+elim2)));

        printf("%f,%i,%i,%i",x,elim,elim2, elim+elim2);

getch();

/*      free(samplesin);
        free(samplesout);

        free(headerc);
        free(tam);

        free(C);
        free(D);*/

        return 0;
}
```

Apéndice B

Tablas

Tabla C

C[0]= 0.000000000 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

Tabla D

D[0]= 0.000000000 D[1]=-0.000015259 D[2]=-0.000015259 D[3]=-0.000015259
D[4]=-0.000015259 D[5]=-0.000015259 D[6]=-0.000015259 D[7]=-0.000030518
D[8]=-0.000030518 D[9]=-0.000030518 D[10]=-0.000030518 D[11]=-0.000045776
D[12]=-0.000045776 D[13]=-0.000061035 D[14]=-0.000061035 D[15]=-0.000076294
D[16]=-0.000076294 D[17]=-0.000091553 D[18]=-0.000106812 D[19]=-0.000106812

D[20]=-0.000122070 D[21]=-0.000137329 D[22]=-0.000152588 D[23]=-0.000167847
D[24]=-0.000198364 D[25]=-0.000213623 D[26]=-0.000244141 D[27]=-0.000259399
D[28]=-0.000289917 D[29]=-0.000320435 D[30]=-0.000366211 D[31]=-0.000396729
D[32]=-0.000442505 D[33]=-0.000473022 D[34]=-0.000534058 D[35]=-0.000579834
D[36]=-0.000625610 D[37]=-0.000686646 D[38]=-0.000747681 D[39]=-0.000808716
D[40]=-0.000885010 D[41]=-0.000961304 D[42]=-0.001037598 D[43]=-0.001113892
D[44]=-0.001205444 D[45]=-0.001296997 D[46]=-0.001388550 D[47]=-0.001480103
D[48]=-0.001586914 D[49]=-0.001693726 D[50]=-0.001785278 D[51]=-0.001907349
D[52]=-0.002014160 D[53]=-0.002120972 D[54]=-0.002243042 D[55]=-0.002349854
D[56]=-0.002456665 D[57]=-0.002578735 D[58]=-0.002685547 D[59]=-0.002792358
D[60]=-0.002899170 D[61]=-0.002990723 D[62]=-0.003082275 D[63]=-0.003173828
D[64]= 0.003250122 D[65]= 0.003326416 D[66]= 0.003387451 D[67]= 0.003433228
D[68]= 0.003463745 D[69]= 0.003479004 D[70]= 0.003479004 D[71]= 0.003463745
D[72]= 0.003417969 D[73]= 0.003372192 D[74]= 0.003280640 D[75]= 0.003173828
D[76]= 0.003051758 D[77]= 0.002883911 D[78]= 0.002700806 D[79]= 0.002487183
D[80]= 0.002227783 D[81]= 0.001937866 D[82]= 0.001617432 D[83]= 0.001266479
D[84]= 0.000869751 D[85]= 0.000442505 D[86]=-0.000030518 D[87]=-0.000549316
D[88]=-0.001098633 D[89]=-0.001693726 D[90]=-0.002334595 D[91]=-0.003005981
D[92]=-0.003723145 D[93]=-0.004486084 D[94]=-0.005294800 D[95]=-0.006118774
D[96]=-0.007003784 D[97]=-0.007919312 D[98]=-0.008865356 D[99]=-0.009841919
D[100]=-0.010848999 D[101]=-0.011886597 D[102]=-0.012939453 D[103]=-0.014022827
D[104]=-0.015121460 D[105]=-0.016235352 D[106]=-0.017349243 D[107]=-0.018463135
D[108]=-0.019577026 D[109]=-0.020690918 D[110]=-0.021789551 D[111]=-0.022857666
D[112]=-0.023910522 D[113]=-0.024932861 D[114]=-0.025909424 D[115]=-0.026840210
D[116]=-0.027725220 D[117]=-0.028533936 D[118]=-0.029281616 D[119]=-0.029937744
D[120]=-0.030532837 D[121]=-0.031005859 D[122]=-0.031387329 D[123]=-0.031661987
D[124]=-0.031814575 D[125]=-0.031845093 D[126]=-0.031738281 D[127]=-0.031478882
D[128]= 0.031082153 D[129]= 0.030517578 D[130]= 0.029785156 D[131]= 0.028884888

