CAPÍTULO 2 PARAMETRIZACIÓN

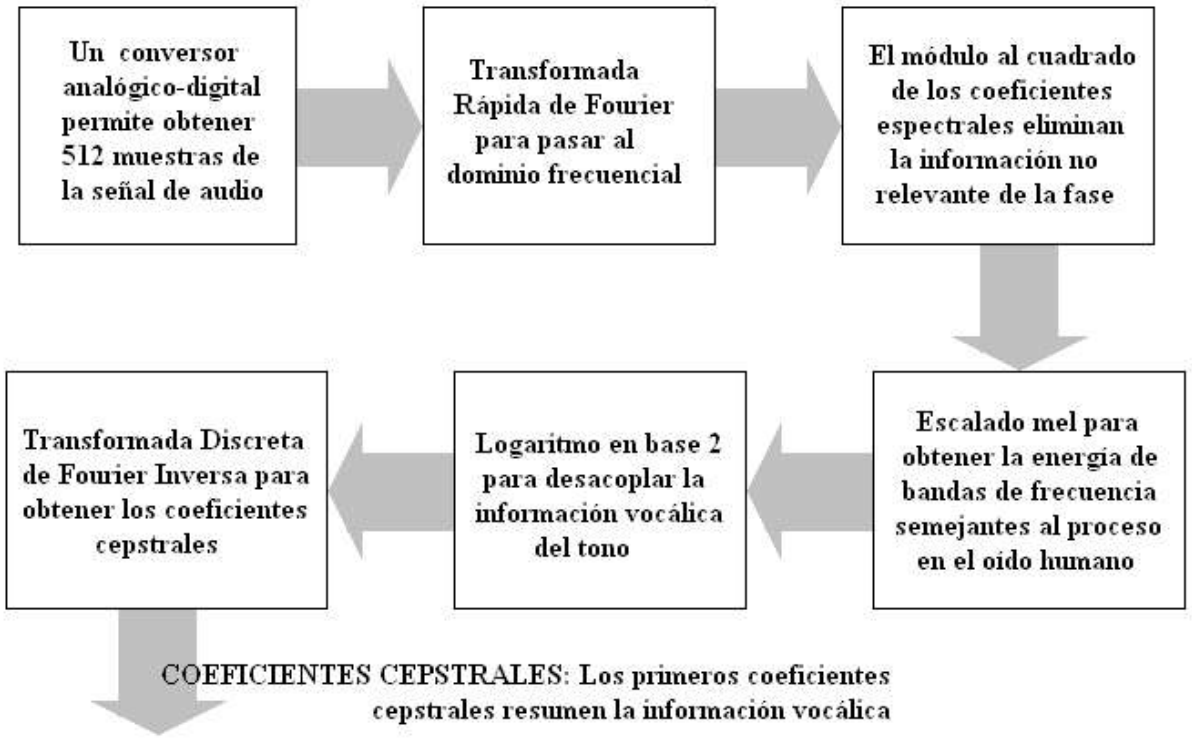
2.1 INTRODUCCIÓN

Es la FPGA la que realiza la ardua labor de reconocimiento, para ello, deben obtenerse características apropiadas de las señales de audio que nos permitan el reconocimiento de las vocales. A partir de la señal digital procedente del muestreo de la señal de voz hemos de extraer una serie de parámetros que nos permitan caracterizar adecuadamente la vocal que estamos pronunciando. El número de parámetros debe ser suficientemente pequeño para eliminar redundancia en la información, pero lo suficientemente amplio como para poder discriminar una vocal entre las otras.

A partir de la señal digital procedente del muestreo de la señal de voz hemos de extraer una serie de parámetros que nos permitan caracterizar adecuadamente la vocal que estamos pronunciando. La parametrización tiene como objetivo fundamental la representación de la señal de voz, previamente muestreada, a través de unos parámetros que resalten las características más importantes del mensaje comprendido en la onda acústica, eliminando parte de la redundancia de la señal.

En la actualidad se prefiere el análisis frecuencial del habla al análisis temporal. Para la realización del proyecto se va a preferir un análisis frecuencial pues nos proporciona más información que la que obtendríamos si tratáramos la señal en el dominio del tiempo. No nos bastará con obtener unas características espectrales de la señal, pues de ella debemos extraer la información relevante que nos permite diferenciar las vocales. Los parámetros que vamos a

extraer son los llamados **coeficientes cepstrales**. El proceso de obtención de los parámetros cepstrales es como sigue:



- El proceso de reconocimiento de la vocal pronunciada empieza con la obtención de 512 muestras de la señal de audio procedentes del conversor analógico-digital.

- Seguidamente se calcula la Transformada Rápida de Fourier de esas 512 muestras para pasar al dominio frecuencial, donde las características vocálicas son más evidentes.

- Se obtiene el módulo de los coeficientes de Fourier ya que la información aportada por la fase no es importante.

- Un escalado mel en el cuadrado del módulo de los coeficientes de Fourier sintetiza la energía de las diversas frecuencias pronunciadas con las vocales de una forma más semejante a la realizada por el oído humano.

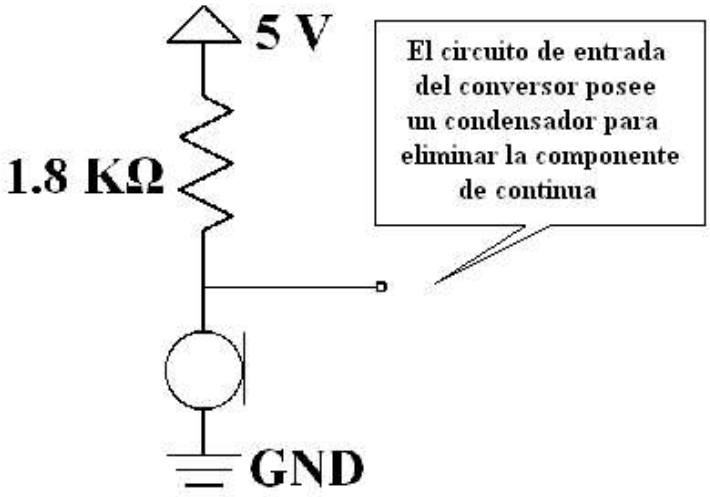
- Al realizar el logaritmo en base 2 a los coeficientes mel se desacopla la información aportada por el tono (frecuencia de vibración de las cuerdas vocales) de la aportada por el tracto vocálico (la que realmente contiene la información útil para la detección vocálica).

- La Transformada Inversa de Fourier sobre los logaritmos anteriores permite obtener los coeficientes cepstrales. Los coeficientes cepstrales bajos resumen la información del tracto

vocálico y los coeficientes cepstrales altos sintetizan la información del tono. Los primeros coeficientes cepstrales constituyen unos parámetros adecuados suficientemente pequeños en número que pueden utilizarse posteriormente en el proceso de reconocimiento.

2.2 EL CONVERSOR ANALÓGICO DIGITAL

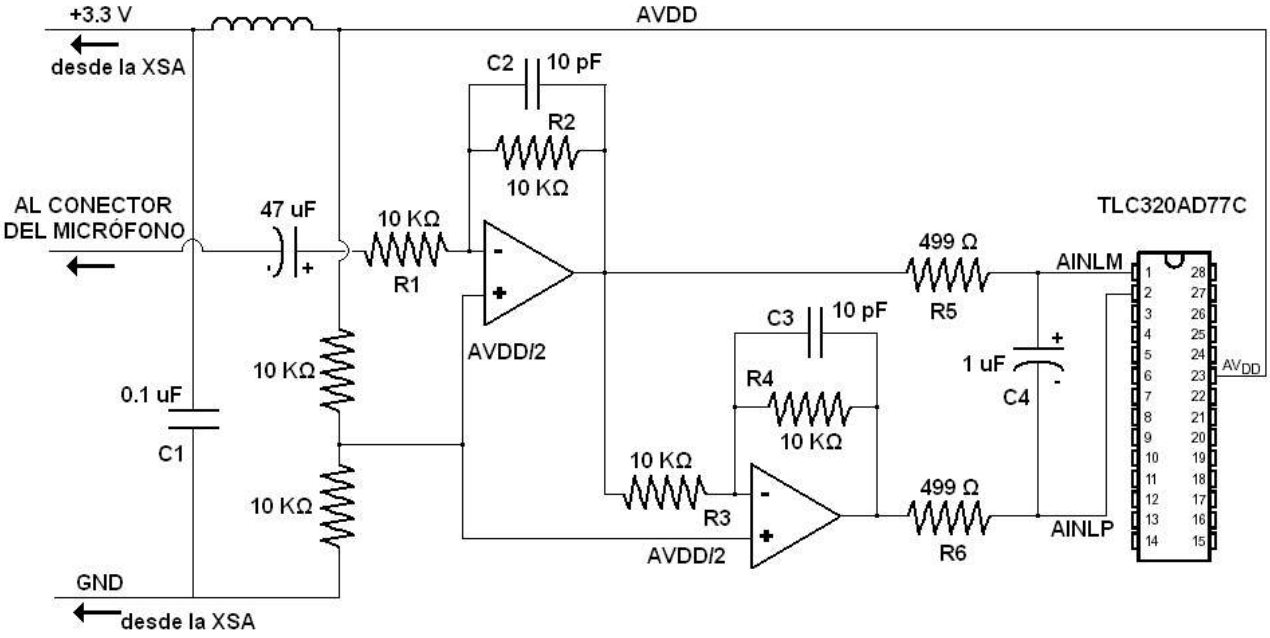
El interfaz del prototipo con el usuario que pronuncia las vocales es el medio acústico que transporta la voz. Un micrófono electret alimentado con 5 voltios de tensión a través de una resistencia de 1.8 KΩ realiza la transducción de la señal mecánica de voz en una señal eléctrica apropiada para ser transformada en muestras digitales por un conversor analógico-digital.



Tras un procesado de la señal eléctrica de audio, el CODEC TLC320AD77C de Texas Instruments realiza la conversión de la señal de audio analógica en muestras digitales comprensibles por la FPGA. El TLC320AD77C es un convertidor analógico-digital y digital-analógico muy competitivo de 24-bits basado en la conversión sigma-delta para aplicaciones de consumidor que demandan una excelente calidad de audio. Tiene una gran variedad de opciones para la señal serie: justificación a la izquierda, justificación a la derecha, IIS o formatos de datos DSP para datos de entrada y de salida de 16, 20 ó 24 bits. Tiene un amplio rango de frecuencias de muestreo que empieza en los 16 KHz y termina en los 96 KHz. Su diseño interno proporciona un voltaje de referencia muy limpio. El TLC320AD77C fue inicialmente diseñado para mini-discos, receptores de audio/vídeo, instrumentos musicales y otros equipos finales que requieren de una alta calidad de conversión de audio digital. Las características del dispositivo pueden resumirse en la siguiente tabla:

TLC320AD77C

Codificador Analógico-Digital y Digital-Analógico estéreo de 24 bits Sigma-Delta.	Datos de entrada y salida de 16, 20 ó 24 bits.
	Amplio rango de frecuencias de muestreo (desde 16 a 96 KHz).
	Reloj maestro de frecuencia: 256 fs o 384 fs.
	Fuente de alimentación de 3.3 V.
	Estabilizador interno de tensión de referencia.
	Económico encapsulado de 28 pines.
Codificador Analógico Digital estéreo.	Entrada diferencial.
	Filtro digital paso-alta.
	Altas prestaciones: 100 dB de relación señal ruido, 100 dB de rango dinámico.
Filtro digital de De-énfasis.	Posibilidad de elegir 32 KHz, 44.1 KHz o 48 KHz.
Excelentes características de rechazo de ruido.	
Temperatura de funcionamiento:	de 0ºC a 70ºC.



Para tener un excelente rechazo en modo común de señales indeseadas, la señal digital es procesada de manera diferencial hasta que es convertida en datos digitales. Una señal de entrada normal debe ser convertida en una señal de entrada diferencial y filtrada con un filtro anti-aliasing de un polo antes de entrar en el CAD. Los procesos de filtrado y amplificación de la

señal de audio se modelan a través de una placa de desarrollo del Departamento de Electrónica de la Escuela Superior de Ingenieros de Sevilla. Sobre esta placa se insertará la tarjeta XSA para poner en comunicación el CODEC con la FPGA.

El Convertidor Analógico Digital acepta una entrada diferencial con un valor máximo que no debe exceder los 4 Vpp. A continuación se muestra el circuito presente en la placa de desarrollo que transforma la señal de entrada normal a una entrada diferencial. Lleva la señal de entrada alrededor de AVDD/2. Esta señal debe tomar un valor máximo de 0.7 Vrms. El dispositivo presenta una señal diferencial a escala completa de 4 Vpp. Para señales de entrada con valores diferentes, la relación R2/R1 debe ser escalada adecuadamente para proporcionar una señal máxima al CAD de 4 Vpp. Las resistencias R5, R6 y el condensador C4 crean un filtro paso-bajo anti-aliasing de un polo para atenuar las frecuencias indeseadas. Si el usuario eligiese introducir una señal normal directamente al dispositivo, de 2 Vpp como máximo, las prestaciones del convertidor disminuirían sensiblemente.

En lo que se refiere a la alimentación del circuito, para tener un buen rechazo del ruido se usa una referencia pseudo-diferencial con capacidades externas conectadas a un filtro paso-bajo diferencial. El siguiente circuito, recomendado en el manual de referencia del CODEC TLC320AD77C de Texas Instruments, alimenta al dispositivo.

La frecuencia de muestreo con la que el CODEC mostrará las muestras digitales de la señal de audio es de 19531 Hz (51.2 us de periodo), tomada esta frecuencia por conveniencia en la obtención de los parámetros que caracterizarán la señal vocálica. El TLC320AD77C soporta dos rangos de frecuencia de muestreo:

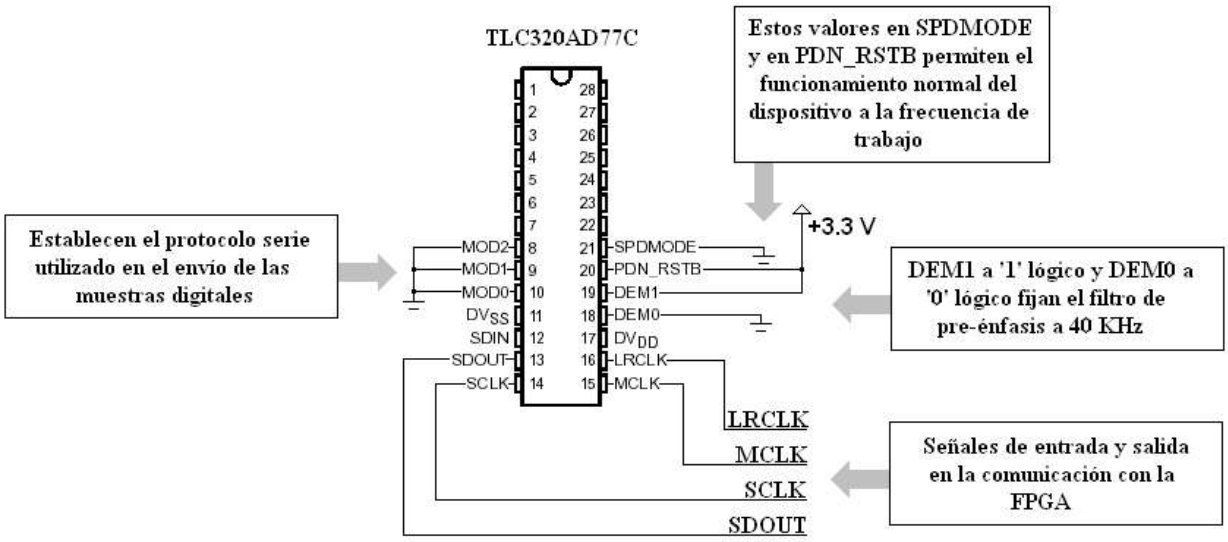
- Cuando trabajamos en el rango normal de funcionamiento de 16 KHz a 48 KHz, el pin de entrada SPDMOD está puesto a nivel bajo. En el caso particular del proyecto, esta opción es la que debe ser elegida.
- Cuando elegimos la opción más rápida el rango está comprendido entre los 48 KHz y los 96 KHz y SPDMOD está a nivel alto.

Los datos son enviados al ordenador con una frecuencia de muestreo (Fs) de 19531 Hz (51.2 us de periodo) a través del pin de salida SDOUT por medio de un protocolo serie. El interfaz serie consiste en un reloj de cambio o de bit (SCLK), un reloj de sincronización de trama izquierdo / derecho (LRCLK) y un canal de salida de datos digitales procedentes del CAD (SDOUT). Podemos seleccionar uno de los ocho protocolos serie posibles (IIS, justificado a la derecha / izquierda y un modo DSP para longitudes de palabras de 16 a 24 bits). El Convertidor Analógico Digital convierte la señal en palabras discretas digitales de salida en complemento a dos. Hay un filtro paso alto para librarse de cualquier desfase que el modulador CAD pueda haber causado. Estas palabras digitales representan los valores de la señal analógica de entrada muestreada.

Para el funcionamiento síncrono del dispositivo es necesario aportarle a través de los pines de entrada tres relojes. La FPGA deberá generar estas señales para el control del conversor.

- **MCLK**: reloj maestro del codificador analógico-digital generado en la FPGA para el funcionamiento síncrono del sistema digital del conversor. Tiene un periodo de 200 ns.

- **SCLK**: reloj generado en la FPGA que establece el sincronismo de bit de los datos transmitidos. Ante una modificación en el nivel del reloj LRCLK, en cada flanco de bajada del reloj SCLK se modificará el nivel de la línea de datos según el bit de la palabra de 20 bits que corresponda, empezando por el bit más significativo. Tiene un periodo de 800 ns.



- **LRCLK**: reloj que establece la frecuencia de muestreo (Fs) de la señal analógica (frecuencia de 19531 Hz ó 51.2 us de periodo). Cuando inicia un nivel alto, en cada flanco de bajada del reloj de sincronismo de bit SCLK se modificará la línea de datos para que se corresponda con el bit actual de la palabra de 20 bits, empezando por el más significativo. Uno tras otro se emitirán los 20 bits de la muestra digital en complemento a dos del canal izquierdo de audio. Cuando se inicia un nivel bajo, se emitirán uno tras otro los 20 bits de la muestra digital en complemento a dos del canal derecho de audio.

El TLC320AD77C opera sólo en modo esclavo. Requiere que le sean aportadas como entradas el reloj maestro (MCLK), el reloj de izquierda / derecha (LRCLK) y el reloj de bits (SCLK). Hay dos opciones al seleccionar las frecuencias del reloj: Si elegimos un reloj maestro (MCLK) con una frecuencia 384*Fs, entonces debemos proporcionar una trama LRCLK de 48 pulsos de reloj SCLK. Si elegimos un reloj maestro (MCLK) con una frecuencia 256*Fs, entonces debemos proporcionar una trama LRCLK de 64 pulsos de reloj SCLK.

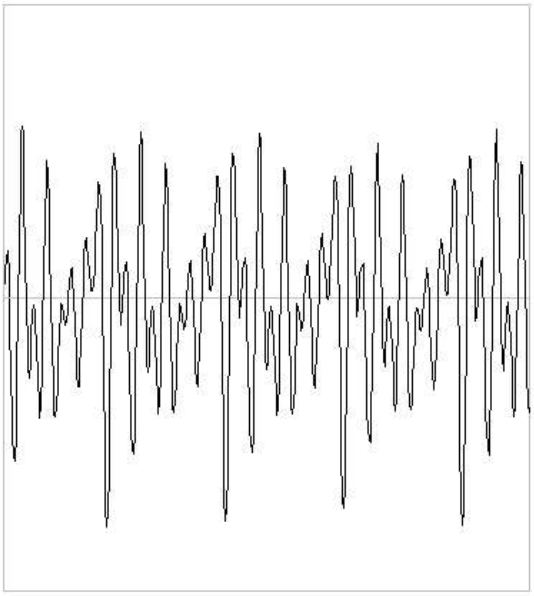
Existen ocho protocolos serie posibles en la transmisión de los datos, pero seleccionamos el siguiente al establecer una correspondencia en los periodos de los relojes de entrada (LRCLK, SCLK y MCLK) y al fijar un '0' lógico en los pines MOD2, MOD1 y MOD0:

- Cada muestra se compone de 20 bits en complemento a dos.
- Se transmite primero el bit más significativo.

