UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

"ESTUDIO DE LAS TÉCNICAS DEL TRATAMIENTO DISCRETO DE LA SEÑAL DE VOZ PARA IMPLEMENTAR UN SISTEMA DE IDENTIFICACIÓN DE LOCUTOR CON MATLAB"

INFORME DE SUFICIENCIA

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

PRESENTADO POR:

HÉCTOR FELIPE VELÁSQUEZ CABANILLAS

PROMOCIÓN 1 992 - I

LIMA – PERÚ

2 002

Dedicado a mi Madre , mis hermanos:

Luis y Eduardo, y, a la memoria de mi

Padre.

“ESTUDIO DE LAS TECNICAS DEL TRATAMIENTO DISCRETO DE LA SEÑAL DE VOZ PARA IMPLEMENTAR UN SISTEMA DE IDENTIFICACIÓN DE LOCUTOR CON MATLAB”

SUMARIO

El presente trabajo pretende concretar una aplicación del tratamiento discreto de la señal de voz , para lo cual se empieza ilustrando el funcionamiento del sistema reproductor de esta señal y conceptuando un modelo del mismo. Posteriormente se hace un análisis del comportamiento de la señal de voz, utilizando representaciones en el tiempo y en frecuencia.

Se presentan las técnicas de codificación , poniendo especial interés en la codificación de la fuente de generación de voz. Este avance nos conduce a explicar la Codificación Predictiva Lineal (LPC), y, a partir de estos conceptos, se tiene un procedimiento para obtener el modelo predictivo lineal del sistema reproductor de voz, propio de cada persona

Para el tema de la "Identificación de locutor", el proceso complementario consiste en "preparar" al sistema de identificación con un patrón de características de la fuente de voz del locutor considerado.

Finalmente, la etapa de identificación consiste en encontrar el patrón de características mas parecido al "patrón de voz" obtenido durante una prueba. Esta búsqueda, determinara al locutor que se corresponde con la secuencia de características de voz obtenidas durante una prueba de identificación.

ÍNDICE

PROLOGO

En los últimos años, todos hemos sido testigos del avasallante desarrollo de las tecnologías de la información y las comunicaciones. Cada vez aparecen nuevos servicios que no dejan de sorprendernos, siendo muchos de ellos, aquellos que involucran la interrelación entre imagen y sonido. Es por esto que el tratamiento de la voz y la imagen resulta de suma importancia para lograr aplicaciones tales como: comunicación hombre-máquina y mejoramiento de la eficiencia en el empleo de recursos (ancho de banda y memoria). El procesamiento discreto de la señal de voz ofrece un gran numero de aplicaciones, y es la identificación de locutor, la aplicación que será tratada en el presente trabajo.

Puesto que la seguridad es un tema de rigor cuando se discute cualquier tecnología, la verificación de locutor nos ofrece una importante "llave de acceso", ya que cae dentro del campo de la "Biometrica", en donde cada ser humano tiene características propias como en el caso de las huellas digitales y las marcas de la retina. En la actualidad existen sistemas comerciales orientados a identificación de locutor que utilizando una adecuada secuencia de entrenamiento logran tremendas precisiones de hasta el 98%.

El modelo algorítmico implementado con MATLAB es un sistema cerrado de identificación de locutor, el cual consiste en reconocer a un locutor de entre 2 locutores conocidos. Se extrajeron las características de la voz, luego se procedió a representarlas vectorialmente en un espacio de 20 dimensiones, a partir de esto, se determino los centroides de las regiones que los vectores de características habían formado, de este modo se obtenían codebooks para cada locutor conocido.

Finalmente, la etapa del reconocimiento propiamente dicha, será un bloque de decisión, el cual buscara en cada codebook la representación más cercana de los vectores fila correspondientes a la palabra sometida a prueba. Es decir, el locutor reconocido como 1 o 2 presentará, en la pronunciación de una palabra de un vocabulario establecido, la menor distorsión de cuantificacion de los vectores de la características de la voz respecto al codebook que lo representa.

En el proceso de obtención del codebook para cada locutor, se itero hasta obtener una cuantificacion de las palabras de la secuencia de entrenamiento próximas al 100%. Este resultado se logro con un codebook de 7 bits. La funcion de busqueda para el reconocimiento utiliza el codebook de 7 bits .

Para la comprobación del sistema se ha implementado un algoritmo que simula una verificación en casi tiempo real, previa grabación de la voz a probar.

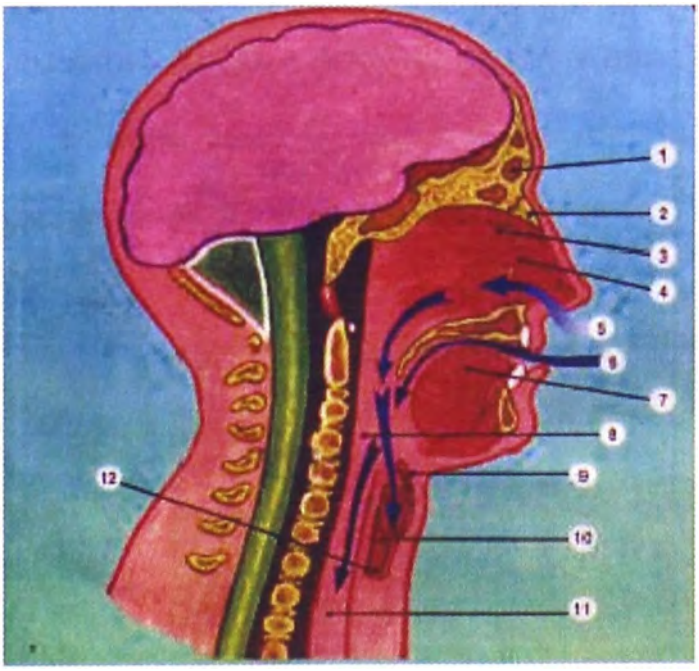
En la parte del fundamento teórico se muestran gráficas reales referidas a la voz y su tratamiento, obtenidas por medio del programa MatLab.

Además, en las conclusiones se indican las conseguidas a través de esta experiencia y otras de origen teórico.

CAPÍTULO I
ESTUDIO DEL SISTEMA FONADOR HUMANO Y LA SEÑAL DE VOZ

1.1 Anatomia del sistema fonador humano

La señal de voz es una onda de presión acústica sonora que se origina voluntariamente a partir de movimientos de la estructura anatómica del sistema humano de producción de voz. Los componentes principales son los pulmones, la traquea, la laringe, la cavidad de la faringe, la cavidad oral y bucal y la cavidad nasal (ver Fig. 1.a). El tracto vocal empieza a la salida de la laringe y termina a la entrada de los labios. El tracto nasal empieza en el paladar y termina en los orificios nasales.



1. SENO FRONTAL
2. HUESO NASAL
3. MEATO
4. CORNETE
5. VIA RESPIRATORIA
6. VIA DIGESTIVA
7. LENGUA
8. FARINGE
9. EPIGLOTIS
10. LARINGE.
11. ESOFAGO
12. TRAQUEA

FIGURA 1.a .- Anatomia del aparato fonador

Los parámetros principales del sistema articulatorio son las cuerdas vocales, el paladar, la lengua, los dientes, los labios, y las mandíbulas. Los distintos sonidos se producen al pasar el aire emitido por los pulmones, a través de todo el sistema de producción, en una determinada posición de cada parámetro articulatorio. Este sistema físico puede modelarse como un filtro, cuya función de transferencia depende del sonido articulado y, por tanto, de la posición de los diversos órganos involucrados en la generación del habla. La entrada al filtro se puede modelar mediante una señal de excitación, que se corresponde con el paso del aire generado por los pulmones a través de la traquea y las cuerdas vocales, y también será dependiente del sonido generado.

1.2 Clases de sonidos

Podríamos comprobar fácilmente que cuando hablamos existen sonidos en los cuales nuestras cuerdas vocales vibran y otros en los que permanecen en reposo(o en todo caso la vibración es mínima). La figura 1.b ilustra algunas posiciones de las cuerdas vocales.

FIGURA 1.b - Formas de la glotis durante la emisión de voz

Esto permite realizar la clasificación de los sonidos en 2 tipos:

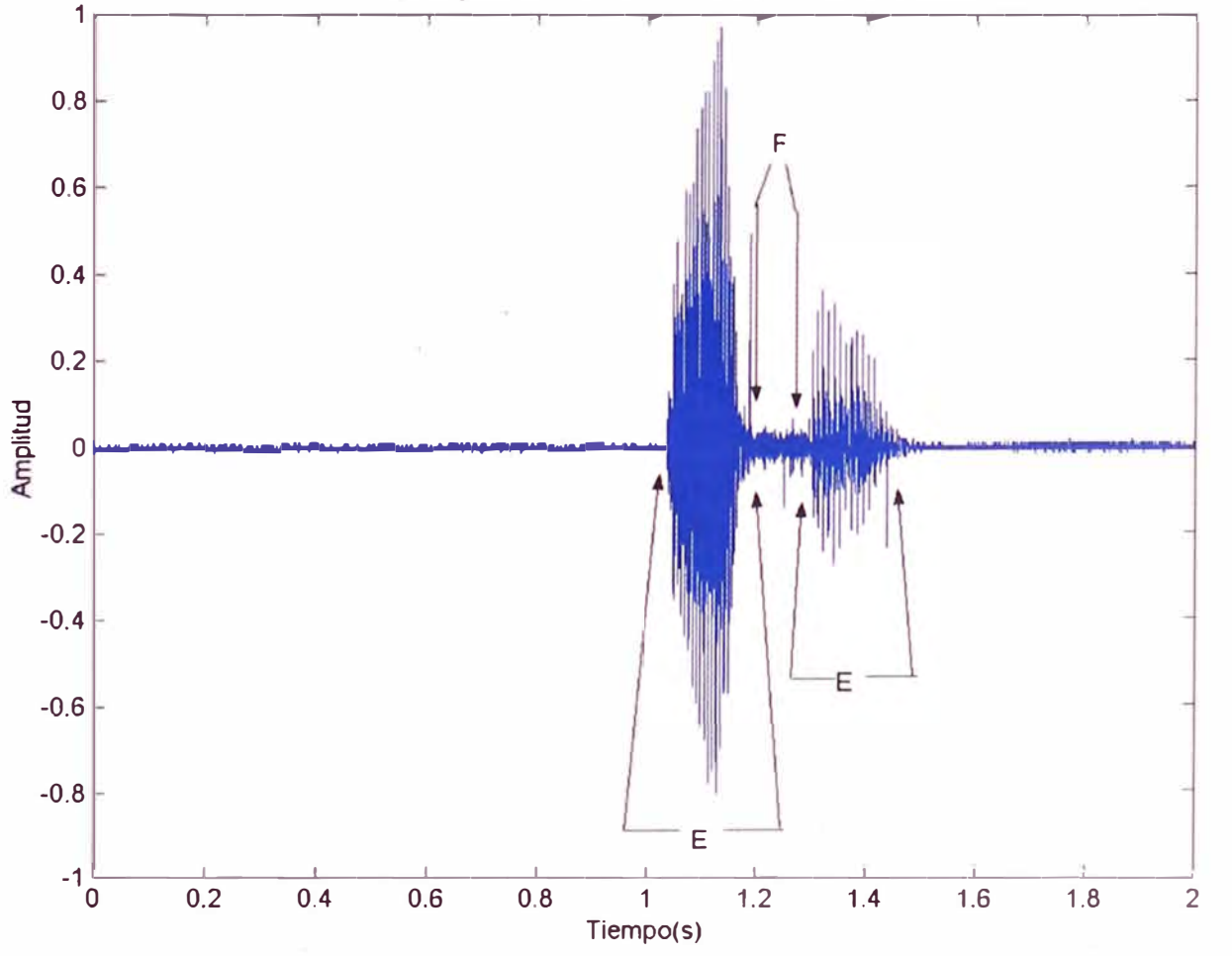
a) SONIDOS SONOROS.- En ellos las cuerdas vocales vibran y el aire pasa a través del tracto vocal sin impedimentos importantes.

b) SONIDOS SORDOS.- En ellos las cuerdas vocales no vibran y existen restricciones importantes al paso del aire que proviene de los pulmones, por lo que son de amplitud menor y normalmente de naturaleza mas ruidosa que los sonoros. También se les conoce como fricativos.

1.3 La señal de voz

En la figura 2, se puede apreciar la diferencia entre ambos tipos de sonidos:

FIG. 2 .- Oscilograma de /EFE/ (Grabación a 8Khz de muestreo)



Esta figura 2, representa la variación en el tiempo de /efe/, donde el fonema /e/ corresponderá a un sonido sonoro y en la gráfica es la zona de mayor magnitud, mientras que el fonema /f/ corresponde a un sonido sordo (menor magnitud). Además se puede observar la característica ruidosa del fonema /f/ y cuasi-periodica de la /e/. Estas observaciones se verifican rapidamente en la ampliación de los fonemas mostrados en las figuras 3 y 4.

FIGURA 3 - Segmento sonoro /E/

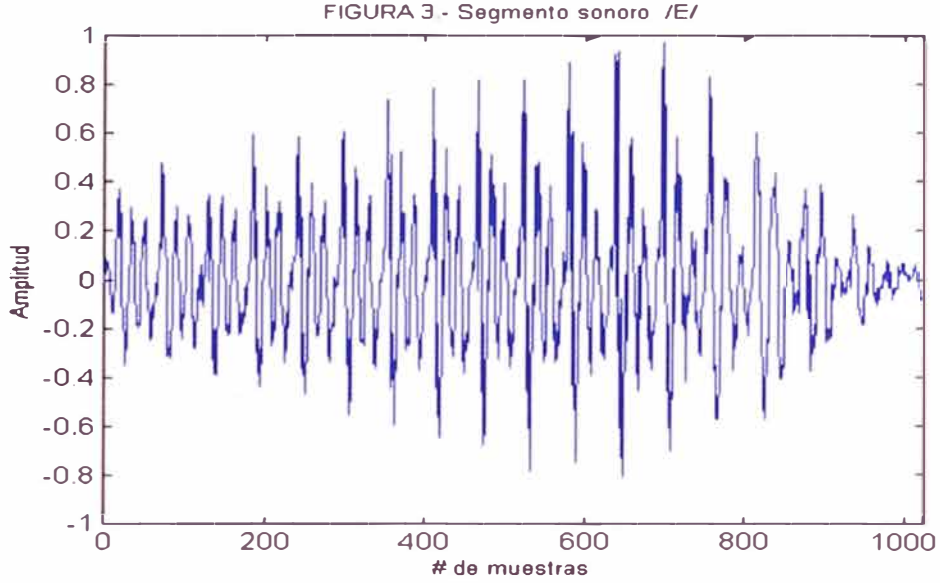
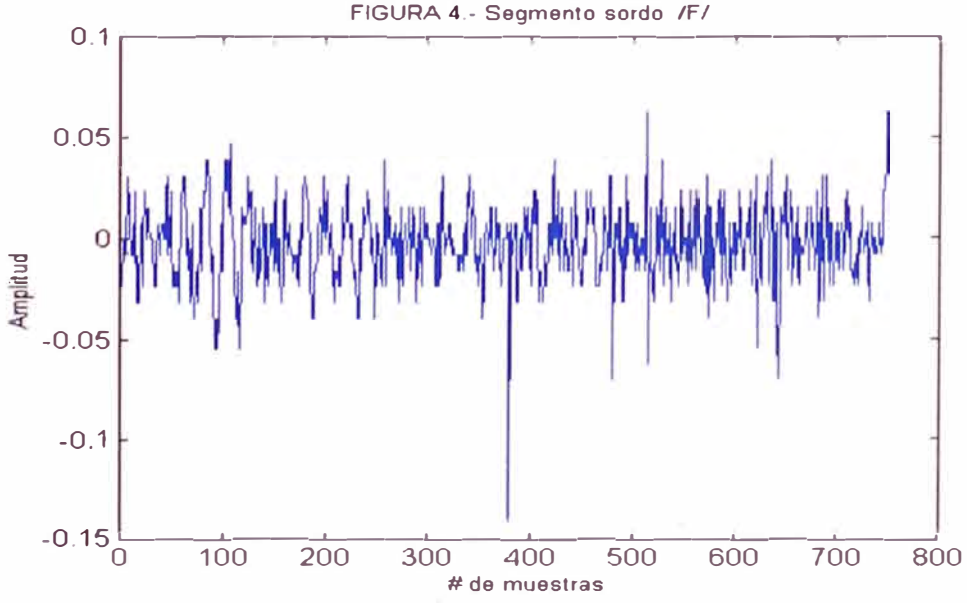


FIGURA 4.- Segmento sordo /F/



Una representación en el dominio de la frecuencia mediante la Transformada Rápida de Fourier(FFT), nos permitirá obtener mas información acerca de estos fonemas.

FIGURA 5.- Espectro de /E/

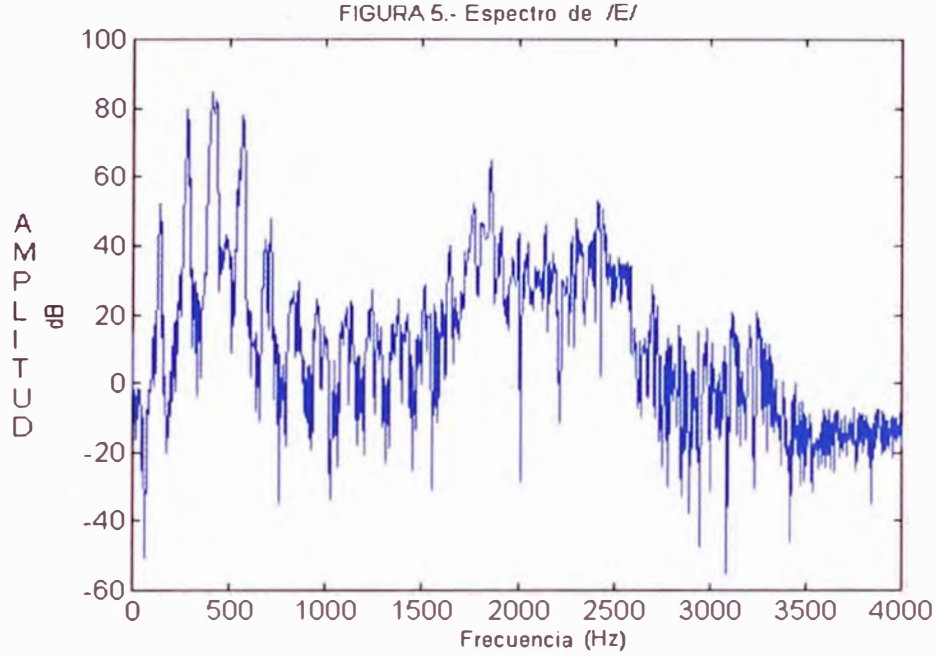
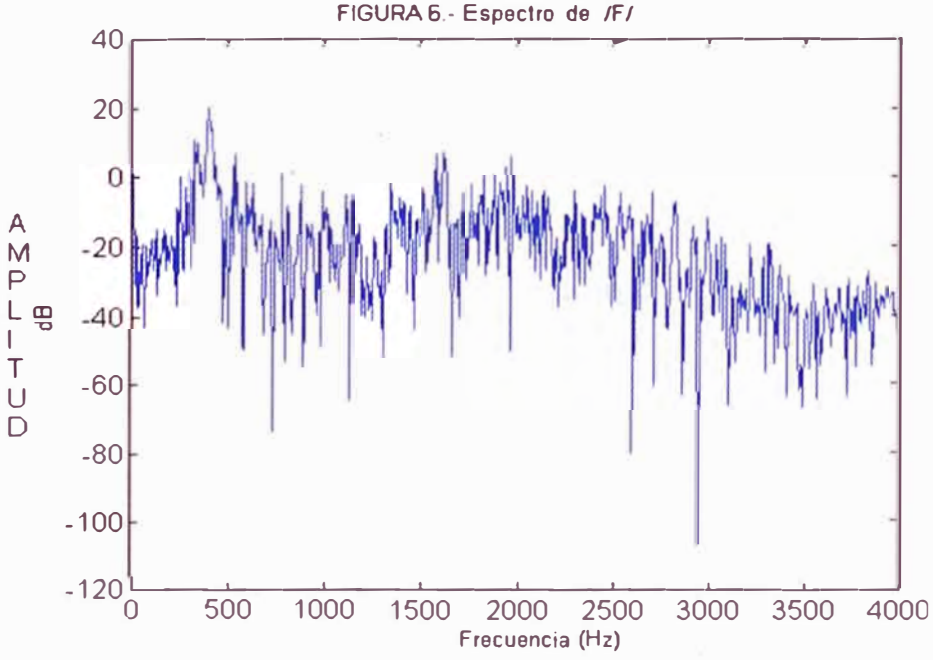


FIGURA 6.- Espectro de /F/



Las representaciones obtenidas en el dominio de la frecuencia (modulo de la FFT) reciben el nombre de espectros (ver figuras 5 y 6). Se esta utilizando un ancho de banda de 4 Khz para la señal de voz como el utilizado para comunicaciones por vía telefónica.

En la figura 5 (trama sonora) se puede apreciar la existencia de una frecuencia fundamental alrededor de 150 Hz, y algunos armónicos equiespaciados esta misma frecuencia, la cual recibe el nombre de PITCH (tono), y esta relacionada directamente con la frecuencia de vibración principal de las cuerdas vocales. El margen habitual para locutores masculinos es de 50 a 300 Hz (periodo de 20 a 3 ms o 160 a 24 muestras al trabajar con frecuencias de muestreo de fs=8KHz).

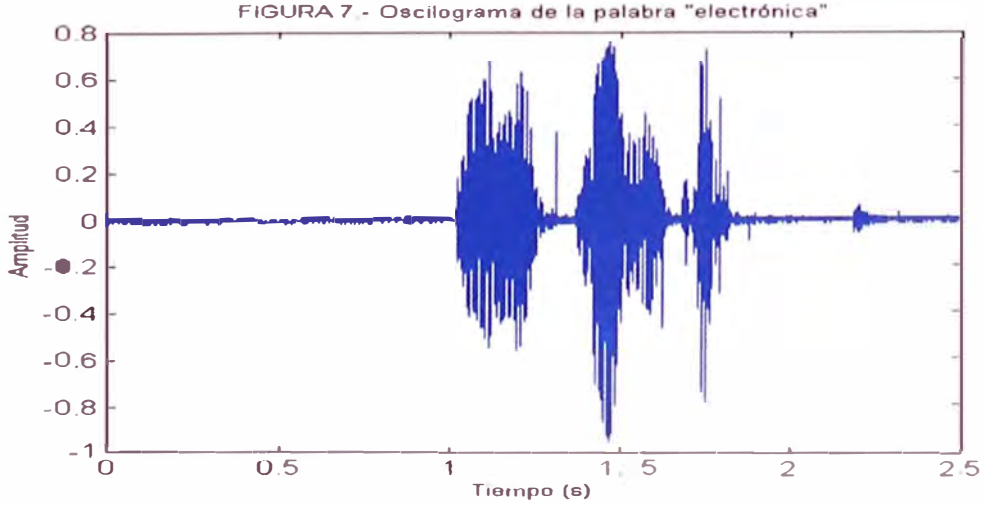
Otro aspecto importante de los ESPECTROS correspondientes a fonemas sonoros es la presencia de zonas enfatizadas, o resonancias, y zonas desenfatizadas o antirresonancias. Estas zonas se deben a la configuración física de las diferentes cavidades del tracto configurado por el sistema articulador. La posición de las frecuencias de resonancia depende de la forma, tamaño y características físicas del tracto vocal al producir cada sonido. De modo que cada cada forma del tracto vocal, cuando hablamos, podria modelarse por medio de un filtro con un conjunto de frecuencias de resonancia que lo caracterizarían. En el modelo del tracto vocal a cada una de estas frecuencias de resonancia se le denomina “frecuencia formante” o simplemente “formante”. Debido a que las señales sonoras presentan una característica típica de paso bajo, los formantes mas significativos serán los primeros. Habitualmente se utiliza entre 3 y 5 formantes al considerar una

frecuencia de muestreo(fs) de 8KHz. Como el tracto vocal evoluciona en el tiempo para producir los diferentes sonidos, la caracterizacion espectral de la señal de voz es variante en el tiempo. Esta evolución temporal puede representarse mediante un espectrograma de la señal de voz o sonograma, el cual es una representacion bidimensional que muestra la evolución temporal de la caracterización espectral.

Los formantes aparecen como franjas horizontales, mientras que los valores de amplitud en función de la frecuencia se representan en sentido vertical.

Existen 2 tipos de sonogramas: de banda ancha y de banda angosta, en función del ancho de banda del filtro que se haya utilizado para realizar el análisis frecuencial. En el caso de los espectrogramas de banda ancha se obtendrá una buena resolución temporal. En cambio en el caso de espectrogramas de banda angosta se obtendrá una buena resolución frecuencial. A continuación se ilustra el análisis por medio de espectrogramas de banda angosta y banda ancha para la señal mostrada en el oscilograma de la figura 7.

FIGURA 7 - Oscilograma de la palabra "electrónica"



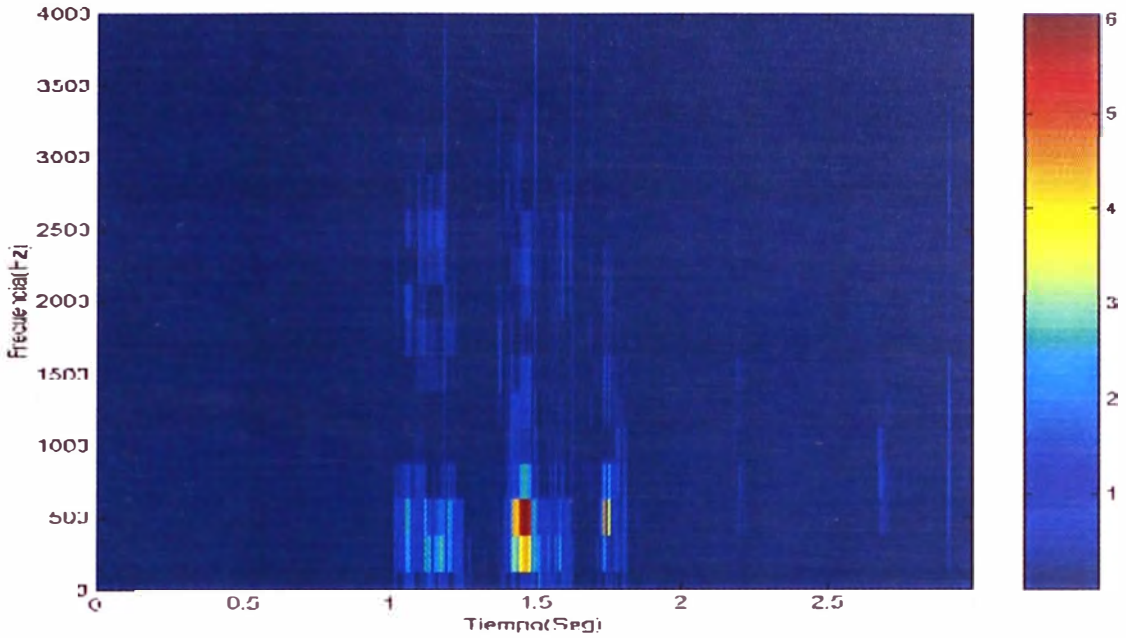


FIGURA 8.- Espectrograma de banda ancha "Electrónica"

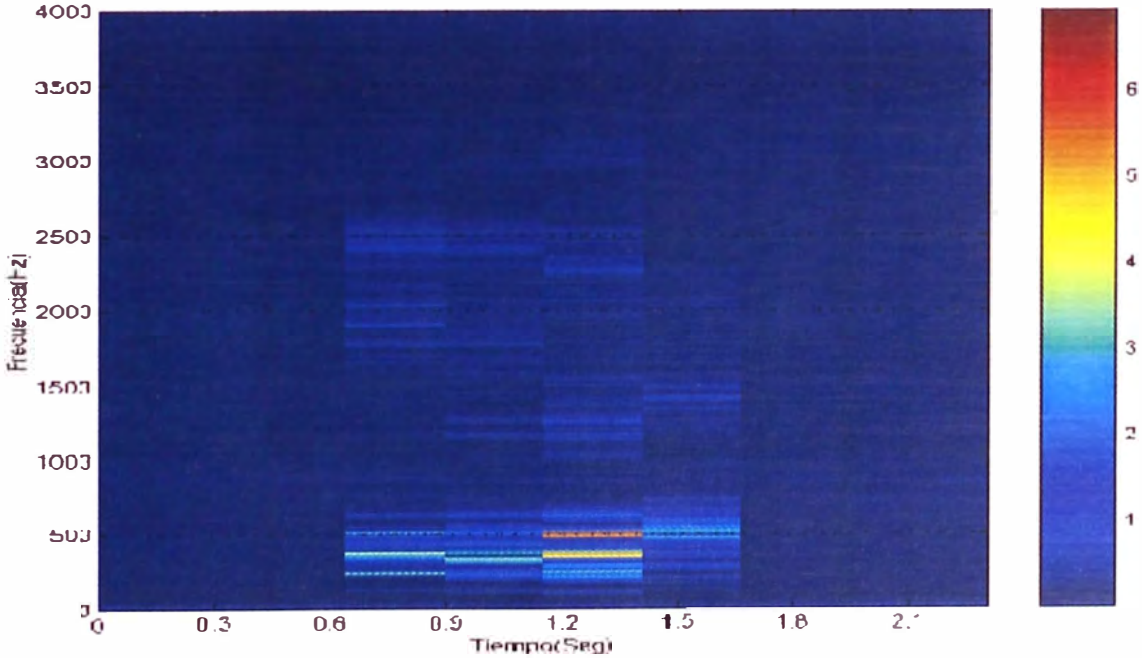


FIGURA 9.- Espectrograma de banda angosta "Electrónica"

1.4 Digitalización de la voz

El procesado digital de la señal de voz mediante un Procesador Digital de Señales (DSP), computador,etc., requiere previamente la conversión de la señal acústica a eléctrica mediante un micrófono, y la conversión de la señal analógica resultante a señal digital.

Para realizar la conversión de analógico a digital es necesario realizar un muestreo o discretizacion de los valores de señal cada cierto intervalo de tiempo, denominado periodo de muestreo, Ts, cuyo inverso será la frecuencia de muestreo, fs. En este punto hay que considerar el teorema de Nyquist, el cual, para evitar fenómenos de "aliasing" en señales de paso bajo, es necesario muestrear como mínimo al doble del ancho de banda de la señal. Esto supone incorporar antes del muestreado un filtro de paso bajo "antialiasing" que limite el ancho de banda de la señal analógica, y elimine el ruido presente mas allá del ancho de banda de la señal útil.

Una vez realizada la discretizacion en el tiempo, es necesario discretizar cada muestra en amplitud, a este paso se le conoce como CUANTIFICACION, de modo que las muestras que conforman la señal procesada tendrán un correspondiente valor binario (CODIFICACION) cuya longitud en bits dependerá de la aplicación que se este tratando (por ejemplo: Telefonía digital, Audio con calidad de CD, mezclas digitales,etc).

1.4.1 Cuantificación

Es el proceso mediante el cual se asigna un valor de salida perteneciente a un conjunto discreto de valores , a un cierto margen de valores posibles de entrada.

Al aplicar cuantificacion a una señal se obtiene compresión, ya que el conjunto de posibles valores de salida es menor al de entrada. La desventaja de este paso radica en el error que se introduce, llamado también ruido de cuantificacion, el cual lo convierte en un proceso irreversible.

El compromiso que existe entre la compresión y el error introducido por la cuantificacion, se puede expresar del siguiente modo: "Cuanto mayor sea el conjunto de posibles valores de salida, menor será el error de cuantificacion, pero la compresión resultante será menor, ya que en la codificación, el numero de bits necesarios para representar todos los posibles valores de salida será mayor". Además se tiene la siguiente relación que evidencia este compromiso:

$$Pot_{RUIDO}(dB) = 20log(Am) - 4.77 - 6.02(b) \quad \text{..... (1.1)}$$

Donde:

Am : es el valor absoluto máximo del cuantificador.

b : es el número de bits de la codificación.

