UNIVERSIDAD NACIONAL DE INGENIERIA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ANÁLISIS DE LA VOZ USANDO CODIFICACIÓN PREDICTIVA LINEAL

INFORME DE SUFICIENCIA

PARA OPTAR EL TÍTULO PROFESIONAL DE

INGENIERO ELECTRÓNICO

PRESENTADO POR:

WILLIAM CASTRO GRIJALVA

PROMOCIÓN
1995 – I

LIMA – PERÚ
2002

A mis padres, Olga y Bernardo, por ser la luz que con sus enseñanzas ,dedicación y esfuerzo, guiaron cada paso que recorrí, durante mis largos años de estudio. A mis hermanos ,Roberto y Maritza por el apoyo y sus consejos. A mi esposa Nelly por su paciencia, comprensión y su compañía durante mis horas de investigación. A mis tíos Pablo y Reina por sus consejos y por haberme apoyado. A mi asesor Dr. Jorge Del Carpio, por sus grandes ideas ,su dedicación y por el tiempo compartido para guiarme en la realización de esta tesis.
A todos mis profesores por su enseñanzas.

ANÁLISIS DE LA VOZ USANDO CODIFICACIÓN PREDICTIVA LINEAL.

SUMARIO

La codificación predictiva lineal (LPC), es un método de análisis de una señal digital, el cual ha sido adoptado para el procesamiento de la señal vocal, dado que se adapta al modelamiento matemático del tracto vocal, que tiene como uno de sus aspectos más importantes el hecho de que la alta correlación existente entre muestras de la señal, permite calcular un valor particular como una combinación lineal de los valores anteriores de la misma, este aspecto es la base principal del estudio del modelamiento predictivo lineal.

El modelo LPC fue propuesto como un método para la codificación de la señal vocal de los seres humanos por el Departamento de Defensa de los Estados Unidos, en su norma federal 1015, publicado en 1984. La codificación predictiva lineal ha cobrado importancia gracias a su capacidad de compresión de la información contenida en la señal vocal, la cual en circunstancias normales tiene una tasa de 64000 bits/segundos, como resultado de ser muestreada a 8000 muestras/segundo y usando 8 bits para representar cada muestra. La codificación predictiva lineal, reduce esta tasa a 2400 bits/segundos, a éste nivel el habla tiene un sonido sintético característico y hay una apreciable pérdida de calidad. Sin embargo, la voz es todavía audible y fácilmente entendible. Dado que existe una pérdida de información en la codificación predictiva lineal, se le conoce como un método de compresión del habla con pérdida.

En el presente informe, se describirá el fundamento teórico necesario para entender el modelamiento matemático del tracto vocal. Y también se mostrará como la codificación predictiva lineal modela matemáticamente los parámetros del tracto vocal

para reducir la señal a un estado que es apreciablemente sintético pero comprensible. Al final se concluirá discutiendo otros esquemas de codificación que se han basado en el modelo LPC y además se discutirán algunas desventajas y aplicaciones del modelo LPC.

ÍNDICE

PRÓLOGO

Existen muchos tipos de métodos de compresión de la señal vocal que hacen uso de una gran variedad de técnicas, la mayoría de los cuales hacen el uso del hecho que la producción de la señal vocal ocurre a través de movimientos anatómicos lentos en el tracto vocal y que la señal así producida tiene un rango de frecuencia que va desde 300 Hz a 3400 Hz aproximadamente. La compresión de la señal vocal es también conocido como codificación y se define como método para reducir la cantidad de información necesaria para representar la señal vocal. La mayoría de estos métodos se basan en algoritmos que producen pérdidas; estos algoritmos se consideran aceptables siempre y cuando la pérdida de calidad producida por la codificación sea inapreciable por el oído humano.

Existen muchas características inherentes a la producción de la señal vocal que pueden ser explotados por los algoritmos de codificación. Una característica que es frecuentemente utilizado es que el periodo de silencio en una conversación es mayor que el 50% de la duración de la misma. Una forma fácil de ahorrar ancho de banda y reducir la cantidad de información requerida para representar la señal de voz, es evitar enviar este silencio. Otra característica de la producción de la señal vocal que debe tomarse en cuenta es que mecánicamente existe una alta correlación entre muestras adyacentes de la señal vocal.

La codificación Predictiva Lineal (LPC), es uno de los métodos de compresión que modela matemáticamente el proceso de la producción del habla. Específicamente, el

método LPC modela este proceso como una suma de muestras anteriores usando un filtro digital. El modelo LPC, fue propuesto como un método para codificar la señal vocal por el Departamento de Defensa de los Estados Unidos en su norma federal 1015, publicado en 1984.

La compresión o codificación de la señal vocal es usualmente llevado a cabo por codificadores de voz o vocoders. Hay dos tipos de vocoders: Los codificadores que "siguen la señal" y los codificadores "basados en modelos". Los codificadores del primer tipo se caracterizan por que en ausencia de errores de cuantificación reproducen exactamente la señal vocal original. De otro lado los codificadores basados en modelos aún en ausencia de errores de cuantificación nunca reproducirán la señal vocal original, ésto se debe a que el modelo paramétrico de producción del habla implica la codificación y transmisión de los parámetros del modelo y no de la señal. Los vocoders LPC, son considerados codificadores basados en modelos, lo cual significa que la codificación LPC es con perdida aún asi no hubiesen errores de cuantificación.