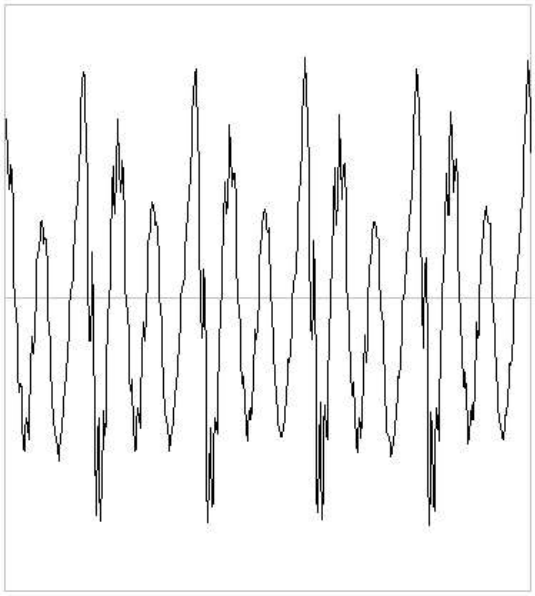
- El reloj LRCLK fija la frecuencia de muestreo (19531 Hz).
- Cada muestra es enviada en el flanco de subida del reloj LRCLK a través del pin de salida SDOUT.
- La salida SDOUT está justificada a la izquierda.
- Cada bit de datos cambia en el flanco de bajada del reloj SCLK. El reloj SCLK establece el sincronismo de bit.
- El bit de datos es estable en el flanco de subida del reloj SCLK, por lo que se propone este instante para la lectura del bit.
- Cada ciclo del reloj LRCLK contiene 64 ciclos del reloj SCLK.
- Cada ciclo del reloj LRCLK contiene 256 ciclos del reloj maestro MCLK.

La FPGA generará los relojes de control del CODEC (LRCLK, SCLK y MCLK) y recibirá las muestras digitales bit a bit en el formato serie anterior a través de la línea SDOUT. Presento a continuación ejemplos de muestras digitales de cinco pronunciaciones vocálicas obtenidas a través del CODEC TLC320AD77C. Gracias a un diseño en VHDL implementado de forma particular sobre la FPGA para la comprobación del sistema conseguí almacenar las muestras en la memoria FLASH externa AT49F002 de ATMEL presente en la tarjeta XSA. Con una función especial ejecutada en lenguaje MATLAB® leí la información contenida en los archivos hexadecimales extraídos desde la memoria FLASH. Pude así representar gráficamente las muestras digitales de las distintas pronunciaciones y comprobar el correcto funcionamiento del CODEC TLC320AD77C.

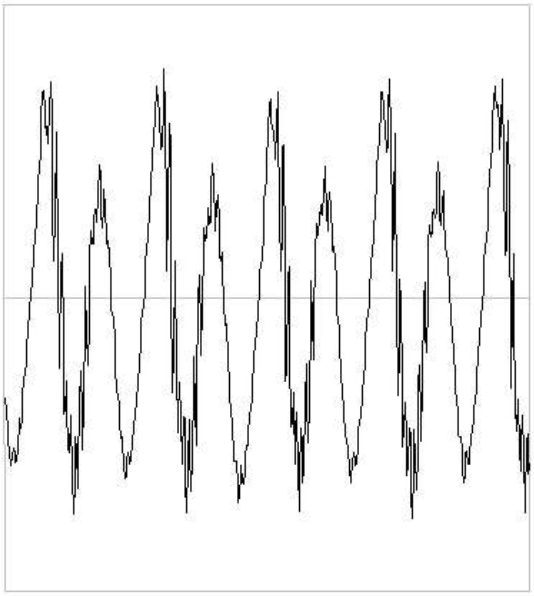
REPRESENTACIÓN GRÁFICA DE
LAS MUESTRAS DIGITALES DE UNA VOCAL A



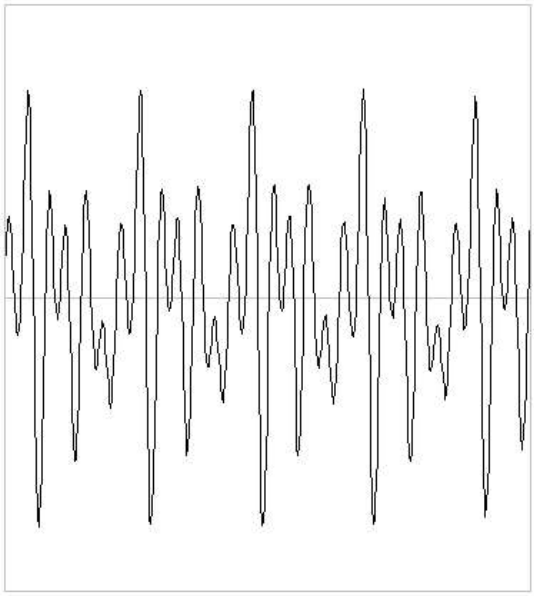
REPRESENTACIÓN GRÁFICA DE
LAS MUESTRAS DIGITALES DE UNA VOCAL E



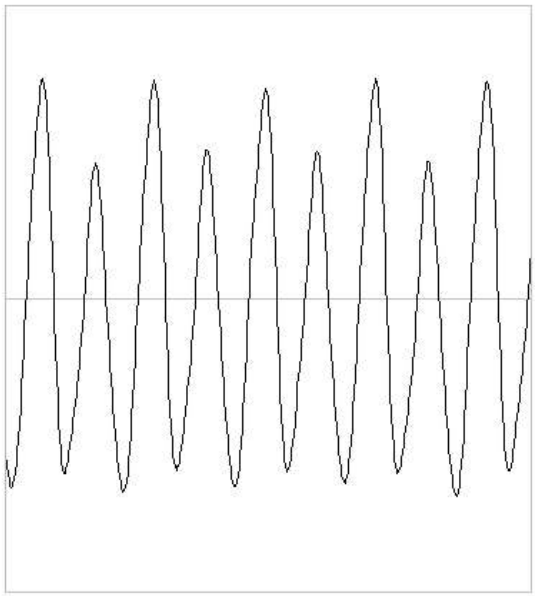
REPRESENTACIÓN GRÁFICA DE LAS MUESTRAS DIGITALES DE UNA VOCAL I



REPRESENTACIÓN GRÁFICA DE LAS MUESTRAS DIGITALES DE UNA VOCAL O



REPRESENTACIÓN GRÁFICA DE LAS MUESTRAS DIGITALES DE UNA VOCAL U



2.3 TRANSFORMADA RÁPIDA DE FOURIER

La Transformada Discreta de Fourier (DFT) desempeña un papel muy importante en numerosas aplicaciones del procesado de señales digitales, incluyendo el filtrado lineal, el análisis de la correlación y el análisis espectral. Una de las razones fundamentales de su importancia radica en la existencia de algoritmos eficientes para el cálculo de la DFT.

En nuestro caso vamos a utilizar un algoritmo basado en la idea de "divide y vencerás", así, una DFT de tamaño N, donde N es un número compuesto, se reduce al cálculo de DFTs más pequeñas a partir de las cuales se obtiene la DFT total. En particular usamos un algoritmo muy importante desde el punto de vista computacional, el algoritmo de la Transformada Rápida de Fourier (FFT, Fast Fourier Transform), para calcular la DFT cuando el tamaño N de la ventana es potencia de 2.

En vista de la importancia de la DFT en distintas aplicaciones del procesado de señales digitales como el filtrado lineal, el análisis de la correlación o el análisis espectral, el cálculo eficiente de la misma ha sido objeto de la atención de numerosos matemáticos, ingenieros y científicos.

Básicamente, el problema del cálculo de la DFT es calcular la secuencia {X(k)} de N números complejos dada la secuencia de datos {x(n)} de longitud N según la fórmula:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn} \qquad 0 \le k \le N-1$$

donde

$$W_N = e^{-j\,2\pi / N}$$

Vemos que para cada valor de k, el cálculo directo de X(k) supone realizar N multiplicaciones complejas (4N multiplicaciones reales) y N-1 sumas complejas (4N-2 sumas reales). En consecuencia, para calcular los N valores de la DFT necesitamos N^2 multiplicaciones complejas y N^2–N sumas complejas.

El cálculo directo de la DFT es básicamente ineficiente debido, fundamentalmente, a que no explota las propiedades de simetría y periodicidad del factor de fase W_N. En particular estas propiedades son:

El algoritmo computacionalmente eficiente de la Transformada Rápida de Fourier (FFT) usa estas dos propiedades básicas del factor de fase.

2.3.1 CÁLCULO DIRECTO DE LA DFT

Para una secuencia compleja x(n) de N puntos, la DFT se puede expresar como

El cálculo directo de la DFT exige:

- 2 N^2 cálculos de funciones trigonométricas.
- 4 N^2 multiplicaciones reales.
- 4N(N-1) sumas reales.
- Numerosas operaciones de direccionamiento e indexado.

Estas operaciones son típicas de los algoritmos computacionales de la DFT. Las operaciones de direccionamiento e indexado son necesarias para ir a buscar los datos x(n), $0 \leq n \leq N-1$, y los factores de fase y los resultados. El algoritmo FFT optimiza cada uno de estos procesos computacionales.

2.3.2 METODOLOGÍA DE 'DIVIDE Y VENCERÁS' PARA EL CÁLCULO DE LA DFT

El desarrollo de algoritmos computacionalmente eficientes par la DFT es posible adoptando la estrategia de 'divide y vencerás'. Este método se basa en la descomposición de una DFT de N puntos en DFTs más pequeñas. Este método lleva a una familia de algoritmos computacionalmente eficientes, conocidos como algoritmos FFT.

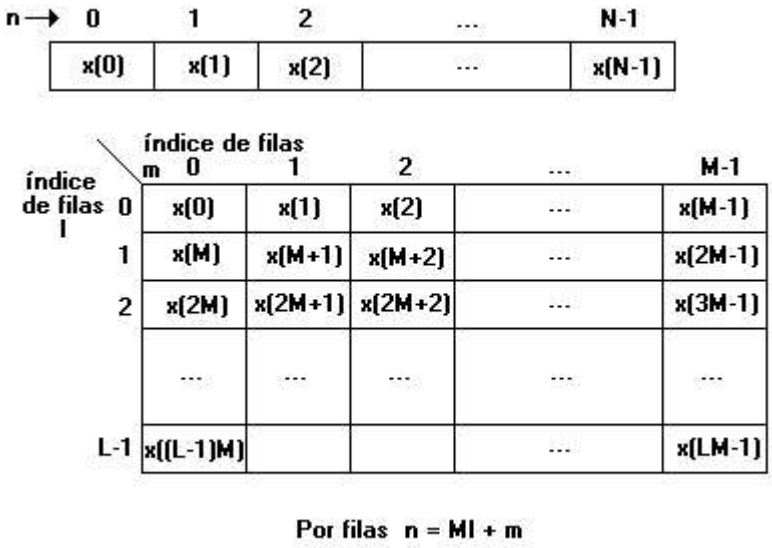
Para ilustrar las ideas básicas, consideremos el cálculo de una DFT de N puntos, donde N se puede representar como el producto de dos enteros, esto es:

$$N = L\,M$$

La suposición de que N no es un número primo no es restrictiva, ya que podemos rellenar cualquier secuencia con ceros para asegurar una factorización de la forma anterior.

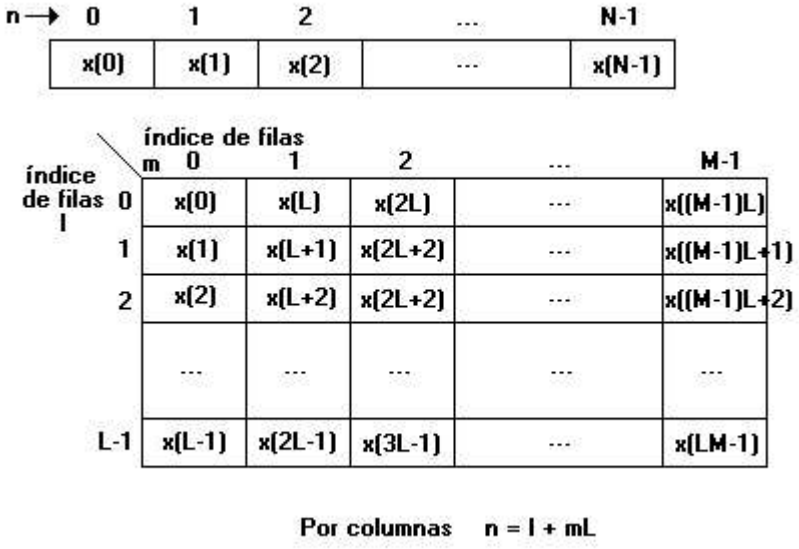
Ahora se puede almacenar la secuencia x(n), $0 \leq n \leq N-1$, tanto en una matriz unidimensional indexada por n o en una matriz bidimensional indexada por l y m, donde, $0 \leq l \leq L-1$ y, $0 \leq m \leq M-1$. Observe que l es el índice de las filas y m el de las columnas. Por tanto, la secuencia x(n) puede almacenarse en una matriz rectangular de diferentes maneras, cada una de las cuales depende de la correspondencia entre el índice n y los índices l y m.

Por ejemplo, si se elige la correspondencia n = Ml + m, nos lleva a una disposición en la que la primera fila contiene los primeros M elementos de x(n), la segunda contiene los M siguientes elementos de x(n), y así sucesivamente, como se muestra en la figura siguiente.



Por filas n = Ml + m

Por otra parte, la correspondencia n = l + m L almacena los primeros L elementos de x(n) en la primera columna, los siguientes L elementos en la segunda columna, y así sucesivamente, como se muestra en la figura siguiente:



Por columnas n = l + mL

Una disposición similar puede usarse para almacenar los valores calculados de la DFT. En particular, consideraremos la correspondencia desde el índice k a la pareja de índices (p,q), donde

$0 \le p \le L-1$ y $0 \le q \le M-1$. Si elegimos la correspondencia $k = Mp + q$ la DFT se almacena por filas, donde la primera fila contiene los primeros M elementos de la DFT X(k), la segunda fila los siguientes M elementos, y así sucesivamente. Por otra parte, la correspondencia $k = qL + p$ da lugar a un almacenamiento por columnas de X(k), donde los primeros L elementos se almacenan en la primera columna, los siguientes L elementos en la segunda columna y así sucesivamente.

Suponga ahora que x(n) se lleva a la matriz rectangular x(l,m), y que X(k) se lleva a la matriz rectangular correspondiente X(p,q). Entonces la DFT puede expresarse como el sumatorio doble de los elementos de la matriz rectangular multiplicados por los factores de fase correspondientes. Concretamente, consideremos la correspondencia que da lugar al almacenamiento por columnas de x(n). Entonces:

$$X(p,q) = \sum_{m=0}^{M-1} \sum_{l=0}^{L-1} x(l,m)\, W_N^{(Mp+q)(mL+l)}$$

$$W_N^{(Mp+q)(mL+l)} = W_N^{MLmp}\, W_N^{mLq}\, W_N^{Mpl}\, W_N^{Nlq}$$

$$W_N^{Nmp} = 1 \qquad W_N^{mqL} = W_{N/L}^{mq} = W_M^{mq}$$

$$W_N^{Mpl} = W_{N/M}^{pl} = W_L^{pl}$$

Con estas simplificaciones se puede expresar como:

$$X(p,q) = \sum_{l=0}^{L-1} \left\{ W_N^{lq} \left[\sum_{m=0}^{M-1} x(l,m)\, W_M^{mq} \right] \right\} W_L^{lp}$$

La expresión anterior implica el cálculo de DFTs de longitudes M y L. Subdividamos el cálculo en tres pasos:

- Primero calculamos las DFTs de M puntos para cada una de las filas l = 0, 1, ..., L-1

$$F(l,q) = \sum_{m=0}^{M-1} x(l,m)\, W_M^{mq} \qquad , \quad 0 \le q \le M-1$$

- Segundo, calculamos la nueva matriz rectangular G(l,q) definida como

$$G(l,q) = W_N^{lq}\, F(l,q) \qquad \begin{matrix} 0 \le l \le L-1 \\ 0 \le q \le M-1 \end{matrix}$$

- Finalmente, calculamos las DFTs de L puntos para cada columna q = 0, 1, ..., M-1 de la matriz G(l,q)

$$X(p,q) = \sum_{l=0}^{L-1} G(l,q)\, W_L^{lp} \qquad \begin{matrix} 0 \le l \le L-1 \\ 0 \le q \le M-1 \end{matrix}$$

A primera vista puede parecer que el procedimiento computacional descrito más arriba es más complejo que el cálculo directo de la DFT. Sin embargo, calculemos la complejidad computacional de este método. El primer paso implica el cálculo de L DFTs, cada una de M puntos. Por lo tanto, este paso requiere LM^2 multiplicaciones complejas y LM(M-1) sumas complejas. El segundo paso requiere LM multiplicaciones complejas. Finalmente, el tercer paso exige ML^2 multiplicaciones complejas y ML(L-1) sumas complejas. Por tanto, la complejidad computacional es:

Multiplicaciones complejas: N(ML+1)
Sumas complejas: N(M+L-2)

Sabemos que N=ML. Por lo tanto, el número de multiplicaciones se ha reducido de N^2 a N (M+L+1) y el número de sumas se ha reducido de N(N-1) a N(M+L-2).

Por ejemplo, supongamos que N=1000 y elijamos L=2 y M=500. Entonces, en vez de tener que realizar 10^6 multiplicaciones complejas mediante el cálculo directo de la DFT, con este método haremos 503.000 multiplicaciones complejas. Esto representa una reducción por un factor 2 aproximadamente. El número de sumas también se reduce en un factor cercano a 2.

Cuando N es un número compuesto alto, esto es, N puede factorizarse como el producto de números primos de la forma:

$$N = r_1 r_2 \ldots r_s$$

Entonces la descomposición de arriba puede repetirse (s-1) veces más. Este procedimiento da lugar a DFTs más pequeñas que, en cambio, nos llevan a un algoritmo eficiente computacionalmente.

En efecto, la primera fragmentación de la secuencia x(n) en una matriz rectangular de M columnas y L elementos en cada columna nos lleva a la fragmentación de cada fila (o columna) en matrices rectangulares más pequeñas, lo que da lugar a DFTs más pequeñas. Este procedimiento termina cuando se ha factorizado N en sus factores primos.

Para ilustrar este procedimiento computacional, consideremos el cálculo de una DFT de N = 15 puntos. Dado que N = 5 x 3 = 15, elegimos L = 5 y M = 3. Almacenamos la secuencia de 15 puntos x(n) en columnas como sigue:

Fila 1:	**x(0,0) = x(0)**	**x(0,1) = x(5)**	**x(0,2) = x(10)**
Fila 2:	**x(1,0) = x(1)**	**x(1,1) = x(6)**	**x(1,2) = x(11)**
Fila 3:	**x(2,0) = x(2)**	**x(2,1) = x(7)**	**x(2,2) = x(12)**
Fila 4:	**x(3,0) = x(3)**	**x(3,1) = x(8)**	**x(3,2) = x(13)**
Fila 5:	**x(4,0) = x(4)**	**x(4,1) = x(9)**	**x(4,2) = x(14)**

Ahora calculamos DFTs de tres puntos de cada una de las 5 filas. Esto nos lleva a la siguiente matriz 5 x 3:

F(0,0)	**F(0,1)**	**F(0,2)**
F(1,0)	**F(1,1)**	**F(1,2)**
F(2,0)	**F(2,1)**	**F(2,2)**
F(3,0)	**F(3,1)**	**F(3,2)**
F(4,0)	**F(4,1)**	**F(4,2)**

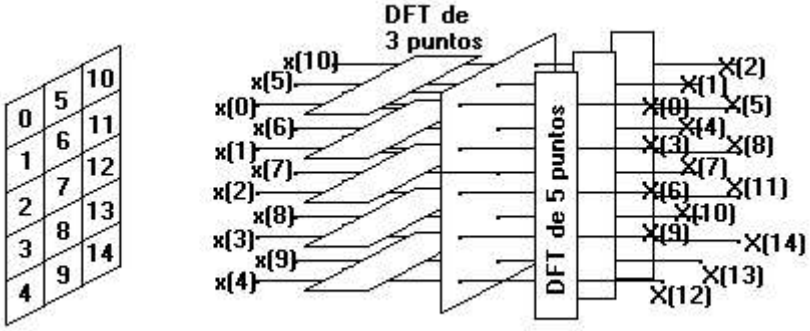
El siguiente paso consiste en multiplicar cada uno de los términos F(l,q) por los factores de fase $W_N^{lq} = W_{15}^{lq}$, $0 \le l \le 4$ y $0 \le q \le 2$. Este cálculo da lugar a la matriz 5 x 3:

Columna 1	***Columna 2***	***Columna 3***
G(0,0)	**G(0,1)**	**G(0,2)**
G(1,0)	**G(1,1)**	**G(1,2)**
G(2,0)	**G(2,1)**	**G(2,2)**
G(3,0)	**G(3,1)**	**G(3,2)**
G(4,0)	**G(4,1)**	**G(4,2)**

El último paso consiste en calcular las DFTs de 5 puntos para cada una de las tres columnas. Este cálculo da los valores deseados de la DFT en la forma

X(0,0) = X(0)	**X(0,1) = X(1)**	**X(0,2) = X(2)**
X(1,0) = X(3)	**X(1,1) = X(4)**	**X(1,2) = X(5)**
X(2,0) = X(6)	**X(2,1) = X(7)**	**X(2,2) = X(8)**
X(3,0) = X(9)	**X(3,1) = X(10)**	**X(3,2) = X(11)**
X(4,0) = X(12)	**X(4,1) = X(13)**	**X(4,2) = X(14)**

Mostramos en la siguiente figura los pasos de este cálculo.