D[132]= 0.027801514 D[133]= 0.026535034 D[134]= 0.025085449 D[135]= 0.023422241
D[136]= 0.021575928 D[137]= 0.019531250 D[138]= 0.017257690 D[139]= 0.014801025
D[140]= 0.012115479 D[141]= 0.009231567 D[142]= 0.006134033 D[143]= 0.002822876
D[144]=-0.000686646 D[145]=-0.004394531 D[146]=-0.008316040 D[147]=-0.012420654
D[148]=-0.016708374 D[149]=-0.021179199 D[150]=-0.025817871 D[151]=-0.030609131
D[152]=-0.035552979 D[153]=-0.040634155 D[154]=-0.045837402 D[155]=-0.051132202
D[156]=-0.056533813 D[157]=-0.061996460 D[158]=-0.067520142 D[159]=-0.073059082
D[160]=-0.078628540 D[161]=-0.084182739 D[162]=-0.089706421 D[163]=-0.095169067
D[164]=-0.100540161 D[165]=-0.105819702 D[166]=-0.110946655 D[167]=-0.115921021
D[168]=-0.120697021 D[169]=-0.125259399 D[170]=-0.129562378 D[171]=-0.133590698
D[172]=-0.137298584 D[173]=-0.140670776 D[174]=-0.143676758 D[175]=-0.146255493
D[176]=-0.148422241 D[177]=-0.150115967 D[178]=-0.151306152 D[179]=-0.151962280
D[180]=-0.152069092 D[181]=-0.151596069 D[182]=-0.150497437 D[183]=-0.148773193
D[184]=-0.146362305 D[185]=-0.143264771 D[186]=-0.139450073 D[187]=-0.134887695
D[188]=-0.129577637 D[189]=-0.123474121 D[190]=-0.116577148 D[191]=-0.108856201
D[192]= 0.100311279 D[193]= 0.090927124 D[194]= 0.080688477 D[195]= 0.069595337
D[196]= 0.057617187 D[197]= 0.044784546 D[198]= 0.031082153 D[199]= 0.016510010
D[200]= 0.001068115 D[201]=-0.015228271 D[202]=-0.032379150 D[203]=-0.050354004
D[204]=-0.069168091 D[205]=-0.088775635 D[206]=-0.109161377 D[207]=-0.130310059
D[208]=-0.152206421 D[209]=-0.174789429 D[210]=-0.198059082 D[211]=-0.221984863
D[212]=-0.246505737 D[213]=-0.271591187 D[214]=-0.297210693 D[215]=-0.323318481
D[216]=-0.349868774 D[217]=-0.376800537 D[218]=-0.404083252 D[219]=-0.431655884
D[220]=-0.459472656 D[221]=-0.487472534 D[222]=-0.515609741 D[223]=-0.543823242
D[224]=-0.572036743 D[225]=-0.600219727 D[226]=-0.628295898 D[227]=-0.656219482
D[228]=-0.683914185 D[229]=-0.711318970 D[230]=-0.738372803 D[231]=-0.765029907
D[232]=-0.791213989 D[233]=-0.816864014 D[234]=-0.841949463 D[235]=-0.866363525
D[236]=-0.890090942 D[237]=-0.913055420 D[238]=-0.935195923 D[239]=-0.956481934
D[240]=-0.976852417 D[241]=-0.996246338 D[242]=-1.014617920 D[243]=-1.031936646

D[244]=-1.048156738 D[245]=-1.063217163 D[246]=-1.077117920 D[247]=-1.089782715
D[248]=-1.101211548 D[249]=-1.111373901 D[250]=-1.120223999 D[251]=-1.127746582
D[252]=-1.133926392 D[253]=-1.138763428 D[254]=-1.142211914 D[255]=-1.144287109
D[256]= 1.144989014 D[257]= 1.144287109 D[258]= 1.142211914 D[259]= 1.138763428
D[260]= 1.133926392 D[261]= 1.127746582 D[262]= 1.120223999 D[263]= 1.111373901
D[264]= 1.101211548 D[265]= 1.089782715 D[266]= 1.077117920 D[267]= 1.063217163
D[268]= 1.048156738 D[269]= 1.031936646 D[270]= 1.014617920 D[271]= 0.996246338
D[272]= 0.976852417 D[273]= 0.956481934 D[274]= 0.935195923 D[275]= 0.913055420