El concepto dado corresponde a la cuantificacion escalar . No se hace una mayor explicación de este caso, puesto que es muy conocido y no será de aplicación en el presente trabajo. La cuantificacion escalar puede ser utilizada cuando se traten señales unidimensionales tales como la voz(señal función del tiempo) sin embargo cuando se pretende discretizar señales de dos o mas componentes, por ejemplo para imágenes, es necesario utilizar una CUANTIFICACION VECTORIAL. A pesar de ser la voz una señal

unidimensional, si la representamos a través de sus características(formantes, pitch, etc), podríamos obtener a partir de ella una cuantificacion vectorial, lo cual produciría además una compresión adicional muy útil por ejemplo en Voz sobre IP.

Para diferenciar la cuantificacion escalar de la vectorial , podríamos decir que para el primer caso la relación salida vs entrada se representa mediante una función de tipo escalera o un arreglo que contiene los márgenes de entrada y su correspondiente valor de salida, en cambio, en la cuantificacion vectorial se utiliza un libro de códigos, llamado también CODEBOOK. El codebook, es una tabla que contiene los vectores posibles a la salida del cuantificador. Cada vector del codebook se conoce como CODEVECTOR y su correspondiente índice en la tabla (ubicación) se denomina CODEWORD.

Al igual que en la cuantificacion escalar, para obtener una representación eficiente, se almacena el índice de la tabla para cada una de las muestras de la señal de entrada y la tabla que permite obtener el correspondiente valor de salida.

En el diseño de un cuantificador vectorial hay que escoger el numero de bits del codebook (#codewords=$2^{\#bits\ codebook}$), la dimensión de los vectores de entrada, y los valores de los codevectores para que la cuantificacion implique el mínimo error posible de la señal cuantificada respecto a la original.

1.4.2 Proceso de cuantificación vectorial

Cuantificar un vector de entrada (x) consiste en encontrar su "vecino mas cercano" (y_j) dentro del codebook. Para ello hay que calcular la distancia euclídea entre (x) y todos los vectores del codebook, y escoger el que proporcione un valor menor (el más cercano).

El algoritmo para este proceso podría plantearse del modo siguiente:

1. Inicializar la distancia mínima (d) igual a la distancia inicial (D_1), y los contadores : (j) ; índice del codevector correspondiente e (i).

   $$d = D_1 \; ; j=1 \; ; \; i=1$$

2. $D_{j+1} = d(x, y_{j+1})$: Distancia Euclidiana entre 2 vectores.
3. Si: $D_{j+1} < d \Rightarrow d = D_{j+1} ; i = j+1$
4. Si: $j < k \Rightarrow j = j+1$; y volver al paso 2.
5. Si: $j = k$; el resultado es el indice "i" (codeword) y la distancia mínima será (d). Donde (k) es el numero de codevectores.

CAPÍTULO II
TÉCNICAS DE CODIFICACIÓN DE LA VOZ

2.1 Codificadores de forma de onda y VOCODERS

En principio los codificadores de forma de onda, están diseñados para ser independientes del tipo de señal, es decir ellos pueden ser igualmente de buenos para para una amplia variedad de señales,por ejemplo: voz, música, tonos, datos en la banda de voz. Además que, tienden a ser robustos para un amplio rango de características del locutor y para entornos ruidosos. Para mantener estas ventajas con una mínima complejidad, los codificadores de forma de onda típicamente apuntan por una economía moderada en la velocidad de transmisión binaria. Los codificadores de forma de onda pueden ser optimizados si se realizan para señales mas especificas, lo cual produciría una mejor eficiencia en la codificación. Usualmente la optimización se logra observando las variables estadísticas de una señal dada (por ejemplo la voz). La columna vertebral de estos codificadores esta basada en una caracterización estadística de formas de onda de la voz.

Otro tipo de codificadores de voz necesitan el soporte de una descripción de la voz usando un conocimiento previo de cómo la señal fue generada en la fuente, la idea es que las restricciones físicas de la generación de la señal sean cuantificadas para lograr una ventaja en eficiencia en la descripción de

la señal. Estos codificadores también se conocen como “codificadores de fuente” o VOCODERS.

Dado que este ultimo tipo de codificadores se basan en la obtención de los parámetros de un modelo de producción de señal de voz, también se les conoce como codificadores parametricos. En el emisor se analiza la señal de voz, correspondiente a un segmento temporal considerado estacionario, y se obtiene el conjunto de parámetros que lo han generado. Estos parámetros son los que se transmiten y el receptor reproduce con ellos una señal de características análogas a la señal original.

Los codificadores parametricos, analizan, por predicción lineal, el filtro correspondiente al modelo de generación de señal de voz. Este modelo, muestreando a 8 Khz, presenta cinco cavidades resonantes, que se corresponden con un filtro de orden 10, se obtiene un primer conjunto de parámetros. El resto de parámetros se utiliza para modelar la señal de excitación del filtro y existen diversos esquemas, entre los que destacan:

a) Vocoder LPC (Linear Predictive Coding).- En el cual , la señal de excitación se puede representar como un tren de impulsos para las tramas sonoras y por ruido blanco en las tramas sordas.

b) CELP (Predicción Lineal Excitada por Codebook).- Al igual que con los coeficientes del filtro,puede restringirse el numero de excitaciones posibles creando una tabla con las excitaciones mas representativas. En este caso hay que escoger la excitación optima para cada trama y enviar el índice que marca la posición en el codebook de la excitación seleccionada.

c) MULTIPULSO.- Una de sus variantes esta implementada en el estándar de codificación europeo de GSM (Sistemas Globales para comunicaciones Móviles). Consiste en guardar la posición y la amplitud de una serie de impulsos(típicamente 8 o 12), obtenidos mediante un proceso de analisis-sintesis, de forma que se minimice el error de la señal sintetizada.

En todos estos casos, la señal reconstruida mantiene el contenido espectral de la señal original, pero no su forma de onda. Por ello la evaluación de este tipo de sistemas es siempre subjetiva.

2.2 Predicción lineal

En el análisis de series temporales el modelado de señal mas utilizado consiste en expresar una señal "*S*" en el instante "n" ("*S* [n]") como una combinación lineal de muestras de "*S* [n]" en instantes previos. Es decir el modelo corresponde a un sistema lineal que predice valores futuros, y para el cual se tiene la siguiente recursion:

$$\check{S}[n] = \sum_{k=1}^{p} a_k S[n-k] + \sum_{r=1}^{q} b_r \check{S}[n-r] \cong S[n] \qquad \text{.....(2.1)}$$

Normalmente se escoge $b_r = 0$ ($1 \leq r \leq q$), de tal modo que el predictor lineal se reduce a un filtro FIR (todo ceros):

$$\check{S}[n] = \sum_{k=1}^{p} a_k S[n-k] \qquad \text{.....(2.2)}$$

Los parámetros de este modelo son: a_k (coeficientes de predicción lineal) y P (orden del predictor), y pueden calcularse a partir de la minimizacion del error entre la señal original y la señal predecida. El error de esta prediccion se ilustra en la figura 10 :

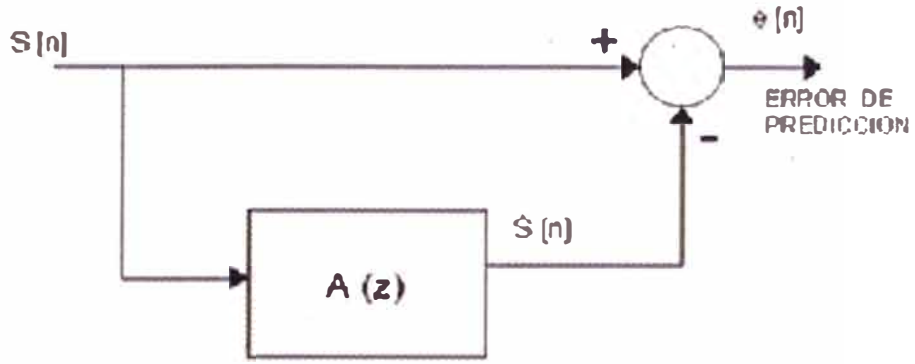


FIGURA 10.- Esquema de predicción

Donde $A(z) = a_1 z^{-1} + a_2 z^{-2} + \ldots\ldots + a_p z^{-p}$

Para la voz, se debe hacer un tratamiento localizado, puesto que si consideramos toda la señal, seria intratable a través de esta tecnica. Como se estudio en el capitulo I, el comportamiento de la voz es cuasi estacionario para bloques de 20 a 30 ms. Podemos aplicar el análisis predictivo para cada uno de estos bloques (previamente se aplica una ventana de hamming a las muestras que se analizaran, para evitar discontinuidades en los extremos de dichos bloques) y tener un esquema de predicción en cada bloque.

El orden P, del análisis, será el punto de partida para evaluar la precisión del sistema, sin embargo se puede comprobar que a partir de cierto valor no se consigue una importante mejoría, esto se justifica en que, a

medida que las muestras son mas distantes, estas son menos parecidas (disminuye la correlación entre muestras). La experiencia ha demostrado que debe existir una relación entre el orden P y la frecuencia de muestreo Fs :

$$P \geq Fs\ (KHz) + 4 \quad \text{.....(2.3)}$$

2.3 Codificación predictiva lineal (LPC)

La Codificación Predictiva Lineal(LPC) es una de las mas potentes técnicas de análisis de voz y uno de los métodos mas usados para codificar/decodificar buena calidad de voz a una baja velocidad binaria. Este método provee estimaciones de los parámetros de voz bastante precisas y una relativa eficiencia para su calculo computacional

La técnica LPC parte de asumir que la señal de voz es producida por un "zumbador" en el extremo de un tubo. La glotis produce el zumbido, el cual esta caracterizado por su intensidad y su frecuencia (pitch). El tracto vocal forma el tubo, el cual esta caracterizado por sus resonancias (formantes).

LPC, analiza la señal de voz mediante la estimación de los formantes, retirando sus efectos de la señal de voz y estimando la intensidad y frecuencia del zumbido remanente. El proceso de retirar los formantes es llamado "filtrado inverso", y la señal remanente es llamada "residuo".

Los numeros que describen los formantes y el residuo podran luego ser almacenados o transmitidos. La técnica, LPC, sintetiza la señal de voz a través de un procedimiento inverso; para lo cual usa el residuo para crear

una fuente de señal, usa los formantes para crear un filtro(el tubo), y luego procesa la fuente a través del filtro, para producir la voz.

Como la señal de voz varia con el tiempo, este proceso se realiza sobre segmentos de la señal de voz, las cuales se llaman “frames”, “bloques” o “ventanas”.

2.3.1 Estimación de los formantes

El problema del sistema LPC es determinar los formantes de la señal de voz. La solución mas utilizada se obtiene a partir de la ecuación del predictor lineal.

Los coeficientes de la ecuación del predictor (coeficientes de predicción) caracterizan los formantes. La estimación de estos coeficientes se realiza a partir de la minimizacion del error cuadratico entre la señal predicha y la señal actual. Para lo cual se establecen algoritmos que implementan métodos de la autocorrelacion, covarianza, etc, los cuales aseguran la convergencia a una sola solución y una eficiencia en el computo.

2.3.2 Estimación del pitch

Existen diversos métodos para estimar la frecuencia de pitch. Uno de ellos se fundamenta en la detección del primer pico de la autocorrelación dentro del margen de valores de periodo de pitch posibles(16 a 160 muestras para una frecuencia de muestreo de 8Khz). Además habrá que comprobar que la relación entre la amplitud de dicho pico y la autocorrelación en el origen sea superior a un cierto umbral.

Es lógico, que debemos estimar el pitch solo en “frames” (o tramas) sonoras, lo cual puede ser discriminado mediante un umbral de energía. El

método de la autocorrelación será bastante preciso si previamente realizamos un filtrado inverso para eliminar la información del tracto vocal.

2.4 CEPSTRUM

Sea x[n] una secuencia causal y estable cuya transformada Z es X(z). El cepstrum de x[n] se define como la transformada inversa del logaritmo de X(z), es decir:

$$\text{Cepstrum}\{ x[n] \} = F^{-1} \{\log | X(z) |\} \quad \text{.....(2.4)}$$

Donde:

$$X(z)=F\{x[n]\} \quad \text{.....(2.5)}$$

Dado que el sistema de producción del habla se puede modelar mediante una señal de excitación y un filtro lineal invariante en el tiempo, para cada trama, entonces se tiene el siguiente modelo:

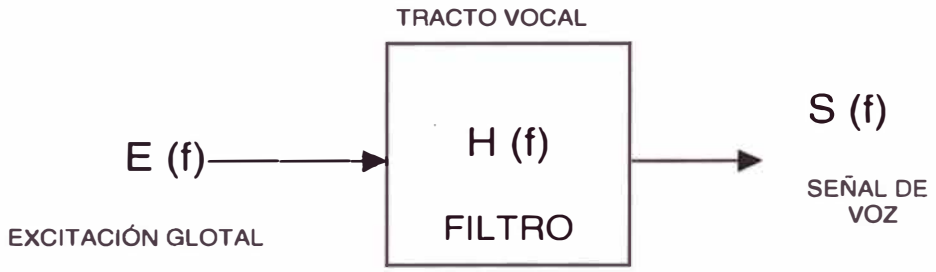


FIGURA 11.- Modelo del sistema reproductor de voz

A partir de la figura 11, en el dominio de la frecuencia, tendríamos:

$$S(f) = E(f)\, H(f) \quad \text{.....(2.6)}$$

Tomando logaritmos:

$$\log|S(f)| = \log|E(f)| + \log|H(f)| \quad \text{.....(2.7)}$$

De modo que ambas informaciones están contenidas en el cepstrum pero se encuentran separadas, puesto que son de frecuencias distintas. La presencia del tracto vocal en el cepstrum se mostrara a través de una variación lenta,mientras que la de la señal de excitación se mostrara a través de unos picos periódicos. En conclusión, se puede obtener un método de detección del pitch de una determinada trama de voz a partir del cepstrum.

Existe un procedimiento bastante simple para calcular los coeficientes cepstrum a partir de los coeficientes LPC (a_i), con una complejidad computacional menor que la definición del cepstrum. Los coeficientes cepstrum obtenidos a partir de los coeficientes LPC reciben el nombre de Coeficientes Cepstrum LPC (LPCC) y se obtienen a partir de la siguiente recursión:

$$C_n = -a_n - \sum_{m=1}^{n-1} \frac{m}{n} c_m \, a_{n-m} \; ; \; 1 \leq n \leq P \qquad \text{.....(2.8)}$$

Donde “P” es el orden del predictor y “c_i“ son los coeficientes LPCC.

2.5 Cuantificacion vectorial de coeficientes LPC ó LPCC (LPC-QV)

Como ya se explico, el tracto vocal puede modelarse con la tecnica de los coeficientes LPC , los cuales se actualizan cada 20ms, de modo tal que para una duración de algunos segundos de la voz, tendríamos una gran cantidad de valores correspondientes a los coeficientes LPC, esto conducirá a recurrir a una representación vectorial para tener una forma mas eficiente del manejo de toda la información adquirida a través del predictor.

La cuantificacion vectorial de los coeficientes LPC , se refiere a la asignación de una etiqueta de dirección, a cada uno de los juegos de coeficientes LPC acomodados como vectores, luego de una búsqueda en el CODEBOOK del codevector mas cercano al vector que intentamos cuantificar.

CAPÍTULO III
IDENTIFICACIÓN DE LOCUTOR

3.1 Identificación de locutor

Para toda aplicación que involucre la toma de decisiones, ejecutar ordenes,etc a partir de una comunicación oral hombre-maquina, será necesario afrontar el problema de reconocimiento o identificación automático del habla. Dependiendo del caso, el sistema deberá confirmar previamente que el usuario esta autorizado a acceder a la información, dar ordenes, etc, y es aquí donde nos enfrentamos a otro problema: " La identificación de locutor", de manera que el sistema pueda determinar quien es la persona que pretende accesarlo. De hecho,en situaciones donde la seguridad sea un factor prioritario, la voz será la característica biometrica personal e intransferible mas fácil de enviar a través de las redes telefónicas.

Existen aplicaciones que requieren el reconocimiento del habla y de locutor simultáneamente. Sin embargo para lograr un mejor estudio del tema, estas aplicaciones se dividen en 2 grupos:

- Reconocimiento del habla y
- Reconocimiento de locutor

Otros grupos derivados podrían incluir: reconocimiento del idioma de una determinada frase hablada, variantes dialectales,etc.

3.2 Aplicaciones

Las principales aplicaciones en el reconocimiento de locutor son:

- Acceso a lugares o informaciones: A través de la voz es posible identificar a las personas y permitirles accesos o no a residencias, embajadas, fabricas, informaciones bancarias privadas o de grupos de trabajo, etc.
- Reconocimiento de sospechosos: A partir de grabaciones policiales, se puede determinar a que individuo pertenece la voz de la grabación.
- Transcripción automática de reuniones: En reuniones con diversos oradores, donde interese disponer de la transcripción de las frases habladas(reconocimiento del habla), así como de las identidades de las personas que han realizado los discursos.

3.2.1 Clasificación

Dentro de estas aplicaciones en el reconocimiento de locutor se pueden utilizar diferentes sistemas de identificación, cuyos procedimientos se pueden clasificar en:

a) Identificación.- Consiste en determinar la identidad de un locutor mediante un computador. Para esto, la voz de entrada deberá ser conocida previamente por el computador, del mismo modo que cuando se comunican invidentes. El conocimiento de las diferentes voces se adquirirán a través de un proceso de entrenamiento, en el cual se recogerá un conjunto suficientemente grande de frases habladas de cada uno de los locutores que formaran parte de la base de locutores conocidos por el sistema. Luego, en el proceso de identificación de un

locutor de entrada, se realizara un test consistente en comparar los datos de entrada con todos los existentes en la memoria del sistema. A partir de la comparación se decidirá la identidad del locutor de entrada.

b) Verificación.- Para este caso, el locutor de entrada suministra su identidad, y el sistema deberá comprobar los datos de entrada con los almacenados en memoria de ese mismo locutor, para verificar si es quien dice ser. Para prevenir suplantaciones de personalidad mediante grabaciones, será necesario realizar un reconocimiento dependiente del texto, el cual deberá ser diferente en cada sesión, y comprobar mediante reconocimiento del habla que el texto pronunciado es el correcto.

Se puede considerar otra clasificación de estos sistemas:

a) Sistemas independientes del texto.- El proceso de identificación debe funcionar correctamente para cualquier texto usado en los procesos de entrenamiento y prueba.

b) Sistemas dependientes del texto.- El texto debe ser el mismo en los procesos de entrenamiento y prueba. Es claro que la dependencia del texto permitirá mejorar el comportamiento, pero en algunas aplicaciones esto puede ser un inconveniente, o imposible, por desconocer el locutor que esta siendo sometido a una identificación.

Una tercera clasificación que se refiere a como se ha tomado el conjunto de locutores, es la siguiente:

a) Conjunto cerrado.- Consiste en identificar a un locutor de entre un conjunto de “M” locutores conocidos.

b) Conjunto abierto.- Consiste en decidir si un locutor determinado pertenece a un conjunto de "M" locutores conocidos. En este caso, no es necesario decidir cuál es de los "M" locutores. Cuando M=1 (caso particular), se trata de una identificación de locutor.

En la producción del habla intervienen elementos como la semántica, lingüística, articulación y acústica. Todos ellos condicionan las propiedades acústicas de la señal de voz. Las diferencias debidas intrínsecamente al locutor son el resultado de una combinación de diferencias anatómicas del tracto vocal y de la forma de hablar adquirida por cada persona. En el reconocimiento de locutor se utilizan estas diferencias para discriminar entre locutores.

3.3 Pasos a seguir en el procedimiento de identificación de locutor

1. Adquisición de la señal de voz.
2. Extracción de las características (coeficientes LPC, cepstrum, etc.).
3. Comparación de las características sometidas a verificación contra las almacenadas en la etapa de entrenamiento y la consecuente toma de decisión.

3.3.1 Adquisición de la señal de voz

Consiste en captar la onda acústica producida por el locutor por medio de un micrófono y digitalizarla para poder tratarla. En este paso se debe tener presente que tanto el entorno (ambiente) donde se tome la muestra en la etapa de entrenamiento como en la verificación deberían ser los mismos, de igual modo el micrófono utilizado en captar la voz de un locutor debe ser el mismo, de otro modo las tasas de reconocimiento podrían verse afectadas.

3.3.2 Extracción de características

Para poder obtener las características del modelo de reproducción humano, debemos dividir la señal de voz en tramas de 10 a 30 ms de forma de onda para construir un vector de características de N dimensiones.

Los coeficientes LPCC como ya se demostró contienen una información completa del sistema de producción de voz, por esta razón, son de bastante utilidad en el proceso de extracción de características de voz de un locutor.

3.3.3 Decisión

Finalmente una secuencia de vectores se compararan con los modelos almacenados en la secuencia de entrenamiento para decidir el resultado del reconocimiento.

Por medio de la parametrización, es posible reducir el numero de datos a tratar además que se consigue transformar la señal de voz a un nuevo espacio de características , en el cual será mas sencillo diferenciar entre locutores.

Los vectores de características producidos por cada individuo se pueden considerar muestras de una función de densidad de probabilidad. Las distribuciones de los distintos locutores se solaparan en algunas zonas del espacio N-dimensional, pero será posible distinguir los locutores entre si, si las características extraídas presentan un alto grado de capacidad discriminativa. Entre las características mas usadas destacan:

- Coeficientes LPC
- Coeficientes Cepstrum

Luego, a partir de los vectores de características se puede extraer un modelo de locutor, que pertenecerá a una de las siguientes clases:

- Modelo Parametrico: Implica una estructura caracterizada por unos parámetros, como se explica en el parrafo anterior.
- Modelo no parametrico: No supone ningún modelo de la función de densidad de probabilidad (Utiliza codebooks).

El modelo parametrico presenta la ventaja de necesitar pocos datos para especificar la función de densidad de probabilidad, presentando algunas restricciones, como por ejemplo una gran variabilidad entre locutores.

El modelo no parametrico, es menos restrictivo, por lo que puede permitir un mejor modelo pero requiere un mayor numero de vectores de características. Esto restringe el uso de los modelos no parametricos y de vectores de características con un numero elevado de componentes.

CAPÍTULO IV
IMPLEMENTACIÓN DE ALGORITMOS CON MATLAB PARA LA IDENTIFICACIÓN DE LOCUTOR

4.1 Objetivo

El objetivo de este capitulo es implementar funciones en MatLab para el pre-procesamiento de la voz , basándose en el desarrollo teórico de los capítulos previos, para luego desarrollar programas que permitan manejar los datos obtenidos en esta etapa de pre-procesamiento, con miras a discriminar un locutor de otro, con solamente las características de la voz extraídas en cada prueba.

Empezare mostrando un diagrama de bloques descriptivo del proceso de identificación de locutor (Figura 12), para luego ilustrar la implementación de programas en MATLAB, por medio de diagramas de flujo y algoritmos de procedimiento que permitirán inclusive ser usados por algún otro software de elección. Los resultados obtenidos del sistema bajo prueba se muestran en el siguiente capitulo.

Son quince (15) los programas que se prepararon para la implementacion de este sistema, el pre-procesamiento de la voz se simplifico bastante, gracias a la existencia de funciones que vienen con el MATLAB V5.3 (versión estudiante), tal como lpc.m , filter.m , etc. Todos los programas se muestran en el anexo A, con los siguientes nombres:

1) idlocutor.m : Reconoce a los locutores
2) busca.m : Cuantifica vectorialmente la grabación de voz
3) dist.m : Calculo de la distancia euclidea entre 2 vectores
4) hector.m : Pre-procesamiento de la voz
5) acotar.m : Detección y eliminación del silencio de la sala de grabación
6) cepstrum.m : Calcula los coeficientes cepstrales
7) entrena.m : Entrega la secuencia de entrenamiento
8) cbookh.m : Obtención del codebook para el locutor 1
9) cbookl.m : Obtención del codebook para el locutor 2
10) mejora1.m : Entrega un codebook de 2 bits
11) mejora2.m : Entrega un codebook de 3 bits
12) mejora3.m : Entrega un codebook de 4 bits
13) mejora4.m : Entrega un codebook de 5 bits
14) mejora5.m : Entrega un codebook de 6 bits
15) mejora6.m : Entrega un codebook de 7 bits

Los codebooks obtenidos para cada locutor tambien se muestran en el anexo A. Se incluye un archivo de texto: "leer.doc", el cual indica los pasos previos a seguir para ejecutar el programa de identificación y el procedimiento que permite obtener el CODEBOOK de un locutor, a partir de un vocabulario elegido. Este archivo ,tambien sugiere una forma de ampliar el numero de locutores a identificar, en este sistema,creando funciones de generacion del CODEBOOK para cada uno de ellos y modificando la rutina de identificación.

Diagrama de bloques completo del proceso de identificación del locutor implementado.

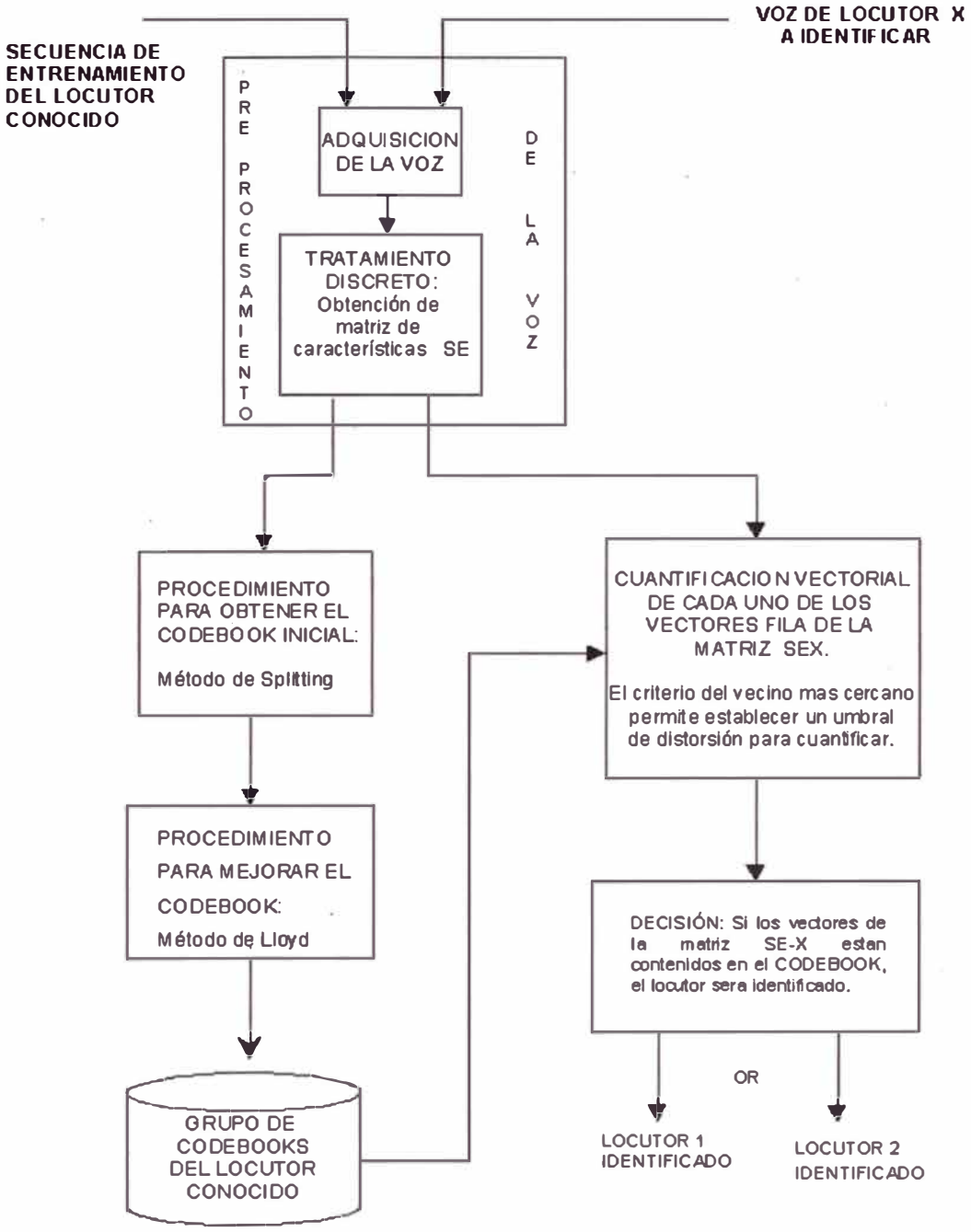


FIGURA 12

4.2 Pre-procesamiento de la voz

Esta etapa corresponde a la fase inicial del trabajo implementado, tal como se indica en el diagrama de bloques anterior. A continuacion se detallan las caracteristicas relativas a la adquisicion de voz y el posterior tratamiento discreto de las muestras de voz.

4.2.1 Bloque de adquisición de la voz

Las muestras de voz a procesar han sido grabadas en una computadora portátil con procesador INTEL PENTIUM I, para lo cual se utilizo la tarjeta electrónica de sonido y el micrófono incorporados. La sala de grabación fue siempre la misma , tanto en la etapa de entrenamiento como para las pruebas de identificación, tratando en lo posible que las condiciones de silencio del entorno sean las mas adecuadas para evitar el ruido. El software de grabación utilizado fue el Grabador de Sonidos de WINDOWS , en el cual se escogió las siguientes características de grabación:

- Formato: PCM
- Frec. de muestreo: 8KHz
- 8bits de codificación

Cada palabra grabada es almacenada en un directorio del disco duro (*.wav), desde el cual será leída con la función **WAVREAD ('nombre de archivo')**, previo direccionamiento a través del comando **PATH (PATH, 'ruta del archivo')**, cuando sea necesario su procesamiento (segmentación y extracción de características).

4.2.2 Tratamiento discreto de la voz

El siguiente diagrama (Figura 13) detalla el proceso a seguir en el tratamiento discreto de la voz :

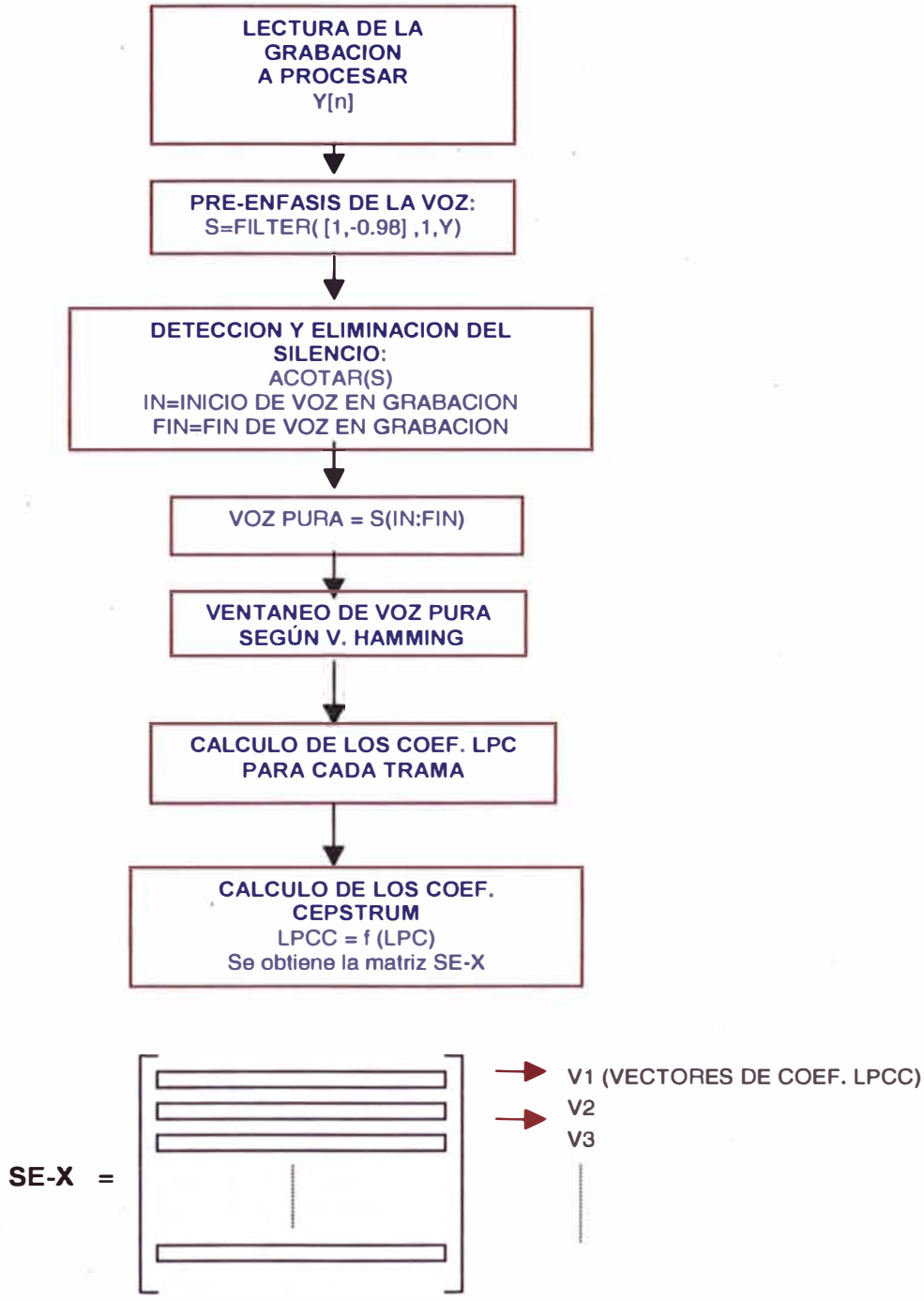


FIGURA 13

Las premisas para el tratamiento discreto de la voz son:

La señal de voz al ser adquirida por la tarjeta de sonido (A/D), ha sido cortada aproximadamente a 4KHz para evitar el "aliasing" (Frec. de muestreo es 8KHz), por lo que frecuencias en las proximidades de la frecuencia de corte, dentro de la banda base de la señal, de este filtro no ideal, también se verán afectadas en amplitud. Para evitar perder la información que estas frecuencias llevan, se recomienda un procedimiento de pre-enfasis o pre-acentuación, como compensación en la etapa del tratamiento discreto de la señal de voz.