Es interesante ver la secuencia de datos fragmentada y la DFT resultante en función de matrices unidimensionales. Cuando la secuencia de entrada x(n) y la DFT de salida X(k) de las matrices bidimensionales se leen por filas desde la fila 1 a la fila 5, obtenemos las siguientes secuencias:

Matriz de entrada
x(0) x(5) x(10) x(1) x(6) x(11) x(2) x(7) x(12) x(3) x(8) x(13) x(4) x(9) x(14)

Matriz de salida
X(0) X(1) X(2) X(3) X(4) X(5) X(6) X(7) X(8) X(9) X(10) X(11) X(12) X(13) X(14)

Se observa que la secuencia de datos de entrada ha sido mezclada con respecto a su orden normal para el cálculo de la DFT. Por otra parte, la secuencia de salida se encuentra ordenada. En este caso, la reordenación de la secuencia de entrada se debe a la segmentación de la matriz unidimensional en una matriz rectangular y al orden en que se calculan las DFTs. Este mezclado de los datos de entrada o los datos DFT de salida es característico de la mayoría de los algoritmos FFT.

Resumiendo, el algoritmo que hemos introducido implica los siguientes cálculos:

Algoritmo 1

- Almacenamiento de la señal por columnas.
- Cálculo de la DFT de M puntos de cada fila.
- Multiplicación de la matriz resultante por los factores de fase W_N^{lq}.
- Cálculo de la DFT de L puntos de cada columna.
- Lectura por filas de la matriz resultante.

Otro algoritmo con estructura computacional similar se obtiene si la señal de entrada se almacena por filas y la transformación resultante se hace por columnas. En este caso elegimos:

$$N = Ml + m$$
$$K = qL + p$$

Esta elección de los índices nos lleva a la siguiente forma de la fórmula de la DFT:

$$X(p,q) = \sum_{m=0}^{M-1} \sum_{l=0}^{L-1} x(l,m)\ W_N^{pm}\ W_L^{pl}\ W_M^{qm} =$$

$$\sum_{m=0}^{M-1} W_M^{mq} \left[\sum_{l=0}^{L-1} x(l,m)\ W_L^{lp} \right] W_N^{mp}$$

Por lo tanto, obtenemos un segundo algoritmo:

Algoritmo 2

- Almacenamiento de la señal por filas.
- Cálculo de la DFT de L puntos de cada columna.
- Multiplicación de la matriz resultante por los factores W_N^{pm}.
- Cálculo de la DFT de M puntos de cada fila.
- Lectura por columnas de la matriz resultante.

Los dos algoritmos dados arriba tienen la misma complejidad. Sin embargo, se distinguen en el orden de los cálculos. Vamos a explotar la metodología de 'divide y vencerás' para obtener algoritmos rápidos cuando el tamaño de la DFT está restringido a potencias de 2.

2.3.3 ALGORITMO PARA LA FFT BASE-2

El método 'divide y vencerás' proporciona algoritmos eficientes para el cálculo de la DFT. Tal método es aplicable cuando el número de puntos de datos N no es primo. En particular, el método es muy eficiente cuando N es compuesto, es decir, cuando N puede factorizarse como $N= r_1\ r_2\ .\ .\ .\ r_s$, donde los $\{r_j\}$ son primos.

El caso en el que $r_1= r_2 = r_3= ... = r_s= r$, de manera que $N = r^s$, es de particular importancia. En dicho caso, las DFTs son de tamaño r, de manera que el cálculo de las DFTs de N puntos sigue un patrón regular. El número r se denomina base del algoritmo para la FFT.

Vamos a describir el algoritmo en base 2, que con diferencia es el más utilizado por los algoritmos para la FFT. Consideremos el cálculo de la DFT de $N = 2^s$ mediante el método de 'divide y vencerás'. Escogemos M = N / 2 y L = 2. Esta selección da lugar a la división de la secuencia de datos de N puntos en dos secuencias de datos de N / 2 puntos, $f_1(n)$ y $f_2(n)$, correspondientes a las muestras pares e impares de x(n), respectivamente, esto es:

$$f_1(n) = x(2n)$$
$$f_2(n) = x(2n+1) \quad , \quad n = 0, 1, ..., N/2 - 1$$

Por lo tanto, $f_1(n)$ y $f_2(n)$ se obtienen diezmando x(n) por 2 y, en consecuencia, el algoritmo para la FFT resultante se denomina algoritmo de diezmado en tiempo.

La DFT de N puntos puede expresarse ahora en términos de las DFTs de las secuencias diezmadas como sigue:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn} = \sum_{n\ par} x(n)\, W_N^{kn} + \sum_{n\ impar} x(n)\, W_N^{kn}$$

$$= \sum_{m=0}^{(N/2)-1} x(2m)\, W_N^{2mk} + \sum_{m=0}^{(N/2)-1} x(2m+1)\, W_N^{k(2m+1)}$$

$$k = 0, 1, ..., N-1$$

Pero $W_N^2 = W_{N/2}$. Sustituyendo esta igualdad, puede expresarse como:

$$X(k) = \sum_{m=0}^{(N/2)-1} f_1(m)\, W_{N/2}^{km} + W_N^k \sum_{m=0}^{(N/2)-1} f_2(m)\, W_{N/2}^{km}$$

$$= F_1(k) + W_N^k F_2(k)$$

$$k = 0, 1, ..., N-1$$

$F_1(k)$ y $F_2(k)$ son las DFTs de N/2 puntos de las secuencias $f_1(m)$ y $f_2(m)$, respectivamente. Además, al ser periódicas, de periodo N / 2, tenemos que $F_1(k + N/2) = F_1(k)$ y $F_2(k + N/2) = F_2(k)$. Además, $W_N^{k+N/2} = - W_N^k$. De aquí que se pueda expresar como:

$$X(k) = F_1(k) + W_N^k \; F_2(k) \qquad k = 0, 1, ..., N/2 - 1$$

$$X(k + N/2) = F_1(k) - W_N^k \; F_2(k) \qquad k = 0, 1, ..., N/2 - 1$$

Se observa que el cálculo directo de $F_1(k)$ requiere $(N/2)^2$ multiplicaciones complejas. La misma exigencia se aplica al cálculo de $F_2(k)$. Además, se requieren N/2 multiplicaciones complejas más para calcular $W_N^k \; F_2(k)$. De aquí que el cálculo de X(k) requiera $2(N/2)^2 + N/2 = N^2/2 + N/2$ multiplicaciones complejas. El primer paso da lugar a una reducción en el número de multiplicaciones de N^2 a $N^2/2 + N/2$, que equivale aproximadamente a dividir por 2 el número de multiplicaciones cuando N es grande.

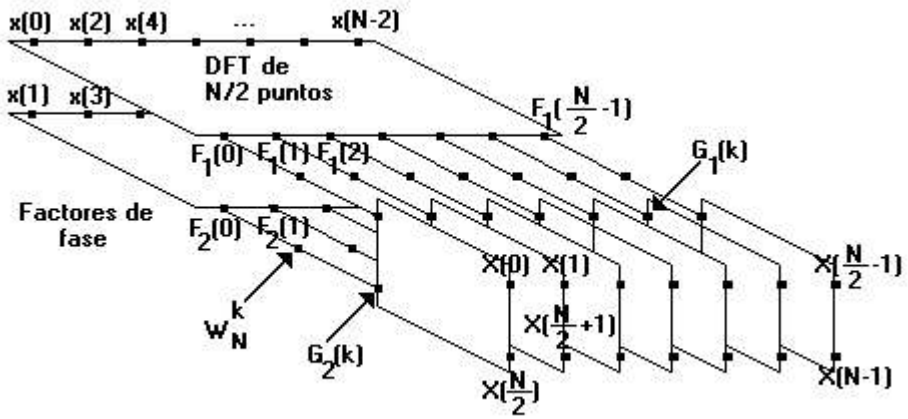
Para ser consistentes con la notación previa definimos:

$$G_1(k) = F_1(k) \quad k = 0, 1, ..., N/2 - 1$$
$$G_2(k) = W_N^k \; F_2(k) \quad k = 0, 1, ..., N/2 - 1$$

Así, la DFT X(k) puede expresarse como:

$$X(k) = G_1(k) + G_2(k) \quad k = 0, 1, ..., N/2 - 1$$
$$X(k + N/2) = G_1(k) - G_2(k) \quad k = 0, 1, ..., N/2 - 1$$

Este cálculo se muestra en la figura siguiente:



Habiendo realizado el diezmado en tiempo una vez, podemos repetir el proceso para cada una de las secuencias $f_1(n)$ y $f_2(n)$. Por lo tanto, $f_1(n)$ dará lugar a las dos secuencias de N/4 puntos:

$$v_{11}(n) = f_1(2n) \quad n = 0, 1, ..., N/4 -1$$

$$v_{12}(n) = f_1(2n+1) \quad n = 0, 1, ..., N/4 -1$$

$f_2(n)$ dará lugar a:

$$\mathbf{v_{21}(n) = f_2(2n) \quad n = 0, 1, ..., N/4 -1}$$
$$\mathbf{v_{22}(n) = f_2(2n+1) \quad n = 0, 1, ..., N/4 -1}$$

Calculando las DFTs de N/4 puntos obtendremos las DFTs de N/2 puntos $F_1(k)$ y $F_2(k)$ a partir de las relaciones:

$$\mathbf{F_1(k) = V_{11}(k) + W_{N/2}^k V_{12}(k) \qquad k = 0, 1, ..., N/4 -1}$$
$$\mathbf{F_1(k + N/4) = V_{11}(k) - W_{N/2}^k V_{12}(k) \qquad k = 0, 1, ..., N/4 -1}$$
$$\mathbf{F_2(k) = V_{21}(k) + W_{N/2}^k V_{22}(k) \qquad k = 0, 1, ..., N/4 -1}$$
$$\mathbf{F_2(k + N/4) = V_{21}(k) - W_{N/2}^k V_{22}(k) \qquad k = 0, 1, ..., N/4 -1}$$

V_{ij} son las DFTs de N/4 puntos de las secuencias $\{v_{ij}(n)\}$.

Se observa que el cálculo de $\{V_{ij}(k)\}$ requiere $4(N/4)^2$ multiplicaciones y, por lo tanto, el cálculo de $F_1(k)$ y $F_2(k)$ puede realizarse con $N^2/4 + N/2$ multiplicaciones complejas. Se requieren N/2 multiplicaciones complejas más para calcular X(k) a partir de $F_1(k)$ y $F_2(k)$. En consecuencia, el número total de multiplicaciones necesarias $N^2/4 + N$ se reduce otra vez por aproximadamente 2.

El diezmado de la secuencia de datos se puede repetir una y otra vez hasta que las secuencias resultantes sean secuencias de un punto. Para $N = 2^s$, el diezmado puede realizarse $s = \log_2 N$ veces. Por lo tanto, el número total de multiplicaciones complejas se reduce a $(N/2) \log_2 N$. El número de sumas complejas es $N \log_2 N$. En la tabla siguiente se presenta una comparación entre el número de multiplicaciones complejas usando la FFT y el cálculo directo de la DFT.

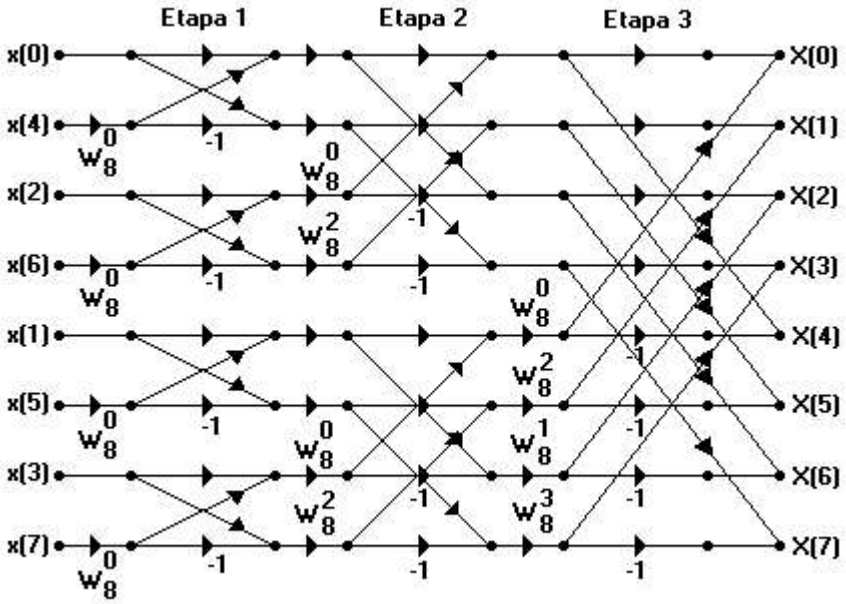
Número de puntos N	Multiplicaciones complejas N^2 en el cálculo directo	Multiplicaciones complejas con el algoritmo FFT $(N/2) \log_2 N$	Factor de mejora de la velocidad
4	16	4	4.0
8	64	12	5.3
16	256	32	8.0
32	1024	80	12.8
64	4096	192	21.3
128	16384	448	36.6

256	65536	1024	64.0
512	265144	2304	113.8
1024	1048576	5120	204.8

La mejora es notable para un número de puntos elevado.

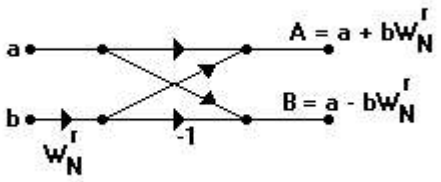
Con fines ilustrativos, en la figura siguiente se representa el cálculo de una DFT de N = 8 puntos. Se observa que el cálculo se realiza en tres etapas, se comienza con el cálculo de cuatro DFTs de dos puntos, después dos de cuatro puntos, y finalmente, una de ocho puntos.

La forma de combinar las DFTs más pequeñas para obtener la DFT mayor se muestra a continuación para N = 8.

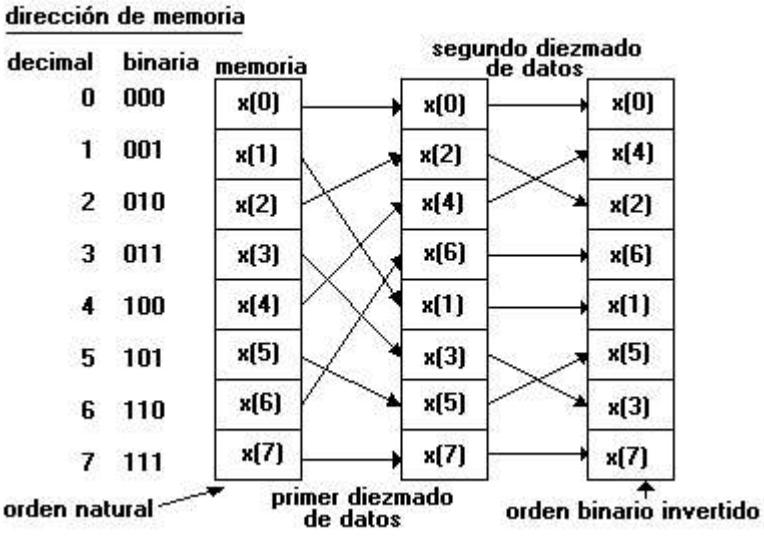


Obsérvese que el cálculo básico que se realiza en cada etapa, como se muestra en la figura, consiste en coger dos números complejos, digamos (a , b), multiplicar b por W_N^r, y sumar y restar el producto obtenido de a para obtener los dos nuevos números complejos (A , B). Este cálculo básico se denomina mariposa, dado que el diagrama de flujo recuerda a una mariposa.

En general, cada mariposa implica una multiplicación y dos sumas complejas. Para $N = 2^s$, tenemos N/2 mariposas por cada etapa del proceso de cálculo y $\log_2 N$ etapas. Por lo tanto, como indicamos anteriormente, el número total de multiplicaciones complejas es $(N/2) \log_2 N$ y el de sumas complejas es $N \log_2 N$. El diagrama de flujo de la mariposa es el siguiente:



Cada vez que se realiza una mariposa sobre un par de números complejos (a , b) para obtener (A , B), no es necesario guardar el par de entrada (a , b). Por tanto, podemos guardar el resultado (A , B) en las mismas posiciones que (a , b). En consecuencia, necesitamos una cantidad fija de memoria, en concreto, 2N registros de almacenamiento, para guardar los resultados (N números complejos) de los cálculos correspondientes a cada etapa. Dado que durante todo el cálculo de la DFT de N puntos se usan las mismas 2N posiciones de memoria decimos que los cálculos se hacen in situ.



Un segundo aspecto importante tiene que ver con el orden de la secuencia de entrada después de haber sido diezmada (s-1) veces. Por ejemplo, si consideramos N = 8, sabemos que el primer diezmado nos da la secuencia x(0), x(2), x(4), x(6), x(1), x(3), x(5), x(7), y que el segundo diezmado da lugar a la secuencia x(0), x(4), x(2), x(6), x(1), x(5), x(3) y x(7). Este mezclado de

la secuencia de entrada tiene un orden bien determinado. Si se expresa el índice n, de la secuencia x(n), en forma binaria, vemos que podemos obtener el orden de la secuencia diezmada leyendo la representación binaria de n al revés. Por lo tanto, el punto x(3)=x('011') se encontrará en la posición m = '110' o m = 6 de la secuencia diezmada. De esta forma, decimos que la secuencia x(n) después del diezmado se almacena en orden binario invertido.

$n_2\ n_1\ n_0$	→	$n_0\ n_2\ n_1$	→	$n_0\ n_1\ n_2$
000	→	000	→	000
001	→	100	→	100
010	→	001	→	010
011	→	101	→	110
100	→	010	→	001
101	→	110	→	101
110	→	011	→	011
111	→	111	→	111

Si la secuencia de datos se almacena en orden binario invertido, y las mariposas se realizan in situ, la DFT resultante, X(k), se obtiene en orden natural (es decir, k= 0, 1, ..., N-1). Por otra parte, debemos decir que es posible realizar el algoritmo para la FFT de manera que la entrada esté en orden natural y la salida en orden binario invertido. Además, se puede imponer la restricción de que tanto la entrada como la salida estén en orden natural, y deducir un algoritmo para la FFT en el que los cálculos no se hagan in situ. Tal algoritmo requeriría almacenamiento adicional, por lo tanto, no lo vamos a considerar.
La necesidad de memoria es otro de los factores a considerar. Si los cálculos se realizan in situ, el número de posiciones de memoria que se necesitan es 2N dado que los números son complejos. Sin embargo, podemos doblar la memoria, pasando a ser de 4N y, simplificar, por tanto, las operaciones de control y direccionamiento del algoritmo para la FFT. En este caso, simplemente alternamos entre sucesivas etapas del algoritmo el uso de los dos conjuntos de memoria. La doble memoria nos permite, además, tener tanto la secuencia de salida como la de entrada en orden normal.