D[276]= 0.890090942 D[277]= 0.866363525 D[278]= 0.841949463 D[279]= 0.816864014
D[280]= 0.791213989 D[281]= 0.765029907 D[282]= 0.738372803 D[283]= 0.711318970
D[284]= 0.683914185 D[285]= 0.656219482 D[286]= 0.628295898 D[287]= 0.600219727
D[288]= 0.572036743 D[289]= 0.543823242 D[290]= 0.515609741 D[291]= 0.487472534
D[292]= 0.459472656 D[293]= 0.431655884 D[294]= 0.404083252 D[295]= 0.376800537
D[296]= 0.349868774 D[297]= 0.323318481 D[298]= 0.297210693 D[299]= 0.271591187
D[300]= 0.246505737 D[301]= 0.221984863 D[302]= 0.198059082 D[303]= 0.174789429
D[304]= 0.152206421 D[305]= 0.130310059 D[306]= 0.109161377 D[307]= 0.088775635
D[308]= 0.069168091 D[309]= 0.050354004 D[310]= 0.032379150 D[311]= 0.015228271
D[312]=-0.001068115 D[313]=-0.016510010 D[314]=-0.031082153 D[315]=-0.044784546
D[316]=-0.057617187 D[317]=-0.069595337 D[318]=-0.080688477 D[319]=-0.090927124
D[320]= 0.100311279 D[321]= 0.108856201 D[322]= 0.116577148 D[323]= 0.123474121
D[324]= 0.129577637 D[325]= 0.134887695 D[326]= 0.139450073 D[327]= 0.143264771
D[328]= 0.146362305 D[329]= 0.148773193 D[330]= 0.150497437 D[331]= 0.151596069
D[332]= 0.152069092 D[333]= 0.151962280 D[334]= 0.151306152 D[335]= 0.150115967
D[336]= 0.148422241 D[337]= 0.146255493 D[338]= 0.143676758 D[339]= 0.140670776
D[340]= 0.137298584 D[341]= 0.133590698 D[342]= 0.129562378 D[343]= 0.125259399
D[344]= 0.120697021 D[345]= 0.115921021 D[346]= 0.110946655 D[347]= 0.105819702
D[348]= 0.100540161 D[349]= 0.095169067 D[350]= 0.089706421 D[351]= 0.084182739
D[352]= 0.078628540 D[353]= 0.073059082 D[354]= 0.067520142 D[355]= 0.061996460

D[356]= 0.056533813 D[357]= 0.051132202 D[358]= 0.045837402 D[359]= 0.040634155

D[360]= 0.035552979 D[361]= 0.030609131 D[362]= 0.025817871 D[363]= 0.021179199

D[364]= 0.016708374 D[365]= 0.012420654 D[366]= 0.008316040 D[367]= 0.004394531

D[368]= 0.000686646 D[369]=-0.002822876 D[370]=-0.006134033 D[371]=-0.009231567

D[372]=-0.012115479 D[373]=-0.014801025 D[374]=-0.017257690 D[375]=-0.019531250

D[376]=-0.021575928 D[377]=-0.023422241 D[378]=-0.025085449 D[379]=-0.026535034

D[380]=-0.027801514 D[381]=-0.028884888 D[382]=-0.029785156 D[383]=-0.030517578

D[384]= 0.031082153 D[385]= 0.031478882 D[386]= 0.031738281 D[387]= 0.031845093

D[388]= 0.031814575 D[389]= 0.031661987 D[390]= 0.031387329 D[391]= 0.031005859

D[392]= 0.030532837 D[393]= 0.029937744 D[394]= 0.029281616 D[395]= 0.028533936

D[396]= 0.027725220 D[397]= 0.026840210 D[398]= 0.025909424 D[399]= 0.024932861

D[400]= 0.023910522 D[401]= 0.022857666 D[402]= 0.021789551 D[403]= 0.020690918

D[404]= 0.019577026 D[405]= 0.018463135 D[406]= 0.017349243 D[407]= 0.016235352

D[408]= 0.015121460 D[409]= 0.014022827 D[410]= 0.012939453 D[411]= 0.011886597

D[412]= 0.010848999 D[413]= 0.009841919 D[414]= 0.008865356 D[415]= 0.007919312