Todas las grabaciones contienen áreas de señal que no presentan información de voz, es decir son irrelevantes, por lo que se hace necesario eliminar estas áreas que se encuentran antes y después de la palabra pronunciada. Este procedimiento se denominara detección y eliminación del silencio de la sala de grabación.

La voz es una señal no estacionaria, sin embargo es cuasi-estacionaria en intervalos pequeños de tiempo, por lo cual se realiza un análisis localizado para cada intervalo pequeño de tiempo (aprox. 20 ms). Este proceso se conoce como segmentación en tramas o ventaneo.

En 20 ms a 8KHz de muestreo, se tienen 160 muestras por ventana, luego utilizando un predictor lineal, se puede calcular los coeficientes del filtro FIR que modelaría el tracto vocal (Coeficientes LPC), para las 160 muestras producidas.

4.2.2.1 Pre-enfasis de voz

Un método de pre-enfasis simple y ampliamente usado es el filtrado lineal con un filtro descrito por una ecuación en diferencias de primer orden, de la forma:

$$Y[n] = x[n] - \alpha\, x[n-1]; \quad 0.5 < \alpha < 1 \quad \text{.....(4.1)}$$

Donde $x[n]$ es la señal de voz de entrada, $y[n]$ es la "voz pre-enfatizada" de salida y α es un parámetro ajustable , que permitirá se realcen las altas frecuencias de la voz. Se recomienda : α =0.98.

Para nuestro algoritmo se tendrá: $Y[n] = x[n] - 0.98\, x[n-1]$ .....(4.2)

Pasando al dominio de "z" : $Y(z) = (1 - 0.98\, z^{-1})\, X(z)$ .....(4.3)

Para implementar este proceso de compensación, se utilizara la función **FILTER ([1 , -0.98], 1, *X*)** , la cual entregara una secuencia de voz pre-enfatizada.

4.2.2.2 Detección y eliminación del silencio de la sala de grabación

En todo proceso de adquisición de señal de voz, se generan áreas sin información de voz, pero que forman parte de la señal adquirida. Procesarlas seria irrelevante para la extracción de características de la voz. Para detectar el inicio y fin de voz pura, se realiza el proceso de adquisición del silencio (a micrófono abierto), para determinar el nivel máximo del silencio (umbral) en la secuencia de muestras obtenidas , luego se utiliza un criterio de ausencia total de sonido en cada segmento de la señal de voz, otro criterio podría ser el nivel de energía o los cruzes por cero antes y después de la palabra. Para este caso he utilizado el primer criterio, comprobando resultados eficientes.

En la Figura 14 se muestra el diagrama de flujo utilizado para eliminar las zonas irrelevantes en una grabación de voz, este algoritmo tendra como objetivo calcular el instante de tiempo en que se inicia la señal de voz.

Para obtener el **FIN DE VOZ** , tenemos que hacer una exploración de la secuencia grabada en sentido inverso, de modo tal que en el diagrama de flujo, cambiaríamos el inicio del bucle por: FOR i= # TRAMAS : 1 y el calculo del **FIN DE VOZ** seria: "20*(i)" en ms.

Se verifico que toda trama, ya sea sonora o fricativa. es conservada al eliminar el silencio de la sala de grabación, si se considera la relación planteada en el bloque condicional, para el vocabulario de la secuencia de entrenamiento.

4.2.2.3 Analisis discreto con ventanas deslizantes: "VENTANEO"

La señal de voz se caracterizan por tener cambios en las características frecuenciales respecto al tiempo (es no estacionaria). Por tanto, su análisis debe ser local y dependiente del tiempo. Para este estudio necesitamos de la transformada localizada de Fourier (STFT), definida como:

$$X_n(e^{j\lambda}) = \sum_{m=-\infty}^{\infty} w[n-m]\, x[m]\, e^{-j\lambda m} \qquad \text{.....(4.4)}$$

$$X_n(e^{j\lambda}) = e^{-j\lambda n} \sum_{m=-\infty}^{\infty} w[-m]\, x[n+m]\, e^{-j\lambda m} = e^{-j\lambda n}\, \tilde{X}_n(e^{j\lambda}) \qquad \text{.....(4.5)}$$

Donde $-\infty < n < \infty$ y $0 \le \lambda \le 2\Pi$ (o cualquier otro intervalo de longitud 2Π).

El concepto de espectro variante con el tiempo es la base de muchos de los algoritmos mas útiles de procesado en tiempo discreto para señales de voz. La ecuación (4.5) indica que es posible obtener el espectro de la señal de voz superponiendo espectros de segmentos de señal de voz..

DETECCION DEL SILENCIO ANTES DEL INICIO DE VOZ:

DIAGRAMA DE FLUJO

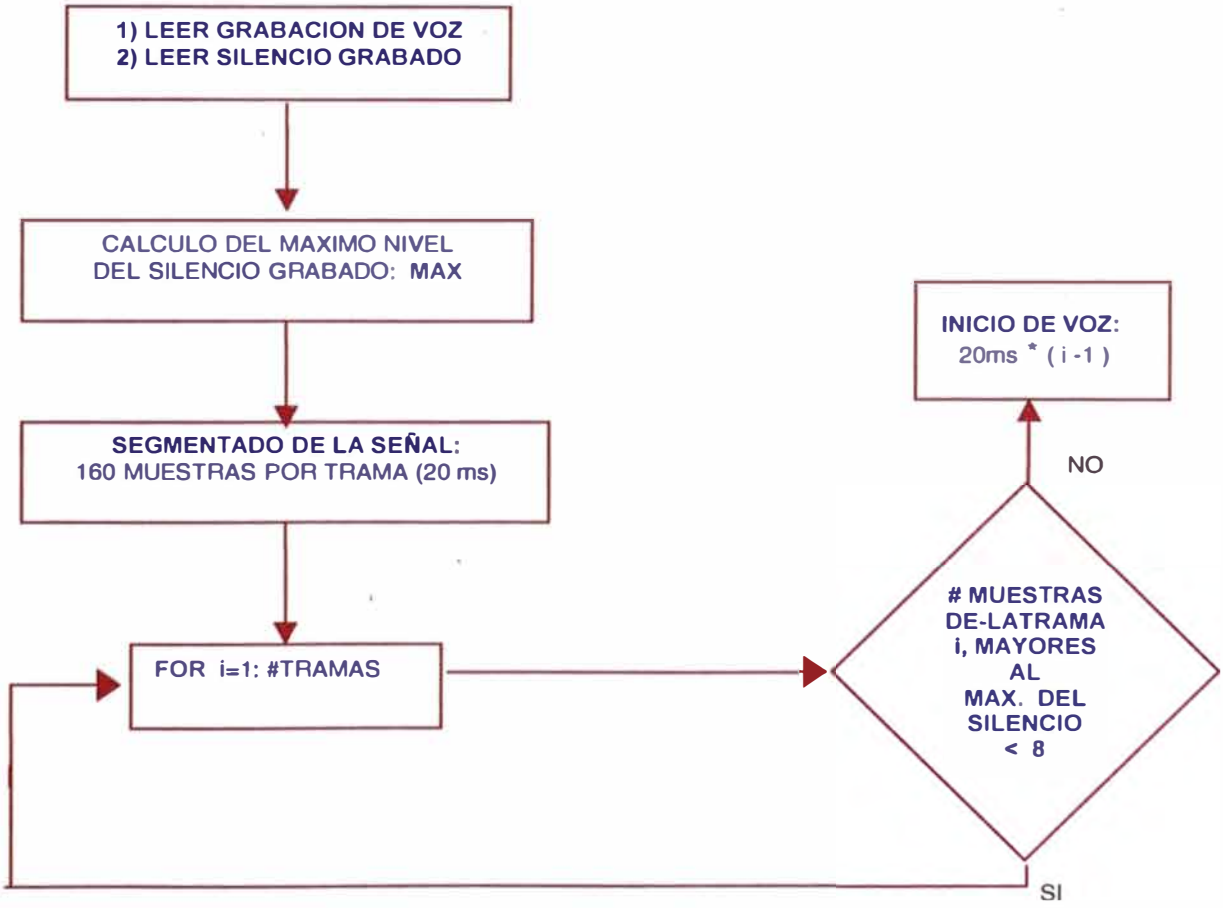


FIGURA 14

Se puede evaluar la STFT en un conjunto discreto de frecuencias $\lambda = 2\Pi K/N$, y en un tiempo fijo n mediante el uso de la DFT.

Sea w una ventana de longitud L, entonces: w[-m] = 0 para todo valor de "m" fuera de esta longitud L. Entonces la ecuación anterior se transformaría en:

$$X_n[k] = e^{-j(2\Pi/N)kn} \sum_{m=0}^{L-1} w[-m]\, x[n+m]\, e^{-j(2\Pi/N)km} = e^{-j(2\Pi/N)kn}\, \tilde{X}_n[k] \qquad \text{.....(4.6)}$$

Si :

$$w[-m] = \tilde{w}[m] \qquad \text{.....(4.7)}$$

$$\tilde{X}_n[k] = \sum_{m=0}^{L-1} \tilde{w}[m]\, x[n+m]\, e^{-j(2\Pi/N)km} \qquad k = 0,1,2,\ldots,(N-1) \qquad \text{....(4.8)}$$

De las ecuaciones (4.7) y (4.8), se nota que :

$$\left|X_n[k]\right| = \left|\tilde{X}_n[k]\right| \qquad \text{.....(4.9)}$$

Entonces, *$X_n[k]$, puede calcularse siguiendo los siguientes pasos:*

1. Se seleccionan L muestras de la señal en el instante n:

$$\{ x[n],\ x[n+1],\ x[n+2],\ldots\ldots,\ x[n+L-1] \} \qquad \text{.....(4.10)}$$

Si se utilizan ventanas simétricas, resultara conveniente que n sea el centro del intervalo de duración de la ventana.

2. Se multiplican las muestras de la señal de voz por las muestras de la ventana, formando la secuencia:

$$\{ \tilde{w}[m]\, x[n+m] \} \quad m = 0,1,2,\ldots,L-1 \qquad \text{.....(4.11)}$$

La secuencia resultante será una matriz, obtenida a partir del producto de elemento a elemento (operador: ".*"), la cual es también conocida como trama o segmento de voz enventanado.

3. Luego podemos calular la DFT para N puntos del "segmento de voz enventanado. Si N>L se rellenara con ceros.
4. Si queremos obtener la STFT., tendríamos que multiplicar por el factor exponencial de la ecuación respectiva.
5. Se repiten los pasos del 1 al 4 para cada valor de n, hasta recorrer toda la señal de voz o el segmento de voz que se quiera estudiar.

Para codificar la fuente de producción de la voz, necesitamos modelar el tracto vocal por medio de los coeficiente LPC para cada ventana generada.

Existen una serie de modelos de "ventanas" que se pueden utilizar, a saber:

-Ventana Rectangular

-Ventana Triangular o de Bartlett

-Ventana de Hanning

-Ventana de Hamming

-Ventana de Blackman

Es usual para el análisis de voz emplear la ventana de Hamming (ver figura 15), ya que la rectangular queda descartada por presentar transiciones abruptas en sus extremos de discontinuidad. Todas las demás no presentan discontinuidad en sus extremos , sino mas bien caen a cero suavemente.

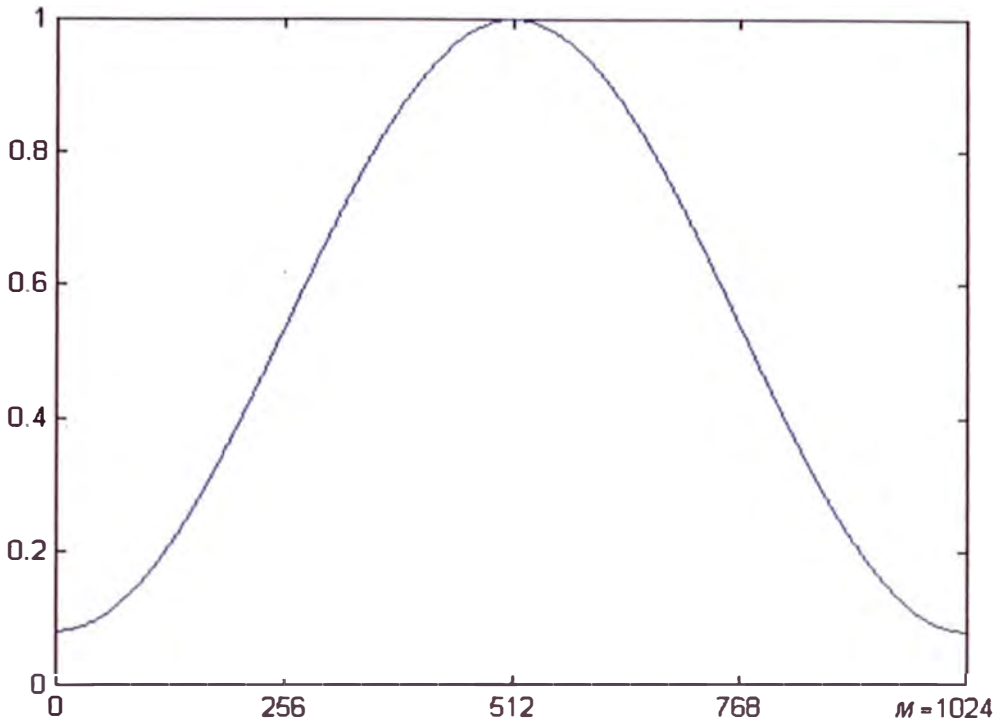


FIGURA 15

4.2.2.4 Consideraciones previas al "ventaneo"

1. Como la ventana de Hamming es simétrica, escojo un numero impar de muestras para el tamaño del bloque de voz a ventanear (m = # muestras), para obtener un traslape al 50%.

   El valor elegido fue: m=161.

2. El inicio de cada ventana es desplazado 81 muestras (aprox. 10ms) con la finalidad de evitar cambios bruscos en los coeficientes LPC (características de voz) de cada trama o ventana. Es decir las ventanas estarán traslapadas al 50%.

   El proceso a través de MATLAB se describe en las siguientes líneas:

```
%VENTANEO POR HAMMING
ham=hamming(muestras);%Ventana de hamming.
for k=1:ventanas,
   inicio=1+((muestras-1)*(k-1))/2;%inicio de cada trama
   trama(1:muestras,k)=voz(inicio:inicio+muestras-1).*ham;
```

```
end
```

Donde se observa que "trama" será la matriz que contendrá los coeficientes LPC de las ventanas de la señal de voz ordenados en columnas.

4.2.2.5 Cálculo de los coeficientes LPC Y LPCC

Esta operación se refiere a la extracción de las características de la voz en cada ventana. Para lo cual se cuenta con la función **LPC (....),** propia del MATLAB, la cual simplificara enormemente los cálculos.

```
%CALCULO DE COEF. LPC PARA CADA VENTANA
coeflpc=lpc(trama,20);%orden del predictor igual a 20
[a,b]=size(coeflpc);
%%Selección de solamente los coeficientes del predictor%%
for j=1:a
   predictor(j,:)=coeflpc(j,2:b);
end
%CALCULO DE LOS COEF. CEPSTRALES PARA CADA VENTANA.
for i=1:ventanas
   LPCC(i,:)=cepstrum(predictor(i,:));%coef. Cepstrum para cada
ventana
end
SEX=LPCC(1:ventanas,:);%matriz con los vectores de características
de la voz.
```

Los coeficientes CEPSTRALES contienen una información completa de las características de la voz, por lo que para esta aplicación son los ideales.

Una vez extraídas las características vocales, podemos optar por un modelo parametrico (o modelo gaussiano), el cual facilitaría mucho los algoritmos de calculo, pero no seria una representación cercana a la real, puesto que en general , las características de la voz no corresponden a este modelo. Razón por la cual he preferido la otra alternativa que consiste en obtener una representación aproximada de la matriz SE-X (características de la voz), denominada CODEBOOK.

4.3 Codebook de un locutor conocido

La obtención del codebook se realiza en 2 pasos:

1. Obtención de un codebook inicial y
2. Mejora del codebook inicial mediante el algoritmo de Lloyd.

4.3.1 Obtención del codebook inicial

El codebook inicial es el punto de partida para lograr un resultado satisfactorio. La creación del codebook inicial y su posterior mejora, se realiza obteniendo en primer lugar una "secuencia de entrenamiento", formada por un grupo de vectores perteneciente al espacio N-dimensional, representativos de la señal a cuantificar.

Se deduce, que el codebook estaría especialmente adaptado a la "secuencia de entrenamiento", puesto que se obtendrá a partir de ella. Para conseguir una cuantificacion eficiente de vectores no contenidos en la secuencia de entrenamiento, dicha secuencia debe ser lo mas representativa posible de la distribución de la señal de entrada. En caso contrario, el codebook se especializara solo en unos determinados vectores.

Entre los principales métodos para la obtención del codebook inicial se tienen:

1. Método aleatorio
2. Método de poda y
3. Método de Splitting.

4.3.2 Método de Splitting

A diferencia de los otros 2 métodos , el numero de codevectores en el codebook es una potencia de 2, esto es porque se trata de un sistema iterativo, y en cada iteración se dobla el tamaño del codebook obtenido en la iteración anterior.

El método puede resumirse en los siguientes pasos:

1. Se crea un codebook con un único vector (C_o) que es el centroide de toda la secuencia de entrenamiento, es decir:

$$C_o = \frac{\sum_{i=1}^{M} V_i}{M} \qquad \text{.....(4.12)}$$

Donde la secuencia de entrenamiento (*SE*), es el conjunto de M vectores:

$$SE = \{ V_1 , V_2 , V_3 , ... , V_M \} \qquad (4.13)$$

2. Luego se divide el centroide en 2 codevectores, obteniendo un codebook de 2 vectores:

$$\text{Codebook Inicial} = \{ C_o + \mathcal{E} , C_o - \mathcal{E} \} \qquad \text{.....(4.14)}$$

Donde ε es una pequeño vector perturbación, de modo tal que su componente i-esima es proporcional a la varianza de la componente i-esima del conjunto de vectores de la secuencia de entrenamiento.

$$V_1 = \{ v_{11} , v_{12} , v_{13} , ..., v_{1N} \}$$

$$V_2 = \{ v_{21} , v_{22} , v_{23} , ..., v_{2N} \}$$

$$\vdots \qquad \text{.....(4.15)}$$

$$V_M = \{ v_{M1} , v_{M2} , v_{M3} , ..., v_{MN} \}$$

$$\mathcal{E} = \{ \sigma_1 , \sigma_2 , \sigma_3 , ... , \sigma_N \}$$

Es decir:

$$\sigma_1 = varianza \{ v_{11} , v_{21} , v_{31} , ..., v_{M1} \}$$

$$\sigma_2 = varianza\ \{ v_{12}, v_{22}, v_{32}, \ldots\ldots, v_{M2} \}$$

$$\vdots \qquad \ldots\ldots\ldots\ldots\ldots\ldots (4.16)$$

$$\sigma_N = varianza\ \{ v_{1N}, v_{2N}, v_{3N}, \ldots\ldots, v_{MN} \}$$

3. Se aplica el algoritmo de mejora de codebooks.
4. Se repite el paso de dividir en 2 cada codevector y la consiguiente mejora del codebook hasta llegar al tamaño que se ha elegido en el diseño (potencia de 2).

El calculo de la perturbación (ε_i) usada en un determinado vector (C_i), se puede realizar a partir del calculo de las varianzas de las componentes de los vectores de la secuencia de entrenamiento asignados a dicho vector (C_i).

4.3.3 Mejora del codebook

A partir de un codebook inicial, independientemente del método utilizado para su obtención, este puede mejorarse siguiendo los siguientes pasos del método de Lloyd:

1. Dividir la secuencia de entrenamiento en grupos o "clusters", a partir de la condición del "vecino mas cercano". De modo que, cada uno de los codevectores del actual codebook tendrá asignado un subconjunto de vectores pertenecientes a la secuencia de entrenamiento.
2. A partir de esta "sectorización" de la secuencia de entrenamiento, se calcula el "mejor representante" de cada "cluster", es decir el centroide.

Para la implementacion practica de este método se necesitan los siguientes elementos:

- La secuencia de entrenamiento: M vectores de dimensión N.
- Un codebook a mejorar formado con K codevectores.
- Una tabla intermedia formada por K filas o clusters de (N+1) posiciones.

El procedimiento a implementar deberá ser el siguiente:

Para cada uno de los M vectores de la secuencia de entrenamiento se busca su vecino mas cercano en el codebook a mejorar y en la correspondiente posición de la tabla intermedia, inicializada previamente a cero en todas sus casillas, se suma el vector de la secuencia de entrenamiento en las correspondientes N primeras casillas de la tabla intermedia. La posición N+1 es un contador del numero de vectores asociados a la respectiva fila.

Se procede de esta forma con todos los vectores de la secuencia de entrenamiento, de manera tal, que al final en la tabla intermedia se tiene: el valor acumulado de todos los vectores asociados a un determinado cluster y el numero de vectores asignados.

Finalmente se actualiza el codebook con cada uno de los centroides y las varianzas de los clusters respectivos. La lógica del siguiente diagrama de flujo (Figura 16), ha sido implementada para cada locutor conocido.

DIAGRAMA DE FLUJO PARA OBTENER EL CODEBOOK DE 1 LOCUTOR

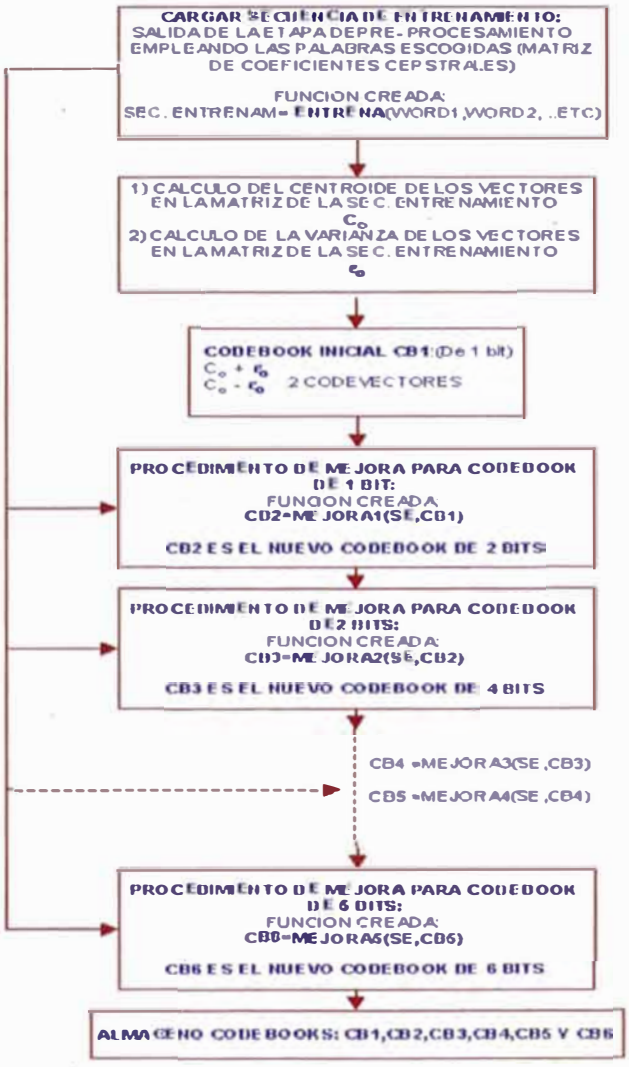


FIGURA 16

Los programas han sido grabados con los nombres: **cbookh.m** y **cbookl.m**. En un principio se trabajo hasta obtener un codebook de 6 bits, sin embargo en las pruebas finales se mejoro el codebook hasta 7 bits, con lo que se logro disminuir la distorsión media de cuantificacion, de todas las palabras de la secuencia de entrenamiento, y una mejora en la tasa de reconocimiento.

La búsqueda del vecino más cercano, se basa en la distancia euclidea entre 2 vectores definida como:

$$d(\bar{X},\bar{Y}) = \sum_{i=1}^{N} (x_i - y_i)^2 \qquad \text{.....(4.17)}$$

Donde:

$d(\bar{X},\bar{Y})$: Distancia euclídea entre vectores $\bar{X}$ e $\bar{Y}$.

x_i : Componente i-esima del vector $\bar{X}$.

y_i : Componente i-esima del vector $\bar{Y}$.

En la ecuación (4.17), se utiliza el cuadrado de la distancia, lo cual facilita el computo, sin perder precisión en la cuantificación.

Para determinar el vecino mas cercano, se ha creado la función **dist(.....)**, la cual se muestra a continuación:

```
función d=dist(A,B)
%d:Distancia Euclidea entre los vectores A y B.
D=A-B;%resta vectorial
D1=D.^2;%
xx=length(D1);
s=0;
for i=1:xx
   s=s+D1(i);
end
d=s;
```

La mejora de cada codebook se realiza siguiendo la lógica del diagrama de flujo, mostrado en la figura 17.

4.4 Identificación de locutor

Esta seria la etapa de prueba, en la cual se obtienen las características de voz de un locutor X, a partir de la pronunciación de una palabra que para nuestro caso será una de las utilizadas en la secuencia de entrenamiento , para esto la voz deberá ser procesada hasta llegar a obtener los coeficientes cepstrales cada 20ms, los que serán acomodados como vectores fila en la matriz SE-x. El siguiente paso corresponde a la cuantificación vectorial,

DIAGRAMA DE FLUJO PARA LA MEJORA DE CODEBOOKS.

Los programas MEJORA1,MEJORA2,.....MEJORA5, utilizan este proceso.

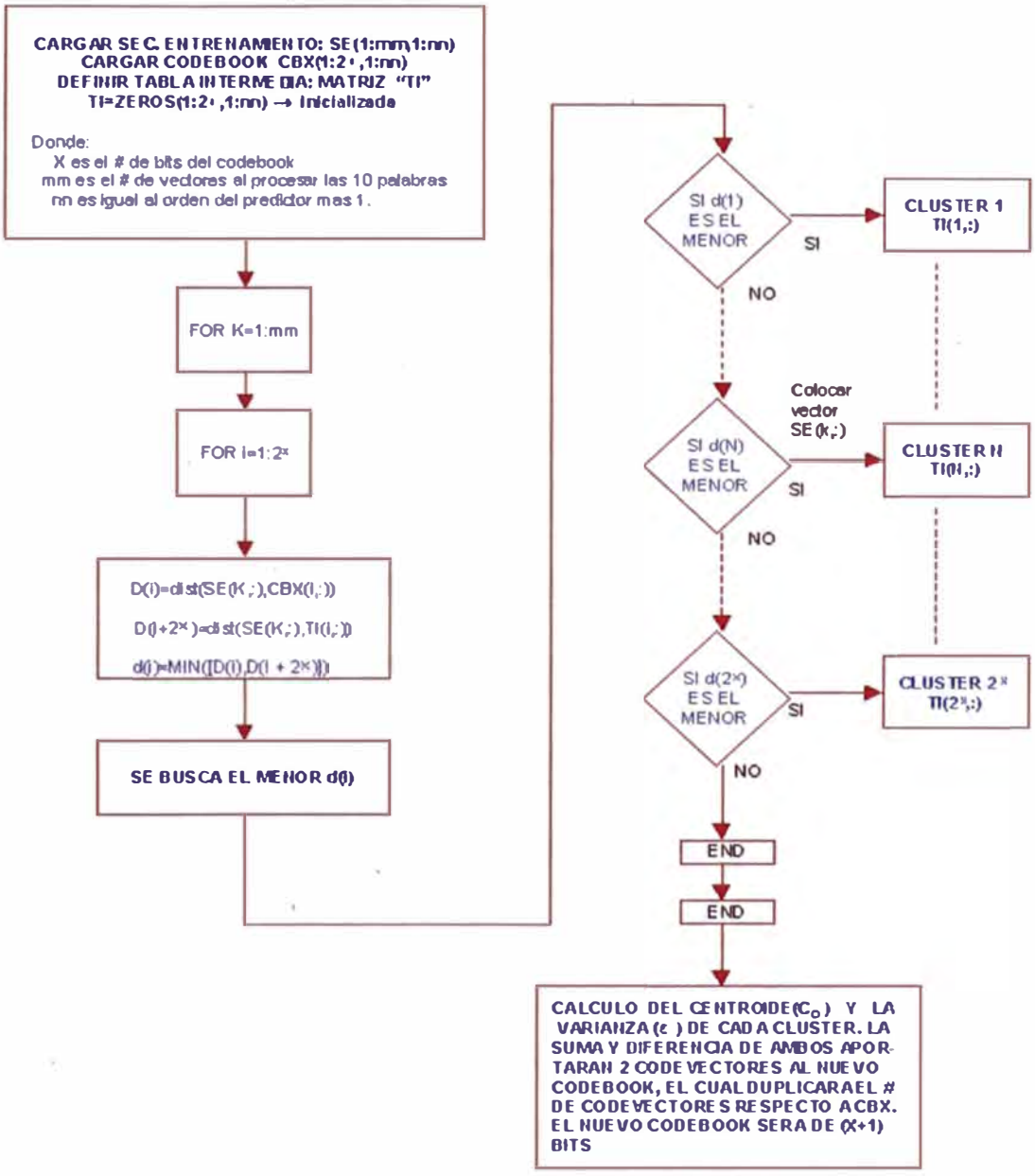


FIGURA 17

el cual consiste en ubicar al codevector mas cercano, en el codebook, respecto al vector fila de SE-x que se intenta cuantificar.

El criterio para discriminar si la matriz de características SE-x corresponde a uno u otro locutor esta basado en determinar la menor distorsión media producida al cuantificar con cada codebook la matriz SE-X. El codebook que se corresponda con la menor distorsión media definirá al locutor identificado. La distorsión media de los vectores de la secuencia de entrenamiento respecto a un codebook , es definida por:

$$Dm = \frac{1}{M} \sum_{i=1}^{M} \min [d(Xi, CB)] \qquad \text{.....(4.18)}$$

Donde:

Dm : es la distorsion media
[d(Xi,CB)] : es un vector formado por las distancias euclideas de un vector de la secuencia de entrenamiento a cada codevector del codebook.
M : es el número de vectores de la secuencia de entrenamiento.

La manera de obtener resultados mas precisos , implica realizar varias pruebas de entrenamiento con las palabras del codebook en diferentes momentos hasta lograr una representación mas distribuida de las características vocales.