2.3.4 UTILIZACIÓN PARTICULAR DEL ALGORITMO FFT EN BASE-2 PARA EL CÁLCULO DE LA TRANSFORMADA DE FOURIER DE 512 MUESTRAS DE LA SEÑAL DE AUDIO

Realizar un traspaso de la información de la señal en el dominio del tiempo al dominio de la frecuencia, resulta fundamental para obtener los parámetros básicos a partir de los cuales se podrán realizar caracterizaciones espectrales de los sonidos. Desde el conversor analógico-digital se reciben 512 muestras de la señal de audio. Sobre estas muestras se aplicará el algoritmo FFT en base dos explicado anteriormente. De esta forma, se conseguirán 512 coeficientes espectrales complejos, que contienen la misma información que las 512 muestras de la señal de audio, pero

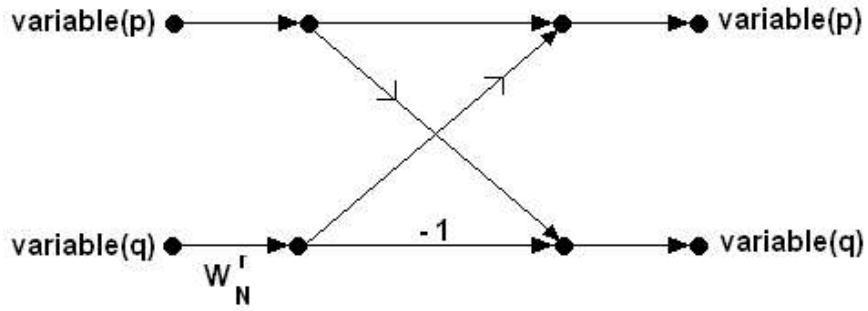
esta información se representa en el dominio de la frecuencia, donde las características vocálicas de la señal de voz quedan patentes de una forma más clara.

El algoritmo FFT toma como vector de entradas las 512 muestras de la señal de audio captadas por el conversor analógico digital presentadas en orden binario inverso, para que así, tras la realización de los cálculos de la Transformada Rápida de Fourier, los coeficientes cepstrales obtenidos tengan el orden correcto. El orden binario inverso es el que resulta de invertir el orden de los bits que indican la posición del número en cuestión. Si tratamos la posición 157, que en binario se corresponde con el número "010011101" de 9 bits (512 posiciones diferentes), el orden binario inverso del número binario anterior es "101110010", que se corresponde con la posición 370. Es decir, 370 es la posición en orden binario inverso en el vector de entrada del algoritmo FFT donde se almacenará la muestra número 157.

Las mismas posiciones de memoria donde se encuentra almacenado el vector de entrada con las muestras digitales de la señal de audio en orden binario inverso se utilizarán para almacenar los resultados intermedios y finales del algoritmo FFT. En estas posiciones de memoria quedarán almacenados al final de la ejecución del algoritmo los 512 coeficientes espectrales complejos de Fourier en forma de vector, donde las posiciones en el vector guardan el mismo orden que los coeficientes.

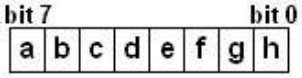
El algoritmo FFT de 512 muestras se puede descomponer en 9 etapas ($\log_2(512) = 9$). En cada una de las etapas se ejecutan 256 unidades de cálculo llamadas mariposas, dado que el esquema operacional de esta unidad de cálculo recuerda a una mariposa. En los cálculos realizados por la mariposa se toman dos variables complejas desde las posiciones de almacenamiento p y q (p y q son dos valores comprendidos entre la posición 0 y la 511 que sólo dependen del número de la mariposa calculada (0-255) y del número de la etapa en el algoritmo FFT (1-9)). A la variable almacenada en la posición q se la multiplica por el factor de fase $W_{512}^{r} = e^{-j\,r\,2\pi / 512}$ (el índice r depende del número de la mariposa calculada (0-255) y del número de la etapa en el algoritmo FFT (1-9)) de modo que no se modifica su módulo, pero sí lo hace su fase. La nueva variable almacenada en la posición p es el resultado de sumar el producto de la variable almacenada en la posición q por el factor de fase W_{512}^{r} a la variable almacenada en la posición p. La nueva variable almacenada en la posición q es el resultado de restar el producto de la variable almacenada en la posición q por el factor de fase W_{512}^{r} a la variable almacenada en la posición p. En cada etapa se realizan los cálculos de 256 mariposas para actualizar las 512 variables complejas almacenadas. Una vez concluidos los cálculos de las 256 mariposas en la etapa novena del algoritmo FFT quedarán en las posiciones de almacenamiento los valores complejos de los 512 coeficientes espectrales de Fourier.

CÁLCULO BÁSICO EN UNA MARIPOSA



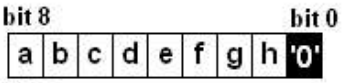
$$W_N^r = e^{-j\,r\,2\pi/512}$$

Las posiciones p y q desde donde se leerán las variables complejas que serán actualizadas por los cálculos de la mariposa dependen del número de la etapa en curso (1-9) en el algoritmo FFT y del número de la mariposa actual en ejecución (0-255). En la etapa primera, el índice p es el resultado de multiplicar por dos el número de la mariposa, y el índice q es el resultado de sumarle uno al número de la mariposa multiplicado por dos.

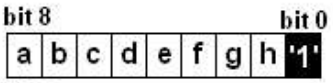


NÚMERO DE LA MARIPOSA EN BINARIO **(0 - 255)**

ÍNDICE P EN BINARIO EN LA PRIMERA ETAPA

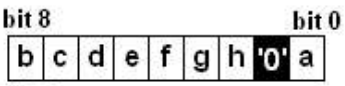


ÍNDICE Q EN BINARIO EN LA PRIMERA ETAPA

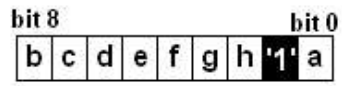


En la etapa segunda los nuevos p y q se calculan a partir de los de la primera etapa, trasladando el bit que está en la posición 8 a la posición 0 y desplazando hacia la izquierda los restantes bits.

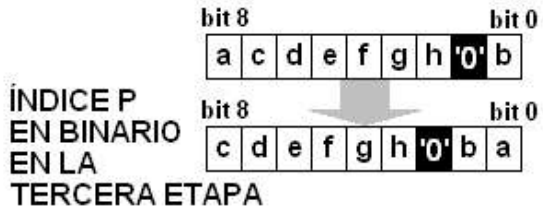
ÍNDICE P EN BINARIO EN LA SEGUNDA ETAPA

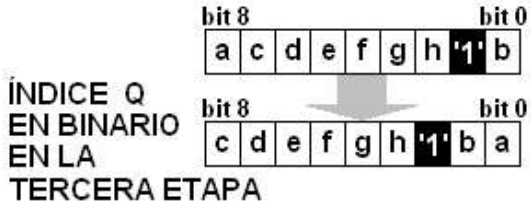


ÍNDICE Q EN BINARIO EN LA SEGUNDA ETAPA

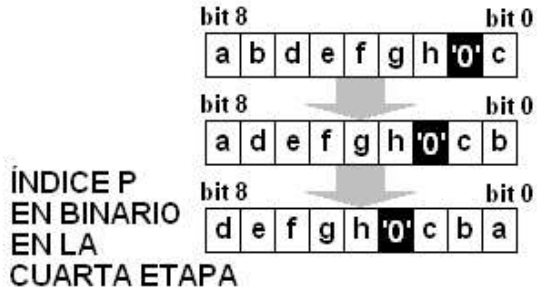


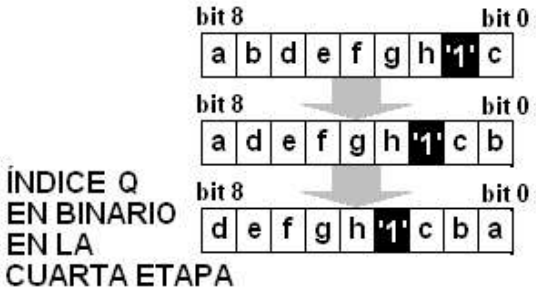
La tercera etapa, los índices p y q se obtienen a partir de los de la primera etapa, manteniendo el bit de la posición 8 y trasladando el bit de la posición 7 a la posición 0. El resto de los bits se trasladan a la izquierda. Al resultado anterior se le aplica el mismo proceso que en la etapa anterior, trasladando el bit que está en la posición 8 a la posición 0 y desplazando hacia la izquierda los restantes bits.



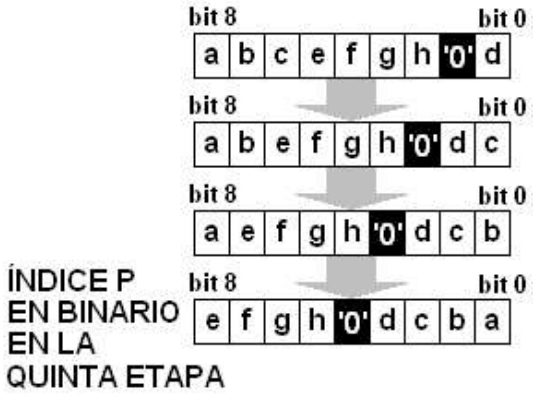


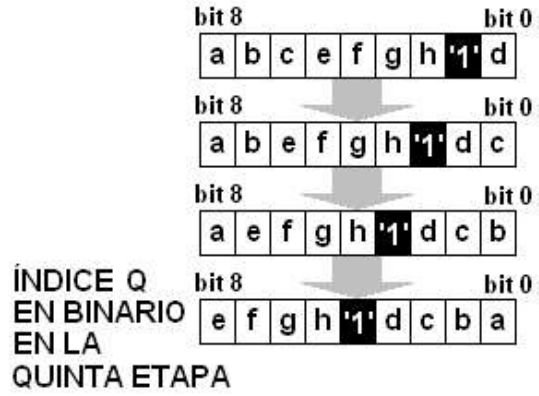
En la etapa cuarta, los índices p y q se obtienen a partir de los de la primera, manteniendo los bits de las posiciones número 8 y 7, y trasladando el bit de la posición 6 a la posición 0. El resto de bits se trasladan a la izquierda. Al resultado anterior se le aplica el mismo proceso que en la etapa anterior: Un primer proceso mantiene el bit de la posición 8 y traslada el bit de la posición 7 a la posición 0, desplazando hacia la izquierda los bits restantes. Un segundo proceso traslada el bit de la posición 8 a la posición 0, desplazando hacia la izquierda los bits restantes.



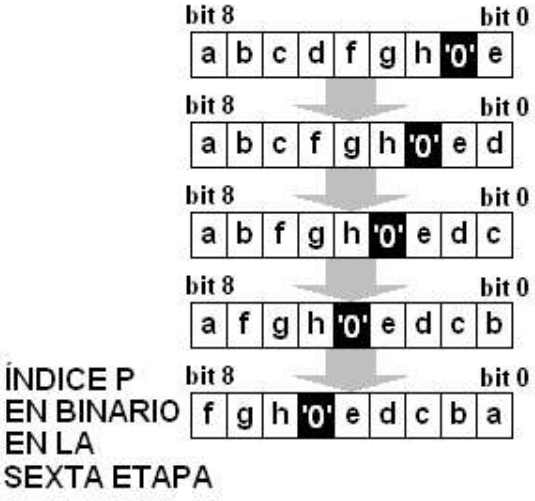


En la etapa quinta, los índices p y q se obtienen a partir de los de la primera, manteniendo los bits de las posiciones número 8, 7 y 6, y trasladando el bit de la posición 5 a la posición 0. El resto de bits se trasladan a la izquierda. Al resultado anterior se le aplica el mismo proceso que en la etapa anterior: Un primer proceso mantiene los bits de las posiciones 8 y 7 y traslada el bit de la posición 6 a la posición 0, desplazando hacia la izquierda los bits restantes. Un segundo proceso mantiene el bit de la posición 8 y traslada el bit de la posición 7 a la posición 0, desplazando hacia la izquierda los bits restantes. Un tercer proceso traslada el bit de la posición 8 a la posición 0, desplazando hacia la izquierda los bits restantes.

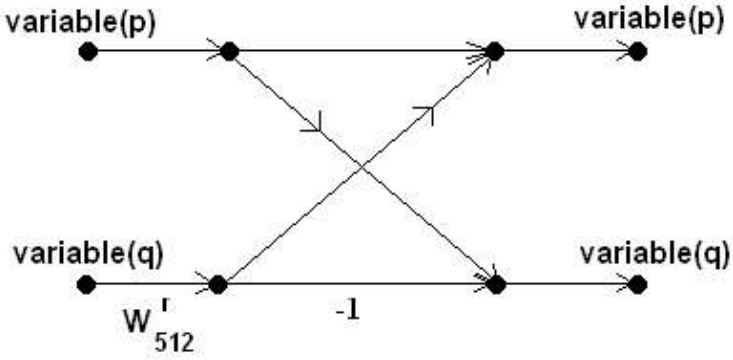




En la etapa sexta, los índices p y q se obtienen a partir de los de la primera, manteniendo los bits de las posiciones número 8, 7, 6 y 5, y trasladando el bit de la posición 4 a la posición 0. El resto de bits se trasladan a la izquierda. Al resultado anterior se le aplica el mismo proceso que en la etapa anterior: Un primer proceso mantiene los bits de las posiciones 8, 7 y 6 y traslada el bit de la posición 5 a la posición 0, desplazando hacia la izquierda los bits restantes. Un segundo proceso mantiene los bits de las posiciones 8 y 7 y traslada el bit de la posición 6 a la posición 0, desplazando hacia la izquierda los bits restantes. Un tercer proceso mantiene el bit de la posición 8 y traslada el bit de la posición 7 a la posición 0, desplazando hacia la izquierda los bits restantes. Un cuarto proceso traslada el bit de la posición 8 a la posición 0, desplazando hacia la izquierda los bits restantes.



El mismo mecanismo se utilizaría para calcular el índice p y q en las distintas mariposas de las etapas siguientes.



$$W_{512}^{r} = e^{-j 2 \pi r / 512}$$

En los cálculos de cada mariposa se realiza una multiplicación por un factor de fase W_{512}^{r}. El factor de fase es un número complejo de módulo unidad que realiza una modificación en la fase del dato al que multiplica. Esta variación en la fase ($- 2 \pi r / 512$ radianes) depende del índice r

y el valor del índice r sólo depende del número de la mariposa calculada (0-255) y del número de la etapa en el algoritmo FFT (1-9)).

- El índice r en la etapa novena es el número que resulta de cambiar el orden en los bits del número de la mariposa actual que se está calculando (0-255). Ejemplo: En la etapa novena y en la mariposa número 105 (que en binario es "01101001") el índice r es el número 150 (que en binario es "10010110").

- El índice r en la etapa octava es el número que resulta de cambiar el orden en los bits del número de la mariposa actual que se está calculando (0-255), desplazándolo luego hacia la izquierda un bit introduciendo un cero por la derecha, el bit más significativo desaparece. Ejemplo: En la etapa octava y en la mariposa número 234 (que en binario es "11101010") el índice r es el número 174 (que en binario es "10101110").

- El índice r en la etapa séptima es el número que resulta de cambiar el orden en los bits del número de la mariposa actual que se está calculando (0-255), desplazándolo luego hacia la izquierda dos bits introduciendo dos ceros por la derecha, los dos bits más significativos desaparecen. Ejemplo: En la etapa séptima y en la mariposa número 234 (que en binario es "11101010") el índice r es el número 92 (que en binario es "01011100").

- El índice r en la etapa sexta es el número que resulta de cambiar el orden en los bits del número de la mariposa actual que se está calculando (0-255), desplazándolo luego hacia la izquierda tres bits introduciendo tres ceros por la derecha, los tres bits más significativos desaparecen. Ejemplo: En la etapa sexta y en la mariposa número 234 (que en binario es "11101010") el índice r es el número 184 (que en binario es "10111000").

- El índice r en la etapa quinta es el número que resulta de cambiar el orden en los bits del número de la mariposa actual que se está calculando (0-255), desplazándolo luego hacia la izquierda cuatro bits introduciendo cuatro ceros por la derecha, los cuatro bits más significativos desaparecen. Ejemplo: En la etapa quinta y en la mariposa número 234 (que en binario es "11101010") el índice r es el número 112 (que en binario es "01110000").

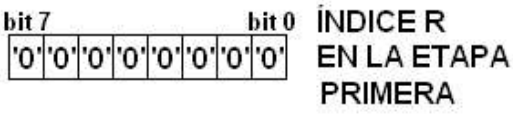
- El índice r en la etapa cuarta es el número que resulta de cambiar el orden en los bits del número de la mariposa actual que se está calculando (0-255), desplazándolo luego hacia la izquierda cinco bits introduciendo cinco ceros por la derecha, los cinco bits más significativos desaparecen. Ejemplo: En la etapa cuarta y en la mariposa número 234 (que en binario es "11101010") el índice r es el número 224 (que en binario es "11100000").

- El índice r en la etapa tercera es el número que resulta de cambiar el orden en los bits del número de la mariposa actual que se está calculando (0-255), desplazándolo luego hacia la izquierda seis bits introduciendo seis ceros por la derecha, los seis bits más significativos desaparecen. Ejemplo: En la etapa tercera y en la mariposa número 234 (que en binario es "11101010") el índice r es el número 192 (que en binario es "11000000").

- El índice r en la etapa segunda es el número que resulta de cambiar el orden en los bits del número de la mariposa actual que se está calculando (0-255), desplazándolo luego hacia la izquierda siete bits introduciendo siete ceros por la derecha, los siete bits más significativos

desaparecen. Ejemplo: En la etapa segunda y en la mariposa número 234 (que en binario es "11101010") el índice r es el número 128 (que en binario es "10000000").

- El índice r en la etapa primera es 0 ("00000000" en binario).



Con el índice r puedo averiguar el desfase provocado por el factor de fase W_{512}^{r} operativo en los cálculos de cada mariposa.

$$W_{512}^{r} = e^{-j\,2\,\pi\,r\,/\,512}$$

$$W_{512}^{r} = \cos(2\,\pi\,r\,/\,512) - j\,\mathrm{sen}(2\,\pi\,r\,/\,512)$$

Tras la ejecución de las 256 mariposas de la etapa novena quedarán almacenados los coeficientes espectrales de Fourier de la señal de audio captada por el conversor analógico-digital. Estos coeficientes se han obtenido gracias al despliegue del algoritmo de cálculo de la Transformada Rápida de Fourier.

2.3.5 SIMULACIÓN MATLAB®.

Antes de la implementación física del algoritmo de cálculo de la Transformada Rápida de Fourier se hace necesaria una simulación externa que permita comprobar el correcto funcionamiento de la solución adoptada. Gracias a un diseño en VHDL implementado de forma particular sobre la FPGA para la comprobación del sistema se almacenaron 512 muestras digitales de cinco pronunciaciones vocálicas diferentes en la memoria FLASH externa AT49F002 de ATMEL presente en la tarjeta XSA. Con una función especial ejecutada en lenguaje MATLAB® es posible leer la información contenida en los archivos hexadecimales extraídos desde la memoria FLASH con las 512 muestras procedentes del CODEC TLC320AD77C de las cinco pronunciaciones vocálicas. La ejecución de diversas funciones en lenguaje MATLAB® sobre las muestras leídas desde los archivos hexadecimales permitirá comprobar el funcionamiento de los

algoritmos que se implementarán sobre la FPGA. Queda así patente el objetivo de la simulación MATLAB®.

El formato de las muestras digitales tomadas por el conversor analógico-digital TLC320AD77C de Texas Instruments era de un número entero binario de 20 bits en complemento a dos. Fueron almacenadas en la memoria FLASH externa AT49F002 de ATMEL presente en la tarjeta XSA y gracias a la aplicación 'gxsload' de XESS Corp. la información fue volcada en archivos hexadecimales para cada una de las posibles pronunciaciones vocálicas. Estas muestras binarias fueron extraídas de los archivos hexadecimales y transcodificadas a un formato flotante de 32 bits cada una para que pudieran ser tratadas como datos en la ejecución de las distintas funciones desarrolladas en MATLAB® para la simulación. También se realizó un escalado para que los valores flotantes de las muestras estuvieran comprendidos entre -1 y +1, como en una señal tipo WAV. Para facilitar el acceso a los datos, fueron almacenados en archivos con extensión '.dav', como por ejemplo el fichero 'vocal.dav'.