D[416]= 0.007003784 D[417]= 0.006118774 D[418]= 0.005294800 D[419]= 0.004486084

D[420]= 0.003723145 D[421]= 0.003005981 D[422]= 0.002334595 D[423]= 0.001693726

D[424]= 0.001098633 D[425]= 0.000549316 D[426]= 0.000030518 D[427]=-0.000442505

D[428]=-0.000869751 D[429]=-0.001266479 D[430]=-0.001617432 D[431]=-0.001937866

D[432]=-0.002227783 D[433]=-0.002487183 D[434]=-0.002700806 D[435]=-0.002883911

D[436]=-0.003051758 D[437]=-0.003173828 D[438]=-0.003280640 D[439]=-0.003372192

D[440]=-0.003417969 D[441]=-0.003463745 D[442]=-0.003479004 D[443]=-0.003479004

D[444]=-0.003463745 D[445]=-0.003433228 D[446]=-0.003387451 D[447]=-0.003326416

D[448]= 0.003250122 D[449]= 0.003173828 D[450]= 0.003082275 D[451]= 0.002990723

D[452]= 0.002899170 D[453]= 0.002792358 D[454]= 0.002685547 D[455]= 0.002578735

D[456]= 0.002456665 D[457]= 0.002349854 D[458]= 0.002243042 D[459]= 0.002120972

D[460]= 0.002014160 D[461]= 0.001907349 D[462]= 0.001785278 D[463]= 0.001693726

D[464]= 0.001586914 D[465]= 0.001480103 D[466]= 0.001388550 D[467]= 0.001296997

D[468]= 0.001205444 D[469]= 0.001113892 D[470]= 0.001037598 D[471]= 0.000961304

D[472]= 0.000885010 D[473]= 0.000808716 D[474]= 0.000747681 D[475]= 0.000686646

D[476]= 0.000625610 D[477]= 0.000579834 D[478]= 0.000534058 D[479]= 0.000473022

D[480]= 0.000442505 D[481]= 0.000396729 D[482]= 0.000366211 D[483]= 0.000320435

D[484]= 0.000289917 D[485]= 0.000259399 D[486]= 0.000244141 D[487]= 0.000213623

D[488]= 0.000198364 D[489]= 0.000167847 D[490]= 0.000152588 D[491]= 0.000137329

D[492]= 0.000122070 D[493]= 0.000106812 D[494]= 0.000106812 D[495]= 0.000091553

D[496]= 0.000076294 D[497]= 0.000076294 D[498]= 0.000061035 D[499]= 0.000061035

D[500]= 0.000045776 D[501]= 0.000045776 D[502]= 0.000030518 D[503]= 0.000030518

D[504]= 0.000030518 D[505]= 0.000030518 D[506]= 0.000015259 D[507]= 0.000015259

D[508]= 0.000015259 D[509]= 0.000015259 D[510]= 0.000015259 D[511]= 0.000015259

Tabla de Huffman

Valor cuantizado de la muestra	Código
0	0
31	1001
-31	1000
-2	111011
1	111010
-3	110111
-1	110110
2	110101
-5	110011
3	110010
-4	101111
-7	101110
-6	101100
6	101011
5	101001
4	101000
7	1111110
-30	1111011
-9	1111010

8	1111001
30	1111000
-10	1110010
9	1110001
-8	1101000
-12	1100010
10	1011011
11	1011010
12	11111110
-13	11111001
-11	11100111
-17	11100001
13	11100000
19	11010010
21	11000111
-16	11000011
-15	11000010
14	11000001
15	10101010
16	10101001
-19	111111111
-24	111111110
-14	111110111
20	111110110
23	111110101
28	111110100
-25	111110000
-18	111001101
-23	111001100
-21	110100111
18	110100110
22	110001101
-26	110001100
-22	110000001
-20	110000000
25	101010111
-27	1111100011
24	1111100010
17	1010101101
-29	1010101100
-28	1010100011
26	1010100010
27	1010100001
29	1010100000

Apéndice C

Glosario

Señal. Es una función que contiene información sobre el estado ó comportamiento de un sistema físico.