Se puede lograr incluso sistemas independientes de la palabra pronunciada, si la etapa de entrenamiento considera diversas y muchas palabras que logren una distribución amplia de los codevectores en el espacio n-dimensional. Sin embargo el sistema planteado cumple el objetivo trazado al inicio del trabajo, por lo que considero además podría servir de referencia para lograr un objetivo mayor.

La función creada en MATLAB para la distorsión media, a partir del diagrama de flujo de cuantificacion mostrado es:

Dm=busca(SE-X,CB)

La Figura 18, muestra un algoritmo cuya objetivo es calcular la distorsion media, luego de una cuantificación:

Diagrama de flujo del procedimiento de busqueda en el codebook (CUANTIFICACION)

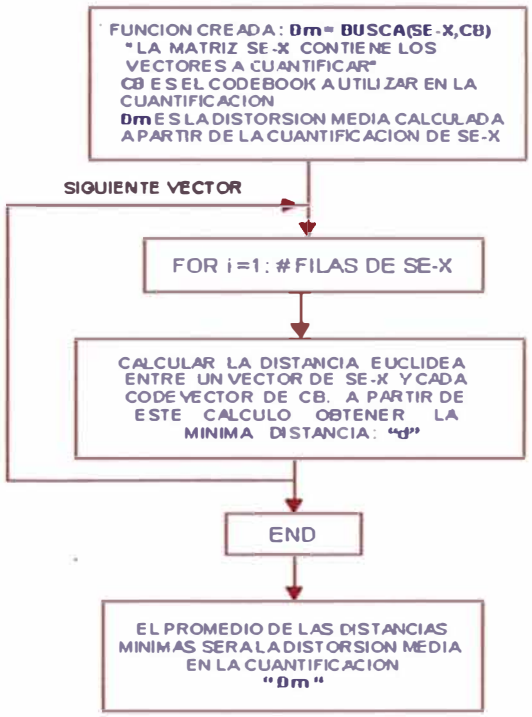


FIGURA 18

Para el sistema cerrado de identificación que se plantea, el algoritmo de identificación consistirá en la comparación de las distorsiones medias obtenidas con cada codebook, para definir al locutor (Luis ó Héctor). El siguiente diagrama de flujo (Figura 19), implementa esta etapa de decisión:

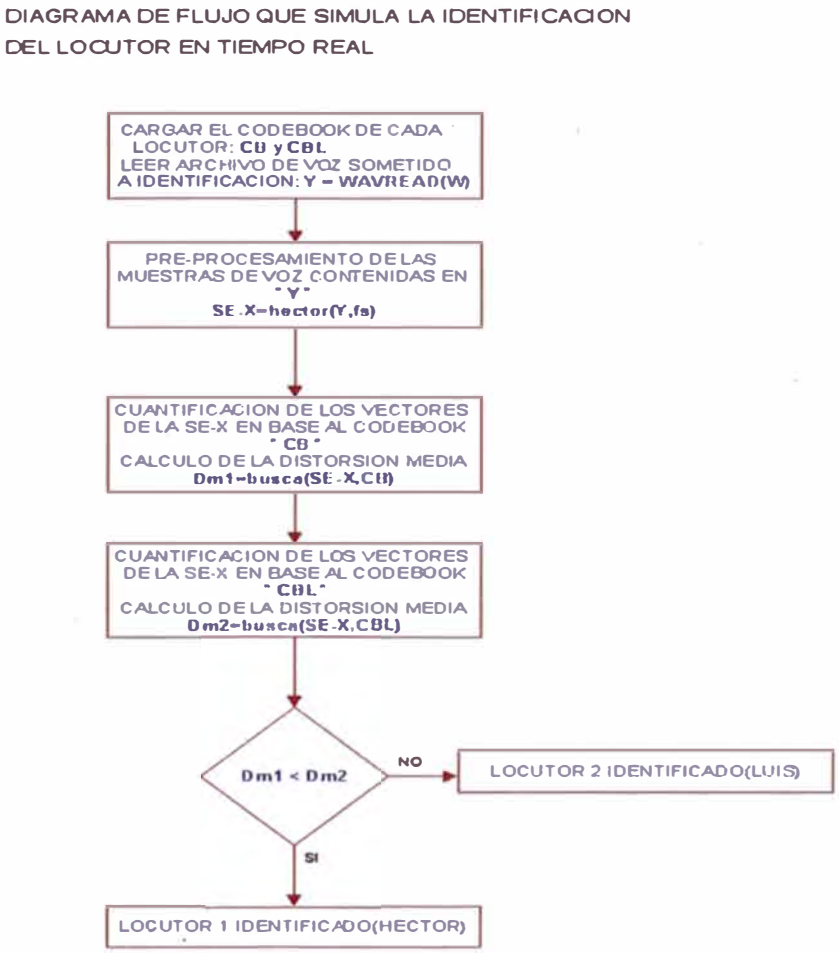


FIGURA 19

Para el caso poco probable, que las distorsiones medias sean iguales, se puede anteponer un bloque de decision, de modo tal que si se presenta este caso, la identificación sea abortada y se solicite al locutor que pronuncie otra vez alguna palabra del vocabulario.

CAPÍTULO V
RESULTADOS, OBSERVACIONES, CONCLUSIONES Y RECOMENDACIONES

5.1 Resultados de las pruebas de identificación

Las palabras utilizadas en la fase de entrenamiento fueron las siguientes:

- Palabra 1 : ENERO
- Palabra 2 : HOLA

  Palabra 3 : UNIVERSIDAD
- Palabra 4 : ABRIL

  Palabra 5 : INGENIERIA
- Palabra 6 : SESENTAIOCHO

  Palabra 7 : ELECTRONICA

  Palabra 8 : TREINTAITRES
- Palabra 9 : SESENTAISIETE
- Palabra 10: VIDEO

Las pruebas de identificación de locutor 1 (llamado Héctor) y de locutor 2 (llamado Luis) han consistido en procesar las muestras de voz originadas al pronunciar en 3 ocasiones diferentes cada una de las palabras de la secuencia de entrenamiento, obteniendo un reconocimiento del 100% para cada locutor. Los resultados se presentan a continuación:

	LOCUTOR 1		LOCUTOR 2	
PALABRAS	DISTORSION MEDIA EN CODEBOOK 1	DISTORSION MEDIA EN CODEBOOK 2	DISTORSION MEDIA EN CODEBOOK 1	DISTORSION MEDIA EN CODEBOOK 2
ENERO	0,2624	0,5313	0,3359	0,2893
HOLA	0,3206	0,5843	0,3800	0,3144
UNIVERSIDAD	0,2195	0,4529	0,5778	0,5130
ABRIL	0,2572	0,4581	0,4212	0,2804
INGENIERIA	0,2601	0,7444	0,4719	0,2522
SESENTAIOCHO	0,2208	0,4483	0,4652	0,1819
ELECTRONICA	0,2506	0,4011	0,3977	0,2469
TREINTAITRES	0,1730	0,2679	0,3706	0,1842
SESENTAISIETE	0,1859	0,3344	0,4292	0,2051
VIDEO	0,2377	0,6216	0,3682	0,2009

Tabla 1. – Resultados de la primera prueba.

	LOCUTOR 1		LOCUTOR 2	
PALABRAS	DISTORSION MEDIA EN CODEBOOK 1	DISTORSION MEDIA EN CODEBOOK 2	DISTORSION MEDIA EN CODEBOOK 1	DISTORSION MEDIA EN CODEBOOK 2
ENERO	0,3747	0,4024	0,3321	0,2233
HOLA	0,3646	0,6380	0,3910	0,3426
UNIVERSIDAD	0,2195	0,4519	0,5217	0,3812
ABRIL	0,4447	0,5276	0,4521	0,3204
INGENIERIA	0,4013	0,4541	0,4719	0,2144
SESENTAIOCHO	0,3360	0,5038	0,4573	0,2013
ELECTRONICA	0,3493	0,4839	0,4135	0,2811
TREINTAITRES	0,3422	0,4304	0,3706	0,1842
SESENTAISIETE	0,3628	0,6209	0,4531	0,2216
VIDEO	0,3078	0,6216	0,3821	0,2115

Tabla 2. – Resultados de la segunda prueba.

	LOCUTOR 1		LOCUTOR 2	
PALABRAS	DISTORSION MEDIA EN CODEBOOK 1	DISTORSION MEDIA EN CODEBOOK 2	DISTORSION MEDIA EN CODEBOOK 1	DISTORSION MEDIA EN CODEBOOK 2
ENERO	0,2960	0,4160	0,4120	0,2431
HOLA	0,3646	0,6380	0,3725	0,3204
UNIVERSIDAD	0,2847	0,4437	0,4519	0,3949
ABRIL	0,3645	0,6645	0,4521	0,2511
INGENIERIA	0,4604	0,4859	0,5320	0,2134
SESENTAIOCHO	0,2658	0,4132	0,4861	0,2111
ELECTRONICA	0,2885	0,3696	0,4185	0,2320
TREINTAITRES	0,4630	0,7086	0,3890	0,1920
SESENTAISIETE	0,3171	0,5385	0,4450	0,2150
VIDEO	0,3309	0,4407	0,3840	0,2135

Tabla 3. – Resultados de la tercera prueba.

Las tablas muestran las distorsiones medias obtenidas al cuantificar una palabra del vocabulario durante la prueba de identificación del locutor 1 ó 2. Con estos resultados se comprueba una identificación valida al 100% para un sistema cerrado de 2 locutores a identificar.

Respecto a la característica de compresión mencionada al inicio, para la propuesta técnica en el presente trabajo, se tiene la siguiente relación:

$$\text{Factor de compresión} = \frac{(\#\text{ muestras por ventana})x(\#\text{vectores de la sec. entrenamiento})}{(\text{orden del predictor LPC})x(\#\text{ de codevectores del codebook final})} \quad \text{.....(5.1)}$$

Para el caso presentado, se tienen los siguientes datos:

a) Número muestras por ventana = 161

b) Orden del predictor=20

c) Número de codevectores del codebook final = 128

d) Número de vectores en la secuencia de entrenamiento:

- Para el locutor 1 : 648 vectores(con traslape al 50%)

349 vectores(sin traslape)

- Para el locutor 2 : 624 vectores(con traslape al 50%)

317 vectores(sin traslape)

Reemplazando los valores para el factor de compresión, se obtiene:

Para el locutor 1 : Factor de compresión = 20,6910

Para el locutor 2 : Factor de compresión = 19,9363

5.1.1 Tiempos de ejecución de programas

Los tiempos de ejecución que se indican fueron obtenidos insertando la función de cronometro **etime(...)** en los programas que obtienen la secuencia de entrenamiento del codebook y en la rutina de identificación. .

5.1.1.2 Rutina de pre-procesamiento de la voz

El promedio de duración de esta rutina depende de la longitud temporal que presente la grabación de voz . Los tiempos medidos fueron:

mínimo: 4 segundos (archivo: “enero.wav”)

máximo: 10,43 segundos (archivo: “67.wav)

5.1.1.3 Rutina de obtención del codebook

- Para el locutor 1: 31,43 minutos
- Para el locutor 2: 29,67 minutos

5.1.1.4 Rutina de identificación

El proceso de identificación independientemente de locutor tomara más o menos tiempo dependiendo de la longitud de la palabra utilizada.

La rutina de cuantificacion en cada codebook tomara de 6 a 29 segundos, por lo que el tiempo para el algoritmo de identificación esta dado por el producto de el tiempo de cuantificacion de la palabra sometida a prueba y el numero de locutores.

5.2 Observaciones

1. El orden del predictor se eligió considerando la frecuencia de muestreo, y la relación (2.3), encontrándose mejoras en la cuantificacion hasta el orden igual a 20, a partir de este valor las mejoras ya no fueron significativas.
2. El proceso iterativo de mejora del codebook tiene como fin de iteración la condición de distorsión de cuantificacion, menor a cierto umbral. De otra forma se puede planear un número de bits para el codebook, y verificar una cuantificacion próxima al 100% de los vectores de la secuencia de entrenamiento para una distorsión menor o igual a la distorsión media correspondiente al codebook considerado. Este ultimo fue el criterio utilizado para definir el codebook de cada locutor.
3. En algunos casos, se tuvieron que desechar algunas pruebas que por efectos de ruido en el entorno del ambiente degeneraban los resultados.
4. En el desarrollo del trabajo se han mantenido algunos términos del habla inglesa tales como CODEBOOK y PITCH puesto que en los textos en español consultados también los utilizan sin llegar a la traducción.
5. Para obtener más vectores en la secuencia de entrenamiento, sin realizar mayores variaciones al programa “entrena.m” , se pueden grabar frases en lugar de palabras e incrementar el tiempo de grabación a procesar.

6. Las grabaciones de las palabras para el entrenamiento o las pruebas de identificación, deben contener una información de voz menor a 2 segundos, ya que los programas de pre-procesamiento consideran este tiempo como máximo, lo cual es suficiente ya que el promedio de duración de una palabra es usualmente de 0.4 a 0.6 segundos.
7. El proceso de pre-procesamiento de la voz no considera el offset superpuesto a la señal de voz, ya que la etapa de entrenamiento y de prueba se realizaron en el mismo lugar y con la misma computadora, como usualmente sucede en una aplicación de identificación de locutor. Sin embargo para la realización de un proceso de codificación para la síntesis de voz, es necesario considerar este offset para su eliminación de la señal de voz antes de la pre-acentuación.
8. Los programas que mejoran cada CODEBOOK ; mejora1.m , mejora2.m , mejora3.m , mejora4.m , mejora5.m y mejora6.m , tienen la siguiente lógica para procurar que los codevectores obtenidos se dispongan de la mejor manera en el espacio de 20 dimensiones:
   - Si un cluster tiene solo 1 vector luego de la sectorización, el centroide de este cluster será dicho vector y el vector perturbación será aquel formado por la varianza del CODEBOOK anterior, es decir se tendrán los siguientes codevectores, producto de este cluster, para el nuevo CODEBOOK:

Vector Unico $^{+}$- Varianza (Codebook anterior)

-Si un cluster tiene cero vectores, luego de la sectorización, el centroide de dicho cluster será, el centroide del siguiente cluster y el vector perturbación será la varianza del CODEBOOK anterior, es decir, los codevectores que se obtendrán, serán:

Centroide del sgte CLUSTER $^{+}_{-}$ Varianza (Codebook anterior)

9. El problema del cluster con 1 ó 0 vectores, ocurrió a partir del CODEBOOK de 6 bits.
10. En una aplicación para implementar un codificador de voz LPC, el pre-procesamiento de la voz, se centrara en determinar la presencia o ausencia del PITCH en cada "trama ventaneada", así como los coeficientes del filtro que representan al tracto vocal, y la ganancia del mismo. Esto significa que las características de la voz deben estar mas evidenciadas en los resultados que se obtengan, que en el caso estudiado, ya que estos, serán los que se envíen al receptor para la síntesis de voz.
11. -Una alternativa diferente al traslape de las ventanas es la interpolación de coeficientes LPC de tramas consecutivas.
12. -Los resultados de las pruebas de identificación, corresponden a palabras pertenecientes al vocabulario de entrenamiento, sin embargo se puede verificar que la tasa de validación correcta es alta para palabras que no pertenecen al vocabulario.

5.3 Conclusiones

1. Los tiempos de ejecución para la rutina del codebook son elevados, a pesar que el vocabulario de la secuencia de entrenamiento no es tan

grande. Esto sustentaría la razón de utilizar un hardware especializado, basado en un Procesador Digital de Señales (DSP) en la mayoría de los casos.

2. Este sistema , presenta una dependencia del numero de locutores que intervienen, puesto que la disposición del universo vectorial para cada locutor podrían estar muy próximas en varios casos y en consecuencia incrementar la tasa de falsa identificación, es decir disminuiría el porcentaje de reconocimiento.
3. No hay que perder de vista que, este trabajo no pretende ser un sistema de aplicación real, sino mas bien un sistema con fines demostrativos , sin embargo los algoritmos desarrollados tienen un planteamiento genérico que permitiría incrementar el número de locutores y entrenando los codebooks con las voces de cada locutor en varios y diferentes tiempos se podría lograr un sistema con un alto porcentaje de reconocimiento.
4. En el bloque que realiza el pre-enfasis:

$$Y[n] = x[n] - \alpha\, x[n-1]; \qquad 0.5 < \alpha < 1 \quad \text{.....(5.2)}$$

Se tiene la posibilidad de implementarlo a través de un bucle FOR, sin embargo esto conduce a una programación muy ineficiente. Por lo que se opto a utilizar una función del Matlab: FILTER(...).

En general para realizar la suma de elementos de un vector se debe buscar otra alternativa diferente al bucle FOR. Por ejemplo la suma de los elementos de un vector fila puede obtenerse utilizando el artificio de multiplicar dicho vector fila por un vector columna de unos.

5. El bloque de pre-enfasis, utilizando un filtro digital pasa alto definido en (5.2), permite atenuar la componente DC en cada segmento ventaneado, es decir esta operación minimiza el offset de la tarjeta de sonido.
6. La lógica empleada en los programas de mejora de los CODEBOOKS, arrojo buenos resultados en la obtención de los codebooks, puesto que las pruebas de identificación así lo comprobaron. Sin embargo , en otras situaciones podría ser necesario cambiar la lógica que resuelva el caso de cluster sin vectores, por ejemplo podría dividirse en 2 el CODEVECTOR del CODEBOOK anterior asociado al cluster en mención.

5.4 Recomendaciones

1. Es recomendable realizar la fase de entrenamiento en el entorno en el que se van a realizar las pruebas de identificación, y que las condiciones de "silencio" del medio sean las mismas, para evitar problemas de cuantificacion y lógicamente de reconocimiento.
2. La eficiencia de todo sistema de reconocimiento depende en gran medida de la fase de entrenamiento, por lo que recomendaría emplear en el vocabulario de la secuencia de entrenamiento, palabras de una duración mayor a 0.5 segundos, con la finalidad de capturar diversos fonemas y obtener un codebook lo mas representativo de la voz de un locutor.
3. En el caso de tener un gran numero de locutores, seria recomendable diseñar un sistema que previo a la identificación de la voz identifique la palabra, con la finalidad de clasificar a los locutores y que la búsqueda en los codebooks se reduzca solo a un grupo de ellos.

4. Si se tienen problemas durante la ejecución de los programas de obtención de CODEBOOKS(cbookh.m y cbookl.m), se deben ejecutar cada uno de estos paso a paso , de modo tal que, se empiece el cálculo hasta: [Dm2,CB2,n1]=mejora1(SE,CB1), con la finalidad de inspeccionar los valores de "n1" y verificar que ningún elemento de este vector columna sea cero, luego incrementar el calculo del programa hasta [Dm3,CB3,n2]=mejora2(SE,CB2) y de igual modo inspeccionar "n2", así sucesivamente hasta: [Dm5,CB5,n4]=mejora4(SE,CB4). Si alguno de estos vectores n1, n2, n3 y/o n4, presenta algún elemento cero, se recomienda, entrenar al sistema variando alguna palabra o frase del vocabulario, con la finalidad de obtener los vectores más representativos de las características de voz de un locutor.

ANEXO A

A.1 Programas desarrollados

Los programas o funciones implementadas utilizan algunos ficheros *.m propios del MatLab, disponibles en las librerías básicas , las cuales están contenidas en la carpeta TOOLBOX. Las rutinas *.m desarrolladas , se adjuntaran contenidas en un medio de almacenamiento, y el listado que sigue los muestra como archivos de texto los cuales conservan la presentación propia del editor de MatLab. Se recomienda leer el archivo "leer.doc", previo a cualquier ejecución de la rutina de identificación o la generacion de algun CODEBOOK.

1)idlocutor.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%                            idlocutor(w)                                      %
%                                                                              %
%Es un programa  que  discrimina  la  voz  de 2 locutores conocidos%
%este   tipo   de   sistemas   cae   dentro   de   la   clasificacion   de%
%sistemas cerrados de identificación.                                          %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%SIMULA IDENTIFICACION EN TIEMPO REAL%%%%%%%%%%

function rr=idlocutor(w)
%        w : Es la palabra pronunciadas por el locutor 'palabra'
load CBhector CB7 ;%carga del codebook del locutor 1
load CBlucho CBl7 ;%carga del codebook del locutor 2
```

```
path(path,'c:\exposicion\palabras');%Ruta de la carpeta que contiene
                                    %las grabaciones
pause on
t=2.0;%tiempo de grabacion en seg.
y=wavread(w);%lectura de la grabacion sometida a prueba
[gg,fs]=wavread('b1');%palabras de bienvenida y solicitud de
palabra.
pause(3)
wavplay(gg,fs)%sonido: "Por favor diga una palabra"
pause(5)
wavplay(y,fs)%sonido: Una de las palabras de la secuencia de
             %entrenamiento.
pause(5)
Y=y(1:t*fs);%Se limita la longitud de la secuencia  sometida a
            %prueba a 2seg.

%%%%%%%%%%Extracción de caracteristicas de voz del locutor%%%%%%%%%%

SEX=hector(Y,fs);%Matriz formada por vectores fila de
                 %caracteristicas de la voz sometida a prueba.

%%%%%%%%Busqueda del locutor a prueba en cada codebook%%%%%%%%

m=busca(SEX,CB7)%Proceso analogo a la cuantificacion vectorial.
n=busca(SEX,CBl7)%Proceso analogo a la cuantificacion vectorial.
%          "m" y "n" son las distorsiones medias respectivas.

                  %%%%%%%DECISION%%%%%%%

if m<n
   %Reconoce al locutor 1
   pause(1)
   [e,fsn]=wavread('b3');
   wavplay(e,fs)%SONIDO:"Bienvenida a Héctor"
else
   %Reconoce al locutor 2
   pause(1)
   [q,fs]=wavread('b4');
   wavplay(q,fs)%SONIDO:"Bienvenida a Luis"
end
```

2)busca.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            Dm=busca(SEX,CB)                           %
%                                                                       %
%Programa que busca el codevector mas cercano, en el codebook de un%
%locutor,  a cada vector de la matriz de caracteristicas sometida a%
%prueba. Para esto el programa determina la minima distancia%
%euclidea y finalmente proporciona la distorsion media de esta%
%cuantificación.                                                        %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Dm=busca(SEX,CB)
```

```
%SEX:Es la matriz de caracteristicas de la palabra sometida a prueba
%    (voz del locutor)
%CB:Es la matriz CODEBOOK del locutor que se esta comprobando.
%Dm:Es la distorsion media para la cuantificacion en el
%   correspondiente codebook.

t0=clock;%inicio del cronometro
[xx,yy]=size(SEX);
[a,b]=size(CB);
sum=0;
for i=1:xx
   for rr=1:a
      d(rr,i)=dist(SEX(i,:),CB(rr,:));%Es la distancia euclidea de
                                      %cada vector de la secuencia
                                      %de entrenamiento a cada
                                      %codevector.
   end
   MIN(i)=min([d(1:a,i)]);%Es la menor distancia euclidea
   sum=sum+MIN(i);%almacena la suma de las distancias minimas
end
Dm=sum/xx;%Es la distorsión media en la cuantificacion
etime(clock,t0)%fin del cronometro
```

3)dist.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
%                           d=dist(A,B)                                 %
%Función que calcula el cuadrado de la distancia euclidea entre 2.%
%vectores A y B. Se utiliza el artificio de multiplicar un vector %
%fila por una columna de unos para calcular la suma de los elemen-%
%tos del vector fila.                                                   %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function d=dist(A,B)
%d:Distancia entre los vectores A y B.
D=A-B;%resta vectorial
D1=D.^2;%
xx=length(D1);
s=0;
Y=ones(xx,1);
d=D1*Y;
```

4)hector.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
%                       SEX=hector(Y,fs)                                %
%                                                                       %
%Programa que realiza el tratamiento dicreto de la voz,para obtener%
%la  matriz  SEX (secuencias) que  contiene las caracteristicas de %
%la  voz de  un  locutor X,  representadas  por  medio  de  los  %
%coeficientes cepstrales LPCC.                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function     SEX=hector(Y,fs)

%SEX:Matriz de caracteristicas de la voz.
%Y : Contiene a las muestras de la señal de voz.
%fs: Frecuencia de muestreo de la voz.


%%%%%%%%%%%ENFASIS DE VOZ:%%%%%%%%%%%

s=filter([1,-0.98],1,Y);%voz enfatizada

%%%%%%%%%DETECCION Y ELIMINACION DE LA SALA DE GRABACION%%%%%%%%%%%

[ti,tf,si,sf]=acotar(s);%Función que elimina el silencio de la sala
%de grabación.
dvoz=tf-ti;%duracion de la voz
voz=s(si:sf);%Solamente voz.
muestras=161;
ventanas=(fix(2*(sf-si+1)/(muestras-1)))-1;%# de ventanas
%traslapadas al 50%.

%%%%%%%%SEGMENTACION DE LA SEÑAL DE VOZ - V.DE HAMMING%%%%%%%%%

ham=hamming(muestras);%Funcion de la ventana de hamming.
for k=1:ventanas,
   inicio=1+((muestras-1)*(k-1))/2;%inicio de cada trama
   trama(1:muestras,k)=voz(inicio:inicio+muestras-1).*ham;%#ventanas
traslapadas al 50%.
end

%%%%%%%%%%%CALCULO DE COEF. LPC PARA CADA VENTANA%%%%%%%%%%%%

coeflpc=lpc(trama,20);
[a,b]=size(coeflpc);

%%Seleccion de solamente los coeficientes del predictor%%
%No se considera la columna de unos en la matriz coeflpc.%
for j=1:a
   predictor(j,:)=coeflpc(j,2:b);
end

%CALCULO DE LOS COEF. CEPSTRUM (LPCC) PARA CADA VENTANA.

for i=1:ventanas
   LPCC(i,:)=cepstrum(predictor(i,:));%coef. cepstrum para cada
                                      %ventana
end
SEX=LPCC(1:ventanas,:);%matriz con los vectores de caract. de la
voz.
```

5)acotar.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%                          [ti,tf,si,sf]=acotar(Y)                             %
%                                                                              %
%Este programa elimina la  zonas  irrelevantes  antes  y  despues              %
%de la información(voz) a procesar, determinando  la  muestra  de              %
%inicio y la muestra  final (cota inf. :cota sup.) de  la señal de             %
%voz, durante 2 seg. de grabación.                                             %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ti,tf,si,sf]=acotar(Y);
%Y: Grabación de voz a procesar
%ti: tiempo de inicio de la voz
%tf: tiempo de fin de la voz
%si: muestra de inicio de voz
%sf: muestra de fin de voz.

path(path,'c:\exposicion\palabras');%ruta de grabacion del silencio.
[SS,fs,Nb]=wavread('silenciosalaconruido');%muestras representativas
                                           %del silencio de la sala
                                           %de grabacion.
t=2.0;%tiempo de grabacion considerado.
S=SS(1:t*fs);%"2 seg de silencio
Y=Y(1:t*fs);%Solo se consideran 2seg. de la grabacion de voz.
s1=max(S);%Amplitud maxima del silencio de la sala.
m=161;%# de muestras por trama/ventana.
Ttrama=m/fs;%Duracion de la trama(en seg.).
n=fix(t/Ttrama);%# de tramas contenidas en la grabacion.(sin
traslape)
iniciovoz=0;%inicio de grabación
finvoz=t;%fin de grabación
j=0;
while j<=n,
   j=j+1;%contador de muestras
   w(1:m,j)=Y((m*(j-1)+1):m*j);
      contsil1=0;
   for i=1:m
       if abs(w(i,j))<=s1,
          contsil1=contsil1+1;%contador de tramas conteniendo solo
                              %silencio y ruido.
       end
    end
    g(j)=m-contsil1;%# de muestras mayores al umbral de silencio en
                    %una trama
   if g(j)<=8,
      iniciovoz=iniciovoz+Ttrama;
   else
      x=j;
      j=n+1;
   end
end
%Calculo de posicion final de muestras de voz, la deteccion ahora es
%en el sentido inverso.
jj=n+1;
```

```
while jj>=1,
   jj=jj-1;
   w(1:m,jj)=Y((m*jj:-1:m*(jj-1)+1));
   contsil2=0;
   for ii=1:m
       if abs(w(ii,jj))<=s1,
          contsil2=contsil2+1;
       end
   end
    h(jj)=m-contsil2;%# de muestras mayores al umbral de silencio en
                     %una trama
   if h(jj)<=8,
      finvoz=finvoz-Ttrama;
   else
      y=jj;
      jj=0;
   end
end
inicio=x*m;
fin=y*m;
ti=(inicio-1)*(1/fs);
tf=(fin-1)*(1/fs);
si=inicio;
sf=fin;
```

6)cepstrum.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
%                        c=cepstrum(a)                                  %
%                                                                       %
%Es una funcion que calcula los coeficientes cepstrales a partir de%
%los coefiecientes LPC. Razon por la que se les denomina LPCC.      %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function c=cepstrum(a)
%a: Vector fila que contiene a los coeficientes LPC de un segmento %
%   ventana de voz.
%c: Vector fila que contiene a los coeficientes LPCC de un segmento
%   ventana de voz.

p=length(a);
c(1)=-a(1);
for j=2:p
   c(j)=-a(j);
   for i=1:j-1
      c(j)=c(j)-(i/j)*(c(i))*(a(j-i));
   end
end
c;
```

7)entrena.m

7)entrena.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
%        [mm,nn,SE]=entrena(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)             %
%                                                                       %
%Programa que procesa las palabras pronunciadas por un locutor%
%para que el sistema "lo conozca". Esta función procesa 10 palabras%
%para obtener una matriz denominada secuencia de entrenamiento, la%
%cual presentara las caracteristicas de voz de un locutor en cada  %
%una de sus filas.                                                      %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [mm,nn,SE]=entrena(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)

%Ai: Son las palabras a utilizar en el entrenamiento (i=1:10)
%mm: #filas de la SE(secuencia de entrenamiento) o # de vectores de
%    caracteristicas de voz.
%nn: #columnas de la SE o dimensiones de los vectores
%SE: Es la secuencia de entrenamiento para un locutor conocido

t=2.0;%tiempo max. de la grabacion en seg.
p=nargin('entrena');;%# de palabras

%%%LECTURA DE LAS GRABACIONES%%%
path(path,'c:\exposicion\palabras');
[X1,fs,Nb]=wavread(A1);
f1=fix(t*fs);
%solo se toman 2seg de grabación para cada palabra.
%Palabras:
word(1:f1,1)=X1(1:f1);
X2=wavread(A2);
word(1:f1,2)=X2(1:f1);
X3=wavread(A3);
word(1:f1,3)=X3(1:f1);
X4=wavread(A4);
word(1:f1,4)=X4(1:f1);
X5=wavread(A5);
word(1:f1,5)=X5(1:f1);
X6=wavread(A6);
word(1:f1,6)=X6(1:f1);
X7=wavread(A7);
word(1:f1,7)=X7(1:f1);
X8=wavread(A8);
word(1:f1,8)=X8(1:f1);
X9=wavread(A9);
word(1:f1,9)=X9(1:f1);
X10=wavread(A10);
word(1:f1,10)=X10(1:f1);

%%%%%%RECOLECCION DE LA SEC. DE ENTRENAMIENTO%%%%%%
r=1;
%BUCLE QUE REALIZA EL PREPROCESAMIENTO A CADA PALABRA Y LOS ACOMODA
EN %LA MATRIZ "SE"
```

```
for ii=1:p
   SEX=hector(word(:,ii),fs);
   [m,n]=size(SEX);%m=#ventanas de cada palabra; n=orden del
predictor
   A(r:m+r-1,:)=SEX;
   r=r+m;
end
SE=A;%Es la sec. de entrenamiento para todas las palabras
[mm,nn]=size(SE);
```

8)cbookh.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
%  PROGRAMA CBOOKH.M PARA OBTENER EL CODEBOOK DEL LOCUTOR 1(HECTOR)%
%                                                                       %
%Rutina que implementa  el  metodo  de  SPLITTING para calcular  el%
%codebook inicial  a partir del   centroide   y  la varianza de los%
%vectores  de la SE(secuencia de entrenamiento). A  partir  de este%
%codebook inicial se utiliza el metodo de Lloyd para la  mejora del%
%codebook, obteniendo finalmente un codebook  de 7 bits, el cual es%
%almacenado en un archivo "*.mat".                                     %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

t0=clock;%inicio del cronometro

%1.-Obtencion de la secuencia de entrenamiento "SE"
%%Palabras utilizadas:

A1='enero';A2='hola';A3='universidad';A4='abril';A5='ingenieria';
A6='68';A7='electronica';A8='33';A9='67';A10='video';

[mm,nn,SE]=entrena(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10); % funcion
creada %para procesar cada palabra y reunir los vectores resultantes
de todas %ellas en una sola matriz SE.

%2.-Obtencion del codebook inicial, utilizando el metodo de
Splitting.
%%%Calculo del centroide de la SE:
suma=0;
for jj=1:mm
   suma=suma+SE(jj,:);
end
C=suma./mm;%Centroide de la SE.
E=(std(SE)).^2;%es la varianza de la SE.

%El codebook inicial formado por 2 vectores sera:
CB1(1,1:nn)=C-E;
CB1(2,1:nn)=C+E;

%CALCULO DE LA DISTORSION MEDIA DEL CODEBOOK INICIAL
%%bucle que calcula las distancias de cada vector de la SE a los
%codevectores.
s=0;
```

```
for i=1:mm
   for j=1:2
      d(j)=dist(SE(i,:),CB1(j,:));
   end
   dd(i)=min(d);
   s=s+dd(i);
end
Dm1=s/mm%Es la distorsion media para 2 codevectores(CODEBOOK
%INICIAL)


%3.-Mejora del codebook y obtencion del segundo codebook.
[Dm2,CB2,n1]=mejora1(SE,CB1);
Dm2%Distorsion media al cuantificar con CB2
CB2%Codebook de 2 bits

%4.-Mejora del codebook y obtencion del tercer codebook.
[Dm3,CB3,n2]=mejora2(SE,CB2);
Dm3%Distorsion media al cuantificar con CB3
CB3%Codebook de 3 bits

%5.-Mejora del codebook y obtencion del cuarto codebook.
[Dm4,CB4,n3]=mejora3(SE,CB3);
Dm4%Distorsion media al cuantificar con CB4
CB4%Codebook de 4 bits

%6.-Mejora del codebook y obtencion del quinto codebook.
[Dm5,CB5,n4]=mejora4(SE,CB4);
Dm5%Distorsion media al cuantificar con CB5
CB5%Codebook de 5 bits

%7.-Mejora del codebook y obtencion del sexto codebook.
[Dm6,CB6,n5]=mejora5(SE,CB5);
Dm6%Distorsion media al cuantificar con CB6
CB6%Codebook de 6 bits

%8.-Mejora del codebook y obtencion del septimo codebook.
[Dm7,CB7,n6]=mejora6(SE,CB6);
Dm7%Distorsion media al cuantificar con CB7
CB7%Codebook de 7 bits


save('CBhector','CB7','Dm7');%Almaceno el codebook.
etime(clock,t0)%fin del cronometro

%NOTAS:
%i)Cada una de las funciones "mejorax", han sido preparadas
%  independientemente.
%ii)El codebook resultante del "locutor 1 (hector)" se almacena
%  con el nombre CBhector.mat
%iii)Las variables de salida n1,n2,n3,n4,n5,n6 proporcionan el # de
%   vectores en cada cluster,durante la mejora de cada codebook, y
%   fueron utilizadas durante la evaluacion del codebook para
%   monitorear un posible cluster con "cero" elementos.
```

9)cbookl.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
%   PROGRAMA CBOOKL.M PARA OBTENER EL CODEBOOK DEL LOCUTOR 2(LUIS)      %
%                                                                       %
%Rutina que implementa el metodo de SPLITTING para calcular el         %
%codebook inicial a partir del calculo del centroide y la              %
%varianza de los vectores de la SE(secuencia de entrenamiento).        %
%A partir de este codebook inicial se utiliza el metodo de Lloyd       %
%parala mejora del codebook, obteniendo finalmente un codebook de 7%   %
%bits,el cual es almacenado en un archivo "*.mat".                     %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

t0=clock;

%1.-Obtencion de la secuencia de entrenamiento "SE"
%%Palabras utilizadas:
A1='enerol';A2='holall';A3='universidadl';A4='abrilll';
A5='ingenierial';A6='68ll';A7='electronical';A8='33l';A9='67l';
A10='videol';
[mm,nn,SE]=entrena(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10);%funcion creada
%para procesar cada palabra y reunir los vectores resultantes de
%todas ellas en una sola matriz SE.