Para obtener los coeficientes espectrales de las muestras almacenadas en uno de los archivos, por ejemplo 'vocal.dav', correspondientes a las muestras digitales tomadas por el CODEC TLC320AD77C sobre una pronunciación vocálica, tendría sólo que leer la información almacenada en el archivo y aplicar sobre estas muestras el algoritmo de cálculo de la Transformada Rápida de Fourier. Si se quisieran obtener los coeficientes espectrales de las muestras de una determinada pronunciación vocálica se ejecutaría sobre la plataforma MATLAB® el siguiente código:

```
%Se proceda a la lectura de 512 datos flotantes almacenados en el archivo 'vocal.dav',
%correspondientes a 512 muestras de una determinada pronunciación vocálica. En el vector 'muestras'
%se almacenarán las muestras para su posterior proceso.
fid=fopen('vocal.dav','r');
preci='float32';
muestras=fread(fid,512,preci);
fclose(fid);

%La función 'mi_fft.m' toma como entrada el vector de 512 muestras y realiza la Transformada Rápida
%de Fourier proporcionando como salida el vector de 'coef_fourier' cuyos elementos ofrecen dicha
%transformada. Este vector nos informa de las características espectrales de la señal introducida.
  coef_fourier=mi_fft(muestras);
```

Es la función 'mi_fft' la que realiza la Transformada Rápida de Fourier sobre las muestras de la señal de audio y obtiene los coeficientes espectrales de Fourier que caracterizan a la señal. El siguiente código sintetiza de forma clara el algoritmo ejecutado tras la llamada a la función.

mi_fft.m

```
function [salida]=mi_fft(entrada)
%La función 'mi_fft.m' toma como entrada un vector de 2^M  muestras y realiza la Transformada Rápida
%de Fourier proporcionando como salida un vector de 2^M  elementos con dicha transformada. Este
%vector nos proporciona las características espectrales de la señal introducida.

dimensiones=size(entrada);
N=max(dimensiones);
%número de etapas de la FFT
M=log2(N);

%En primer lugar, la función guarda las entradas en la memoria, pero ordenando las muestras en las
%posiciones binarias inversas de las posiciones en que son recibidas. El orden binario inverso es el
%que resulta de invertir el orden de los bits que indican la posición del número en cuestión. Si
%tratamos la posición 157, que en binario se corresponde con el número “010011101” de 9 bits (512
%posiciones diferentes), el orden binario inverso del número binario anterior es “101110010”, que se
%corresponde con la posición 370. Es decir, 370 es la posición en orden binario inverso en el vector de
%entrada del algoritmo FFT donde se almacenará la muestra número 157.
for i=1:1:N
  i_binario=pasar_modulo_binario(i-1,M);
  for j=1:1:M
    i_inv_binario(j)=i_binario(M+1-j);
  end
  i_inv=pasar_binario_modulo(i_inv_binario,M);
  memoria(i)=entrada(i_inv+1);
end

%Recorremos las M etapas del algoritmo. El algoritmo FFT de 512 muestras se puede descomponer en
%9 etapas ( log2(512) = 9 ). En cada una de las etapas se ejecutan 256 unidades de cálculo llamadas
%mariposas, dado que el esquema operacional de esta unidad de cálculo recuerda a una mariposa.
for j=1:1:M
  texto=sprintf('Calculando Fase %d-%d MI FFT...',j,M);
  disp(texto)
  %En cada etapa realizamos N/2 cálculos de la mariposa:
  %memoria(p)=memoria(p) + ( W512r * memoria(q) );
  %memoria(q)=memoria(q) + ( W512r * memoria(p) );
  %En los cálculos realizados por la mariposa se toman dos variables complejas desde las posiciones
  %de almacenamiento p y q (p y q son dos valores comprendidos entre la posición 0 y la 511 que sólo
  %dependen del número de la mariposa calculada (0-255) y del número de la etapa en el algoritmo
  %FFT (1-9)). A la variable almacenada en la posición q se la multiplica por el factor de fase W512r =
  %e^(-j r 2 pi / 512) (el índice r depende del número de la mariposa calculada (0-255) y del número de
  %la etapa en el algoritmo FFT (1-9) ) de modo que no se modifica su módulo, pero sí lo hace su fase.
  %La nueva variable almacenada en la posición p es el resultado de sumar el producto de la variable
  %almacenada en la posición q por el factor de fase W512r a la variable almacenada en la posición p.
  %La nueva variable almacenada en la posición q es el resultado de restar el producto de la variable
  %almacenada en la posición q por el factor de fase W512r a la variable almacenada en la posición p.
  %En cada etapa se realizan los cálculos de 256 mariposas para actualizar las 512 variables complejas
  %almacenadas. Una vez concluidos los cálculos de las 256 mariposas en la etapa novena del
  %algoritmo FFT quedarán en las posiciones de almacenamiento los valores complejos de los 512
  % coeficientes espectrales de Fourier.
  for i=1:1:(N/2)
    %En cada mariposa obtengo los índices p y q de las posiciones de memoria que se actualizarán con
    %los valores guardados en esas mismas posiciones de memoria. La posiciones p y q desde donde
```

mi_fft.m

```
    %se leerán las variables complejas que serán actualizadas por los cálculos de la mariposa
    %dependen del número de la etapa en curso (1-9) en el algoritmo FFT y del número de la mariposa
    %actual en ejecución (0-255). La función "obten_pq" resuelve los valores de los índices p y q en
    %función del número de la etapa y del número de la mariposa.
    [p,q]=obten_pq(i-1,j,M);
    %Se almacenan en dos variables auxiliares los valores almacenados en las posiciones p y q de la
    %memoria. Como los vectores en MATLAB® no tienen el índice 0, sino que empiezan con el índice
    %1, se incrementa en una unidad el índice de las posiciones de las variables para poder realizar un
    %equivalente en MATLAB® del direccionamiento adoptado.
    mem_p=memoria(p+1);
    mem_q=memoria(q+1);
    %En los cálculos de cada mariposa se realiza una multiplicación por un factor de fase W512r. El
    %factor de fase es un número complejo de módulo unidad que realiza una modificación en la fase
    %del dato al que multiplica. Esta variación en la fase ( - 2 pi r / 512  radianes ) depende del
    %índice r y el valor del índice r sólo depende del número de la mariposa calculada (0-255) y del
    %número de la etapa en el algoritmo FFT (1-9)). La función "obten_r" resuelve el valor del índice
    %r en función del número de la etapa y del número de la mariposa.
    r=obten_r(i-1,j,M);
    %La función "obten_w_r_N(r,N)" calcula en función del índice r el factor de fase por el que se
    %multiplicará la variable almacenada en la posición q.
    %factor de fase = coseno( 2*pi*r/512 ) - j seno( 2*pi*r/512 );
    w_r_N=obten_w_r_N(r,N);
    %La nueva variable almacenada en la posición p es el resultado de sumar el producto de la variable
    %almacenada en la posición q por el factor de fase W512r a la variable almacenada en la posición
    %p. La nueva variable almacenada en la posición q es el resultado de restar el producto de la
    %variable almacenada en la posición q por el factor de fase W512r a la variable almacenada en la
    %posición p. Como los vectores en MATLAB® no tienen el índice 0, sino que empiezan con el índice
    %1, se incrementa en una unidad el índice de las posiciones de las variables para poder realizar un
    %equivalente en MATLAB® del direccionamiento adoptado.
    memoria(p+1)=mem_p+(w_r_N*mem_q);
    memoria(q+1)=mem_p-(w_r_N*mem_q);
  end
end

%Tras la ejecución de las 256 mariposas de la etapa novena quedarán almacenados los coeficientes
%espectrales de Fourier de la señal de audio captada por el conversor analógico-digital. Estos
%coeficientes se han obtenido gracias al despliegue del algoritmo de cálculo de la Transformada
%Rápida de Fourier. Estos valores son mostrados en el vector de salida de la función.
for i=1:1:N
  salida(i)=memoria(i);
end
```

La función anterior utiliza en su funcionamiento a otras funciones auxiliares. En primer lugar, se utiliza la función “pasar_modulo_binario” utilizada para invertir el orden en el que las muestras son almacenadas en la memoria. Como el orden utilizado es el orden binario inverso, es útil realizar una conversión del índice desde el formato decimal al formato binario.

pasar_modulo_binario.m

```
function [num_binario]=pasar_modulo_binario(modulo,dimension)
%Realiza la conversión del número decimal positivo introducido como parámetro de entrada al formato
%binario.
%INTERFAZ:
%MODULO: número decimal positivo sobre el que la función calculará su correspondiente en binario.
%DIMENSION: número de bits del número en formato binario.
%NUM_BINARIO: cadena de caracteres que contiene el número positivo en formato binario.
%Ejemplo:
%La ejecución de la función: pasar_modulo_binario(187,9)
%Proporciona la cadena de caracteres: "010111011" que se corresponde con la representación binaria
%en 9 bits del número 187

for i=1:1:dimension
  numero=2^(dimension-i);
  if modulo>=numero
    valor='1';
    modulo=modulo-numero;
  else
    valor='0';
  end
  num_binario(i)=valor;
end
```

Es sencillo ahora obtener el orden binario inverso, cambiando la posición de los bits. El orden binario inverso es el que resulta de invertir el orden de los bits de la representación binaria del índice que indica la posición en memoria. Si tratamos la posición 157, que en binario se corresponde con el número “010011101” de 9 bits (512 posiciones diferentes), el orden binario inverso del número binario anterior es “101110010”, que se corresponde con la posición 370. Es decir, 370 es la posición en orden binario inverso en el vector de entrada del algoritmo FFT donde se almacenará la muestra número 157. Para poder utilizar el índice en el orden binario inverso hay que realizar una conversión del nuevo índice en formato binario inverso al formato decimal. La función “pasar_binario_modulo” se utiliza a este propósito.

pasar_binario_modulo.m

```
function [modulo]=pasar_binario_modulo(num_binario,dimension)
%Realiza la conversión del número binario introducido como parámetro de entrada desde el formato
%binario al formato decimal.
%INTERFAZ:
%NUM_BINARIO: cadena de caracteres que contiene el número positivo en binario que se pasará a
%formato decimal.
%DIMENSION: número de bits del número en formato binario.
%MODULO: formato decimal del número binario introducido a la entrada.
```

pasar_binario_modulo.m

```
%MODULO: formato decimal del número binario introducido a la entrada.
%Ejemplo:
%La ejecución de la función: pasar_binario_modulo('00110110101',11)
%Proporciona el número decimal: 437 que se corresponde con la representación decimal del número
%positivo binario de 11 bits '00110110101'

suma=0;
for i=1:1:dimension
  if num_binario(i)=='1'
    suma=suma+(2^(dimension-i));
  end
end
modulo=suma;
```

Una vez iniciado el algoritmo de cálculo de la Transformada Rápida de Fourier la función "obten_pq" calcula las posiciones de memoria implicadas en los cálculos de la mariposa actual.

obten_pq.m

```
function [p,q]=obten_pq(indice_i,indice_j,M)
%Proporciona los valores p y q dependiendo del número de la mariposa que se esté calculand
%(indice_i), del número de etapas totales (M) y de la etapa en curso (indice_j)

i_binario=pasar_modulo_binario(indice_i*2,M);
p_binario=i_binario;
q_binario=i_binario;
q_binario(M)='1';

%La posiciones p y q desde donde se leerán las variables complejas que serán actualizadas por los
%cálculos de la mariposa dependen del número de la etapa en curso (1-9) en el algoritmo FFT  y del
%número de la mariposa actual en ejecución (0-255). En la etapa primera, el índice p es el resultado
%de multiplicar por dos el número de la mariposa, y el índice q es el resultado de sumarle uno al
%número de la mariposa multiplicado por dos.
%En la etapa segunda los nuevos p y q se calculan a partir de los de la primera etapa, trasladando el
%bit que está en la posición 8 a la posición 0 y desplazando hacia la izquierda los restantes bits.
%La tercera etapa, los índices p y q se obtienen a partir de los de la primera etapa, manteniendo el
%bit de la posición 8 y trasladando el bit de la posición 7 a la posición 0. El resto de los bits se
%trasladan a la izquierda. Al resultado anterior se le aplica el mismo proceso que en la etapa anterior,
%trasladando el bit que está en la posición 8 a la posición 0 y desplazando hacia la izquierda los
%restantes bits.
%En la etapa cuarta, los índices p y q se obtienen a partir de los de la primera, manteniendo los bits
%de las posiciones número 8 y 7, y trasladando el bit de la posición 6 a la posición 0. El resto de
%bits se trasladan a la izquierda. Al resultado anterior se le aplica el mismo proceso que en la etapa
```

obten_pq.m

%anterior: Un primer proceso mantiene el bit de la posición 8 y traslada el bit de la posición 7 a la %posición 0, desplazando hacia la izquierda los bits restantes. Un segundo proceso traslada el bit de %la posición 8 a la posición 0, desplazando hacia la izquierda los bits restantes.
%En la etapa quinta, los índices p y q se obtienen a partir de los de la primera, manteniendo los bits %de las posiciones número 8, 7 y 6, y trasladando el bit de la posición 5 a la posición 0. El resto de %bits se trasladan a la izquierda. Al resultado anterior se le aplica el mismo proceso que en la etapa %anterior: Un primer proceso mantiene los bits de las posiciones 8 y 7 y traslada el bit de la posición %6 a la posición 0, desplazando hacia la izquierda los bits restantes. Un segundo proceso mantiene el %bit de la posición 8 y traslada el bit de la posición 7 a la posición 0, desplazando hacia la izquierda %los bits restantes. Un tercer proceso traslada el bit de la posición 8 a la posición 0, desplazando %hacia la izquierda los bits restantes.
%En la etapa sexta, los índices p y q se obtienen a partir de los de la primera, manteniendo los bits de %las posiciones número 8, 7, 6 y 5, y trasladando el bit de la posición 4 a la posición 0. El resto de %bits se trasladan a la izquierda. Al resultado anterior se le aplica el mismo proceso que en la etapa %anterior: Un primer proceso mantiene los bits de las posiciones 8 7 y 6 y traslada el bit de la %posición 5 a la posición 0, desplazando hacia la izquierda los bits restantes. Un segundo proceso %mantiene los bits de las posiciones 8 y 7 y traslada el bit de la posición 6 a la posición 0, %desplazando hacia la izquierda los bits restantes. Un tercer proceso mantiene el bit de la posición 8 %y traslada el bit de la posición 7 a la posición 0, desplazando hacia la izquierda los bits restantes. %Un cuarto proceso traslada el bit de la posición 8 a la posición 0, desplazando hacia la izquierda los %bits restantes.
%El mismo mecanismo se utilizaría para calcular los índices p y q en las distintas mariposas de las %etapas siguientes.
%La posiciones p y q desde donde se leerán las variables complejas que serán actualizadas por los %cálculos de la mariposa dependen del número de la etapa en curso (1-9) en el algoritmo FFT y del %número de la mariposa actual en ejecución (0-255). En la etapa primera, el índice p es el resultado %de multiplicar por dos el número de la mariposa, y el índice q es el resultado de sumarle uno al %número de la mariposa multiplicado por dos.
%En la etapa segunda los nuevos p y q se calculan a partir de los de la primera etapa, trasladando el %bit que está en la posición 8 a la posición 0 y desplazando hacia la izquierda los restantes bits.
%La tercera etapa, los índices p y q se obtienen a partir de los de la primera etapa, manteniendo el %bit de la posición 8 y trasladando el bit de la posición 7 a la posición 0. El resto de los bits se %trasladan a la izquierda. Al resultado anterior se le aplica el mismo proceso que en la etapa anterior, %trasladando el bit que está en la posición 8 a la posición 0 y desplazando hacia la izquierda los %restantes bits.
%En la etapa cuarta, los índices p y q se obtienen a partir de los de la primera, manteniendo los bits %de las posiciones número 8 y 7, y trasladando el bit de la posición 6 a la posición 0. El resto de %bits se trasladan a la izquierda. Al resultado anterior se le aplica el mismo proceso que en la etapa %anterior: Un primer proceso mantiene el bit de la posición 8 y traslada el bit de la posición 7 a la %posición 0, desplazando hacia la izquierda los bits restantes. Un segundo proceso traslada el bit de %la posición 8 a la posición 0, desplazando hacia la izquierda los bits restantes.
%En la etapa quinta, los índices p y q se obtienen a partir de los de la primera, manteniendo los bits %de las posiciones número 8, 7 y 6, y trasladando el bit de la posición 5 a la posición 0. El resto de %bits se trasladan a la izquierda. Al resultado anterior se le aplica el mismo proceso que en la etapa %anterior: Un primer proceso mantiene los bits de las posiciones 8 y 7 y traslada el bit de la posición %6 a la posición 0, desplazando hacia la izquierda los bits restantes. Un segundo proceso mantiene el %bit de la posición 8 y traslada el bit de la posición 7 a la posición 0, desplazando hacia la izquierda %los bits restantes. Un tercer proceso traslada el bit de la posición 8 a la posición 0, desplazando %hacia la izquierda los bits restantes.
%En la etapa sexta, los índices p y q se obtienen a partir de los de la primera, manteniendo los bits de %las posiciones número 8, 7, 6 y 5, y trasladando el bit de la posición 4 a la posición 0. El resto de %bits se trasladan a la izquierda. Al resultado anterior se le aplica el mismo proceso que en la etapa

obten_pq.m

```
%anterior: Un primer proceso mantiene los bits de las posiciones 8 7 y 6 y traslada el bit de la
%posición 5 a la posición 0, desplazando hacia la izquierda los bits restantes. Un segundo proceso
%mantiene los bits de las posiciones 8 y 7 y traslada el bit de la posición 6 a la posición 0,
%desplazando hacia la izquierda los bits restantes. Un tercer proceso mantiene el bit de la posición 8
%y traslada el bit de la posición 7 a la posición 0, desplazando hacia la izquierda los bits restantes.
%Un cuarto proceso traslada el bit de la posición 8 a la posición 0, desplazando hacia la izquierda los
%bits restantes.
%El mismo mecanismo se utilizaría para calcular los índices p y q en las distintas mariposas de las
%etapas siguientes.

vector_1=p_binario;
for i=2:1:M
  for s=1:1:M
    vector_2(s)='0';
  end
  if indice_j>1
    if indice_j>=(M-i+2)
      if i<M
        for j=1:1:M-i
          vector_2(j)=vector_1(j);
        end
      end
      vector_2(M)=vector_1(M-i+1);
      if i>=2
        for j=M-i+2:1:M
          vector_2(j-1)=vector_1(j);
        end
      end
      vector_1=vector_2;
    end
  end
end

p_binario=vector_1;
%El mismo procedimiento se utiliza en la obtención del índice q
vector_1=q_binario;
for i=2:1:M
  for s=1:1:M
    vector_2(s)='0';
  end
  if indice_j>1
    if indice_j>=(M-i+2)
      if i<M
        for j=1:1:M-i
          vector_2(j)=vector_1(j);
        end
      end
      vector_2(M)=vector_1(M-i+1);
      if i>=2
        for j=M-i+2:1:M
```

obten_pq.m

```
          vector_2(j-1)=vector_1(j);
        end
      end

      vector_1=vector_2;
    end
  end
end
q_binario=vector_1;

p=pasar_binario_modulo(p_binario,M);
q=pasar_binario_modulo(q_binario,M);
```

El factor de fase utilizado en los cálculos de las mariposas depende del índice r. La función "obten_r" proporciona el valor del índice r en función del número de la mariposa y de la etapa actual en curso dentro del algoritmo de la Transformada Rápida de Fourier.

obten_r.m

```
function [r]=obten_r(indice_i,indice_j,M)
%El coeficiente r se calcula con la función 'obten_r.m'. que me proporciona el valor
%de r dependiendo del número de la mariposa que se esté calculando (indice_i) (0-255), del número
%de etapas totales (9) y de la etapa en curso (indice_j). Obtenemos un número auxiliar resultado
%de cambiar el orden de los bits del número de la mariposa que estamos calculando (0-255).
%Ejemplo: El número 31 que en binario es '00011111' se corresponde con el número 248 que
%en binario es '11111000'. El coeficiente r es este número para la última etapa (la etapa
%9). Es el número auxiliar desplazado hacia la izquierda introduciendo un cero por la
%derecha para la etapa 8. Es el número auxiliar desplazado hacia la izquierda dos
%posiciones e introduciendo dos ceros por la derecha para la etapa 7. Es el número auxiliar
%desplazado hacia la izquierda tres posiciones e introduciendo tres ceros por la derecha
%para la etapa 6. Y así sucesivamente.
i_binario=pasar_modulo_binario(indice_i,M-1);

for i=1:1:(M-1)
  i_binario_inv(i)=i_binario(M-i);
end

r_binario=i_binario_inv;
r_binario_aux=r_binario;
if indice_j<M
  for i=1:1:(M-indice_j)
    r_binario(M-1)='0';
    for j=1:1:M-2
      r_binario(j)=r_binario_aux(j+1);
```

obten_r.m

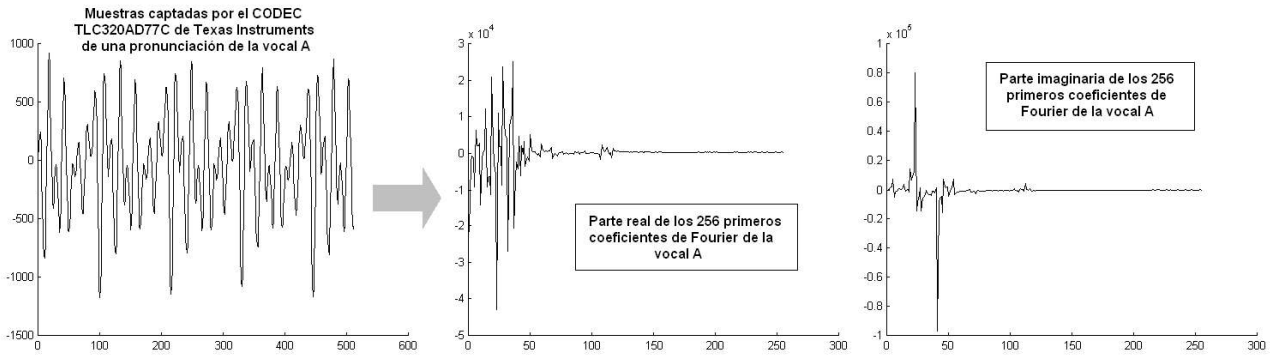
```
    end
    r_binario_aux=r_binario;
  end
end

r=pasar_binario_modulo(r_binario,M-1);
```