Señales de Potencia. Se describen en términos de potencia las señales periódicas, o Aleatorias estacionarias o no limitadas en t.

Sistemas Lineales. Son aquellos que verifican el principio de superposición.

Homogeneidad. Un cambio en la amplitud de la señal de entrada, provoca el mismo cambio de amplitud en la señal de salida.

Filtro digital. Proceso computacional que genera una secuencia discreta a partir de otra, según una regla preestablecida.

Decimador(downsampler). Sistema que reduce la frecuencia de muestreo en una relación de enteros

Interpolador(upsampler). Sistema que incrementa la frecuencia de muestreo en una relación de enteros.

Cuantización. Es el proceso de convertir números con precisión infinita en números de un conjunto finito.

Compresión sin pérdidas o *lossless*. Permite reconstruir el archivo original sin pérdida de información. Si se trata de una imagen la reproduce tal cual era. Utiliza el algoritmo DPCM (*Differential Pulse Code Modulation*).

Compresión con pérdidas o *lossly*. Reconstruye el archivo original de forma aproximada, lo que es válido en aquellos casos en que la información perdida es irrelevante. Sacrifica algunos elementos de información en tanto no se altera perceptiblemente el impacto total del contenido. Se consigue niveles de compresión mucho más altos, del orden de 100.

Loudness Es la intensidad de un sonido pero medida percetualmente es decir es la intensidad con que se perciben los sonidos esta no concuerda con la intensidad en dB SPL se mide usualmente en *sones*.

dB SPL unidad que nos permite medir la intensidad física de los sonidos, las siglas SPL vienen del ingles *Sound Pressure Level.*

Pitch Característica del sonido que se relaciona con la frecuencia y la intensidad del sonido, esta cualidad del sonido solo se puede medir perceptualmente.

Bark Es una unidad que relaciona la frecuencia absoluta con las frecuencias medidas perceptualmente, y puede mapear un sonido en el dominio de la frecuencia en el dominio psicoacústico, facilita la descripción de las curvas de enmascaramiento.

Enmascaramiento. Fenómeno que sucede debido a la manera en que escuchamos y consiste en el hecho de que la existencia física de un sonido no implica que lo percibamos y en muchas no escuchemos todos los sonidos debido a la presencia de otros, cuando esto sucede se dice que el sonido es enmascarado por otro.

Postenmascaramiento Es un tipo de enmascaramiento temporal que sucede cuando un sonido fuerte enmascara a un sonido débil que sucede unos instantes después del sonido fuerte.

Preenmascaramiento Enmascaramiento temporal que sucede cuando un sonido débil es enmascarado por un sonido fuerte que sucede después, la diferencia en tiempo debe ser de algunos milisegundos.

ISO. (International Organization for Standardization). Es una federación internacional encargada de desarrollar estandares para facilitar el intercambio de bienes y servicios, así como promover la cooperación de las actividades en los ámbitos intelectual, científico, tecnológico y comercial.

Trama. Es un conjunto de bit ordenados según ciertos lineamientos, en la que su posición guarda una función según la estructura creada.