%2.-Obtencion del codebook inicial, utilizando el metodo de
Splitting.
%Calculo del centroide de la SE.
suma=0;
for jj=1:mm
   suma=suma+SE(jj,:);
end
C=suma./mm;%Centroide de la SE.
E=(std(SE)).^2;%es la varianza de la SE

%El codebook inicial formado por 2 vectores sera:
CBl1(1,1:nn)=C-E;
CBl1(2,1:nn)=C+E;

%CALCULO DE LA DISTORSION MEDIA DEL CODEBOOK INICIAL
%%calculo de las distancias de cada vector de la SE a los
%%%codevectores:
s=0;
for i=1:mm
   for j=1:2
      d(j)=dist(SE(i,:),CBl1(j,:));
   end
   dd(i)=min(d);
   s=s+dd(i);
end
Dml1=s/mm%Es la distorsion media para 2 codevectores(CODEBOOK
INICIAL)

%3.-Mejora del codebook y obtencion del segundo codebook.
```

```
[Dml2,CBl2,n1]=mejora1(SE,CBl1);
Dml2
CBl2

%4.-Mejora del codebook y obtencion del tercer codebook.
[Dml3,CBl3,n2]=mejora2(SE,CBl2);
Dml3
CBl3

%5.-Mejora del codebook y obtencion del cuarto codebook.
[Dml4,CBl4,n3]=mejora3(SE,CBl3);
Dml4
CBl4

%6.-Mejora del codebook y obtencion del quinto codebook.
[Dml5,CBl5,n4]=mejora4(SE,CBl4);
Dml5
CBl5

%7.-Mejora del codebook y obtencion del sexto codebook.
[Dml6,CBl6,n5]=mejora5(SE,CBl5);
Dml6
CBl6

%8.-Mejora del codebook y obtencion del septimo codebook.
[Dml7,CBl7,n6]=mejora6(SE,CBl6);
Dml7
CBl7

save('CBlucho','CBl7','Dml7');%Almaceno el codebook.
etime(clock,t0)

%NOTA:
%i)Cada una de las funciones "mejorax", han sido preparadas
%  independientemente.
%ii)El codebook resultante del "locutor 2 (luis)" se almacena con el
%  nombre CBlucho.mat
%iii)Las variables de salida n1,n2,n3,n4,n5,n6 proporcionan el # de
%  vectores  en  cada cluster,durante la mejora de cada codebook, y
%  fueron  utilizadas  durante  la  evaluacion  del  codebook  para
%  monitorear un posible cluster con "cero" elementos.
```

10)mejora1.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
%                        [Dm2,CB2,n]=mejora1(SE,CB1)                          %
%                                                                             %
%Función  que  implementa  el  algoritmo  de  mejora de Lloyd para            %
%un codebook de 1 bit y entrega un codebook de 2 bits.                        %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Dm2,CB2,n]=mejora1(SE,CB1)
%CB1: codebook de 1 bit; CB2: codebook de 2 bits.
%SE: secuencia de entrenamiento
```

```
%Dm1 y Dm2:distorsion media del primer y segundo codebook
%n:Vector que acumula el # de vectores en cada cluster

[mm,nn]=size(SE);
n=zeros(1:2);
sum=zeros(2,nn);
for k=1:mm
   for i=1:2
      d(i)=dist(SE(k,:),CB1(i,:));
   end
   if d(1)<d(2),
      n(1)=n(1)+1;
      sum(1,:)=sum(1,:)+SE(k,:);
      SE1(n(1),:)=SE(k,:);
   else
      n(2)=n(2)+1;
      sum(2,:)=sum(2,:)+SE(k,:);
      SE2(n(2),:)=SE(k,:);
   end
end
%Los centroides y las varianzas de cada cluster (SE1 y SE2) son:
Co1=(sum(1,:))./n(1);
Eo1=(std(SE1)).^2;
Co2=(sum(2,:))./n(2);
Eo2=(std(SE2)).^2;
%El codebook de 2 bits sera:
CB2(1,:)=Co1+Eo1;
CB2(2,:)=Co1-Eo1;
CB2(3,:)=Co2+Eo2;
CB2(4,:)=Co2-Eo2;
%calculo de la distorsion media
s=0;
for ii=1:mm
   for j=1:4
      d(j)=dist(SE(ii,:),CB2(j,:));
   end
   dd(ii)=min(d);
   s=s+dd(ii);
end
Dm2=s/mm;%Es la distorsion media para 4 codevectores
n;
```

11)mejora2.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
%                          [Dm3,CB3,n]=mejora2(SE,CB2)                        %
%                                                                             %
%Función que implementa el algoritmo de mejora de Lloyd para un              %
%codebook de 2 bits y entrega un codebook de 3 bits.                          %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Dm3,CB3,n]=mejora2(SE,CB2)
%CB2: codebook de 2 bits, CB3: codebook de 3 bits.
%SE: secuencia de entrenamiento
```

```
%Dm3:distorsion media
%n:Vector que acumula el # de vectores en cada cluster
[mm,nn]=size(SE);
n=zeros(4,1);
sum=zeros(4,nn);
for k=1:mm
   for i=1:4
      d(i)=dist(SE(k,:),CB2(i,:));
   end
   if d(1)==min(d(:)),
         n(1)=n(1)+1;
         sum(1,:)=sum(1,:)+SE(k,:);
         SE1(n(1),:)=SE(k,:);
   else
      if d(2)==min(d(:)),
            n(2)=n(2)+1;
            sum(2,:)=sum(2,:)+SE(k,:);
            SE2(n(2),:)=SE(k,:);
      else
         if d(3)==min(d(:)),
            n(3)=n(3)+1;
            sum(3,:)=sum(3,:)+SE(k,:);
            SE3(n(3),:)=SE(k,:);
         else
            if d(4)==min(d(:)),
                n(4)=n(4)+1;
                sum(4,:)=sum(4,:)+SE(k,:);
                SE4(n(4),:)=SE(k,:);
            end
         end
      end
   end
end

% las varianzas de cada cluster(SE1,SE2,SE3 y SE4) son:
Eo(1,:)=(std(SE1)).^2;
Eo(2,:)=(std(SE2)).^2;
Eo(3,:)=(std(SE3)).^2;
Eo(4,:)=(std(SE4)).^2;

for jj=1:4
   Co(jj,:)=(sum(jj,:))./n(jj);
   CB3(jj,:)=Co(jj,:)+Eo(jj,:);
   CB3(jj+4,:)=Co(jj,:)-Eo(jj,:);
end
%El codebook de 3 bits sera: CB3
%%Calculo de la distorsion media:
s=0;
for h=1:mm
   for j=1:8
      d(j)=dist(SE(h,:),CB3(j,:));
   end
   dd(h)=min(d);
   s=s+dd(h);
end
Dm3=s/mm;%Es la distorsion media para 8 codevectores
n;
```

12)mejora3.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
%                          [Dm4,CB4,n]=mejora3(SE,CB3)                        %
%                                                                             %
%   Función que implementa el algoritmo de mejora de Lloyd para un            %
%   codebook de 3 bits y entrega un codebook de 4 bits.                       %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Dm4,CB4,n]=mejora3(SE,CB3)
%CB3: codebook de 3 bits, CB4: codebook de 4 bits.
%SE: secuencia de entrenamiento
%Dm4:distorsion media
%n:Vector que acumula el # de vectores en cada cluster

[mm,nn]=size(SE);
n=zeros(8,1);
sum=zeros(8,nn);
for k=1:mm
   for i=1:8
      d(i)=dist(SE(k,:),CB3(i,:));
   end
   for ii=1:8
      if d(ii)==min(d(:)),
         sum(ii,:)=sum(ii,:)+SE(k,:);
         n(ii)=n(ii)+1;
         %Se agrupa en clusters segun criterio del vector mas
         %cercano.
         if ii==1,
            SE1(n(ii),:)=SE(k,:);
         end
         if ii==2,
            SE2(n(ii),:)=SE(k,:);
         end
         if ii==3,
            SE3(n(ii),:)=SE(k,:);
         end
         if ii==4,
            SE4(n(ii),:)=SE(k,:);
         end
         if ii==5,
            SE5(n(ii),:)=SE(k,:);
         end
         if ii==6,
            SE6(n(ii),:)=SE(k,:);
         end
         if ii==7,
            SE7(n(ii),:)=SE(k,:);
         end
         if ii==8,
            SE8(n(ii),:)=SE(k,:);
         end
      end
```

```
    end
end

 %Las varianzas de cada cluster(SE1,SE2,...,SE8) son:
Eo(1,:)=(std(SE1)).^2;
Eo(2,:)=(std(SE2)).^2;
Eo(3,:)=(std(SE3)).^2;
Eo(4,:)=(std(SE4)).^2;
Eo(5,:)=(std(SE5)).^2;
Eo(6,:)=(std(SE6)).^2;
Eo(7,:)=(std(SE7)).^2;
Eo(8,:)=(std(SE8)).^2;

for jj=1:8
    Co(jj,:)=(sum(jj,:))./n(jj);
    CB4(jj,:)=Co(jj,:)+Eo(jj,:);
    CB4(jj+8,:)=Co(jj,:)-Eo(jj,:);
end
%El codebook de 4 bits sera: CB4
%calculo de la distorsion media
s=0;
for h=1:mm
    for j=1:16
        d(j)=dist(SE(h,:),CB4(j,:));
    end
    dd(h)=min(d);
    s=s+dd(h);
end
Dm4=s/mm;%Es la distorsion media para 16 codevectores
n;
```

13)mejora4.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%                        [Dm5,CB5,n]=mejora4(SE,CB4)                           %
%                                                                              %
%    Función que implementa el algoritmo de mejora de Lloyd para un            %
%    codebook de 4 bits y entrega un codebook de 5 bits.                       %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Dm5,CB5,n]=mejora4(SE,CB4)
%CB4: codebook de 4 bits, CB5: codebook de 5 bits.
%SE: secuencia de entrenamiento
%Dm5:distorsion media
%n:Vector que acumula el # de vectores en cada cluster

[mm,nn]=size(SE);
n=zeros(16,1);%Contadores del # vectores en cada cluster.
sum=zeros(16,nn);
for k=1:mm
    for i=1:16
        d(i)=dist(SE(k,:),CB4(i,:));
```

```
    end
    for ii=1:16
       if d(ii)==min(d(:)),
          sum(ii,:)=sum(ii,:)+SE(k,:);
          n(ii)=n(ii)+1;
          %Se agrupa en clusters segun criterio del vector mas
          %cercano.
          if ii==1,
             SE1(n(ii),:)=SE(k,:);
          end
          if ii==2,
             SE2(n(ii),:)=SE(k,:);
          end
          if ii==3,
             SE3(n(ii),:)=SE(k,:);
          end
          if ii==4,
             SE4(n(ii),:)=SE(k,:);
          end
          if ii==5,
             SE5(n(ii),:)=SE(k,:);
          end
          if ii==6,
             SE6(n(ii),:)=SE(k,:);
          end
          if ii==7,
             SE7(n(ii),:)=SE(k,:);
          end
          if ii==8,
             SE8(n(ii),:)=SE(k,:);
          end
          if ii==9,
             SE9(n(ii),:)=SE(k,:);
          end
          if ii==10,
             SE10(n(ii),:)=SE(k,:);
          end
          if ii==11,
             SE11(n(ii),:)=SE(k,:);
          end
          if ii==12,
             SE12(n(ii),:)=SE(k,:);
          end
          if ii==13,
             SE13(n(ii),:)=SE(k,:);
          end
          if ii==14,
             SE14(n(ii),:)=SE(k,:);
          end
          if ii==15,
             SE15(n(ii),:)=SE(k,:);
          end
          if ii==16,
             SE16(n(ii),:)=SE(k,:);
          end
       end
    end
end
```

```
% Las varianzas de cada cluster(SE1,SE2,...,SE16) son:
Eo(1,:)=(std(SE1)).^2;
Eo(2,:)=(std(SE2)).^2;
Eo(3,:)=(std(SE3)).^2;
Eo(4,:)=(std(SE4)).^2;
Eo(5,:)=(std(SE5)).^2;
Eo(6,:)=(std(SE6)).^2;
Eo(7,:)=(std(SE7)).^2;
Eo(8,:)=(std(SE8)).^2;
Eo(9,:)=(std(SE9)).^2;
Eo(10,:)=(std(SE10)).^2;
Eo(11,:)=(std(SE11)).^2;
Eo(12,:)=(std(SE12)).^2;
Eo(13,:)=(std(SE13)).^2;
Eo(14,:)=(std(SE14)).^2;
Eo(15,:)=(std(SE15)).^2;
Eo(16,:)=(std(SE16)).^2;

for jj=1:16
   Co(jj,:)=(sum(jj,:))./n(jj);
   CB5(jj,:)=Co(jj,:)+Eo(jj,:);
   CB5(jj+16,:)=Co(jj,:)-Eo(jj,:);
end
%El codebook de 5 bits sera: CB5
%calculo de la distorsion media
s=0;
for h=1:mm
   for j=1:32
      d(j)=dist(SE(h,:),CB5(j,:));
   end
   dd(h)=min(d);
   s=s+dd(h);
end
Dm5=s/mm%Es la distorsion media para 32 codevectores
n;
```

14)mejora5.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                                %
%                        [Dm6,CB6,n]=mejora5(SE,CB5)                             %
%                                                                                %
%    Función que implementa el algoritmo de mejora de Lloyd para un %
%    codebook de 5 bits y entrega un codebook de 6 bits.                         %
%                                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Dm6,CB6,n]=mejora5(SE,CB5)
%CB5:codebook de 5 bits, CB6:codebook de 6 bits.
%SE: secuencia de entrenamiento
%Dm6:distorsion media
%n:Vector que acumula el # de vectores en cada cluster
```

```
[mm,nn]=size(SE);
n=zeros(32,1);%Contadores del # vectores en cada cluster.
sum=zeros(32,nn);
for k=1:mm
   for i=1:32
      d(i)=dist(SE(k,:),CB5(i,:));
   end
   for ii=1:32
      if d(ii)==min(d(:)),
         sum(ii,:)=sum(ii,:)+SE(k,:);
         n(ii)=n(ii)+1;
         %Se agrupa en clusters segun criterio del vector mas
         %cercano.
         if ii==1,
            SE1(n(ii),:)=SE(k,:);
         end
         if ii==2,
            SE2(n(ii),:)=SE(k,:);
         end
         if ii==3,
            SE3(n(ii),:)=SE(k,:);
         end
         if ii==4,
            SE4(n(ii),:)=SE(k,:);
         end
         if ii==5,
            SE5(n(ii),:)=SE(k,:);
         end
         if ii==6,
            SE6(n(ii),:)=SE(k,:);
         end
         if ii==7,
            SE7(n(ii),:)=SE(k,:);
         end
         if ii==8,
            SE8(n(ii),:)=SE(k,:);
         end
         if ii==9,
            SE9(n(ii),:)=SE(k,:);
         end
         if ii==10,
            SE10(n(ii),:)=SE(k,:);
         end
         if ii==11,
            SE11(n(ii),:)=SE(k,:);
         end
         if ii==12,
            SE12(n(ii),:)=SE(k,:);
         end
         if ii==13,
            SE13(n(ii),:)=SE(k,:);
         end
         if ii==14,
            SE14(n(ii),:)=SE(k,:);
         end
         if ii==15,
            SE15(n(ii),:)=SE(k,:);
         end
```

```
            if ii==16,
               SE16(n(ii),:)=SE(k,:);
            end
            if ii==17,
               SE17(n(ii),:)=SE(k,:);
            end
            if ii==18,
               SE18(n(ii),:)=SE(k,:);
            end
            if ii==19,
               SE19(n(ii),:)=SE(k,:);
            end
            if ii==20,
               SE20(n(ii),:)=SE(k,:);
            end
            if ii==21,
               SE21(n(ii),:)=SE(k,:);
            end
            if ii==22,
               SE22(n(ii),:)=SE(k,:);
            end
            if ii==23,
               SE23(n(ii),:)=SE(k,:);
            end
            if ii==24,
               SE24(n(ii),:)=SE(k,:);
            end
            if ii==25,
               SE25(n(ii),:)=SE(k,:);
            end
            if ii==26,
               SE26(n(ii),:)=SE(k,:);
            end
            if ii==27,
               SE27(n(ii),:)=SE(k,:);
            end
            if ii==28,
               SE28(n(ii),:)=SE(k,:);
            end
            if ii==29,
               SE29(n(ii),:)=SE(k,:);
            end
            if ii==30,
               SE30(n(ii),:)=SE(k,:);
            end
            if ii==31,
               SE31(n(ii),:)=SE(k,:);
            end
            if ii==32,
               SE32(n(ii),:)=SE(k,:);
            end
         end
      end
   end

%Las varianzas de cada cluster son:
if n(1)==1,
   Eo(1,:)=(std(CB5)).^2;
```

```
else
   Eo(1,:)=(std(SE1)).^2;
end
if n(2)==1,
   Eo(2,:)=(std(CB5)).^2;
else
   Eo(2,:)=(std(SE2)).^2;
end
if n(3)==1,
   Eo(3,:)=(std(CB5)).^2;
else
   Eo(3,:)=(std(SE3)).^2;
end
if n(4)==1,
   Eo(4,:)=(std(CB5)).^2;
else
   Eo(4,:)=(std(SE4)).^2;
end
Eo(5,:)=(std(SE5)).^2;
Eo(6,:)=(std(SE6)).^2;
if n(7)==0,
   Eo(7,:)=(std(CB5)).^2;
else
   Eo(7,:)=(std(SE7)).^2;
end
if n(8)==1;
   Eo(8,:)=(std(CB5)).^2;
else
   Eo(8,:)=(std(SE8)).^2;
end
if n(9)==0,
   Eo(9,:)=(std(CB5)).^2;
else
   Eo(9,:)=(std(SE9)).^2;
end
Eo(10,:)=(std(SE10)).^2;
Eo(11,:)=(std(SE11)).^2;
Eo(12,:)=(std(SE12)).^2;
Eo(13,:)=(std(SE13)).^2;
Eo(14,:)=(std(SE14)).^2;
Eo(15,:)=(std(SE15)).^2;
Eo(16,:)=(std(SE16)).^2;
Eo(17,:)=(std(SE17)).^2;
Eo(18,:)=(std(SE18)).^2;
Eo(19,:)=(std(SE19)).^2;
Eo(20,:)=(std(SE20)).^2;
Eo(21,:)=(std(SE21)).^2;
Eo(22,:)=(std(SE22)).^2;
Eo(23,:)=(std(SE23)).^2;
Eo(24,:)=(std(SE24)).^2;
Eo(25,:)=(std(SE25)).^2;
Eo(26,:)=(std(SE26)).^2;
Eo(27,:)=(std(SE27)).^2;
Eo(28,:)=(std(SE28)).^2;
Eo(29,:)=(std(SE29)).^2;
if n(30)==0,
   E(30,:)=(std(CB5)).^2;
else
```

```
    Eo(30,:)=(std(SE30)).^2;
end
Eo(31,:)=(std(SE31)).^2;
Eo(32,:)=(std(SE32)).^2;
sum(30,:)=sum(31,:);
n(30)=n(31);
sum(7,:)=sum(8,:);
n(7)=n(8);
sum(9,:)=sum(10,:);
n(9)=n(10);
for jj=1:32
    Co(jj,:)=(sum(jj,:))./n(jj);
    CB6(jj,:)=Co(jj,:)+Eo(jj,:);
    CB6(jj+32,:)=Co(jj,:)-Eo(jj,:);
end
%El codebook de 6 bits sera: CB6
%%calculo de la distorsion media:
s=0;
for h=1:mm
    for j=1:64
        d(j)=dist(SE(h,:),CB6(j,:));
    end
    dd(h)=min(d);
    s=s+dd(h);
end
Dm6=s/mm%Es la distorsion media para 64 codevectores
n;
%Nota:Aquellos clusters con 1 o 0 elementos son tratados
individualmente.
```

15)mejora6.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
%                          [Dm7,CB7,n]=mejora6(SE,CB6)                        %
%                                                                             %
%   Función que implementa el algoritmo de mejora de Lloyd para un           %
%   codebook de 6 bits y entrega un codebook de 7 bits.                       %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Dm7,CB7,n]=mejora6(SE,CB6)
%CB6:codebook de 6 bits, CB7:codebook de 7 bits.
%SE: secuencia de entrenamiento
%Dm7:distorsion media
%n:Vector que acumula el # de vectores en cada cluster

[mm,nn]=size(SE);
n=zeros(64,1);%Contadores del # vectores en cada cluster.
sum=zeros(64,nn);
for k=1:mm
    for i=1:64
        d(i)=dist(SE(k,:),CB6(i,:));
    end
    for ii=1:64
        if d(ii)==min(d(:)),
```