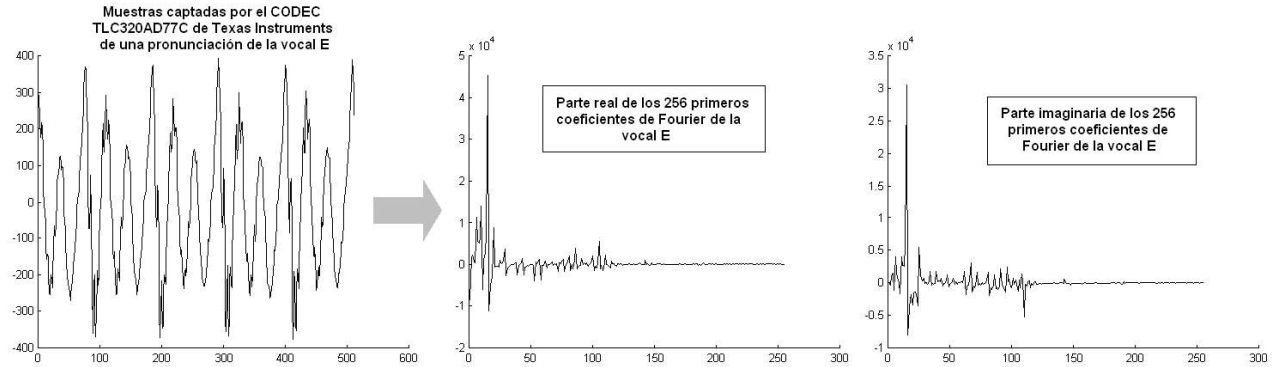
La función "obten_w_r_N" calcula el factor de fase por el que se multiplicará la segunda de las variables de los cálculos de la mariposa en función del índice de r.

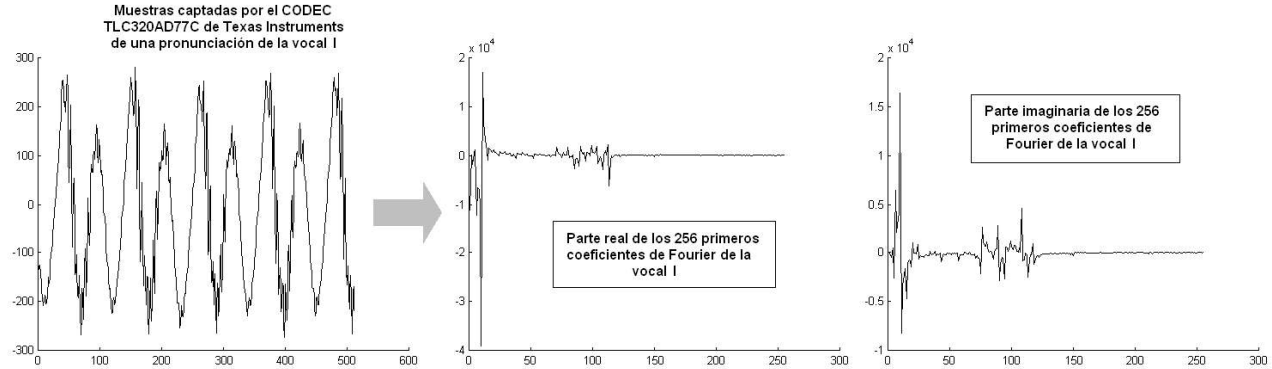
obten_w_r_N.m

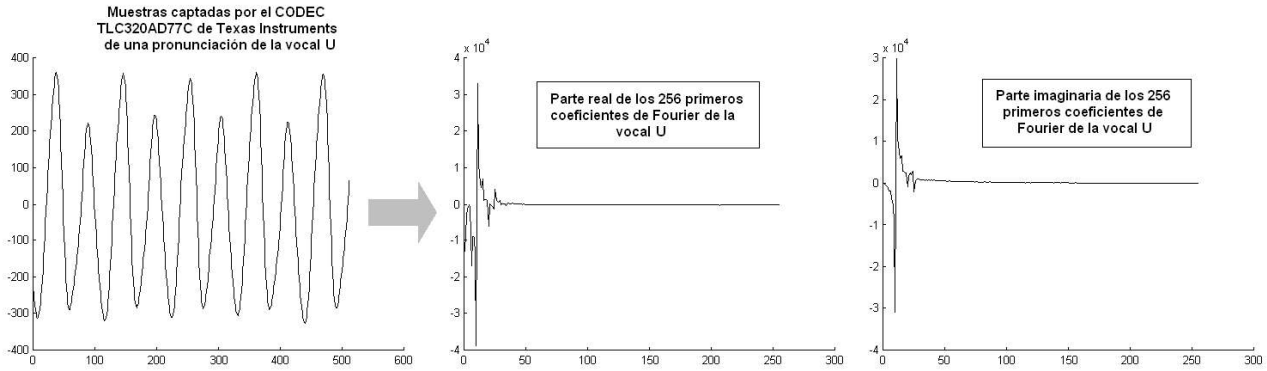
```
function [w_r_N]=obten_w_r_N(r,N)
%La función "obten_w_r_N(r,N)" calcula en función del índice r el factor de fase por el que se
%multiplicará la variable almacenada en la posición q.
%factor de fase w_r_N = coseno( 2*pi*r/512 ) - j seno( 2*pi*r/512 )
%Cuando el número de muestras N=512;

fase=2*pi*r/N;

w_r_N=cos(fase)-(sqrt(-1)*sin(fase));
```

Aplicando la Transformada Rápida de Fourier sobre las muestras vocálicas de distintas pronunciaciones puedo obtener las características espectrales particulares de cada una de las pronunciaciones. Las siguientes gráficas muestran como ejemplo los coeficientes espectrales de cinco pronunciaciones vocálicas diferentes. Los coeficientes de Fourier se han obtenido en la plataforma MATLAB® a través de la ejecución de las funciones descritas en este apartado. Como los coeficientes son complejos, es útil separar la representación de la parte real de la imaginaria. Sólo se considerarán los 256 primeros coeficientes, pues los 256 coeficientes siguientes no aportan más información al ser una imagen especular de los 256 primeros en torno al coeficiente 256. Tómense las gráficas siguientes como resultados en la simulación del algoritmo de la Transformada Rápida de Fourier utilizado para extraer las características espectrales de las muestras captadas por el CODEC TLC320AD77C de Texas Instruments.


Muestras captadas por el CODEC TLC320AD77C de Texas Instruments de una pronunciación de la vocal A
Parte real de los 256 primeros coeficientes de Fourier de la vocal A
Parte imaginaria de los 256 primeros coeficientes de Fourier de la vocal A


Muestras captadas por el CODEC TLC320AD77C de Texas Instruments de una pronunciación de la vocal E
Parte real de los 256 primeros coeficientes de Fourier de la vocal E
Parte imaginaria de los 256 primeros coeficientes de Fourier de la vocal E


Muestras captadas por el CODEC TLC320AD77C de Texas Instruments de una pronunciación de la vocal I
Parte real de los 256 primeros coeficientes de Fourier de la vocal I
Parte imaginaria de los 256 primeros coeficientes de Fourier de la vocal I



2.4 MÓDULO DE LA FFT

La información contenida en la fase de los coeficientes espectrales obtenidos con la Transformada Rápida de Fourier de las muestras de la señal de audio no es relevante desde el punto de vista del reconocimiento vocálico. Para eliminar la información relativa a la fase calculamos el módulo al cuadrado de los coeficientes complejos de Fourier. Conviene dejar el módulo al cuadrado porque así lo utilizaré en el escalado mel. Si se considera la parte real como un parámetro separado de la parte imaginaria, considerada también como parámetro, el hecho de calcular el cuadrado del módulo disminuye a la mitad el número de parámetros que definen la información vocálica. Como la fase no aporta información útil en la caracterización de la vocal, el cuadrado del módulo de los coeficientes espectrales complejos son unos parámetros más eficientes que los coeficientes de Fourier en sí. Sólo se considerarán los 256 primeros coeficientes de Fourier, ya que los siguientes 256 coeficientes son la imagen especular de los 256

primeros. Así se reduce otra vez a la mitad el número de parámetros que sintetizan la información vocálica.

2.4.1 SIMULACIÓN MATLAB®:

En la plataforma MATLAB® ya se calcularon en el apartado correspondiente a la simulación de los coeficientes de Fourier, los coeficientes espectrales obtenidos tras la realización de la Transformada Rápida de Fourier sobre las muestras digitales tomadas por el CODEC TLC320AD77C de una pronunciación vocálica. Si quisiera calcular el cuadrado de los coeficientes de Fourier ejecutaría después sobre la plataforma MATLAB® el siguiente código:

```
%La función 'obten_modulo_fourier.m' toma como entrada el vector de 512 coeficientes espectrales
%obtenidos tras realizar la Transformada Rápida de Fourier sobre las muestras digitales tomadas
%por el CODEC de una pronunciación vocálica.
%Proporciona como salida el vector 'modulo_fourier' cuyos elementos contienen el cuadrado del
%módulo de los coeficientes de Fourier. Este vector nos informa de las características espectrales
%de la señal pronunciada.
modulo_fourier=obten_modulo_fourier(coef_fourier);
```

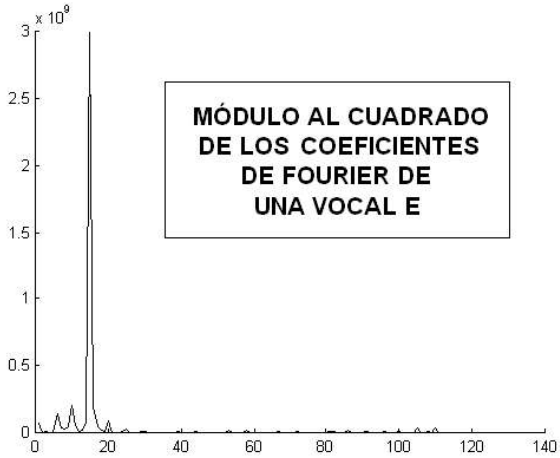
Se ha realizado una llamada a la función "obten_modulo_fourier" con los coeficientes espectrales de Fourier como parámetros de entrada. El siguiente código sintetiza de forma clara el proceso realizado tras la llamada a la función.

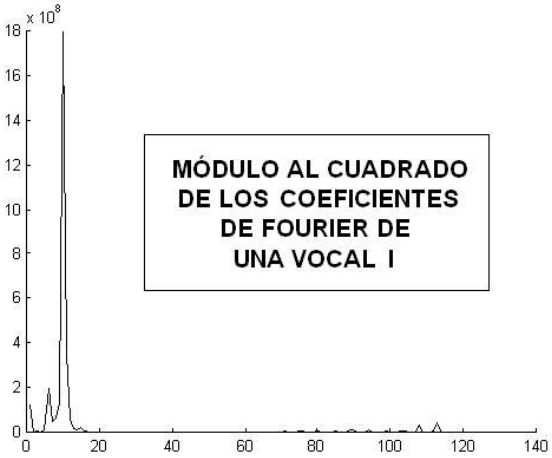
obten_modulo_fourier.m

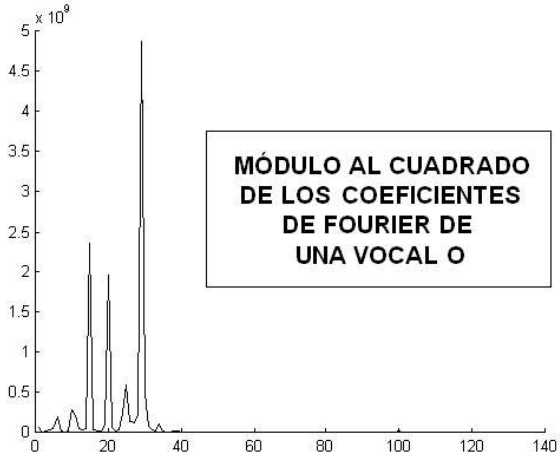
```
function [signal_modulo_fourier]=obten_modulo_fourier(signal_mi_fft)
%Con esta función calculamos el módulo al cuadrado de cada uno de los coeficientes de Fourier
%Como los coeficientes de Fourier son complejos y sólo interesa la información contenida en la
%magnitud y no la de la fase, calculo el módulo al cuadrado de cada uno de los coeficientes y lo
%propongo como parámetro más eficiente que el coeficiente complejo de Fourier en sí.

dimensiones=size(signal_mi_fft);
N=max(dimensiones);

for i=1:1:N
  signal_modulo_fourier(i)=((real(signal_mi_fft(i)))^2)+((imag(signal_mi_fft(i)))^2);
end
```

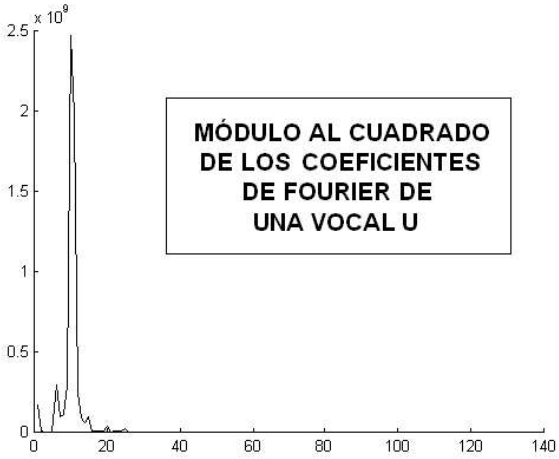
Para tener una mejor impresión del proceso realizado y observar las características espectrales de las diferentes pronunciaciones, las siguientes gráficas resumen de forma clara la información contenida en el cuadrado del módulo de los coeficientes de Fourier de cinco pronunciaciones vocálicas diferentes.


x 10^9
MÓDULO AL CUADRADO DE LOS COEFICIENTES DE FOURIER DE UNA VOCAL A


x 10^9
MÓDULO AL CUADRADO DE LOS COEFICIENTES DE FOURIER DE UNA VOCAL E


x 10^8
MÓDULO AL CUADRADO DE LOS COEFICIENTES DE FOURIER DE UNA VOCAL I


x 10^9
MÓDULO AL CUADRADO DE LOS COEFICIENTES DE FOURIER DE UNA VOCAL O

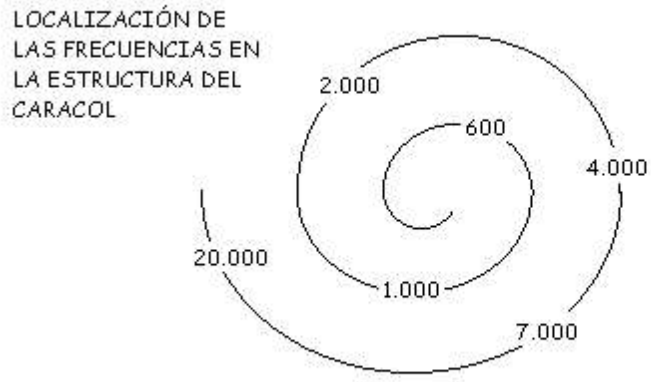

x 10^9
MÓDULO AL CUADRADO DE LOS COEFICIENTES DE FOURIER DE UNA VOCAL U

2.5 ESCALADO MEL

Imitar las habilidades naturales del ser humano es sin duda uno de los retos más complejos con los que se enfrentan los científicos e ingenieros que tratan de construir aparatos y sistemas avanzados. El reconocimiento del habla también es una de esas habilidades que se pretenden imitar para poder crear una interfaz directa entre el hombre y la máquina. La primera etapa de procesamiento que todos los reconocedores de voz deben implementar es el establecimiento de las características espectrales de los sonidos. La etapa de caracterización espectral es necesaria debido a que la información principal de la voz se codifica en el dominio de la frecuencia, mientras que las muestras de sonido que obtenemos con dispositivos electrónicos se encuentran codificadas en el dominio del tiempo (amplitudes de la señal a lo largo del tiempo).

Al pasar del dominio del tiempo al dominio de la frecuencia obtenemos parámetros que nos indican la importancia de cada frecuencia (perteneciente al ancho de banda de la señal) en el intervalo (ventana) considerado. El establecimiento de características espectrales se refiere, en esta primera etapa, al traspaso de la señal de voz desde el dominio del tiempo al dominio de la frecuencia.

El oído humano descompone las señales auditivas que le llegan en sus frecuencias fundamentales. Para ello, situado en el oído interno, se encuentra el caracol, que filtra las frecuencias de forma natural. Las ondas sonoras se introducen en esta estructura helicoidal rebotando en sus paredes y llegando, según sea la longitud de onda de cada frecuencia, más o menos al interior del caracol. La estructura del caracol no es lineal y da más relevancia a las bajas frecuencias. La capacidad de transformar la energía mecánica de la onda de sonido en impulsos eléctricos comprensibles por el cerebro disminuye a medida que aumenta la frecuencia. La estructura física del caracol impone esta restricción y condiciona de esta forma la sensibilidad auditiva humana.



Con la realización del Proyecto Fin de Carrera "Sistema autónomo para accionamiento por voz de un ratón de ordenador" pretendo crear un interfaz entre el hombre y la máquina que emula el funcionamiento de un ratón PS/2 con capacidad de comunicación con el ordenador. El control del dispositivo se realizará mediante la pronunciación de las distintas vocales de acuerdo con un protocolo previamente establecido. Queda patente entonces la necesidad de desarrollar un sistema de reconocimiento de voz donde la primera etapa de procesamiento es el establecimiento de las características espectrales de los sonidos percibidos.

Lo expuesto hasta el momento en este apartado muestra que la información más importante del habla se encuentra en sus frecuencias, sin embargo, la señal de voz se toma en tiempo utilizando convertidores analógicos/digitales en un proceso denominado muestreo. En definitiva, para imitar a la naturaleza debemos aplicar algún mecanismo que traspase a frecuencias la información existente en las muestras de voz que recogemos en el dominio del tiempo. La Transformada de Fourier es una herramienta de análisis muy utilizada en el campo científico (acústica, ingeniería biomédica, métodos numéricos, procesamiento de señal, electromagnetismo, comunicaciones, etc.) que transforma una señal representada en el dominio del tiempo al dominio de la frecuencia sin alterar su contenido de información, sólo es una forma diferente de representarla.

En apartados anteriores se utilizó un algoritmo de cálculo de la Transformada Discreta de Fourier sobre las muestras digitales tomadas por el CODEC TLC320AD77C de una pronunciación vocálica. Los módulos de los coeficientes espectrales son unos parámetros que informan sobre la importancia de cada frecuencia distribuidas de una forma lineal. Si se quiere ir más allá en el proceso de establecimiento de características espectrales se puede intentar emular el funcionamiento del caracol humano y hacer que el sistema de procesamiento del reconocedor de voz actúe de forma similar a la realizada por el oído humano.