Indice de Tablas y Figuras

Bibliografía

Capítulo 1

MURAT KUNT, Digital signal processing, Artech House Inc., 1992
PROAKIS, D.G. MANOLAKIS, Digital signal processing, principles, algorithms and applications, Prentice Hall Inc., 1996.
STEARNS, D.R. HUSH, Digital signal analysis, Prentice Hall Inc., 1990.
EMBREE, B. KIMBLE, C language algorithms for digital signal processing, Prentice Hall, 1991.
O' NEAL, PETER V., Matemáticas avanzadas para Ingeniería, Vol. II, 3ra edición, CECSA, México 1997.
IBARRA PEREYRA, MARIO, Apuntes de Comunicaciones Digitales, FI, UNAM, 2000
http://atc1.aut.alcala.es/~infind Práctica 3 Filtros I
http://dmiwww.cs.tut.fi/intranet/
http://coco.ccu.uniovi.es/immed/compresion/descripcion/spiht/wavelets/wavelets.htm
http://serdis.dis.ulpgc.es/~li-pso/
http://www.revista.unam.mx/vol.1/num2/art3/
http://www.ii.uam.es/~pedro/ccii/teoria/TFDCompresion/TFDCompresion.htm

Capítulo 2

ANÁLISIS DEL COMPORTAMIENTO VERBAL ARTICULATORIO EN CONVERSACIONES GRUPALES ESPONTÁNEAS Tesis doctoral dirigida por la Dra. Pilar González López.
Departamento de Psicología Social. Facultad de Psicología. Universidad de Barcelona.
A Oppenheim y R. Schafer. *Discrete Time Signal Procesing.*
C Grewin y T. Ryden. *"Subjetive Assessments on Low Bit-rate Audio Codecs".*
D. Y. Pan,"Digital Audio Compression", Digital Technical Journal Vol. 5/ No. 2, Spring 1993
D. Y. Pan,"A Tutorial on MPEG/Audio Compression", Institute of Electrical and Electronics Engineers, 1996
F. Baumgarte. A Physiological Ear Model for Auditory Masking Applicable to Perceptual Coding . 103rd AES Convention, New York, NY, September 1997.
Painter, E. M., and Spanias, A. S.. A Review of Algorithms for Perceptual Coding of Digital Audio Signals. Department of Electrical Engeneering, Telecomunications Research Center - Arizona State University, Tempe, Arizona, 1997.
K. Brandenburg and G. Stoll. ISO-MPEG-1 Audio: a generic standard for coding of high quality digital audio. In N. Gilchrist and Ch. Grewin, editors, Col-lected Papers on Digial Audio Bit-Rate Reduction, pages 31 - 42. AES, 1996.
www.helsinki.fi/~ssyreeni/dsound/dsound-t-01
www.helsinki.fi/~ssyreeni/dsound/dsound-c-04
http://www.mp3dev.org/mp3/
http://www.mpeg.org/MPEG/index.html

Capítulo 3

POHLMANN, K, Principles of Digital Audio, 4th ed,McGrawHill,EEUU.
ISO/IEC 11172 - Coding Of Moving Picture And Associated Audio For Digital Storage Media At Up To About 1,5 Mbit/s - Part 3: Audio ISO/IEC
www.mp3-tech.org
Instituto Fraunhoffer. http://www.iis.fhg.de/
Notes on audio compression.
http://fas.sfu.ca/cs/undergrad/CourseMaterials/CMPT479/material/notes
ISO. http://www.iso.ch/ Información sobre las normas ISO-11172.
www.hispamp3.com
www.mpeg.org
ATRAC. http://www.tdk.com/spanish/cdr/6_1.html
http://www.tonytechno.com/BD/makers/makersamp.asp?offset=48
http://www.minidisc.org
ATRAC3 High-Quality Audio Encoding Technology, techno world.
MPEG 4.http://web.media.mit.edu/~eds/mpeg4/sa-tech.html#compresión
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND AUDIO
Overview of the MPEG-4 Standard, Rob Koenen

Capítulo 4

Apuntes sobre el TSM320c6711, Laboratorio de Procesamiento Digital de Voz, Posgrado de Ingeniería, UNAM.