```
            sum(ii,:)=sum(ii,:)+SE(k,:);
            n(ii)=n(ii)+1;
            %Se agrupa en clusters segun criterio del vector mas
            %cercano.
            SE1((ii-1)*100+n(ii),:)=SE(k,:);%ningun cluster supera los
                                            %100 elementos
            ii=64;
         end
      end
   end
%Las varianzas de cada cluster son:
for kk=1:64
   if n(kk)==1,
   Eo(kk,:)=(std(CB6)).^2;
   else
      if n(kk)==0,
      Eo(kk,:)=(std(CB6)).^2;
      sum(kk,:)=sum(kk+1,:);
      n(kk)=n(kk+1);
      else
      Eo(kk,:)=(std(SE1((kk-1)*100+1:(kk-1)*100 +n(kk),:))).^2;
      end
   end
end
for jj=1:64
   Co(jj,:)=(sum(jj,:))./n(jj);
   CB7(jj,:)=Co(jj,:)+Eo(jj,:);
   CB7(jj+64,:)=Co(jj,:)-Eo(jj,:);
end
%El codebook de 7 bits sera: CB7
%calculo de la distorsion media:
s=0;
for h=1:mm
   for j=1:128
      d(j)=dist(SE(h,:),CB7(j,:));
   end
   dd(h)=min(d);
   s=s+dd(h);
end
Dm7=s/mm;%Es la distorsion media para 128 codevectores
n;
```

A.2 Codebooks obtenidos

1)Para el locutor 1 (Héctor)

CB7(1:64,:)

ans =

Columns 1 through 7

0.2917	0.0644	0.5206	0.1499	-0.1157	-0.0767	0.1308
-0.3642	-0.3924	0.4911	-0.0049	-0.1637	-0.0309	-0.0741
0.8749	0.0418	0.5357	0.2059	-0.1691	-0.0401	0.0924
0.4348	-0.2911	0.8653	0.0778	-0.1367	-0.1622	-0.1248
0.2174	-0.0701	0.1275	-0.1540	-0.1063	-0.0032	-0.1025
-0.5589	-0.3709	-0.0012	-0.1102	-0.2757	-0.0461	-0.0741
0.9887	-0.3134	0.0286	-0.0646	-0.0990	0.0421	-0.1579
0.8586	-0.0973	0.2784	-0.3036	-0.2363	0.1264	-0.0871
-0.2771	-0.3110	-0.0261	-0.2386	-0.2688	0.0080	-0.0447
-0.1925	-0.2305	0.0332	-0.1024	-0.4358	-0.1428	-0.0096
0.8714	0.3528	0.2146	-0.2505	-0.4860	-0.2523	-0.0581
0.6234	-0.3270	0.7402	-0.1678	-0.2009	0.0634	-0.1204
-0.3542	0.0259	0.0792	0.0347	-0.0094	-0.1293	-0.1597
-0.6935	-0.5061	-0.2350	-0.1807	-0.1812	-0.0962	-0.1295
0.2696	-0.5891	1.0208	0.2874	-0.3075	-0.2545	0.0065
0.3350	-0.6071	0.3153	-0.1258	-0.1041	-0.1630	-0.1337
0.2089	-0.2884	0.5402	0.3216	0.0387	-0.2163	0.0608
-0.3621	-0.6943	0.4245	-0.2293	-0.4086	-0.0932	-0.0393
0.8324	-0.2550	0.5504	0.2697	-0.3213	-0.0322	0.1833
0.6544	-0.7553	0.6636	0.2520	-0.2478	-0.2774	0.0151
-0.1475	-0.2150	-0.0137	0.0242	-0.1666	-0.1838	-0.0746
-0.5598	-0.3426	-0.2198	-0.2826	-0.2762	-0.1158	-0.0318

0.0626	-0.3724	1.2355	0.0245	-0.2198	0.0569	0.0171
0.8833	-0.6963	0.2863	-0.5426	-0.4279	0.0548	-0.0638
-0.0572	-0.1589	0.4553	0.1554	-0.1631	-0.1038	-0.2153
-0.4204	-0.4306	-0.1955	-0.2738	-0.2973	-0.1056	-0.0580
0.9491	0.0171	0.4832	-0.4313	-0.5023	0.1429	-0.0060
0.3610	-0.3182	0.7025	-0.1155	-0.0625	-0.0332	-0.3803
-0.4115	-0.1452	-0.1566	-0.2122	-0.1177	-0.0966	-0.1099
0.1686	-0.2694	1.0028	0.0092	-0.1387	-0.2486	-0.1960
0.0132	-0.5306	0.7337	-0.0031	-0.0952	-0.2378	0.0476
0.0565	-0.5499	0.1776	-0.3903	-0.0265	-0.0950	-0.0424
0.1683	0.0147	0.4374	0.1366	-0.0961	-0.0488	-0.0357
-0.3747	-0.4869	0.1588	-0.2031	-0.1340	-0.0429	-0.2391
0.7382	0.0309	0.4278	0.2373	-0.1572	-0.0663	0.0443
0.4490	-0.4976	0.8119	0.1377	-0.1457	-0.1859	-0.2411
-0.0295	-0.1131	0.1387	-0.1903	-0.1319	-0.0433	-0.0966
-0.6003	-0.3026	-0.0424	-0.2289	-0.2648	-0.1248	-0.2082
0.9412	-0.4322	0.4290	-0.4484	-0.2125	0.0946	-0.0476
0.7726	-0.3160	0.1387	-0.2555	-0.2133	0.0687	-0.2209
-0.3977	-0.4147	-0.0560	-0.3189	-0.2977	-0.0924	-0.1656
-0.3901	-0.4277	-0.0978	-0.2647	-0.2280	-0.1336	0.0119
0.7981	0.0534	0.3671	-0.1421	-0.2763	-0.2301	-0.1357
0.6351	-0.4143	0.4867	-0.1394	-0.1522	-0.0233	-0.2110
-0.3465	-0.0967	0.0269	-0.1788	-0.1912	-0.1586	-0.1910
-0.8904	-0.4402	-0.2129	-0.1989	-0.1606	-0.1152	-0.1063
0.1813	-0.7141	1.0315	0.0262	-0.3008	-0.3527	-0.1556
-0.0000	-0.9347	0.2891	-0.0919	-0.1674	-0.2082	-0.0612
0.2044	-0.3524	0.3957	0.2103	0.0851	-0.2756	-0.0080
-0.6218	-0.9506	0.4515	-0.2549	-0.3361	-0.1623	-0.0608
0.6301	-0.4282	0.5157	0.2678	-0.2488	-0.1435	0.0837

0.4384	-0.6008	0.6867	-0.0169	-0.2153	-0.3493	-0.1663
-0.1037	-0.1500	-0.0865	-0.1900	-0.1021	-0.1697	-0.1102
-0.6529	-0.4039	-0.2198	-0.3770	-0.2883	-0.1206	-0.0933
-0.2510	-0.5337	1.3696	0.0141	-0.1777	-0.3195	0.1005
0.8556	-0.5893	-0.1727	-0.5164	-0.3978	0.0406	-0.1430
-0.2232	-0.2413	0.3895	-0.0382	-0.1507	-0.1574	-0.3583
-0.4952	-0.4915	-0.2231	-0.2923	-0.3558	-0.1055	-0.0234
0.9215	-0.0622	0.4650	-0.6019	-0.4761	-0.0262	-0.0051
0.4283	-0.5473	0.6227	-0.1870	-0.1041	-0.0740	-0.4924
-0.4507	-0.2984	-0.2813	-0.1812	-0.0912	-0.1182	-0.2386
-0.2779	-0.7260	0.5486	-0.1842	-0.2280	-0.2546	-0.1405
-0.3321	-0.8103	0.9575	-0.1156	-0.2536	-0.2923	-0.0366
0.0842	-0.5530	-0.2280	-0.3856	-0.1206	-0.0865	-0.2698

Columns 8 through 14

-0.1360	-0.1857	-0.2477	-0.2675	-0.1211	0.0356	-0.0441
-0.1427	-0.1039	-0.2479	-0.1974	-0.0563	0.0158	0.0145
-0.0414	-0.1822	-0.3929	-0.3319	-0.0846	-0.0655	0.0606
-0.3514	-0.2417	-0.2550	-0.2857	-0.1151	0.0460	0.0547
-0.2663	-0.2863	-0.1769	-0.2045	-0.0751	0.0612	0.0373
-0.0695	-0.0698	-0.0320	-0.1024	-0.0193	0.0589	-0.0128
-0.4543	-0.4037	-0.2854	-0.0416	0.1232	0.0216	-0.0773
-0.5003	-0.4464	-0.1707	-0.1378	0.0420	0.1182	-0.0565
-0.1334	-0.0860	-0.0388	-0.1768	-0.0052	0.0633	-0.0028
-0.1568	-0.1818	-0.0650	0.0133	-0.0161	0.1923	0.0822
-0.1430	-0.1319	-0.1385	-0.1036	-0.0229	-0.0476	-0.0634
-0.3709	-0.1353	-0.1668	-0.2335	-0.0794	0.0484	0.0090
-0.1819	-0.1556	-0.1839	-0.1832	-0.0589	-0.0132	0.0378

-0.1414	-0.0245	-0.0811	-0.0522	0.0069	0.0819	0.0891
-0.1438	-0.0716	-0.3981	-0.3102	0.0177	-0.0058	-0.1185
-0.1411	-0.1575	-0.0900	-0.0912	-0.0310	-0.0075	-0.0079
-0.1954	-0.2441	-0.0461	-0.3123	-0.1151	-0.0680	-0.0293
-0.0636	-0.1632	-0.1894	-0.0651	-0.0599	-0.0285	-0.0081
-0.1735	-0.1853	-0.2621	-0.3149	-0.0371	0.0172	-0.0146
-0.2968	-0.2595	-0.2056	-0.3020	-0.1064	0.0384	-0.0279
-0.0879	-0.2369	-0.1841	-0.1892	-0.0704	0.0052	0.0434
-0.0895	-0.0365	-0.0553	-0.1420	-0.0614	0.1131	0.0557
-0.2459	-0.2865	-0.1925	-0.2390	-0.1235	-0.0728	-0.0374
-0.3599	-0.2772	-0.1017	0.0217	0.0143	0.2362	0.0776
-0.3491	-0.1795	-0.2469	-0.2678	-0.0979	-0.0011	0.0185
-0.1496	-0.0753	-0.1083	-0.0475	-0.0360	0.0462	0.0906
-0.2358	-0.2505	-0.1838	-0.3204	-0.0898	0.1054	0.0237
-0.3498	-0.1103	-0.3103	-0.2661	-0.1248	-0.0122	0.0902
-0.2301	-0.2210	-0.1161	-0.1164	-0.0677	0.0540	0.0116
-0.3639	-0.3155	-0.4023	-0.0477	-0.0508	-0.0917	-0.1044
-0.2574	-0.0318	-0.2224	-0.2872	-0.0099	0.0707	-0.0295
-0.2454	-0.2274	-0.1858	0.0174	0.0867	-0.0536	0.1118
-0.1832	-0.2266	-0.2154	-0.2937	-0.1477	0.0162	0.0075
-0.2772	-0.1787	-0.2401	-0.0805	0.0157	-0.0474	0.0540
-0.0992	-0.1481	-0.3463	-0.3055	-0.1167	-0.0054	0.0333
-0.3162	-0.1813	-0.3334	-0.3800	-0.0607	0.0231	0.0003
-0.2374	-0.2513	-0.2058	-0.1583	-0.1220	0.0603	0.0485
-0.0843	-0.0799	0.0570	-0.0220	-0.0234	-0.0460	-0.0719
-0.5516	-0.4013	-0.2005	-0.1314	-0.0022	0.0460	-0.0223
-0.4976	-0.3156	-0.1737	-0.0747	0.0250	0.0422	0.0054
-0.1436	-0.1283	-0.0212	-0.1272	-0.0196	0.0784	0.0205
-0.1770	-0.0588	-0.0015	-0.1294	-0.0114	0.0044	0.0163

-0.1180	-0.0718	-0.3287	-0.1978	0.0274	0.0435	0.0064
-0.4862	-0.2448	-0.1365	-0.2077	-0.1190	-0.0206	0.0136
-0.2456	-0.1644	-0.1484	-0.1121	-0.0746	0.0640	0.0802
-0.1199	-0.1085	-0.1008	-0.0707	-0.0174	0.0035	0.0003
-0.1735	-0.0992	-0.4415	-0.2906	0.0675	0.0673	-0.0626
-0.1051	-0.0421	-0.2560	-0.1177	0.0103	-0.0443	0.0584
-0.1986	-0.1441	-0.1282	-0.2371	-0.1413	-0.0576	0.0356
-0.1704	-0.1511	-0.1875	-0.0688	-0.0497	0.0991	0.0209
-0.0719	-0.2515	-0.3020	-0.2780	0.0089	0.0117	-0.0406
-0.3223	-0.2162	-0.1124	-0.2330	-0.1545	0.0712	-0.0310
-0.2195	-0.2894	-0.1512	-0.1119	0.0213	0.1242	0.0358
-0.0797	-0.0505	-0.0856	-0.0646	-0.0317	0.0664	0.1609
-0.1111	-0.2960	-0.3358	-0.1447	-0.0919	-0.2201	0.0177
-0.2858	-0.1739	0.0269	0.0887	0.1427	0.0840	-0.0371
-0.3419	-0.1952	-0.2397	-0.2368	-0.0771	0.0712	0.0339
-0.1389	-0.1348	-0.1546	-0.0707	0.0581	0.0867	0.0891
-0.2355	-0.3098	-0.1928	-0.3247	-0.1019	0.1839	0.0613
-0.3949	-0.1117	-0.3823	-0.1991	-0.0682	0.0121	0.0962
-0.2608	-0.1730	-0.1027	-0.0120	-0.0092	0.0088	0.1485
-0.2102	-0.0618	-0.2366	-0.1325	-0.0608	-0.0258	0.0635
-0.1646	-0.2917	-0.2549	-0.0903	-0.1102	0.0675	0.0233
-0.2309	-0.1862	-0.0011	0.0758	-0.0085	0.1240	0.0461

Columns 15 through 20

-0.0016	0.0292	-0.0152	-0.0497	0.0157	-0.0094
0.0968	0.0038	-0.0029	-0.0206	0.0201	0.0478
0.1250	-0.1100	-0.0128	0.0537	0.0396	0.0392
0.0372	0.0786	0.0710	0.0639	0.0920	0.0521

0.0225	0.0392	0.0575	0.0605	0.0674	0.0209
-0.0097	0.0067	-0.0103	-0.0341	0.0594	0.0219
0.0405	0.1318	0.1123	0.1010	0.0739	0.0190
0.0578	0.0350	0.1136	0.1391	0.1111	0.0095
0.0199	0.0215	0.0188	-0.0476	0.0301	0.0354
0.0410	-0.0096	0.0050	-0.1570	0.1113	0.0742
0.0607	0.0509	0.0380	0.1173	0.0612	-0.0092
0.0628	0.0519	0.0023	0.0314	0.0427	0.0086
0.0558	0.0205	0.0017	0.0136	0.0074	0.0148
0.0473	0.0362	-0.0162	-0.0232	0.0263	0.0179
0.1305	0.0803	-0.0040	0.0099	0.0052	0.0456
0.1018	-0.0330	0.0523	0.0539	0.0228	0.0035
-0.0611	0.0458	0.0679	-0.0218	-0.0097	0.0374
0.0108	0.1307	0.0686	0.0330	0.0707	0.0593
0.0154	0.0266	-0.0388	-0.0176	0.0160	0.0002
0.0539	0.1284	0.1038	0.0073	0.0364	0.0752
0.1033	-0.0208	-0.0553	0.0688	0.0639	0.0502
0.0025	0.0170	-0.0558	-0.0444	0.0518	0.0862
0.0876	0.0183	-0.0089	0.0310	0.0333	0.0652
-0.0212	0.0875	0.0105	-0.0030	0.0402	-0.0109
0.0632	0.0837	0.0998	0.0552	0.0613	0.0298
0.0781	0.0029	-0.0092	0.0193	0.0253	0.0117
0.0829	0.1459	0.0556	0.0593	0.0818	0.0356
0.0799	0.0754	0.0876	0.1223	0.0767	0.0425
0.0205	0.0589	0.0368	-0.0060	0.0454	0.0172
0.1159	0.1093	0.1092	0.0881	0.0483	0.0545
-0.0156	0.0773	-0.0043	-0.0349	-0.0466	-0.0043
0.0476	0.0042	0.0318	0.0744	0.0765	-0.0079
0.0496	0.0097	0.0192	0.0140	0.0092	0.0298

0.1534	0.0557	0.0517	-0.0182	0.0471	0.0920
0.1016	-0.0716	-0.0122	0.0242	0.0144	0.0147
0.0471	0.0997	0.1294	0.0757	0.0920	0.0636
0.0566	0.0389	0.0112	0.0314	0.0369	0.0455
-0.0207	0.0426	0.0627	0.0084	0.0716	0.0356
0.0929	0.1513	0.1225	0.0687	0.0576	0.0414
0.1603	0.1587	0.0523	0.0981	0.0637	-0.0381
0.0906	0.0202	0.0442	0.0010	0.0133	0.0776
0.0456	0.0706	-0.0398	-0.1128	0.0797	0.0082
0.1218	0.0749	0.0024	0.1050	0.0079	-0.0114
0.1103	0.1540	0.1090	0.0469	0.0728	0.0257
0.0985	0.0627	0.0427	0.0224	0.0174	0.0348
0.0218	0.0359	0.0584	0.0552	0.0391	0.0260
0.1536	0.1652	0.0473	0.0492	-0.0078	0.0559
0.0628	0.0832	0.0444	-0.0004	0.0307	0.0445
0.0157	0.0522	0.0416	-0.0385	0.0111	0.0053
0.0624	0.0295	0.0803	0.0697	0.0134	0.0814
0.0239	0.0459	0.0181	-0.0126	0.0031	0.0148
0.0855	0.1445	0.0951	0.0781	0.0593	0.0609
0.0156	-0.0432	0.0019	0.0518	0.1357	0.0143
0.0780	0.0089	-0.0936	-0.0586	0.0470	0.0696
0.0937	-0.0209	0.0164	0.0480	0.0259	0.0379
0.0156	0.0877	0.0064	-0.0316	0.0480	-0.0141
0.1796	0.1071	0.0893	0.0759	0.0439	0.0506
0.0953	0.0149	0.0067	-0.0317	0.0001	-0.0211
0.1257	0.1682	0.0328	0.0200	0.0625	0.0200
0.1990	0.0650	0.0958	0.1626	0.0341	0.0313
0.0461	0.0085	0.0812	0.0235	0.0490	-0.0275
0.0641	0.1332	0.0494	-0.0128	0.0225	0.0827

0.1358	0.1465	0.0306	0.0197	-0.0191	0.0456
0.0698	0.0375	-0.0305	0.0633	0.0411	-0.0230

CB7(65:128,:)

ans =

Columns 1 through 7

0.2373	0.0145	0.4009	0.1053	-0.1587	-0.0949	0.1005
-0.4370	-0.5071	0.4788	-0.0405	-0.2187	-0.0775	-0.1008
0.8290	-0.0347	0.5272	0.2033	-0.1972	-0.0528	0.0660
0.4063	-0.3291	0.7958	0.0358	-0.1661	-0.1995	-0.1379
0.1859	-0.1273	0.0813	-0.1874	-0.1530	-0.0362	-0.1504
-0.5726	-0.3946	-0.0088	-0.1412	-0.3140	-0.0662	-0.0949
0.9290	-0.4635	-0.0056	-0.0970	-0.1635	0.0101	-0.2058
0.8172	-0.1318	0.2405	-0.3413	-0.2436	0.0707	-0.1116
-0.2960	-0.3323	-0.0466	-0.2800	-0.3029	-0.0166	-0.0747
-0.7236	-0.3454	-0.2364	-0.1829	-0.4567	-0.1642	-0.0306
0.7861	0.3108	0.1921	-0.2958	-0.4976	-0.3168	-0.1030
0.5584	-0.3579	0.6789	-0.1951	-0.2719	-0.0166	-0.1689
-0.3766	0.0114	0.0622	0.0246	-0.0209	-0.1489	-0.1834
-0.7133	-0.5406	-0.2402	-0.2134	-0.1902	-0.1012	-0.1402
0.2121	-0.6263	0.9804	0.2068	-0.3585	-0.2824	-0.0442
0.1820	-0.6432	0.2081	-0.1697	-0.1746	-0.2330	-0.1635
0.1789	-0.3442	0.4545	0.2369	0.0111	-0.2936	-0.0309
-0.4148	-0.7032	0.4024	-0.2893	-0.4814	-0.1516	-0.0863
0.7737	-0.2890	0.5022	0.2168	-0.3554	-0.0495	0.1682
0.6324	-0.7715	0.6089	0.1952	-0.2767	-0.3005	-0.0259
-0.1532	-0.2680	-0.0285	-0.0170	-0.2012	-0.1997	-0.1019
-0.5723	-0.3574	-0.2351	-0.2982	-0.2869	-0.1332	-0.0635
0.0272	-0.4340	1.1866	-0.0906	-0.2433	-0.0456	-0.0328

0.8374	-0.7577	0.2315	-0.6614	-0.4660	0.0173	-0.0932
-0.1012	-0.2058	0.3966	0.1238	-0.1998	-0.1484	-0.2470
-0.4358	-0.4425	-0.2104	-0.2985	-0.3113	-0.1181	-0.0745
0.8407	-0.0111	0.3580	-0.5183	-0.5574	0.0285	-0.0499
0.3370	-0.3618	0.6198	-0.1692	-0.1097	-0.0574	-0.4004
-0.4340	-0.1595	-0.1631	-0.2278	-0.1457	-0.1247	-0.1183
-0.3625	-0.3842	0.7333	-0.0713	-0.1596	-0.2700	-0.2170
-0.0231	-0.5436	0.7287	-0.0153	-0.1208	-0.3837	0.0140
-0.0579	-0.6140	0.0281	-0.4248	-0.0740	-0.1121	-0.1718
0.0836	-0.0097	0.3839	0.1032	-0.1427	-0.0760	-0.0730
-0.4549	-0.4932	0.1426	-0.2033	-0.1344	-0.0503	-0.3073
0.7134	-0.0237	0.3895	0.2187	-0.1760	-0.0757	0.0217
0.3655	-0.5521	0.7747	0.1089	-0.2114	-0.2266	-0.2553
-0.0645	-0.1458	0.0942	-0.2310	-0.1519	-0.0652	-0.1429
-0.6165	-0.3244	-0.0614	-0.2594	-0.2868	-0.1572	-0.2155
0.9059	-0.4596	0.4063	-0.4733	-0.2485	0.0883	-0.1057
0.6884	-0.3641	0.1074	-0.2875	-0.2301	0.0524	-0.3000
-0.4112	-0.4317	-0.0714	-0.3317	-0.3022	-0.1026	-0.1740
-0.4045	-0.4316	-0.1107	-0.2728	-0.2664	-0.1339	-0.0036
0.6656	-0.1444	0.3331	-0.1928	-0.3382	-0.2513	-0.1700
0.5860	-0.4842	0.4498	-0.1724	-0.1631	-0.0470	-0.2292
-0.3806	-0.1184	0.0175	-0.2027	-0.2159	-0.1846	-0.2644
-0.8975	-0.4559	-0.2238	-0.2086	-0.1724	-0.1245	-0.1087
0.1397	-0.7407	1.0076	-0.0097	-0.3396	-0.3925	-0.2163
-0.0502	-1.0259	0.2519	-0.1127	-0.1817	-0.2440	-0.1728
0.1382	-0.3989	0.3375	0.1165	0.0028	-0.3587	-0.0388
-0.6961	-1.0335	0.3958	-0.2943	-0.3796	-0.1794	-0.0995
0.5567	-0.4927	0.4727	0.1845	-0.2729	-0.1862	0.0165
0.4240	-0.6240	0.6510	-0.0648	-0.2407	-0.3901	-0.2069

-0.1287	-0.1785	-0.1433	-0.2194	-0.1220	-0.1960	-0.1426
-0.6685	-0.4400	-0.2381	-0.3882	-0.3227	-0.1448	-0.1162
-0.3937	-0.5808	1.2764	-0.0209	-0.1897	-0.3767	0.0558
0.7573	-0.6954	-0.2591	-0.5551	-0.4725	0.0019	-0.1635
-0.2514	-0.2751	0.3666	-0.0695	-0.1613	-0.1747	-0.3770
-0.5017	-0.5097	-0.2326	-0.2939	-0.3987	-0.1179	-0.0411
0.8294	-0.1267	0.4472	-0.6893	-0.4976	-0.0949	-0.0198
0.3988	-0.5738	0.5539	-0.2125	-0.1086	-0.0918	-0.5073
-0.4585	-0.3231	-0.2959	-0.2241	-0.1299	-0.1717	-0.2877
-0.2918	-0.8100	0.5447	-0.1990	-0.2591	-0.2624	-0.1600
-0.3663	-0.9036	0.8737	-0.1200	-0.2540	-0.5160	-0.0492
0.0119	-0.6676	-0.3195	-0.4683	-0.1521	-0.0921	-0.3377

Columns 8 through 14

-0.1649	-0.2319	-0.2584	-0.2814	-0.1400	0.0099	-0.0572
-0.1598	-0.1064	-0.3018	-0.2348	-0.0956	-0.0113	-0.0164
-0.0495	-0.1955	-0.3954	-0.3338	-0.0946	-0.0765	0.0424
-0.4034	-0.3096	-0.2847	-0.3267	-0.1284	0.0038	0.0252
-0.2893	-0.3070	-0.2258	-0.2287	-0.0954	0.0452	0.0264
-0.0844	-0.0795	-0.0738	-0.1256	-0.0449	0.0300	-0.0287
-0.5000	-0.4139	-0.3050	-0.0839	0.0673	-0.0333	-0.0825
-0.5187	-0.5294	-0.2102	-0.1637	0.0136	0.0945	-0.0623
-0.1661	-0.1159	-0.0653	-0.1918	-0.0258	0.0466	-0.0178
-0.1847	-0.1976	-0.0867	-0.0063	-0.0220	0.1870	0.0798
-0.1779	-0.1525	-0.1587	-0.1304	-0.0507	-0.0497	-0.0783
-0.4048	-0.1800	-0.2013	-0.2624	-0.0990	0.0233	-0.0121
-0.1945	-0.1701	-0.2070	-0.1941	-0.0679	-0.0207	0.0316
-0.1545	-0.0320	-0.0990	-0.0536	0.0028	0.0253	0.0847

-0.1789	-0.1229	-0.4274	-0.3260	0.0052	-0.0204	-0.1359
-0.1935	-0.1733	-0.1691	-0.1148	-0.0984	-0.0827	-0.0156
-0.2213	-0.2509	-0.0652	-0.3274	-0.1258	-0.0946	-0.0487
-0.1207	-0.2456	-0.2432	-0.1020	-0.0606	-0.0397	-0.0123
-0.1947	-0.2324	-0.2893	-0.3317	-0.0574	-0.0030	-0.0282
-0.3650	-0.2898	-0.2411	-0.3209	-0.1186	0.0247	-0.0487
-0.1043	-0.2663	-0.1959	-0.2056	-0.0930	-0.0130	0.0288
-0.0996	-0.0533	-0.0776	-0.1526	-0.0753	0.1027	0.0382
-0.3423	-0.3091	-0.2906	-0.2518	-0.1588	-0.1132	-0.0517
-0.4009	-0.3314	-0.1217	-0.0105	-0.0014	0.2204	0.0605
-0.3731	-0.2130	-0.2699	-0.2971	-0.1078	-0.0216	0.0043
-0.1622	-0.0899	-0.1343	-0.0621	-0.0496	0.0281	0.0704
-0.2724	-0.3298	-0.2573	-0.3486	-0.0969	0.0882	-0.0005
-0.3826	-0.1398	-0.3536	-0.2855	-0.1462	-0.0459	0.0736
-0.2473	-0.2432	-0.1481	-0.1290	-0.0948	0.0398	-0.0108
-0.3917	-0.3313	-0.4239	-0.0672	-0.0568	-0.0969	-0.1068
-0.2695	-0.0792	-0.2641	-0.2966	-0.0148	0.0592	-0.0327
-0.3039	-0.2625	-0.2046	-0.0008	0.0242	-0.0773	0.0893
-0.2043	-0.2420	-0.2524	-0.3166	-0.1677	-0.0153	-0.0076
-0.3142	-0.1933	-0.2489	-0.1018	-0.0025	-0.0496	0.0176
-0.1078	-0.1587	-0.3529	-0.3215	-0.1234	-0.0073	0.0287
-0.3326	-0.2260	-0.3562	-0.3949	-0.0817	0.0098	-0.0152
-0.2772	-0.2780	-0.2298	-0.1892	-0.1523	0.0519	0.0357
-0.1061	-0.0881	0.0487	-0.0401	-0.0389	-0.0551	-0.0860
-0.5632	-0.4358	-0.2295	-0.1964	-0.0465	0.0348	-0.0319
-0.5209	-0.3419	-0.1868	-0.1211	-0.0082	-0.0168	-0.0050
-0.1695	-0.1657	-0.0410	-0.1374	-0.0286	0.0700	0.0149
-0.2217	-0.0668	-0.0361	-0.1342	-0.0388	-0.0406	-0.0344
-0.1491	-0.1203	-0.4158	-0.2341	0.0061	0.0201	-0.0007

-0.5159	-0.2840	-0.1587	-0.2466	-0.1540	-0.0307	0.0031
-0.2704	-0.1965	-0.1553	-0.1222	-0.0902	0.0383	0.0738
-0.1276	-0.1137	-0.1132	-0.0714	-0.0218	-0.0066	-0.0104
-0.1923	-0.1290	-0.5014	-0.3063	0.0531	0.0501	-0.0727
-0.2091	-0.0882	-0.2805	-0.1417	-0.0020	-0.0582	0.0477
-0.2631	-0.1850	-0.1855	-0.2617	-0.1522	-0.0790	0.0196
-0.2056	-0.1688	-0.2091	-0.1007	-0.0793	0.0763	0.0051
-0.1069	-0.2644	-0.3120	-0.2955	-0.0109	-0.0004	-0.0471
-0.3832	-0.2669	-0.1360	-0.2618	-0.1620	0.0647	-0.0443
-0.2936	-0.2963	-0.1591	-0.1244	0.0088	0.1155	0.0173
-0.0940	-0.0808	-0.1062	-0.0813	-0.0418	0.0235	0.1488
-0.1271	-0.3103	-0.3824	-0.1616	-0.1123	-0.2298	0.0085
-0.3430	-0.2389	0.0014	0.0710	0.1212	0.0737	-0.0511
-0.3559	-0.2082	-0.2674	-0.2662	-0.0930	0.0550	0.0094
-0.1482	-0.1514	-0.1774	-0.0843	0.0425	0.0644	0.0714
-0.2704	-0.3378	-0.2145	-0.3420	-0.1177	0.1635	0.0338
-0.4319	-0.1372	-0.4160	-0.2177	-0.0810	0.0004	0.0850
-0.2729	-0.2025	-0.1118	-0.0133	-0.0241	0.0041	0.0487
-0.2358	-0.0941	-0.2706	-0.1459	-0.0752	-0.0386	0.0498
-0.2150	-0.3046	-0.2895	-0.2697	-0.1888	0.0250	0.0184
-0.2799	-0.2479	-0.0180	0.0189	-0.0643	0.0902	0.0008

Columns 15 through 20

-0.0199	0.0223	-0.0285	-0.0557	0.0054	-0.0163
0.0896	-0.0103	-0.0297	-0.0328	0.0136	0.0156
0.1125	-0.1162	-0.0190	0.0531	0.0344	0.0364
0.0210	0.0613	0.0488	0.0539	0.0794	0.0421
0.0104	0.0232	0.0476	0.0491	0.0555	0.0117

-0.0156	-0.0114	-0.0140	-0.0495	0.0561	0.0099
0.0180	0.1120	0.0923	0.0986	0.0624	0.0103
0.0413	0.0298	0.0953	0.1315	0.1064	0.0006
0.0030	0.0097	-0.0075	-0.0640	0.0202	0.0297
0.0374	-0.0152	0.0021	-0.1609	0.1098	0.0728
0.0482	0.0213	0.0125	0.1119	0.0484	-0.0135
0.0427	0.0207	-0.0188	0.0181	0.0320	0.0017
0.0507	0.0147	-0.0089	0.0098	-0.0001	0.0081
0.0442	0.0277	-0.0267	-0.0478	0.0172	0.0151
0.1095	0.0702	-0.0215	-0.0043	-0.0056	0.0325
0.0837	-0.0441	0.0418	0.0516	0.0176	-0.0008
-0.0892	0.0336	0.0650	-0.0272	-0.0266	0.0345
-0.0002	0.1283	0.0638	0.0243	0.0665	0.0575
-0.0098	0.0184	-0.0476	-0.0278	0.0094	-0.0070
0.0337	0.1213	0.0980	-0.0021	0.0329	0.0679
0.0784	-0.0277	-0.0737	0.0552	0.0317	0.0428
-0.0093	0.0096	-0.0659	-0.0607	0.0302	0.0689
0.0814	0.0099	-0.0176	0.0221	0.0257	0.0595
-0.0324	0.0827	0.0054	-0.0077	0.0364	-0.0170
0.0506	0.0685	0.0806	0.0486	0.0565	0.0219
0.0699	-0.0075	-0.0206	0.0067	0.0200	0.0009
0.0794	0.1277	0.0471	0.0509	0.0762	0.0255
0.0705	0.0629	0.0768	0.1178	0.0682	0.0394
0.0010	0.0406	0.0298	-0.0116	0.0323	0.0051
0.1122	0.1038	0.1064	0.0843	0.0468	0.0532
-0.0308	0.0423	-0.0159	-0.0446	-0.0576	-0.0063
0.0197	-0.0060	0.0151	0.0705	0.0666	-0.0140
0.0404	-0.0003	0.0047	0.0096	-0.0040	0.0118
0.1228	0.0508	0.0443	-0.0350	0.0415	0.0659

0.0925	-0.0761	-0.0216	0.0152	0.0052	0.0111
0.0392	0.0841	0.1239	0.0718	0.0859	0.0593
0.0425	0.0117	-0.0011	0.0196	0.0217	0.0365
-0.0405	0.0258	0.0523	-0.0081	0.0691	0.0168
0.0725	0.1369	0.1195	0.0620	0.0546	0.0227
0.1287	0.1498	0.0389	0.0921	0.0600	-0.0469
0.0558	0.0149	0.0405	-0.0045	0.0017	0.0564
-0.0060	0.0674	-0.1115	-0.1190	0.0502	0.0013
0.1154	0.0705	-0.0040	0.0892	-0.0078	-0.0148
0.1025	0.1302	0.1024	0.0422	0.0687	0.0163
0.0892	0.0306	0.0317	0.0101	0.0158	0.0235
0.0166	0.0313	0.0556	0.0478	0.0353	0.0183
0.1460	0.1587	0.0192	0.0384	-0.0131	0.0488
0.0477	0.0659	0.0194	-0.0099	0.0242	0.0428
-0.0065	0.0404	0.0244	-0.0465	-0.0084	-0.0019
0.0399	0.0102	0.0659	0.0539	0.0084	0.0675
-0.0051	0.0328	0.0063	-0.0268	-0.0046	0.0080
0.0756	0.1255	0.0728	0.0659	0.0534	0.0489
0.0059	-0.0494	-0.0291	0.0348	0.1227	0.0068
0.0709	-0.0010	-0.1044	-0.0755	0.0377	0.0670
0.0789	-0.0276	0.0126	0.0425	0.0134	0.0292
-0.0026	0.0687	0.0003	-0.0387	0.0344	-0.0233
0.1545	0.0976	0.0791	0.0702	0.0364	0.0403
0.0887	-0.0041	-0.0065	-0.0384	-0.0056	-0.0359
0.1095	0.1602	0.0257	0.0095	0.0606	0.0136
0.1922	0.0562	0.0905	0.1591	0.0278	0.0277
0.0258	-0.0122	0.0776	-0.0225	0.0440	-0.0304
0.0409	0.1048	0.0448	-0.0141	0.0120	0.0748
0.1347	0.1459	0.0193	0.0190	-0.0238	0.0444

0.0501 0.0255 -0.0314 0.0584 0.0397 -0.0456

2)Para el locutor 2 (Luis)

CBl7(1:64,:)

ans =

Columns 1 through 7

0.1966	0.2389	1.5060	-0.4926	-0.4317	0.0877	-0.3222
0.0545	0.0007	0.0169	-0.2509	-0.0850	-0.3854	-0.1398
1.1431	-0.5112	0.8632	0.4940	-0.2224	-0.1588	-0.3448
1.1342	0.1712	0.5897	-0.1542	-0.3444	0.0607	-0.3070
0.3459	0.1971	0.4696	0.1202	-0.1278	-0.0633	-0.1062
-0.6369	-0.1911	-0.0912	-0.0987	-0.1329	-0.1636	-0.1843
0.5218	-0.0839	0.3114	-0.0136	0.0448	-0.1351	-0.3742
0.6187	-0.0950	0.2243	0.1722	0.1069	0.0079	-0.1505
0.1086	0.0037	0.2290	-0.0914	-0.2715	-0.2807	-0.3171
0.0833	-0.4916	-0.0545	-0.3074	-0.2120	-0.2207	-0.2698
0.6297	0.0483	0.8823	-0.0023	-0.2789	-0.2574	-0.4169
1.1880	-0.3746	0.4723	-0.3106	0.2071	0.1065	-0.4879
-0.0843	-0.1476	0.4502	-0.1556	-0.0178	-0.3099	-0.2943
-0.4296	-0.4426	-0.1963	-0.3705	-0.3027	-0.1988	-0.1881
0.2391	-0.3347	1.0458	0.0080	-0.0348	-0.2333	-0.3715
0.7424	-0.2023	0.0773	-0.3332	-0.0959	-0.1180	-0.5631
0.2485	0.1387	0.8284	-0.1896	-0.0397	-0.1057	-0.2076
-0.2361	-0.0634	0.0112	-0.0006	-0.0859	-0.0981	-0.2792
0.6270	-0.2761	1.0508	0.1539	-0.2242	-0.3217	-0.2850
1.2380	0.0860	0.1750	-0.4628	-0.3823	0.0548	-0.2498
-0.0046	-0.0380	0.3418	0.0241	-0.1381	-0.1007	-0.3210
-0.7790	-0.3825	-0.0232	-0.2951	-0.2689	-0.1349	-0.0930
0.3954	-0.2318	1.1044	-0.0322	-0.1945	-0.1239	-0.1871

0.4488	-0.2161	0.2738	-0.2093	0.0669	0.0360	-0.3884
-0.5609	0.1056	1.1997	-0.2679	0.0555	-0.0824	-0.0045
-0.3937	-0.3338	0.1965	-0.4426	-0.2468	-0.4299	-0.2806
0.7883	-0.1634	0.8769	-0.0708	-0.1880	-0.3881	-0.6517
0.8729	-0.5080	0.3992	-0.3673	-0.0287	0.1344	-0.4027
0.0914	-0.3191	0.3818	-0.5854	0.0647	-0.3690	-0.5722
0.2469	-0.2914	0.5399	-0.0804	0.0162	-0.2917	-0.5112
0.2607	-0.3760	0.5045	-0.1712	0.1413	-0.1888	-0.4149
0.5870	-0.2825	-0.2813	-0.2459	-0.2834	0.0053	-0.4397
0.0849	-0.1211	1.2423	-0.2735	-0.2243	-0.2095	-0.1834
0.0267	-0.0522	-0.1206	-0.3728	-0.2091	-0.1144	-0.2749
0.8855	-0.4588	0.9982	0.5046	-0.0973	-0.1900	-0.4804
1.0407	-0.1128	0.4004	-0.1799	-0.2111	0.1213	-0.1319
0.3597	0.0671	0.3554	-0.0131	-0.2175	-0.1651	-0.2922
-0.7106	-0.3758	-0.0726	-0.1627	-0.1765	-0.1548	-0.1533
0.4518	-0.1886	0.0423	-0.1901	0.1174	0.0902	-0.2609
0.5119	-0.2569	0.3019	-0.0546	0.1381	-0.1753	-0.4875
-0.5113	-0.4151	-0.2514	-0.2874	-0.3461	-0.1240	-0.2610
-0.3865	-0.4579	-0.0717	-0.2286	-0.3223	-0.0908	-0.3489
0.5179	0.0547	0.6266	0.0104	-0.2880	-0.2417	-0.3996
0.6314	-0.3501	0.5452	-0.1848	0.1098	-0.0469	-0.5655
-0.3369	-0.2177	0.4131	-0.2619	-0.0436	-0.2655	-0.3083
-0.6205	-0.5595	-0.1661	-0.3884	-0.3553	-0.1836	-0.1304
-0.0047	-0.3924	0.9275	-0.0887	-0.1246	-0.2853	-0.3082
0.6400	-0.1484	-0.1577	-0.2817	-0.1344	-0.1173	-0.4957
-0.0117	-0.1484	0.7698	-0.1340	-0.2448	-0.0445	-0.2778
-0.2984	-0.0908	-0.0647	-0.1380	-0.1115	-0.1351	-0.2584
0.6000	-0.5789	0.8725	0.0975	-0.1280	-0.1982	-0.4194
1.0569	-0.1506	0.0106	-0.5077	-0.3422	0.0010	-0.2459

-0.2161	-0.1093	0.3373	0.0097	-0.1599	-0.1253	-0.2821
-0.7499	-0.4525	-0.1450	-0.1974	-0.3364	-0.1606	-0.1814
0.2214	-0.4301	1.0973	-0.1366	-0.2140	-0.1848	-0.1354
0.4614	-0.2268	0.2756	-0.5362	0.1305	0.1108	-0.4347
-0.5053	0.0147	1.0574	-0.3067	0.1812	-0.2817	-0.0381
-0.4870	-0.4315	0.1048	-0.4908	-0.1247	-0.5178	-0.4236
0.6033	-0.3669	0.7690	-0.0748	-0.1365	-0.3128	-0.5898
0.7868	-0.5574	0.4368	-0.4176	-0.1505	0.0967	-0.3341
-0.3445	-0.2347	0.3452	-0.4092	-0.0066	-0.3139	-0.3174
0.0323	-0.4376	0.6825	-0.1921	-0.1392	-0.0997	-0.3080
0.1105	-0.3729	0.4691	-0.1124	-0.1515	-0.3531	-0.4674
0.4186	-0.2382	-0.1958	-0.3675	-0.2333	0.0568	-0.5225

Columns 8 through 14

-0.2254	-0.1645	-0.1634	-0.1834	-0.1416	-0.2136	0.0315
-0.2130	-0.2208	-0.1772	-0.0939	-0.0710	0.1420	0.0686
-0.2826	-0.1825	-0.2136	-0.1728	-0.0571	0.0333	-0.1964
-0.3003	-0.0258	-0.1358	-0.1665	-0.1341	-0.0546	-0.1171
-0.2414	-0.2426	-0.1605	-0.1782	-0.0938	0.0051	0.0170
-0.1767	-0.1724	-0.1204	-0.0734	-0.0348	0.0046	0.0126
-0.4664	-0.1872	-0.1597	-0.0898	-0.0861	0.0080	0.0464
-0.5591	-0.1552	-0.0550	-0.1951	-0.1386	-0.0607	0.0519
-0.2078	0.0217	0.0489	-0.0490	0.0317	0.0813	0.0697
-0.2615	-0.1235	-0.0323	0.1146	-0.2414	0.3423	0.0821
-0.1026	-0.0659	-0.2507	-0.0324	-0.0443	-0.0487	-0.0110
-0.3310	-0.2064	-0.1357	-0.0774	-0.1218	0.0312	-0.0058
-0.0393	-0.2799	-0.3102	-0.1433	-0.1003	0.1274	0.1002
-0.1254	-0.0353	-0.0398	0.0214	-0.0146	0.0944	0.0942

-0.1745	-0.2973	-0.3295	-0.0322	-0.0520	-0.0867	-0.0027
-0.4549	-0.0774	-0.1309	0.0532	0.0355	0.0140	0.0956
-0.1806	-0.4589	-0.2504	-0.1648	-0.2242	0.0026	0.0902
-0.3946	-0.3424	-0.2468	-0.0571	0.0450	0.0333	0.0433
-0.2560	-0.2002	-0.2363	-0.1710	0.0427	-0.0521	-0.0168
-0.3454	-0.2249	-0.2814	-0.1021	0.0994	0.0442	0.0229
-0.3170	-0.2335	-0.2282	-0.0894	-0.0211	-0.0211	0.0333
-0.1455	-0.1139	-0.0689	0.0011	-0.0046	0.0540	0.0731
-0.4376	-0.2474	-0.2066	-0.1814	0.0034	0.0819	-0.0577
-0.6204	-0.2671	-0.2162	-0.0998	-0.0323	0.0075	0.0483
-0.4174	-0.3334	-0.1128	-0.1771	-0.1731	-0.0166	-0.0085
-0.1906	-0.2657	0.0077	-0.1566	0.0415	0.1914	0.1540
-0.1554	-0.1502	-0.2591	-0.0348	0.0518	-0.0428	-0.0219
-0.4130	-0.4018	-0.1896	0.0807	-0.0317	0.0282	0.0910
-0.1777	-0.2744	-0.1682	-0.0682	0.0411	0.2453	0.2363
-0.2271	-0.3048	-0.3234	-0.0236	0.0573	-0.0152	0.0106
-0.2071	-0.1322	-0.3884	-0.0779	0.0171	-0.0246	0.0166
-0.5004	-0.0948	-0.0012	0.0318	0.1717	0.1012	0.0635
-0.2207	-0.3612	-0.1680	-0.0158	-0.1805	-0.0812	0.0482
-0.3842	-0.3295	-0.1662	0.0061	0.0577	0.0772	0.1361
-0.3457	-0.1207	-0.2431	-0.0968	-0.0539	0.0267	-0.1032
-0.4472	-0.1582	-0.0336	-0.2357	-0.0695	-0.0928	-0.0277
-0.2523	-0.2261	-0.2092	-0.1768	-0.0420	-0.0045	0.0450
-0.1296	-0.1363	-0.1350	-0.0752	-0.0202	0.0604	0.0099
-0.4078	-0.3462	-0.2277	-0.1511	-0.0643	0.0060	-0.0161
-0.4635	-0.1465	-0.1539	-0.1530	-0.1320	0.0221	0.0788
-0.2729	-0.0462	-0.0099	-0.0497	0.0047	0.1824	0.0118
-0.0970	-0.0655	0.0345	-0.1173	0.0052	0.1195	0.0249
-0.0929	-0.0840	-0.2896	-0.0802	-0.0195	-0.0500	0.0142

-0.3145	-0.1829	-0.2450	-0.1875	-0.1186	0.0375	0.1410
-0.0363	-0.2225	-0.3325	-0.0658	-0.0734	0.0382	0.1519
-0.1678	-0.0915	-0.0355	-0.0234	-0.0043	0.1179	0.1100
-0.1790	-0.3608	-0.3228	-0.0848	-0.0026	-0.0093	0.0392
-0.4438	-0.1792	-0.1119	0.0098	0.0693	-0.0456	0.0889
-0.1920	-0.3798	-0.1720	-0.0728	-0.1079	-0.0726	0.0523
-0.3230	-0.3138	-0.2247	-0.1215	0.0276	0.0886	0.0638
-0.5664	-0.0882	-0.3122	-0.2734	-0.0129	0.0708	-0.0807
-0.3718	-0.1996	-0.1662	-0.0229	0.0577	0.0951	0.0520
-0.3344	-0.3085	-0.2405	-0.1257	-0.0195	-0.0106	0.0307
-0.1261	-0.1267	-0.0299	-0.0295	-0.0221	0.0823	0.0648
-0.5132	-0.3159	-0.2042	-0.1561	0.0376	-0.0277	0.0330
-0.7054	-0.2718	-0.1302	-0.0734	-0.0169	0.0550	0.0323
-0.4423	-0.3779	-0.0492	-0.2738	-0.1626	0.0547	0.1146
-0.1114	-0.1676	-0.0799	-0.0840	0.0328	0.3366	0.1540
-0.1882	-0.1343	-0.2691	-0.0353	0.0354	-0.0372	-0.0282
-0.4481	-0.3722	-0.1750	0.0143	-0.0095	0.1546	0.1198
-0.1957	-0.2549	-0.1300	-0.0867	-0.0220	0.0943	0.1745
-0.3473	-0.2507	-0.3183	-0.1310	-0.0051	0.0421	0.0477
-0.4900	-0.0630	-0.2420	-0.0179	0.0741	0.0914	0.0507
-0.5984	-0.0608	-0.0407	0.0427	0.1101	0.1623	0.0822

Columns 15 through 20

-0.0057	0.0695	0.0743	0.0714	0.0475	0.0321
0.1072	0.0978	0.0704	0.0112	0.0057	0.0052
0.0035	0.0673	-0.0309	-0.0683	0.0333	-0.0250
0.0139	-0.0227	0.1012	0.0703	0.0507	-0.0102
-0.0102	0.0167	0.0078	-0.0081	-0.0055	-0.0242

0.0359	0.0432	0.0362	0.0401	0.0429	0.0383
0.1135	0.0471	0.1558	0.0420	0.0657	0.0580
-0.0219	0.1039	-0.0462	-0.0303	0.0907	0.0020
0.1312	0.0326	0.0075	0.0807	0.0481	-0.0864
0.0952	0.0840	0.0378	0.0116	-0.0083	-0.0248
-0.0240	-0.0218	0.0781	0.0490	0.0531	0.0229
-0.0309	0.0959	0.0757	0.0751	0.0782	-0.0306
0.0103	0.0691	0.0979	0.0222	-0.0071	0.0269
0.0317	0.0533	0.0324	0.0715	-0.0186	-0.0276
0.0738	0.0486	0.0867	0.0533	0.0193	0.0429
0.1984	0.1590	0.0681	-0.0226	-0.0055	-0.0171
-0.0032	0.0830	0.1170	0.0385	0.0571	0.0888
0.1152	0.1650	0.0889	0.0451	0.0488	0.0184
0.0547	0.0571	0.0786	0.0348	0.0273	0.0096
0.1498	0.1117	0.1489	0.0866	0.0599	0.0056
0.0875	0.0852	0.1149	0.0267	0.0168	0.0684
0.0785	0.0778	0.0152	0.0447	-0.0097	-0.0464
0.0429	0.0751	0.0680	0.0637	0.0525	0.0268
0.1626	0.1026	0.0960	0.1256	0.0663	0.0383
-0.0402	0.0295	0.0201	-0.0290	-0.0050	0.0273
0.2542	0.0253	0.0750	-0.0039	-0.0910	-0.0539
0.0098	0.0477	0.1243	0.0176	0.0507	0.1042
0.0523	0.0640	0.1440	0.0934	0.0386	0.0203
0.0601	0.1198	0.1217	0.0141	-0.0517	-0.0433
0.1071	0.1284	0.1867	0.0525	0.0014	0.0364
0.0702	0.0672	0.0790	0.0721	0.0002	0.1098
0.2158	0.0806	0.0471	-0.0093	-0.0643	-0.0734
-0.0010	0.0380	0.1132	0.0645	0.0610	0.0601
0.0894	0.0934	0.1488	0.0053	-0.0117	-0.0059

0.0035	0.1134	0.0152	0.0266	0.0304	-0.0110
-0.0053	0.0035	0.0290	-0.0009	0.0844	-0.0059
0.0277	0.0676	0.1197	0.0714	0.0631	-0.0124
0.0473	0.0672	0.0160	0.0274	0.0520	0.0327
0.1236	0.1227	0.0958	0.1212	0.1500	0.0060
0.0723	0.1943	0.0419	0.0664	0.0883	0.0499
0.0414	0.1876	0.0691	-0.1427	0.1006	-0.1096
0.0735	0.0688	0.1124	-0.1921	-0.0117	0.0258
-0.0011	0.0107	0.1092	0.0440	0.0455	0.0188
0.0442	0.0634	0.1216	0.0871	0.0539	0.0270
0.0310	0.0679	0.1523	0.0646	-0.0304	0.0386
0.0970	0.0465	0.0118	0.0008	0.0060	-0.0323
0.1091	0.0793	0.1052	0.0566	0.0486	0.0540
0.1888	0.1266	0.0910	0.0180	0.0168	-0.0496
0.0348	0.0188	0.0980	0.0556	0.0369	0.0218
0.1744	0.0801	0.0517	0.0973	0.0507	-0.0113
0.1514	0.1510	0.0859	0.0907	0.0790	0.0389
0.1006	0.0929	0.0619	0.0865	0.0371	-0.0191
0.1032	0.1090	0.1025	0.0630	0.0478	0.0048
0.1264	0.0483	0.0104	0.0428	0.0031	-0.0249
0.1206	0.0810	0.1033	0.1061	0.0663	0.0030
0.1654	0.1161	0.1217	0.1160	0.0104	0.0302
0.0646	0.0394	0.0651	0.0646	0.0337	0.0447
0.1205	0.0603	0.0607	-0.0426	-0.0519	-0.1048
0.0722	0.0602	0.1400	0.0454	0.0339	0.0970
0.0296	0.0701	0.1408	0.0570	0.0569	0.0617
0.1024	0.0936	0.0862	0.0168	-0.0147	-0.0239
0.0634	0.1139	0.1375	0.0439	0.0341	0.0686
0.2028	0.1048	0.0945	0.1310	-0.0416	-0.0190

0.1993 0.0835 -0.0069 0.0437 -0.0414 -0.0862

CBI7(65:128,:)

ans =

Columns 1 through 7

0.1449	0.2165	1.4022	-0.5032	-0.4879	0.0525	-0.3252
0.0211	-0.1193	-0.0420	-0.2893	-0.1134	-0.3970	-0.2988
1.1339	-0.5143	0.8442	0.4727	-0.2263	-0.1633	-0.3472
1.1281	-0.0878	0.5530	-0.1713	-0.3709	0.0105	-0.3555
0.2791	0.1519	0.4312	0.0616	-0.1631	-0.0984	-0.1572
-0.6452	-0.1959	-0.0955	-0.1123	-0.1364	-0.1692	-0.1889
0.4731	-0.1118	0.1688	-0.0272	0.0235	-0.1390	-0.3797
0.5158	-0.2137	0.1447	0.1213	0.0680	-0.0290	-0.2326
-0.4423	-0.0585	-0.1324	-0.1588	-0.3033	-0.3127	-0.3498
-0.4676	-0.5538	-0.4159	-0.3748	-0.2438	-0.2527	-0.3026
0.5541	0.0026	0.8233	-0.0348	-0.3109	-0.2735	-0.4321
1.0965	-0.4226	0.3969	-0.4349	0.1447	0.0740	-0.5354
-0.1525	-0.1870	0.4235	-0.1853	-0.0385	-0.3405	-0.3520
-0.4764	-0.4771	-0.2234	-0.3779	-0.3079	-0.2092	-0.2177
0.1575	-0.3736	0.9650	-0.0393	-0.0623	-0.2957	-0.3846
0.7234	-0.2787	0.0230	-0.3838	-0.1117	-0.1497	-0.6299
0.1949	0.0947	0.7959	-0.2553	-0.1091	-0.1587	-0.2446
-0.2650	-0.0907	-0.0101	-0.0146	-0.1064	-0.1231	-0.3104
0.6006	-0.3181	1.0318	0.1122	-0.2594	-0.3466	-0.3624
1.1621	-0.0797	0.1257	-0.5957	-0.4267	-0.1069	-0.3191
-0.0506	-0.0665	0.2924	-0.0183	-0.1736	-0.1394	-0.3758
-0.7948	-0.4120	-0.0684	-0.3101	-0.3218	-0.1515	-0.1225
0.3592	-0.2704	1.0793	-0.0989	-0.2770	-0.2312	-0.2496
0.3697	-0.2917	0.2272	-0.2389	0.0265	0.0018	-0.4367

-0.6411	-0.0449	1.1762	-0.3394	0.0188	-0.0915	-0.0573
-0.4666	-0.3914	0.1878	-0.4485	-0.2590	-0.4437	-0.2964
0.7389	-0.1979	0.7834	-0.1021	-0.2043	-0.4144	-0.6782
0.7969	-0.5841	0.3173	-0.4557	-0.1271	0.0792	-0.4744
0.0019	-0.3663	0.3704	-0.6069	0.0167	-0.4139	-0.6081
0.2338	-0.3823	0.5150	-0.1109	-0.1082	-0.3064	-0.5412
0.2541	-0.4721	0.5021	-0.1998	0.1056	-0.1984	-0.4469
0.5070	-0.3047	-0.3129	-0.3653	-0.3202	-0.0300	-0.6032
0.0050	-0.1443	1.2242	-0.3211	-0.3134	-0.2548	-0.2327
0.0073	-0.0598	-0.1590	-0.4188	-0.2292	-0.1341	-0.2910
0.8698	-0.5071	0.7375	0.4164	-0.1259	-0.2186	-0.4971
1.0124	-0.2449	0.3094	-0.1918	-0.2669	0.0678	-0.1504
0.3212	0.0371	0.3055	-0.0472	-0.2488	-0.1924	-0.3150
-0.7264	-0.3837	-0.1027	-0.1713	-0.2040	-0.1832	-0.1680
0.4338	-0.2613	0.0194	-0.2032	0.0361	-0.0022	-0.3096
0.4652	-0.2961	0.2682	-0.0772	0.0840	-0.2665	-0.5003
-0.5811	-0.4214	-0.2523	-0.3176	-0.3471	-0.1271	-0.2796
-0.3871	-0.4671	-0.1094	-0.2431	-0.3600	-0.0938	-0.3610
0.4554	0.0382	0.5585	-0.0176	-0.3154	-0.2624	-0.4282
0.6086	-0.3915	0.4924	-0.2285	0.0533	-0.1032	-0.5784
-0.3723	-0.3287	0.4076	-0.3151	-0.0752	-0.3033	-0.3359
-0.6386	-0.5729	-0.1845	-0.4129	-0.3746	-0.1953	-0.1434
-0.0468	-0.4389	0.8817	-0.1505	-0.1552	-0.3233	-0.3258
0.5502	-0.2029	-0.1843	-0.3018	-0.1592	-0.1293	-0.5188
-0.0792	-0.1705	0.7284	-0.1511	-0.2801	-0.0782	-0.3180
-0.3468	-0.1066	-0.0805	-0.1513	-0.1176	-0.1816	-0.2932
0.5292	-0.7008	0.6817	0.0891	-0.1508	-0.2059	-0.4289
0.9356	-0.2412	-0.0491	-0.5857	-0.3797	-0.0554	-0.3011
-0.2670	-0.1372	0.3014	-0.0060	-0.1913	-0.1429	-0.3148

-0.7696	-0.4688	-0.1628	-0.2017	-0.3665	-0.1717	-0.1925
0.1497	-0.4783	1.0526	-0.1673	-0.2672	-0.2222	-0.2059
0.4324	-0.2488	0.2197	-0.6132	0.0874	0.0916	-0.4648
-0.7575	-0.0356	1.0015	-0.3629	0.1746	-0.5550	-0.0873
-0.5624	-0.4450	0.0477	-0.5027	-0.1328	-0.5464	-0.4527
0.5163	-0.4270	0.7021	-0.1469	-0.1843	-0.3770	-0.6086
0.6531	-0.5857	0.3964	-0.6313	-0.2230	-0.0526	-0.3600
-0.3882	-0.2594	0.3312	-0.4174	-0.0545	-0.3800	-0.3805
0.0271	-0.4400	0.6364	-0.1972	-0.1610	-0.1381	-0.3223
0.0797	-0.3884	0.4248	-0.1215	-0.1653	-0.4000	-0.5676
0.3734	-0.2515	-0.2428	-0.5169	-0.2558	-0.0073	-0.6365

Columns 8 through 14

-0.2275	-0.1683	-0.1688	-0.1889	-0.1437	-0.2167	0.0179
-0.2133	-0.2279	-0.1888	-0.1010	-0.0714	0.0760	0.0683
-0.2841	-0.1862	-0.2180	-0.1802	-0.0647	0.0121	-0.2029
-0.3107	-0.0636	-0.1733	-0.1984	-0.1509	-0.0781	-0.1185
-0.2822	-0.2633	-0.1909	-0.2056	-0.1015	-0.0094	0.0081
-0.1824	-0.1762	-0.1230	-0.0766	-0.0403	-0.0058	0.0054
-0.5615	-0.2074	-0.2031	-0.1112	-0.1063	-0.0214	0.0460
-0.5989	-0.2180	-0.1147	-0.2035	-0.1474	-0.0687	0.0494
-0.2473	0.0057	0.0337	-0.0571	0.0214	0.0709	0.0634
-0.3009	-0.1395	-0.0476	0.1065	-0.2517	0.3320	0.0758
-0.1101	-0.1101	-0.2789	-0.0464	-0.0558	-0.0935	-0.0288
-0.4106	-0.2413	-0.1598	-0.1071	-0.1451	-0.0263	-0.0185
-0.0587	-0.3200	-0.3402	-0.1487	-0.1167	0.1018	0.0903
-0.1450	-0.0489	-0.0590	0.0032	-0.0352	0.0640	0.0842
-0.2021	-0.3203	-0.3507	-0.0528	-0.0595	-0.1030	-0.0211

-0.5334	-0.1221	-0.1602	0.0279	0.0249	-0.0120	0.0752
-0.2004	-0.4829	-0.2692	-0.1754	-0.2474	-0.0178	0.0759
-0.4259	-0.3788	-0.2756	-0.0635	0.0381	0.0313	0.0387
-0.3326	-0.2262	-0.2683	-0.2053	0.0289	-0.0780	-0.0228
-0.4683	-0.2507	-0.3209	-0.1196	0.0571	0.0181	0.0110
-0.3588	-0.2623	-0.2591	-0.1385	-0.0449	-0.0475	0.0185
-0.1550	-0.1364	-0.0807	-0.0090	-0.0160	0.0458	0.0664
-0.4875	-0.2647	-0.2364	-0.2430	-0.0063	0.0164	-0.0660
-0.6472	-0.3030	-0.2487	-0.1237	-0.0409	-0.0026	0.0325
-0.4750	-0.3457	-0.1247	-0.2138	-0.1811	-0.0307	-0.0279
-0.2483	-0.2973	-0.0049	-0.1683	0.0177	0.1904	0.1457
-0.1634	-0.1865	-0.2761	-0.0587	0.0382	-0.0514	-0.0352
-0.4528	-0.4554	-0.2265	0.0644	-0.0432	-0.0030	0.0816
-0.1969	-0.2811	-0.1880	-0.0900	0.0305	0.2259	0.2199
-0.2463	-0.3591	-0.3551	-0.0851	0.0357	-0.0285	0.0038
-0.2102	-0.1496	-0.4356	-0.1581	0.0027	-0.0526	0.0126
-0.6249	-0.1350	-0.0337	0.0079	0.1295	0.0949	0.0571
-0.2363	-0.4355	-0.1829	-0.0460	-0.2200	-0.1205	0.0292
-0.3859	-0.3553	-0.1801	0.0029	0.0565	0.0651	0.1240
-0.3588	-0.1567	-0.2732	-0.2231	-0.0541	0.0050	-0.1054
-0.4668	-0.1663	-0.0767	-0.2728	-0.0924	-0.1058	-0.0461
-0.2737	-0.2479	-0.2277	-0.1939	-0.0639	-0.0351	0.0184
-0.1392	-0.1368	-0.1434	-0.0846	-0.0323	0.0333	0.0017
-0.4542	-0.4578	-0.2407	-0.1633	-0.1117	-0.0142	-0.0199
-0.4818	-0.1744	-0.1600	-0.1760	-0.1403	0.0139	0.0628
-0.2737	-0.0468	-0.0121	-0.0499	0.0031	0.1794	0.0105
-0.1093	-0.0840	0.0316	-0.1441	-0.0114	0.1122	-0.0071
-0.1410	-0.1081	-0.3145	-0.0931	-0.0344	-0.0598	-0.0069
-0.3923	-0.2048	-0.2603	-0.2008	-0.1359	0.0233	0.1282

-0.0839	-0.3041	-0.3472	-0.1218	-0.0937	0.0141	0.1197
-0.1882	-0.1086	-0.0483	-0.0370	-0.0150	0.1039	0.0989
-0.2063	-0.3749	-0.3529	-0.0975	-0.0115	-0.0397	0.0171
-0.4764	-0.1928	-0.1504	-0.0260	0.0577	-0.0495	0.0645
-0.2262	-0.4300	-0.1951	-0.0861	-0.1304	-0.0977	0.0487
-0.3420	-0.3774	-0.2311	-0.1263	0.0168	0.0701	0.0572
-0.5750	-0.1206	-0.3167	-0.2781	-0.0157	0.0626	-0.0834
-0.4392	-0.2378	-0.2240	-0.0479	0.0318	0.0750	0.0228
-0.3534	-0.3632	-0.2660	-0.1518	-0.0322	-0.0195	0.0234
-0.1375	-0.1509	-0.0400	-0.0339	-0.0259	0.0735	0.0518
-0.5649	-0.3311	-0.2204	-0.1727	0.0255	-0.0501	0.0263
-0.7146	-0.2951	-0.1482	-0.0862	-0.0291	0.0458	0.0213
-0.5038	-0.3870	-0.0640	-0.2780	-0.1841	0.0452	0.0734
-0.1231	-0.1809	-0.0914	-0.0984	0.0212	0.3180	0.1457
-0.2079	-0.1545	-0.2873	-0.0905	0.0148	-0.0545	-0.0481
-0.4526	-0.4614	-0.2659	0.0021	-0.0194	0.1185	0.0927
-0.2619	-0.3057	-0.1744	-0.1091	-0.0534	0.0664	0.1336
-0.3681	-0.2874	-0.3355	-0.1803	-0.0914	0.0159	0.0170
-0.5102	-0.2177	-0.2783	-0.0258	0.0591	0.0800	0.0411
-0.6311	-0.0889	-0.0484	0.0196	0.0995	0.1242	0.0529

Columns 15 through 20

-0.0070	0.0456	0.0699	0.0608	0.0459	0.0178
0.0522	0.0855	0.0704	-0.0115	0.0055	-0.0008
0.0019	0.0640	-0.0338	-0.0709	0.0302	-0.0255
0.0003	-0.0306	0.0495	0.0691	0.0488	-0.0114
-0.0260	-0.0054	-0.0033	-0.0210	-0.0160	-0.0441
0.0271	0.0386	0.0328	0.0366	0.0410	0.0377

0.1110	0.0405	0.1535	0.0398	0.0358	0.0468
-0.0293	0.0925	-0.0582	-0.0426	0.0774	-0.0035
0.1252	0.0305	0.0051	0.0781	0.0458	-0.0893
0.0892	0.0819	0.0354	0.0090	-0.0106	-0.0277
-0.0391	-0.0353	0.0742	0.0481	0.0496	0.0067
-0.0641	0.0763	0.0670	0.0668	0.0722	-0.0358
0.0055	0.0511	0.0926	0.0163	-0.0212	0.0219
0.0277	0.0346	0.0182	0.0613	-0.0366	-0.0403
0.0620	0.0355	0.0768	0.0448	0.0101	0.0374
0.1887	0.1434	0.0575	-0.0264	-0.0192	-0.0215
-0.0119	0.0551	0.1114	0.0280	0.0533	0.0818
0.1020	0.1567	0.0759	0.0327	0.0389	0.0171
0.0382	0.0451	0.0741	0.0233	0.0226	-0.0014
0.1356	0.0871	0.1324	0.0754	0.0458	0.0009
0.0665	0.0764	0.1039	0.0203	0.0056	0.0599
0.0743	0.0638	0.0083	0.0395	-0.0127	-0.0545
0.0311	0.0538	0.0529	0.0497	0.0502	0.0247
0.1489	0.0900	0.0898	0.1151	0.0596	0.0313
-0.0524	0.0128	0.0136	-0.0364	-0.0202	0.0235
0.2412	0.0056	0.0746	-0.0162	-0.0941	-0.0599
0.0015	0.0340	0.1155	0.0097	0.0442	0.0947
0.0428	0.0361	0.1390	0.0840	0.0328	0.0127
0.0503	0.1066	0.1123	0.0127	-0.0715	-0.0545
0.1017	0.1228	0.1762	0.0418	-0.0077	0.0311
0.0552	0.0659	0.0574	0.0712	-0.0078	0.1051
0.1750	0.0705	0.0408	-0.0179	-0.0675	-0.0806
-0.0190	0.0238	0.1082	0.0576	0.0578	0.0569
0.0808	0.0899	0.1408	0.0018	-0.0270	-0.0143
-0.0329	0.1130	0.0067	0.0256	0.0271	-0.0399

-0.0284	-0.0034	0.0171	-0.0041	0.0638	-0.0117
0.0186	0.0564	0.1021	0.0602	0.0497	-0.0208
0.0435	0.0665	0.0110	0.0270	0.0474	0.0124
0.1111	0.1142	0.0890	0.1076	0.1273	-0.0010
0.0673	0.1840	0.0394	0.0522	0.0776	0.0431
0.0409	0.1796	0.0665	-0.1526	0.0961	-0.1206
0.0649	0.0688	0.1087	-0.1921	-0.0122	0.0090
-0.0128	0.0017	0.1014	0.0315	0.0380	0.0115
0.0341	0.0501	0.1163	0.0817	0.0495	0.0206
0.0282	0.0552	0.1310	0.0598	-0.0320	0.0303
0.0776	0.0332	0.0039	-0.0066	-0.0033	-0.0420
0.0970	0.0689	0.0911	0.0461	0.0450	0.0504
0.1743	0.1080	0.0724	0.0098	0.0072	-0.0557
0.0270	0.0103	0.0944	0.0496	0.0297	0.0105
0.1695	0.0776	0.0484	0.0945	0.0433	-0.0160
0.1120	0.1496	0.0833	0.0899	0.0671	0.0360
0.0855	0.0764	0.0519	0.0750	0.0270	-0.0248
0.0897	0.0998	0.0896	0.0557	0.0399	0.0011
0.1158	0.0241	0.0003	0.0358	-0.0050	-0.0379
0.1038	0.0741	0.0939	0.0996	0.0612	-0.0018
0.1571	0.1137	0.1088	0.1104	0.0015	0.0238
0.0488	0.0211	0.0541	0.0545	0.0278	0.0340
0.0660	0.0576	0.0474	-0.0560	-0.0675	-0.1090
0.0534	0.0529	0.1245	0.0368	0.0292	0.0919
0.0101	0.0697	0.1351	0.0463	0.0529	0.0586
0.0880	0.0736	0.0756	0.0084	-0.0230	-0.0322
0.0570	0.0982	0.1352	0.0369	0.0166	0.0683
0.1977	0.0976	0.0867	0.1254	-0.0555	-0.0289
0.1775	0.0725	-0.0169	0.0393	-0.0432	-0.0891

A.3 Pautas para ejecutar los programas

Los programas desarrollados, junto con los codebooks obtenidos y algunas grabaciones de voz, que pérmiten probar el sistema implementado, estan contenidos en un diskette que se adjunta al presente trabajo. Ademas se incluye un archivo de texto : leeme.doc ; con las instrucciones a seguir para operar este sistema, el cual se muestra a continuación:

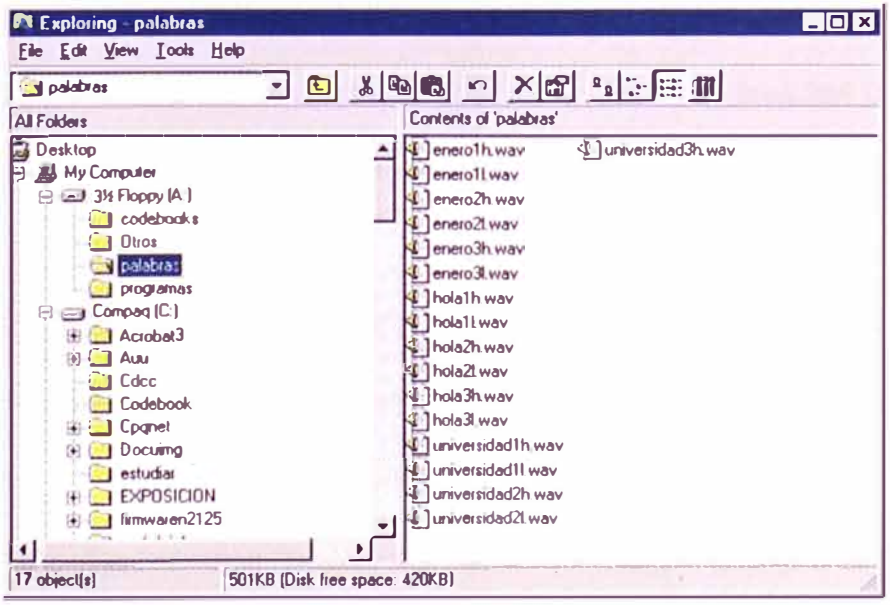
Archivo instructivo: Leeme.doc

a)Identificacion de locutor:

1) En la linea de comandos de MATLAB introducir lo siguiente, antes de ejecutar el programa de identificación:

```
path(path,'a:\programas');
path(path,'a:\palabras');
path(path,'a:\codebooks');
path(path,a:\otros');
```

2) Las palabras utilizadas para las pruebas de identificación han sido grabadas en la carpeta "palabras". Algunas de ellas se muestran a continuación:



Los archivos de voz "*.wav", presentan en la última letra del nombre una l (para el locutor LUIS) o una h (para el locutor HECTOR), con la finalidad de corroborar la identificación del sistema, en alguna prueba donde los locutores no esten presentes. Por ejemplo: Podemos ingresar alguna de las grabaciones de voz mostradas a la funcion de identificación, de la siguiente manera:

idlocutor('universidad2l')

El sistema respondera con el nombre de locutor identificado.

b)Entrenamiento del sistema:

El entrenamiento del sistema se puede realizar con 10 palabras o frases, las cuales son ingresadas al sistema, a partir de grabaciones (*.wav),de 2 segundos como mínimo, las cuales deberan presentar un contenido de voz menor o igual a 2 segundos. En caso contrario, se debera incrementar este tiempo, el cual esta definido en cada una de las funciones de pre-procesamiento y puede ser identificado dentro de los programas por un comentario adjunto. Los programas cbookl.m ó cbookh.m generaran independientemente el CODEBOOK de cada locutor. En cada uno de estos programas las palabras o frases del vocabulario, son ingresadas utilizando el nombre del archivo WAV correspondiente, en cada una de las variables A1, A2, ... y A10.

Si quisiéramos incrementar el número de locutores, se incrementara la cantidad de CODEBOOKS . Estos CODEBOOKS se pueden obtener de similar forma , para lo cual se recomienda un método ordenado, el cual comenzaría por hacer una copia de uno de los archivos cbookh.m ó

cbookl.m y generar un nuevo archivo cbookx.m, en el cual se deberá hacer el cambio de nombre a las variables que almacenan el CODEBOOK y su distorsión media, para evitar la perdida de algún CODEBOOK ya calculado previamente.

[Dmx7,CBx7,n6]=mejora6(SE,CB6);
Save(´CBnuevo locutor', CBx7,Dmx7);

En las recomendaciones, dadas en el Capitulo V del informe se indica como resolver algunos problemas que se pudieran presentar en esta etapa de entrenamiento. Para utilizar este sistema con otros locutores, se debe grabar un archivo que represente el silencio de la sala de grabación y almacenarlo en la carpeta **"A:\otros"** con el nombre:

"silenciosalaconruido.wav"

Finalmente, la función "idlocutor ( )" deberá ser ampliada con la "carga" del CODEBOOK del nuevo locutor: **load CBnuevo locutor CBx7** , una línea adicional de búsqueda en el nuevo CODEBOOK : **p=busca(SEX,nuevo CODEBOOK)** , y un bloque condicional que averigüe si "p" (la nueva distorsión media de cuantificación) es menor que las otras distorsiones medias ("m" ó "n") para poder identificar al nuevo locutor . Este bloque de decisión puede ser planteado de diversas maneras, a continuación se muestra una forma genérica de realizarlo:

```
m=busca(SEX,CB7);%Proceso analogo a la cuantificacion
%vectorial.
n=busca(SEX,CBl7);%Proceso analogo a la cuantificacion
%vectorial.
p=busca(SEX,CBx7);%Proceso analogo a la cuantificacion
%vectorial.
DM=[m n p]; %"m" , "n"     "p" ,.... son las distorsiones
%medias
```

```
%%%%%%%DECISION%%%%%%%
if m==min(DM)
   %Reconoce al locutor 1
   pause(1)
   [e,fsn]=wavread('b3');
   wavplay(e,fs)%SONIDO:"Bienvenida a Héctor"
end
if n==min(DM)
   %Reconoce al locutor 2
   pause(1)
   [q,fs]=wavread('b4');
   wavplay(q,fs)%SONIDO:"Bienvenida a Luis"
end
if p==min(DM)
   %Reconoce al locutor 3
   pause(1)
   [r,fs]=wavread('Bienvenida al locutor 3');
   wavplay(r,fs)%SONIDO
end
```

ANEXO B

Sistemas comerciales de reconocimiento de voz

Actualmente se tiene un buen numero de productos que se comercializan y que en los últimos años han mejorado sus prestaciones y reducido su precio de forma notable.

Si bien es cierto que los productos de reconocimiento del habla son bastante estándar, para el caso de la verificación del locutor depende de la cantidad de locutores que deba procesar, y su implementacion practica dependerá del caso particular que se este resolviendo, por esto, no es posible fijar un costo del sistema y prestaciones , sin embargo a manera de referencia se puede considerar un precio de 10 dólares por año y usuario. La siguiente tabla muestra algunos productos de sistemas de reconocimiento.

Tabla 1.- Sistemas de reconocimiento comerciales.

PRODUCTO	FABRICANTE	APLICACIÓN	CARACTERÍSTICAS	HTTP://
Viavoice Millenium Pro	IBM	Reconocimiento del habla continua	>100 palabras/min, 100.000 palabras activas, conv. Texto-voz	www-4.ibm.com/software/speech/es/
Viavoice Millenium Web	IBM	Reconocimiento del habla continua	>100 palabras/min, 100.000 palabras activas, conv. Texto-voz	www-4.ibm.com/software/speech/es/
Viavoice Millenium Standard	IBM	Reconocimiento del habla continua	>100 palabras/min, 100.000 palabras activas, conv. Texto-voz	www-4.ibm.com/software/speech/es/
Naturally Speaking Professional	Dragon Systems	Reconocimiento del habla continua	Hasta 160 palabras/min, 50.000 palabras activas, precisión > 95%, conv. Texto-voz	www.dragonsys.com/international/sp/productos/index.html
Naturally Speaking Mobile	Dragon Systems	Reconocimiento del habla continua	Hasta 160 palabras/min, 50.000 palabras activas, precisión > 95%, conv. Texto-voz	www.dragonsys.com/international/sp/productos/index.html
Naturally Speaking Preferred	Dragon Systems	Reconocimiento del habla continua	Hasta 160 palabras/min, 50.000 palabras activas, precisión > 95%, conv. Texto-voz	www.dragonsys.com/international/sp/productos/index.htm
Voice Xpress	Lernout&Hauspie	Reconocimiento del habla continua	Hasta 140 palabras/min., 98% precisión.	www.lhsl.com/voicexpress/pro/
Voice Guardian	Keyware technologies	Verificación de locutor		www.keyware.com
SpeakerKey	ITT industries	Verificación de locutor		www.speakerkey.com
Verivoice	Verivoice inc.	Verificación de locutor		www.verivoice.com
SpeakEZ Voice Print	T-NETIX	Verificación de locutor		www.t-netix.c

A continuacion se presenta un articulo recogido de internet (www.servitel.es) que ilustra las bondades de un producto comercial

HAL 2000

Con HAL 2000 en tu ordenador personal, tu puedes ¡por fin! acceder a tu propio « hogar del futuro »... y controlar este con tu propia voz !

Prepárese para una nueva dimensión en informática personal. Nosotros no hablaremos sobre otro juego o alguna imaginativa aplicación multimedia. Esta historia trata sobre como poner su PC a trabajar en todo lo que rodea su hogar.

¡Es parte de nuestras vidas!

No hay ninguna duda que la tecnología continuará integrando nuestros estilos de vida modernos. En el pasado, electricidad, teléfono, televisión, radio y muchos otros artículos electrónicos encontraron su manera de entrar en nuestras casas y tuvieron un impacto dramático en la calidad de vida. En los primeros momentos de estas tecnologías, la mayoría de las personas creyó que estos productos sólo estaban creados para unos "pocos privilegiados". ¡Obviamente, ellos estaban equivocados!

El tiempo ha demostrado que la tecnología es asequible cuando se comienza a producir en masa. La tecnología no es ya un privilegio. Eso es exactamente lo que está pasando en el mundo de hoy en la informática personal. El crecimiento de la “Informática en el hogar " oscurece cualquier otra penetración de tecnología en el pasado. Los niños aprenden a usar los ordenadores en la escuela, mientras muchos adultos lo utilizan como una herramienta habitual en sus trabajos. La red Internet le permite a las personas comunicarse y descubrir nuevos mundos de información y servicios con un “pulse el botón”. Hoy, los ordenadores personales cuestan menos de una quinta parte de lo que costaban hace diez años. Al mismo tiempo, el aumento en potencia y capacidades han sido tremendos. La normalización y su comercialización masiva han hecho de PC´s y montones de aplicaciones útiles, algo disponible para la inmensa mayoría. ¡Más de una cuarta parte de las casas europeas ya poseen un ordenador personal, y el número aumenta vertiginosamente.

HAL2000 representa un verdadero hito tecnológico y lleva la “Informática en el hogar" con paso de gigante, un poquito más lejos, su vocación actual es ser una herramienta para el entretenimiento y la productividad. HAL2000 ofrece una plataforma que posibilita una verdadera integración electrónica en la casa, para beneficio de nuestra conveniencia personal.

Descubra HAL2000…

Este producto de software asombroso le permite integrar una amplia gama de dispositivos eléctricos y electrónicos en la casa. Le permite

programar estos dispositivos, fijar su funcionamiento, y dejar que ellos interactuen recíprocamente entre sí. Por ejemplo, usted se ha ido por la tarde y le gustaría simular una presencia para persuadir a "visitantes no deseados" y mantenerlos alejados de su residencia.

Usted puede programar HAL2000 simplemente para encender luces, radios o cualquier otro aparato eléctrico mientras está lejos.

Bien, podría hacer esto con cualquier dispositivo cronometrado barato que puede comprar en cualquier tienda de electrónica o centro comercial. Pero tales dispositivos no pueden hacer que sus aparatos interactuen entre sí. Con HAL2000, usted puede tener un sensor inalámbrico de detección de movimiento y puede hacer que su PC de respuesta a ese movimiento detectado. Puede enviar un mensaje a un "beeper" o teléfono móvil, encender la TELEVISION en un canal específico (su videoportero), hacer sonar una sirena, encender todas las luces en el jardín, y así sucesivamente.

"Esto cambiará la idea que tengas acerca de los ordenadores. Esto también podría cambiar tu forma de vida. ¡No te pierdas la próxima revolución electrónica!"

HAL2000 también ofrece servicios de telefonía totalmente integrados. Recibe voz, facsímil (Fax) y mensajes electrónicos (e-mail) y puede notificarlos en su teléfono móvil. También transmite toda clase de información de la red Internet, el tiempo, la programación de televisión, noticias... y puede advertirlo en su "beeper" si sus acciones en bolsa han adquirido un nivel prefijado.

HAL2000 es verdaderamente excepcional e innovador. Por primera vez, la Automatización de la casa (domótica) y la convergencia electrónica está disponible en plataforma Windows. Y excede con mucho la funcionalidad de otros sistemas tradicionales… a un fragmento de su costo.

Con HAL2000, su casa "del futuro" está aquí hoy.

Una de las características más impresionante de HAL2000 es que usted puede hacer uso de su propia voz para operar el sistema.

HAL2000 confía en la tecnología de reconocimiento de voz innovadora de Lernout & Hauspie (Marca Registrada de productos de software, algunos de ellos orientados al reconocimiento automático de voz. Incluyen un micrófono con cancelación de ruidos). ¡Usted puede usar micrófonos al aire libre, un teléfono inalámbrico, o incluso puede llamar desde cualquier parte del mundo para operar todos los dispositivos de su casa!, No necesita entrenar el sistema o aprender órdenes complicadas. HAL2000 usa frases naturales, simplemente como «Atenúa la luz del salón en un 65 por ciento», «Pon el termostato a 21º grados», «Todos los Días de la semana a las 5 PM, encienda la luz de la entrada durante dos horas», o «Hoy a las 8 PM, graba en vídeo "Las Noticias"». Es tan simple como, eso.

HAL2000 es una solución muy económica y escalable que integra todos los dispositivos eléctricos y electrónicos en su hogar. Y no necesita re-cablear su casa. HAL2000 confía en la tecnología X-10, una tecnología probada y normalizada, que controla dispositivos eléctricos haciendo uso de los cables de corriente. Controla su televisión y otros equipos de audio/vídeo a través de las emisiones de infrarrojos usuales. También puede integrarlo con

sistemas de seguridad ya existentes o controladores de automatización avanzados para los que HAL2000 también ofrece compatibilidad. Y éste software, continuamente va añadiendo nuevas compatibilidades, así que el sistema siempre continuará evolucionando con el fin de agregar nueva funcionalidad. En cierto sentido, usted puede ver HAL2000 como «el pegamento» entre la electrónica de su casa y los dispositivos eléctricos. Usted puede agregar o quitar dispositivos todas las veces que quiera. Y si no le gusta interferir con su armario eléctrico principal o caja de fusibles, puede pedir a su electricista que haga el trabajo por usted. Es fácil de instalar y ampliar, así podrá manejar el presupuesto de automatización de su hogar perfectamente.

¿Qué necesita?

Echemos una mirada más íntima a cómo puede empezar y puede modernizar su casa con HAL2000.

En primer lugar, necesita un PC. Ya puede empezar con el que tiene. Sin embargo, encontrará que una vez empieza haciendo uso de los rasgos más avanzados de HAL2000 como él reconocimiento de voz, seguridad, el

control térmico de su hogar, preferirá ejecutar HAL2000 en un PC dedicado (no querrá que los niños desordenen el sistema). ¡Cualquier buen PC con un Pentium 120 de procesador, 32 MB RAM y 60 Mb de disco duro hará el trabajo; usted se planteará ¿Por qué no? Adquirir un PC de segunda mano a un precio económico, y ¡no necesita la mejor pantalla para su sistema!

¿Qué encuentra en la caja del HAL2000? El software –por supuesto– entregado en un CD-ROM. No hay ningún grueso manual en papel, encontrará todas las instrucciones en la Ayuda del CD (puede imprimirlo sí así lo desea). El archivo de Ayuda es sumamente rápido e interactivo, y cuando se sitúe encima de las muchas opciones de pantalla, accederá a las ayudas contextuales que le ayudaran a encontrar lo que esta buscando.

La caja incluye un pequeño manual de "Comienzo Rápido" que lo guía a través de un proceso de instalación simple. Los diseñadores del software pensaron en alguna protección extra, así que, contra la piratería del software necesitará llamar a un centro de asistencia técnica HAL2000 para darle su código de acceso personal (es un número nacional, así que no le costará mucho). ¡Una vez hecho esto, todo está listo para comenzar. En la caja HAL2000, usted encontrará también un dispositivo Interface X-10 entre su PC y la línea de corriente y un módulo de lámpara X10. La interface de PC -> X10 se conecta entre el puerto serie de su ordenador y cualquier toma de corriente de pared. Este dispositivo transmite las señales X-10 desde su PC a la instalación eléctrica, y su "línea de corriente" cuida del resto enviando las instrucciones X-10 a los receptores de X10 (en principio el módulo de lámpara incluido). Necesitará conectar este módulo de lámpara en una toma

de corriente de pared, y después conectará una lámpara (incandescente) al módulo.

¿Cómo trabaja el X10?

Realmente es muy simple. Cada dispositivo posee su propia dirección única que usted define rodando los dos diales en el dispositivo. Hay 256 combinaciones, así que puede extender su instalación hasta 256 puntos de control X10 (Muchísimo más de lo que necesitaremos en una casa media). En HAL2000, encontrará los mismos diales que aparecen en los módulos X10, por lo que lo único que cambiará a la hora de su identificación será que aquí los diales los gira con el puntero del ratón. Lleva sólo un par de segundos hacerlo: Usted define un nombre para el dispositivo (usará este nombre al dirigir el sistema por voz), le pone el código correspondiente, prueba el dispositivo en tiempo real, y puede asignar el dispositivo a un grupo para que pueda operar un rango entero de dispositivos y luces incluso con una sola orden. Eso es todo. No hay nada más. ¡Puede agregar nuevos dispositivos o puede renombrar los existentes todas las veces que Ud. quiera, puede fijar su funcionamiento a lo largo de las horas de un día, una semana, etc. ¡ Puede ser tan creativo como quiera ser!.

Podrá adquirir módulos X10 adicionales en el establecimiento donde adquirió el HAL2000 (Se asombrará de las muchas posibilidades que se abren ante sus ojos). Por ejemplo, puede adquirir el kit X10sion HAL2000, (equipo que incluye tres módulos de lámpara adicionales y un módulo de aparato), o puede adquirir módulos individualmente empaquetados, interruptores X10 de pared, sensores, módulos raíl DIN, timbres X10,

interruptores de persianas X10, módulos universales, módulos "Powerflash", etc... Una larga lista de dispositivos del bus domótico mas extendido mundialmente "X10".

Además de éstos dispositivos de control, también puede agregar sensores que le permiten programaciones del tipo «**SI** el sensor **ENTONCES** haz...» . De hecho, puede querer que su casa responda de una cierta manera cuando ciertas condiciones se den.

¡Por ejemplo, usted está entrando en su garaje o en un cuarto trastero, y quiere que las luces se activen solo con su presencia, que la radio se conecte , o ¿por que no? que su ordenador le diga la buena cara que tiene hoy "Manolo, tienes buena cara, no trabajes mucho hoy! ».

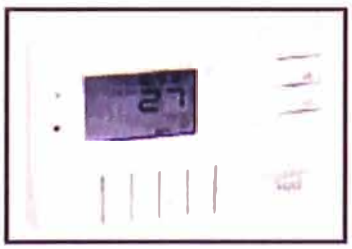
Muy fácil con HAL2000. Tan solo necesita un kit de Sensores de movimiento optativo. Este viene con un módulo transceptor RF-X10 qué conectará en una toma de corriente de pared y tres sensores de movimiento inalámbricos, operados por pilas que atornillará en una pared o en un techo. Cada sensor tiene dos diales para que pueda darles un código X10 disponible. Cuando detecten su entrada, el transceptor captura el código por medio de las señales de radio frecuencia enviadas por los sensores, y lo convierte en señales X10 que pasa a la linea eléctrica; estas señales llegan a HAL2000 a través del interface de su ordenador, y el sistema responderá de la manera que habia previamente especificado, quizás de diferente forma en función de unas determinadas condiciones, como la hora, la fecha... Preparar un sensor X10 en HAL2000 es tan simple como preparar cualquier otro módulo X10.

El abanico de controladores que soporta HAL2000 es impresionante, y usted podría ya disponer de alguno en su casa. De todas maneras cuando compra HAL2000, ya puede empezar a desarrollar su sistema, con ideas (de seguro) más originales, funcionales, y creativas que las que pueda leer en este documento.

Ampliando su sistema

Ya hemos visto como HAL2000 usa la tecnología X10 para controlar algo que es eléctrico en nuestra casa. Llevémoslo un paso más allá. De hecho usted también puede controlar y puede automatizar cosas tales como termostatos, acondicionadores de aire, videos, instalaciones de alta fidelidad, etc...

Por supuesto, controlar estos dispositivos necesita algo más que HAL2000, un PC, y X10. Necesitará periféricos adicionales. Pero lo grande es que la mayoría de estos periféricos son "Plug&Play", y no le costarán una fortuna.



Calefacción. Muchas casas ofrecen termostatos electrónicos estos días. HAL2000 interactua recíprocamente con el termostato preguntando a su sensor de temperatura y controlando su calefacción a los niveles deseados. Sin embargo, usted probablemente no podrá usar su termostato actual. La

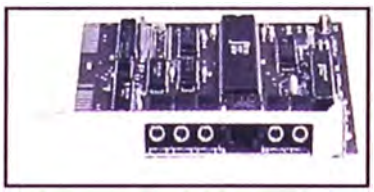
razón es muy simple: la mayoría de los termostatos nunca se ha diseñado para comunicar con un ordenador (o con cualquier otro dispositivo controlador). Así que, su termostato necesita ser uno de los que si pueden trabajar con HAL2000.

Necesita reemplazar su termostato por uno soportado por el sistema HAL2000. La lista incluye termostatos de RCS, Enerzone y Xencom. Éstas no son marcas europeas típicas. Pida a su establecimiento de compra o instalador un termostato bajo la marca COCONZ. COCONZ ofrece termostatos que son soportados a través de HAL2000 y tienen certificado su uso en Europa. Una vez tiene uno, puede reemplazar su termostato actual simplemente conectando el nuevo de la misma manera. El equipo puede venir con una caja adicional que necesitará conectar en la instalación eléctrica que va del sistema calorífico al termostato; esta caja recibe y envia señales a través de la tecnología X10.

Otra opción es usar una Central de Automatización Coconz. Estas centrales se instalan en su garaje, sótano o cualquier pequeño rincón de su casa. Ellas le permiten que conecte termostatos múltiples, todos controlados por HAL2000 a través de comunicaciones serie RS-232 (en este caso, su PC se conecta con un cable serie a la central). La misma central también le permite que use una amplia gama de sensores sin hilos o sensores cableados (movimiento, humo, luz, inundación, rotura, gas,...).

Instalar esta central requiere la intervención de un profesional. Por otro lado, los kits ofrecidos para HAL2000 son bastante sencillos y fáciles de instalar por usted mismo.

Una vez el termostato se instala, lleva unos segundos el configurarlo en HAL2000. Puede llamar al sistema desde donde quiera que usted este, y pedir al termostato la temperatura actual, aumentar o bajar la temperatura un par de grados, o a la temperatura que desee. Y, por supuesto, puede programar el termostato perfectamente en su PC, y fijarlo para horas específicas o fechas específicas. Usted encontrará que es mucho más fácil de programar el termostato de esta manera, incluso más que en el propio termostato. Además de todo esto, ¿ Ha pensado que esto lo puede hacer a través de un móvil desde su coche, o a través de cualquier teléfono desde la oficina..? Nunca más se encontrará que su termostato lleva 4 horas calentando su hogar, justo el día que Ud. retraso su llegada.



Programe su vídeo por voz

Seamos honrados: muchas personas tienen problemas a la hora de programar su vídeo. Los fabricantes de VCR siguen agregando rasgos para darles más valor a sus clientes. Pero, también se pone más difícil operar estas máquinas.

Con HAL2000, programar su vídeo es tan fácil como decirle «Graba esto». ¡Realmente, ésa es de hecho la única cosa que usted necesita decir! Coja su teléfono (local o remoto), y diga algo como «viernes, a las 5 PM, graba el Canal 5 durante dos horas». Así, si usted está en la oficina y tiene la

programación delante, puede llamar a casa simplemente y puede decirle a HAL2000 qué hacer. Por supuesto, usted todavía necesita poner un videocassette en el registrador, pero eso no es una sorpresa. (Sin embargo, en el futuro cercano, puede esperar que HAL2000 grabe su programa favorito en su disco duro o en un DVD reescribible).

Otra manera de programar su VCR es usando la propia información que HAL2000 es capaz de proveer. Si tiene una cuenta de Internet en su PC, usted puede hacer que HAL2000 recoja en intervalos definidos y desde Internet, información útil para su conveniencia personal: correo, noticias, datos de bolsa... y programaciones de televisión. Esto significa que puede preguntar a HAL2000 (simplemente) en cualquier momento: «Lo que hacen en Eurosport a las 6 PM», y el sistema le responderá dandole el nombre del programa que está previsto en ese horario. Usted responderá con «Graba este», y el sistema grabará la transmisión . ¿Cómo hace HAL2000 esto? Todo lo que necesita es el Equipo económico de A/V Inalámbrico IRX10 (o cualquier otro sistema soportado por HAL2000 como el JDS Xpander Infrarrojo o el IR Blaster de Sunbelt).

El equipo IRX10 es una tarjeta ISA que se instala en su PC, y tiene un receptor infrarrojo y dos emisores infrarrojos. El receptor es capaz de “aprender “ las frecuencias usadas por su telemando de VCR, TELEVISIÓN, alta fidelidad... En HAL2000, hay una aplicación especial que habilita a HAL2000 para ‘aprender las señales apropiadas que se corresponden a las varios órdenes posibles, como " Sube Volumen", "Graba", "Canal 1", etc... Una vez hecho esto, HAL2000 está listo para operar todo el equipo que

usted ha configurado. Cada vez que usted da una instrucción a HAL2000, este enviará la señal apropiada a través de los emisores infrarrojos de IRX10. Por supuesto, con este panorama , se requiere que que su PC este instalado en el mismo cuarto que su equipo de A/V, ya que las señales infrarojas necesitan ser visibles por parte de los receptores del equipo. Pero, hay algunos dispositivos, en el mercado que convierten esas señales infrarojas en señales de radio frecuencia capaces de atravesar paredes, para después ser convertidas de nuevo en infrarojos con lo que haremos posible poder tener los equipos no necesariamente en el mismo cuarto, y de esta manera solventar las limitaciones de las señales infrarojas.

Comunicándose con su sistema HAL2000

Usted ha visto como HAL2000 le permite que controle, programe y automatice varios dispositivos eléctricos y electrónicos en su casa, incluidas luces, aparatos, termostatos, equipo de A/V, seguridad... Usted también vió como HAL2000 es algo más que un sistema de automatización de su hogar: permite la verdadera convergencia de todos los dispositivos, incluido algo muy importante ,la telefonía y la información que desea. Además, HAL2000 interactua con Internet y le permite pedir boletines meteorológicos, lee las noticias, le dice lo que está previsto ver en su TELEVISIÓN y le advierte cuando sus acciones preferidas van más allá de los límites que usted ha definido.

Y esto es justo el principio. HAL2000 es un producto VIVO y continuará evolucionando con nuevos rasgos que usted puede descargar

(gratuitamente) desde varias paginas web. HAL2000 realmente es el "Sistema Operativo de su Hogar".

Ahora echemos una mirada a cómo usted interactua con el sistema. Una vez ha instalado HAL2000 en su PC, puede automatizar, controlar y fijar toda esta funcionalidad en sus pantallas. Aun cuando decida no usar la innovadora tecnología de reconocimiento de voz que incluye el sistema HAL2000, usted puede sacar un valor increible del sistema a través de su interface de usuario gráfica, de muy fácil uso. Desde sus pantallas gráficas es posible comunicarse de una forma intuitiva y muy visual con su sistema. En este caso, su pantalla de PC actuará como una verdadera consola de control interactiva. Usted no necesita ser un programador, todo es simple "señale y pulse" El único momento en el que tiene que hacer uso de su teclado es cuando de nombre a sus dispositivos, o cuando teclee un texto que le gustaría que HAL2000 leyera en momentos predefinidos (tecnología de texto-a-voz).

Para reproducir y grabar audio en HAL2000, necesita una tarjeta de sonido. Con un micrófono simple, puede pasar sus instrucciones de voz a HAL2000, y puede pedirle que confirme todo lo ordenado. De manera que a través de los altavoces externos de su PC le pedirá confirmación de todo lo que ud. decidió ejecutar. En esta aproximación de uso, necesita estar delante de su ordenador, excepto si usted instala un sistema de micrófonos por toda la casa "open air", esto le permitiría comunicarse con HAL2000 por toda la casa . Sin embargo, no es fácil encontrar sistemas de buena calidad. Después de todo, no había antes de HAL2000 necesidad de comunicarse

con su PC desde cualquier punto de la casa con voz; es de hecho la primera aplicación que lleva el reconocimiento de voz al servicio de la casa y del bienestar personal.

Muchos fabricantes están trabajando actualmente en sistemas que le permiten este tipo de comunicación con su ordenador, desde donde quiera que usted este en la casa. Nuestra sugerencia es seguir al tanto de nuevos desarrollos en esta área. Realmente, esta es la grandeza de HAL2000 , usted puede seguir ampliando su sistema con nueva funcionalidad y nuevos periféricos hasta ver cumplidas sus expectativas o justo hasta el punto donde le lleve su presupuesto.

Actualmente, la mejor manera y más fiable de comunicarse con su ordenador es usando un módem con soporte de voz. En primer lugar, usted necesita un módem si quiere aprovechar las funcionalidades de HAL2000 para transmitir información desde Internet, como boletines meteorológicos y noticias. Para este rasgo, su módem actual hará probablemente la labor. Sin embargo, si usted quiere interactuar con su sistema desde cualquier teléfono con su voz, necesita un módem con soporte de voz.

Hay dos cosas que usted tiene que ver a la hora de asegurarse que el modém con soporte de voz es el adecuado para trabajar con HAL2000; asegúrese que tiene dos conectores RJ-11, uno de ellos lo usaremos para conectar el módem a su linea telefónica, y el otro para conectar un teléfono local. Si su casa dispone de instalación más compleja con supletorios , puede hacer uso de cualquier teléfono en la casa para hablar con el

sistema. Sin embargo, usted encontrará que un teléfono inalámbrico es la manera más práctica de hacer esto: puede usarlo desde cualquier punto de la casa. En este caso, conectará la base del inalámbrico al módem, y listo .

Pero, antes necesita asegurarse que su módem con soporte de voz cumple unos requisitos que desgraciadamente no todos los modems con soporte de voz ni todos los fabricantes cumplen, posiblemente porque en un pasado muy reciente no daba ningún valor añadido toda esta funcionalidad (p.ej. soporte DTMF). En su establecimiento o central de soporte tienen un pequeño programa que evalua si su módem es adecuado para sacarle el máximo partido a HAL2000. De todas maneras y si ud. tiene que adquirir un nuevo modem pregunte en su establecimiento o centro de soporte por la lista de modems compatibles con HAL2000 (Modemblaster de Creative Labs, algunos modems de Aztech, Zoltrix, Zoom, Wisecom...).

BIBLIOGRAFÍA

1. **Tratamiento digital de señales,** autores: John G. Proakis & Dimitris G. Mano akis; Editorial: Prentice Hall, Inc., 3ª Edición – año 2000.

2. **Tratamiento de señales en tiempo discreto,** autores: Alan V.Oppenheim & Ronald W Schafer; Editorial: PRENTICE HALL, Inc., 2ª Edición – año 2000.

3. **Tratamiento digital de la señal. Una introducción experimental,** autores: José B. Mariño Acebal, Francesc Vallverdú Bayés, José A. Rodriguez Fonollosa & Asunción Moreno Bilbao; Editor : ALFAOMEGA GRUPO EDITOR , 2ª edición – año 2001.

4. **Tratamiento digital de voz e imagen y aplicacion a la multimedia ,** autor: Marcos Faúndez Zanuy ;Obra publicada por ALFAOMEGA GRUPO EDITOR, 1ª Edición – año 2001

5. **Reconocimiento de voz y Fonética acústica,** autores: Jesús Bernal, Jesús Bobadilla Sancho & Pedro Gomez Vilda; Editorial: Alfaomega, 2 edición – año 2000.

6. **Solución de problemas de ingeniería con matlab,** autor: Delores M. Etter; Editorial: PRENTICE HALL, Inc.;2ª Edición – año 1998.

7. **Tatamiento de la señal utilizando matlab v.4,** autores: C. Sidney Burrus , James H. McClellan, Alan V. Oppenheim, Tomas W. Parks,

Ronald W.Schafer & Hans W. Schuessler Obra publicada por PRENTICE HALL, Inc.,1ª Edición - Año 1998.