El caracol informa al cerebro a través de impulsos eléctricos de la energía mecánica de las ondas acústicas percibida en las diversas frecuencias. La precisión en la medida de la energía es máxima en las bajas frecuencias y la capacidad de discriminar entre una frecuencia y otra es casi lineal. Pero a medida que aumenta la frecuencia, disminuye la sensibilidad, y de forma casi logarítmica aumenta la distancia entre frecuencias que el cerebro será capaz de discriminar. Si se pretende construir un sistema similar, se debe proporcionar una medida de la energía en cada una de las bandas del espectro que el sistema sea capaz de reconocer. Los coeficientes espectrales dan una medida de la amplitud y de la fase de una señal individual con la frecuencia que se corresponde con la banda de frecuencia asociada al coeficiente, donde la señal original podría generarse sumando todas las señales individuales asociadas a cada frecuencia. De esta forma, el módulo al cuadrado de cada uno de los coeficientes representa la energía que la señal original tiene en la frecuencia asociada al coeficiente. El módulo al cuadrado de los coeficientes espectrales representa de una forma más natural la información en el dominio de la frecuencia.

Si quiero representar la información en frecuencia de una forma todavía más semejante a la realizada por el oído humano, divido el espectro lineal que conforman los coeficientes espectrales en un espectro en el que la diferencia entre las frecuencias que serán capaces de discriminarse será cada vez mayor de acuerdo con una escala logarítmica (Escala Mel). Se divide el espectro en una serie de bandas de frecuencia (en nuestro caso 32), y para cada una de las

bandas de frecuencia se le proporcionará un coeficiente (coeficiente mel) que da una idea de la energía acumulada en todas las frecuencias que conforman esa banda de frecuencia. El tamaño de las bandas de frecuencia es mínimo para las bajas frecuencias, además, no aumenta en las 8 primeras bandas de frecuencia (8 primeros coeficientes mel). Así se consigue simular el funcionamiento casi lineal en la capacidad de captación de las frecuencias más bajas por el oído humano. A medida que aumenta la frecuencia, el tamaño de las bandas de frecuencia aumentará de forma logarítmica, hasta que las 32 bandas cubran todo el espectro a utilizar. Se simula así el aumento con la frecuencia casi logarítmico de la diferencia entre frecuencias que el oído humano es capaz de discriminar.

Se obtienen así los 32 coeficientes mel, donde cada coeficiente es una medida de la energía contenida en la banda de frecuencia a la que representa. Su valor se corresponde con la suma de los módulos al cuadrado de los coeficientes de Fourier asociados a las frecuencias incluidas en la banda de frecuencias que representa el coeficiente mel.

Coeficiente mel 1: Suma de los módulos al cuadrado de los coeficientes FFT de índice 1 al 2
Coeficiente mel 2: Suma de los módulos al cuadrado de los coeficientes FFT de índice 3 al 4
Coeficiente mel 3: Suma de los módulos al cuadrado de los coeficientes FFT de índice 5 al 6
Coeficiente mel 4: Suma de los módulos al cuadrado de los coeficientes FFT de índice 7 al 8
Coeficiente mel 5: Suma de los módulos al cuadrado de los coeficientes FFT de índice 9 al 10
Coeficiente mel 6: Suma de los módulos al cuadrado de los coeficientes FFT de índice 11 al 12
Coeficiente mel 7: Suma de los módulos al cuadrado de los coeficientes FFT de índice 13 al 14
Coeficiente mel 8: Suma de los módulos al cuadrado de los coeficientes FFT de índice 15 al 16
Coeficiente mel 9: Suma de los módulos al cuadrado de los coeficientes FFT de índice 17 al 20
Coeficiente mel 10: Suma de los módulos al cuadrado de los coeficientes FFT de índice 21 al 24
Coeficiente mel 11: Suma de los módulos al cuadrado de los coeficientes FFT de índice 25 al 30
Coeficiente mel 12: Suma de los módulos al cuadrado de los coeficientes FFT de índice 31 al 36
Coeficiente mel 13: Suma de los módulos al cuadrado de los coeficientes FFT de índice 37 al 42
Coeficiente mel 14: Suma de los módulos al cuadrado de los coeficientes FFT de índice 43 al 48
Coeficiente mel 15: Suma de los módulos al cuadrado de los coeficientes FFT de índice 49 al 56
Coeficiente mel 16: Suma de los módulos al cuadrado de los coeficientes FFT de índice 57 al 64
Coeficiente mel 17: Suma de los módulos al cuadrado de los coeficientes FFT de índice 65 al 72
Coeficiente mel 18: Suma de los módulos al cuadrado de los coeficientes FFT de índice 73 al 80
Coeficiente mel 19: Suma de los módulos al cuadrado de los coeficientes FFT de índice 81 al 90
Coeficiente mel 20: Suma de los módulos al cuadrado de los coeficientes FFT de índice 91 al 100
Coeficiente mel 21: Suma de los módulos al cuadrado de los coeficientes FFT de índice 101 al 110
Coeficiente mel 22: Suma de los módulos al cuadrado de los coeficientes FFT de índice 111 al 120
Coeficiente mel 23: Suma de los módulos al cuadrado de los coeficientes FFT de índice 121 al 132
Coeficiente mel 24: Suma de los módulos al cuadrado de los coeficientes FFT de índice 133 al 144

Coeficiente mel 25: Suma de los módulos al cuadrado de los coeficientes FFT de índice 145 al 156

Coeficiente mel 26: Suma de los módulos al cuadrado de los coeficientes FFT de índice 157 al 168

Coeficiente mel 27: Suma de los módulos al cuadrado de los coeficientes FFT de índice 169 al 182

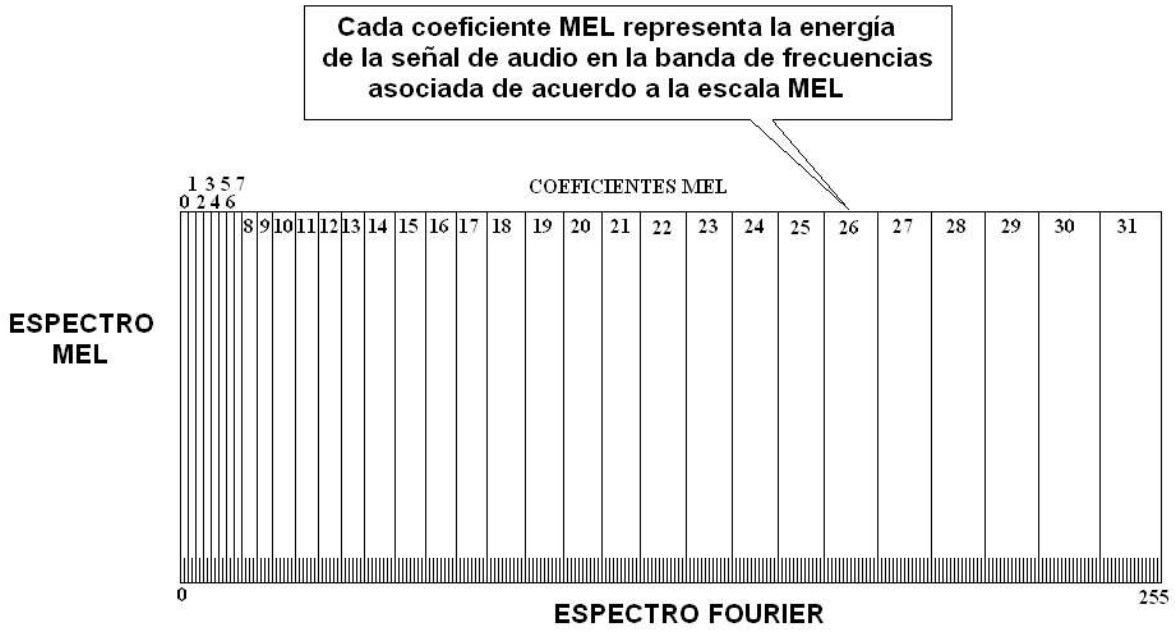
Coeficiente mel 28: Suma de los módulos al cuadrado de los coeficientes FFT de índice 183 al 196

Coeficiente mel 29: Suma de los módulos al cuadrado de los coeficientes FFT de índice 197 al 210

Coeficiente mel 30: Suma de los módulos al cuadrado de los coeficientes FFT de índice 211 al 224

Coeficiente mel 31: Suma de los módulos al cuadrado de los coeficientes FFT de índice 225 al 240

Coeficiente mel 32: Suma de los módulos al cuadrado de los coeficientes FFT de índice 241 al 256



Los coeficientes MEL constituyen unos parámetros que representan de una forma más natural la información vocálica de la señal de audio percibida.

2.5.1 SIMULACIÓN MATLAB®:

En el apartado anterior de simulación MATLAB® ya se calculó el módulo al cuadrado de cada uno de los coeficientes de Fourier obtenidos tras la realización de la Transformada Rápida de Fourier sobre las muestras digitales tomadas por el CODEC TLC320AD77C de una pronunciación vocálica. Para obtener los coeficientes MEL de los parámetros espectrales calculados ejecutaría a continuación sobre la plataforma MATLAB® el siguiente código:

```
%La función "obten_mel.m" calcula los coeficientes mel (parámetros que representan la energía de
%la señal en las bandas de frecuencia descritas por la escala mel) en función del módulo al
%cuadrado de los coeficientes complejos de Fourier introducidos en la función como parámetros de
%entrada. En el vector "coef_mel" se almacenarán los valores de los coeficientes mel.
[coef_mel]=obten_mel(modulo_fourier);
```

Se ha realizado una llamada a la función "obten_mel" que calculará los coeficientes mel que caracterizan de forma natural la energía de la señal de audio en la escala de frecuencias mel. Toma como parámetros de entrada el vector de módulos al cuadrado de los coeficientes espectrales de Fourier. El siguiente código sintetiza de forma clara el proceso realizado tras la llamada a la función.

obten_mel.m

```
function [coef_mel]=obten_mel(espectro)
%Función para obtener los coeficientes mel. Para ello sumo los módulos al cuadrado de la FFT
%en los puntos que se encuentren contenidos en cada una de dichas bandas. Se han utilizado
%32 bandas, cada banda se corresponde con los siguientes índices

%En el vector "elem_mel" se incluyen el número de coeficientes espectrales FFT cuyos módulos
%al cuadrado se sumarán para conformar el coeficiente mel asociado al índice del vector.
%Coeficiente mel 1: Suma de los módulos al cuadrado de los coeficientes FFT de índice 1 al 2
elem_mel(1)=2;
%Coeficiente mel 2: Suma de los módulos al cuadrado de los coeficientes FFT de índice 3 al 4
elem_mel(2)=2;
%Coeficiente mel 3: Suma de los módulos al cuadrado de los coeficientes FFT de índice 5 al 6
elem_mel(3)=2;
%Coeficiente mel 4: Suma de los módulos al cuadrado de los coeficientes FFT de índice 7 al 8
elem_mel(4)=2;
%Coeficiente mel 5: Suma de los módulos al cuadrado de los coeficientes FFT de índice 9 al 10
elem_mel(5)=2;
%Coeficiente mel 6: Suma de los módulos al cuadrado de los coeficientes FFT de índice 11 al 12
elem_mel(6)=2;
%Coeficiente mel 7: Suma de los módulos al cuadrado de los coeficientes FFT de índice 13 al 14
elem_mel(7)=2;
```

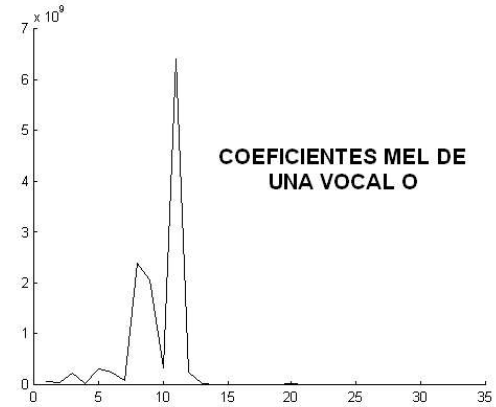
obten_mel.m

%Coeficiente mel 8: Suma de los módulos al cuadrado de los coeficientes FFT de índice 15 al 16
elem_mel(8)=2;
%Coeficiente mel 9: Suma de los módulos al cuadrado de los coeficientes FFT de índice 17 al 20
elem_mel(9)=4;
%Coeficiente mel 10: Suma de los módulos al cuadrado de los coeficientes FFT de índice 21 al 24
elem_mel(10)=4;
%Coeficiente mel 11: Suma de los módulos al cuadrado de los coeficientes FFT de índice 25 al 30
elem_mel(11)=6;
%Coeficiente mel 12: Suma de los módulos al cuadrado de los coeficientes FFT de índice 31 al 36
elem_mel(12)=6;
%Coeficiente mel 13: Suma de los módulos al cuadrado de los coeficientes FFT de índice 37 al 42
elem_mel(13)=6;
%Coeficiente mel 14: Suma de los módulos al cuadrado de los coeficientes FFT de índice 43 al 48
elem_mel(14)=6;
%Coeficiente mel 15: Suma de los módulos al cuadrado de los coeficientes FFT de índice 49 al 56
elem_mel(15)=8;
%Coeficiente mel 16: Suma de los módulos al cuadrado de los coeficientes FFT de índice 57 al 64
elem_mel(16)=8;
%Coeficiente mel 17: Suma de los módulos al cuadrado de los coeficientes FFT de índice 65 al 72
elem_mel(17)=8;
%Coeficiente mel 18: Suma de los módulos al cuadrado de los coeficientes FFT de índice 73 al 80
elem_mel(18)=8;
%Coeficiente mel 19: Suma de los módulos al cuadrado de los coeficientes FFT de índice 81 al 90
elem_mel(19)=10;
%Coeficiente mel 20: Suma de los módulos al cuadrado de los coeficientes FFT de índice 91 al 100
elem_mel(20)=10;
%Coeficiente mel 21: Suma de los módulos al cuadrado de los coeficientes FFT de índice 101 al 110
elem_mel(21)=10;
%Coeficiente mel 22: Suma de los módulos al cuadrado de los coeficientes FFT de índice 111 al 120
elem_mel(22)=10;
%Coeficiente mel 23: Suma de los módulos al cuadrado de los coeficientes FFT de índice 121 al 132
elem_mel(23)=12;
%Coeficiente mel 24: Suma de los módulos al cuadrado de los coeficientes FFT de índice 133 al 144
elem_mel(24)=12;
%Coeficiente mel 25: Suma de los módulos al cuadrado de los coeficientes FFT de índice 145 al 156
elem_mel(25)=12;
%Coeficiente mel 26: Suma de los módulos al cuadrado de los coeficientes FFT de índice 157 al 168
elem_mel(26)=12;
%Coeficiente mel 27: Suma de los módulos al cuadrado de los coeficientes FFT de índice 169 al 182
elem_mel(27)=14;
%Coeficiente mel 28: Suma de los módulos al cuadrado de los coeficientes FFT de índice 183 al 196
elem_mel(28)=14;
%Coeficiente mel 29: Suma de los módulos al cuadrado de los coeficientes FFT de índice 197 al 210
elem_mel(29)=14;
%Coeficiente mel 30: Suma de los módulos al cuadrado de los coeficientes FFT de índice 211 al 224
elem_mel(30)=14;
%Coeficiente mel 31: Suma de los módulos al cuadrado de los coeficientes FFT de índice 225 al 240
elem_mel(31)=16;
%Coeficiente mel 32: Suma de los módulos al cuadrado de los coeficientes FFT de índice 241 al 256
elem_mel(32)=16;

obten_mel.m

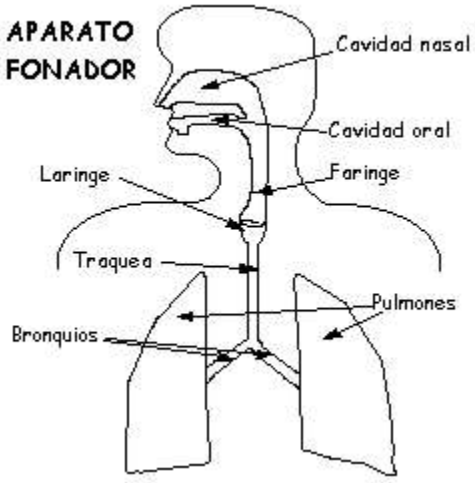
```
k=1;
for i=1:1:32
  suma=0;
  for j=1:1:elem_mel(i)
    suma=suma+espectro(k);
    k=k+1;
  end
  coef_mel(i)=suma;
end
```

Las siguientes gráficas resumen los resultados obtenidos tras el procesamiento de las muestras obtenidas de cinco pronunciaciones vocálicas diferentes. Puede apreciarse la información introducida por el tono (frecuencia de vibración de las cuerdas vocálicas que se caracteriza en la inclusión de picos separados entre sí por una distancia que caracteriza a la frecuencia del tono) y por el tracto vocálico (envolvente de los picos en la gráfica).



2.6 LOGARITMO EN BASE 2 DE LOS COEFICIENTES MEL

Directamente relacionado con la forma en la que nuestro oído recibe la información, nos encontramos la manera en la que nuestro aparato articulatorio crea el habla. El mecanismo de producción del habla, brevemente resumido es el siguiente:



1. El diafragma empuja los pulmones, haciendo que se expulse el aire.

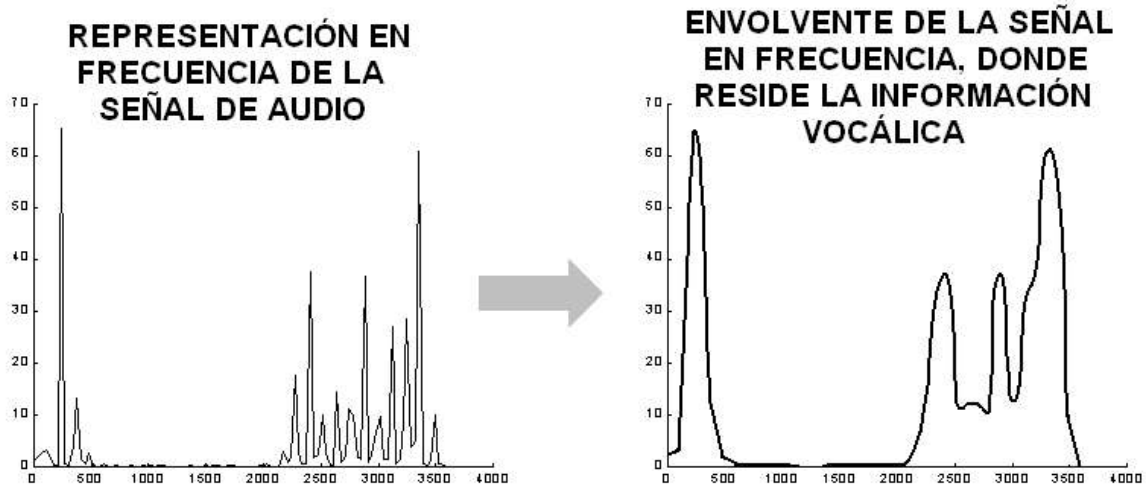
2. El aire circula por la tráquea y laringe, pasando por las cuerdas vocales y haciendo que vibren con un tono fundamental.

3. El tono fundamental producido por las cuerdas vocales pasa, a través de la laringe, a la caja de resonancia que forman las cavidades nasales y orales.

4. Algunas frecuencias entran en resonancia en las cavidades nasales y orales, saliendo hacia el exterior como la información más importante del habla.

Los coeficientes MEL obtenidos en el apartado anterior a partir de la suma del cuadrado del módulo de los coeficientes espectrales de la banda de frecuencias correspondiente en la escala MEL resumen de forma natural la información frecuencial de la señal de voz detectada. Los coeficientes MEL dan una idea de la energía captada en la banda de frecuencias con la que se identifican de acuerdo con la escala MEL de incremento logarítmico frecuencial.

El usuario que utilice el emulador del ratón utilizará la pronunciación de las vocales para comunicar al dispositivo los eventos de movimientos que quiere transmitir al ordenador a través del protocolo serie PS/2. Para generar una vocal, el usuario expulsará aire de los pulmones haciendo vibrar las cuerdas vocálicas a una determinada frecuencia (tono). En la pronunciación de cada una de las vocales, el tracto vocálico se modifica de forma distinta para inferir características particulares a la señal de voz según la vocal pronunciada. Al constituir el tracto vocálico un sistema no lineal, modificará la frecuencia fundamental (tono) repitiéndola en todos sus múltiplos, atenuada en algunas frecuencias y amplificada por resonancias en otras frecuencias. De ahí la utilidad de la representación de la señal en el dominio frecuencial, donde queda patente la información aportada por el tracto vocálico (envolvente de la señal representada en frecuencia) sobre la frecuencia de las cuerdas vocálicas (portadora que al actuar sobre el sistema no lineal se repetirá en los múltiplos de la frecuencia del tono). Si tratamos la señal vocálica desde el punto de vista frecuencial, es en la envolvente de la señal donde reside la información del tracto vocálico, y por consiguiente, la información vocálica.



El tono o frecuencia utilizado en la pronunciación no es relevante a la hora de identificar la vocal pronunciada. Sería interesante extraer la información que reside en la envolvente de la señal en el dominio frecuencial y eliminar la parte de información relativa al tono. Cada coeficiente mel podría verse como el producto de un multiplicando en el que reside toda la información del tono por un multiplicador en el que estaría toda la información del tracto vocálico.

$$\text{COEFICIENTE MEL} = \begin{matrix}\text{INFORMACIÓN}\\ \text{DEL TONO}\end{matrix} \times \begin{matrix}\text{INFORMACIÓN DEL}\\ \text{TRACTO VOCÁLICO}\end{matrix}$$

La información del tono está en íntima relación con la información del tracto vocálico por lo que resultaría adecuado desacoplar los dos términos. Realizando el logaritmo en base 2 a cada uno de los coeficientes mel, puedo desacoplar en un sumando la información relativa al tono de las cuerdas vocálicas y en otro sumando la información relativa al tracto vocálico. El desacoplamiento de la información permitirá eliminar la parte correspondiente al tono cuando se calculen los coeficientes cepstrales y así disponer sólo de la información vocálica.

$$\text{Log}_2(\text{COEFICIENTE MEL}) = \text{Log}_2\left(\begin{matrix}\text{INFORMACIÓN}\\ \text{DEL TONO}\end{matrix}\right) + \text{Log}_2\left(\begin{matrix}\text{INFORMACIÓN DEL}\\ \text{TRACTO VOCÁLICO}\end{matrix}\right)$$

SE DESACOPLA LA INFORMACIÓN EN DOS SUMANDOS, LO QUE PERMITIRÁ ELIMINAR LA INFORMACIÓN DEL TONO CUANDO SE CALCULEN LOS COEFICIENTES CEPSTRALES

2.6.1 SIMULACIÓN MATLAB®:

En el apartado anterior de simulación MATLAB® se calcularon los coeficientes MEL que resumen el nivel de energía de la señal captada en los espectros de frecuencia MEL. Realizo ahora el logaritmo en base dos a cada uno de los coeficientes para desacoplar la información presente relativa al tono de la información proporcionada por el tracto vocálico. Esta operación permitirá tras el cálculo de los coeficientes cepstrales eliminar la parte concerniente al tono. Para calcular el logaritmo en base dos a los coeficientes MEL ejecutaría sobre la plataforma MATLAB® el siguiente código:

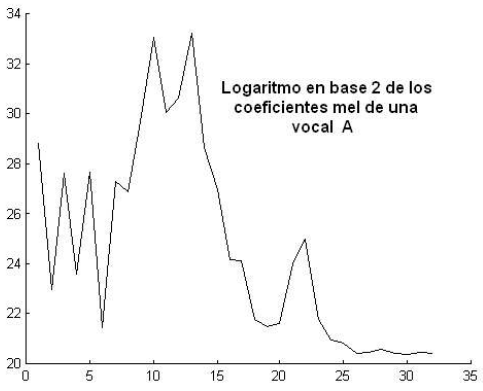
```
%La función "obten_log2.m" calcula el logaritmo en base 2 de los coeficientes mel pasados como
%parámetros de entrada a la función. En el vector "coef_log2" se almacenarán los valores de los
%logaritmos en base dos de los coeficientes mel.
[coef_log2]=obten_log2(coef_mel);
```

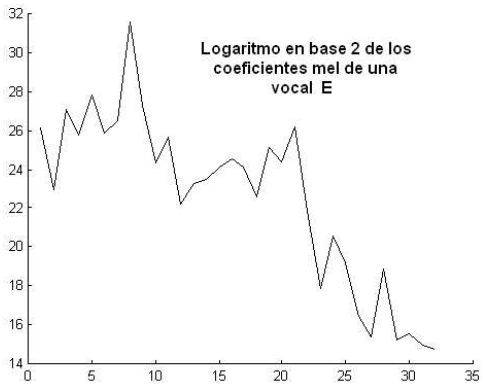
Se ha realizado una llamada a la función "obten_log2" que calculará el logaritmo en base dos de los coeficientes mel que caracterizan de forma natural la energía de la señal de audio. Este proceso permite desacoplar en cada uno de los coeficientes el término asociado al tono del término asociado al tracto vocálico, donde reside la información vocálica relevante en el proceso de reconocimiento. Este hecho permitirá eliminar el término asociado al tono cuando se calculen los coeficientes cepstrales. Toma como parámetros de entrada el vector de coeficientes mel y creará un vector con el logaritmo en base dos de cada uno de los coeficientes mel. El siguiente código sintetiza de forma clara el proceso realizado tras la llamada a la función.

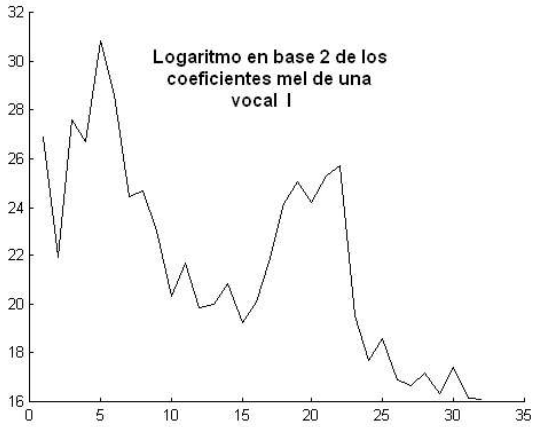
obten_log2.m

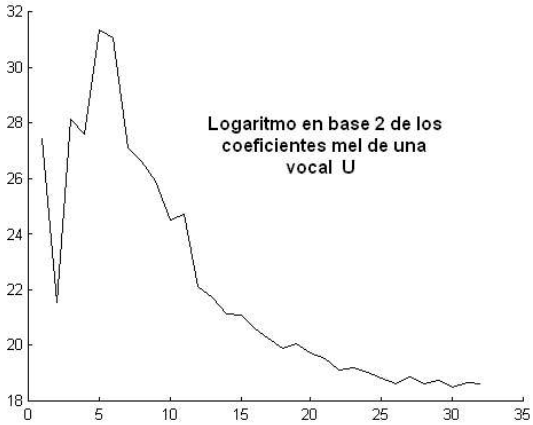
```
function [coef_log2]=obten_log2(coef_mel)
%Función para obtener el logaritmo en base 2 de los coeficientes mel pasados como parámetros
%de entrada a la función. Devolverá un vector con el logaritmo en base 2 de todos los
%coeficientes mel

for i=1:1:32
  coef_log2(i)=log2(coef_mel(i));
end
```

Las siguientes gráficas resumen los resultados obtenidos tras el procesamiento de las muestras obtenidas de cinco pronunciaciones vocálicas diferentes para obtener el logaritmo en base dos de los coeficientes mel. La información relativa al tono se encuentra presente todavía en los coeficientes calculados, pero cuando se calculen los coeficientes cepstrales se podrá discriminar la información del tono de la información asociada al tracto vocálico, donde verdaderamente residen las características vocálicas que permitirán el reconocimiento de la vocal pronunciada. Una red neuronal de 32 neuronas detectará la vocal pronunciada de forma similar a como lo realizaría el cerebro humano. Con el cálculo de los coeficientes MEL, que proporcionan información de la energía en las diversas frecuencias en una escala similar a la existente en el caracol del oído humano y el uso de una red neuronal que intenta emular el funcionamiento del cerebro en un proceso de reconocimiento, se pretende realizar un sistema de reconocimiento lo más natural posible.









2.7 COEFICIENTES CEPSTRALES

Los coeficientes mel representan la información en frecuencia de las muestras captadas por el CODEC TLC320AD77C de Texas Instruments externo a la FPGA, pero presente en la tarjeta XSA y lo hacen de una forma más natural que la obtenida directamente con los coeficientes de Fourier. Cada coeficiente da una medida de la energía captada en una parcela del espectro. En los coeficientes menores el espectro es menor y va creciendo de forma logarítmica. Este proceso es similar al realizado en el oído humano. Pero en los coeficientes mel se encuentra fundida la información aportada por el tono (frecuencia de vibración de las cuerdas vocálicas) con la información aportada por el tracto vocálico (donde reside realmente la información que permitirá reconocer la vocal). Este hecho se puede modelar con una multiplicación donde el multiplicando encierra toda la aportación del tono y el multiplicador encierra toda la información relativa al tracto vocálico.

$$\text{Coeficiente Mel (i)} = \text{Aportación del tono (i)} \times \text{Aportación del tracto vocálico (i)}$$

$$\text{i : índice del espectro mel (1-32)}$$

Con la realización del logaritmo en base 2 de los coeficientes mel se consigue desacoplar en dos sumandos cada una de las aportaciones. Un sumando englobará toda la información del tono y el otro sumando englobará toda la información del tracto vocálico.

$$\mathbf{Log}_2(\text{Coeficiente Mel (i)}) = \mathbf{Log}_2(\text{Aportación del tono (i)}) + \mathbf{Log}_2(\text{Aportación del tracto vocálico (i)})$$

$$\text{i : índice del espectro mel (1-32)}$$

Los coeficientes cepstrales se obtienen al realizar la Transformada Coseno Discreta al logaritmo en base dos de los coeficientes MEL. Ahora, en el dominio cepstral, el índice del coeficiente es de nuevo el tiempo, ya que se ha aplicado una Transformada de Fourier Inversa, pero se suele llamar a este índice cuefrencia, y así diferenciar de los valores temporales de las muestras de la señal de audio original. No son equivalentes puesto que antes de la Transformada de Fourier Inversa se han realizado una serie de procesos que provocan enormes diferencias con la señal de audio original.

$$\text{coeficiente cepstral (i)} = \mathbf{IDFT}(\mathbf{Log}_2(\text{Aportación del tono (i)})) + \mathbf{IDFT}(\mathbf{Log}_2(\text{Aportación del tracto vocálico (i)}))$$

$$\text{coeficiente cepstral (i)} = \text{cepstral del tono (i)} + \text{cepstral del tracto vocálico (i)}$$

$$\text{i : índice del espectro mel (1 - 32)}$$

La utilidad práctica de los coeficientes cepstrales se descubre ahora, ya que en los coeficientes cepstrales de bajas cuefrencias el sumando cepstral correspondiente al tono es despreciable, y por otro lado, en los coeficientes cepstrales de altas cuefrencias el sumando cepstral correspondiente a la información del tracto vocálico es despreciable. Así, los coeficientes cepstrales de las cuefrencias más bajas resumen toda la información que el tracto vocálico insertó en la señal de audio, y los coeficientes cepstrales de las cuefrencias más altas resumen la información del tono. Ahora se pone de manifiesto la utilidad de descomponer, gracias al logaritmo en base dos, las componentes de información en dos sumandos para poder así calcular la Transformada Discreta de Fourier Inversa de cada uno de los sumandos por separado. Aprovechando la peculiaridad de los coeficientes cepstrales, utilizaré los ocho primeros coeficientes cepstrales como parámetros que definen de forma adecuada la señal de voz. Estos coeficientes resumen la información aportada por el tracto vocálico y en ellos la aportación del tono es despreciable, como ya se ha

mencionado. Al tratar sólo los primeros coeficientes cepstrales, se consigue extraer la información del tracto vocálico y eliminar la parte de información que no merece atención en el proceso de reconocimiento.

Los coeficientes cepstrales se calculan como la Transformada Coseno Discreta (DCT), que hace las veces de transformada inversa, de las energías logarítmicas obtenidas con anterioridad. En concreto, los coeficientes cepstrales se obtienen a partir del muestreo de dicha transformada. El cálculo de los coeficientes responde a la expresión:

$$\text{Coeficiente Cepstral}(i) = \sum_{k=1\ldots L} \text{Log}_2(\text{Coeficiente Mel}(k)) \times \cos\left(i\left(k-\frac{1}{2}\right)\frac{\pi}{L}\right) \qquad \text{donde } i=1\ldots M$$

Los elementos de la ecuación representan:

- **k es el índice del coeficiente mel.**
- **L es el número de bandas mel o de coeficientes mel, 32 en el caso del proyecto.**
- **M es el número de coeficientes cepstrales calculados, 8 en el caso del proyecto.**

No se toma el índice 0 para obtener el primer coeficiente ya que éste representa una media de la energía de la señal vocálica. Como este valor depende mucho del emplazamiento del micrófono y de las condiciones del canal de comunicación, no se utilizará este valor como parámetro de la señal en el proceso de reconocimiento.

2.7.1 SIMULACIÓN MATLAB®:

En el apartado anterior de simulación MATLAB® se calculó el logaritmo en base dos de los coeficientes MEL. Estos coeficientes resumen el nivel de energía de la señal captada en los espectros de frecuencia MEL. Con el logaritmo en base dos se consigue desacoplar la información presente relativa al tono de la información proporcionada por el tracto vocálico. Aplicando la Transformada Coseno Discreta, que hace las veces de Transformada de Fourier Inversa, se obtienen los coeficientes cepstrales donde gracias al desacoplo, la información asociada al tracto vocálico toma presencia sólo en los coeficientes cepstrales bajos y la información asociada al tono toma presencia sólo en los coeficientes cepstrales altos. Se propondrán los ocho primeros coeficientes cepstrales como parámetros significativos de la señal de audio pronunciada. Estos coeficientes serán los que se utilicen en el proceso de reconocimiento para detectar la vocal pronunciada. Para calcular los ocho primeros coeficientes cepstrales se ejecutaría sobre la plataforma MATLAB® el siguiente código:

```
%La función "obten_cepstrales.m" calcula los coeficientes cepstrales de la señal de audio en
%función de las energías logarítmicas pasadas como parámetros de entrada a través del vector
%"coef_log2". En el vector "cepstrales" se almacenarán los valores de los ocho primeros
%coeficientes cepstrales, sin incluir el índice 0, que se utilizarán como parámetros
%identificativos de la vocal en el proceso de reconocimiento.
cepstrales=obten_cepstrales(coef_log2);
```

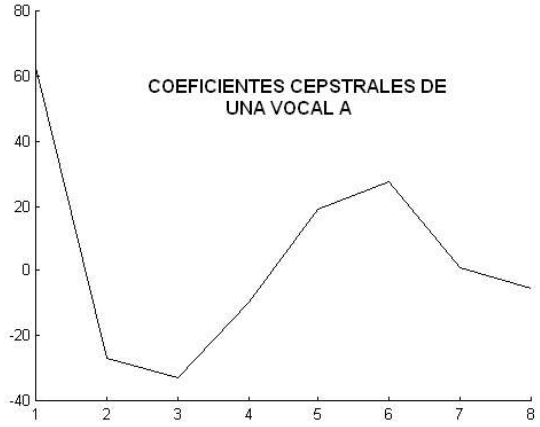
Se ha realizado una llamada a la función "obten_cepstrales" que calculará la Transformada Coseno Discreta sobre los parámetros de entrada de la función, que se corresponden con las energías logarítmicas en la escala MEL de la señal de audio procesada. Los coeficientes cepstrales de las primeras cuefrencias resumen la información que el tracto vocálico aporta al tono, por eso se escogen los 8 primeros coeficientes cepstrales como los parámetros más característicos de la señal vocálica. Los coeficientes cepstrales de las cuefrencias más altas engloban la información relativa al tono y se desechan como fuente de información vocálica. Creará un vector con los coeficientes cepstrales que se utilizarán como parámetros de la vocal pronunciada.

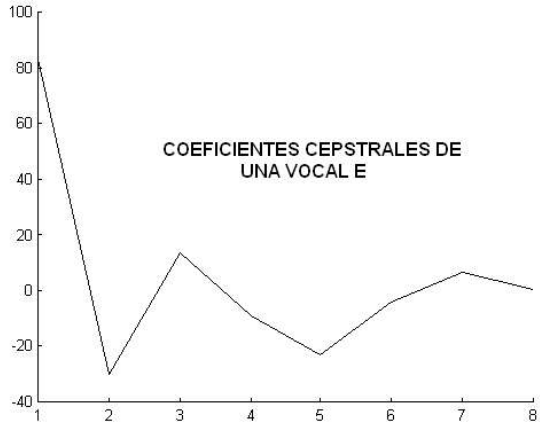
obten_cepstrales.m

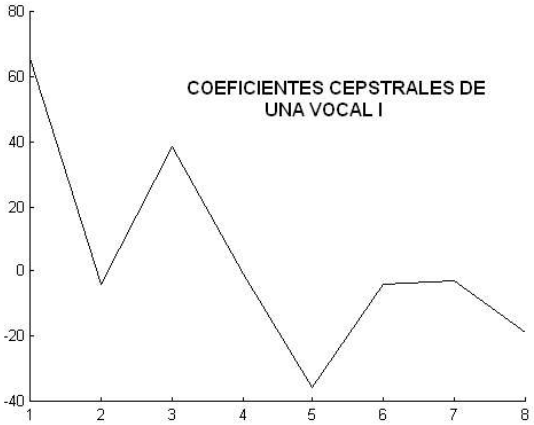
```
function [cepstrales]=obten_cepstrales(coef_log2)
% Con esta función obtenemos los coeficientes cepstrales.
% Los coeficientes cepstrales se calculan como la transformada
% coseno discreta (DCT), que hace las veces de transformada inversa,
% de las energías logarítmicas obtenidas con anterioridad.
% Los ocho primeros valores de esta transformada inversa me
% proporcionarán los coeficientes cepstrales.

%Número de coeficientes mel
L=32;

%Número de coeficientes cepstrales
M=8;

for i=1:1:M
  suma=0;
  for k=1:1:L
    suma=suma+(coef_log2(k)*cos(i*(k-0.5)*pi/L));
  end
  cepstrales(i)=suma;
end
```

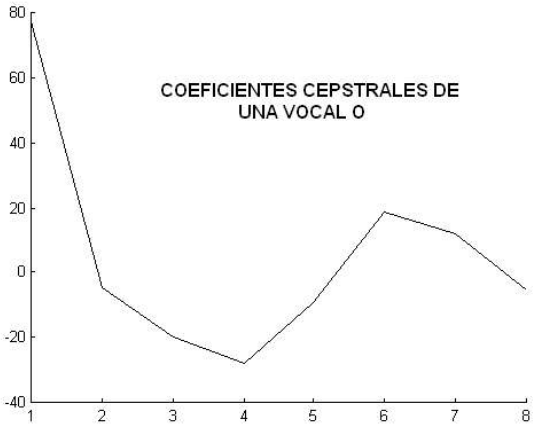
Las siguientes gráficas resumen los resultados obtenidos tras el procesamiento de las muestras obtenidas de cinco pronunciaciones vocálicas diferentes para obtener los coeficientes cepstrales.

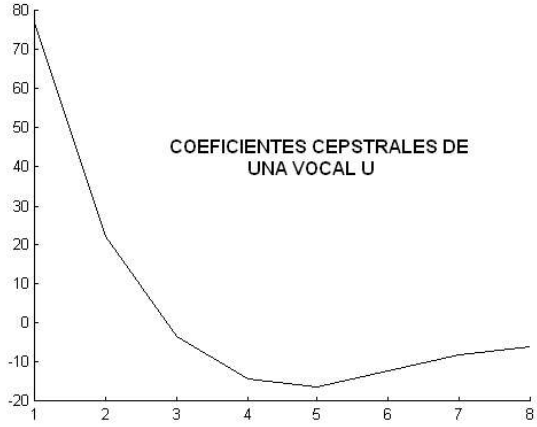
Una red neuronal de 32 neuronas en la topología de un mapa autoorganizado de cuatro filas y ocho columnas utilizará los ocho coeficientes cepstrales como parámetros que alimentarán la red neuronal. Una vez expuestas todas las neuronas a través de las sinapsis con las neuronas de entrada a los parámetros cepstrales, se activará sólo una de las neuronas. Esta neurona tiene asociada una vocal, que será la vocal que se proponga como vocal detectada en el proceso de reconocimiento. Los parámetros cepstrales tienen un número lo suficientemente pequeño como para facilitar el reconocimiento, habiendo eliminado en la medida de lo posible las redundancias existentes en la información, pero lo suficientemente elevado como para permitir discriminar las distintas vocales.










COEFICIENTES CEPSTRALES DE
UNA VOCAL U
80
70
60
50
40
30
20
10
0
-10
-20
1
2
3
4
5
6
7
8