Todos los vocoders, incluyendo los vocoders LPC, tienen cuatro características principales, que son: Tasa de bits, retraso, complejidad y calidad. Cualquier codificador de voz, sin importar el algoritmo que este use, tiene que realizar un compromiso entre estos atributos.

La primera característica de los vocoders, la tasa de bits, se usa para determinar el grado de compresión que se logra con un vocoder. La señal vocal sin compresión es usualmente transmitida a una tasa de 64 Kbits/s usando 8 bits/muestra y una frecuencia de muestreo de 8 kHz. Cualquier tasa de bits por debajo de 64 Kbits/s se considera como

compresión. La codificación predictiva lineal transmite la señal vocal a una tasa de 2.4 Kbits/s, lo cual significa una excelente tasa de compresión.

El retraso, es otra característica importante para los vocoders que están asociados a la transmisión de la señal vocal codificada. Los vocoders asociados al almacenamiento de la señal vocal comprimida, no están afectados por este atributo. El estándar general, para el retraso en el caso de la transmisión de una conversación es que cualquier retraso mayor que 300 ms es considerado inaceptable.

La tercera característica, es la complejidad del algoritmo usado. La complejidad afecta tanto al costo, como a la potencia del vocoder. El algoritmo de codificación lineal predictiva debido a su alta tasa de compresión es muy complejo e implica la ejecución de millones de instrucciones por segundo. La implementación en hardware del algoritmo LPC, a menudo requiere más de un procesador para correr en tiempo real.

La característica final de los vocoders es la calidad. La calidad es un atributo subjetivo y depende como el sonido de la señal vocal es captado por un oyente determinado. Uno de los test más comunes para determinar la calidad del habla es el test de valoración de categoría absoluta (absolute category rating - ACR). Este test implica que un grupo determinado de personas, escuche un par de oraciones y luego califiquen los mismos en una de las siguientes categorías: excelente, bueno, regular, pobre, malo. Los codificadores LPC, sacrifican calidad a fin de lograr una tasa baja de bits y como resultado se obtiene un sonido sintético. Un método alterno de compresión del habla es llamado modulación por pulsos codificados diferencial adaptivo (ADPCM) que sólo

reduce la tasa de bits por una factor de 2 a 4, pero tiene una calidad mucho mayor del habla que el método LPC.

El algoritmo general para la codificación predictiva lineal implica dos etapas, la parte de análisis o codificación y la parte de decodificación o síntesis. En la parte de codificación, el método LPC toma la señal del habla en bloques o segmentos del habla y determina el tipo de señal de entrada , sonora o no sonora, y los coeficientes del filtro que serán capaces de reproducir el segmento analizado. Esta información es cuantificada y transmitida. En la parte de decodificación, el modelo LPC reconstruye el filtro basado en los coeficiente recibidos.

CAPÍTULO I

PERSPECTIVA HISTÓRICA DE LA CODIFICACIÓN PREDICTIVA LINEAL

La historia de la compresión de audio y música comienza en la década de los 30's con la modulación codificada por pulsos (PCM) y la codificación PCM. La compresión de audio digital fue iniciado en la década de los 60's por las compañías de teléfono interesadas en el costo del ancho de banda de transmisión. Los orígenes de la codificación predictiva lineal comienza en la década de los 70's con el desarrollo de los primeros algoritmos LPC. Existe también otro método de codificación del habla, que es la Modulación por Pulsos Codificados Diferencial Adaptiva (ADPCM), que también fue concebido en la década de los 70's. En 1984, el Departamento de Defensa de Estados Unidos creó la norma federal 1015 en la cual se detallan los principios del método LPC. También existen algoritmos que son extensiones del método LPC, tales como el Code Excited Linear Predictive (CELP) y el Vector Selectable Excited Linear Predictive (VSELP) que fueron desarrollados a mediados de los 80's y usados para codificación de audio en la última parte de esa década. En la década de los 90's se hicieron mejoras a estos últimos algoritmos mejorando su tasa de compresión y su nivel de calidad.

La historia de la codificación de la señal vocal no hace mención del método LPC hasta la década de los 70's. Sin embargo, la historia de la síntesis del habla muestra que los comienzos de la codificación predictiva lineal ocurrieron 40 años antes en la década de los 30's. El primer vocoder fue descrito por Homer Dudley en 1939 en los

Laboratorios Bell. Dudley desarrolló su vocoder, llamado Vocoder Pasabanda Paralelo o Vocoder de Canal, para realizar análisis y síntesis del habla. El método LPC, es descendiente del vocoder de canal. El esquema de análisis/síntesis usado por Dudley es el esquema de compresión que es usado en muchos tipos de métodos de compresión de la señal vocal tal como el LPC. La parte de síntesis de este esquema fue usado por primera vez aun antes de la década de los 30's por Kempelen Farkas Lovag (1734 – 1804). El usó este método, para realizar la primera máquina que podía hablar.

Los esquemas de análisis y síntesis están basados en el desarrollo de un modelo paramétrico durante el análisis de la señal original, el cual es usado mas tarde para la síntesis de la señal a la salida. El transmisor analiza la señal original y adquiere los parámetros para el modelo los cuales son enviados al receptor. El receptor luego usa el modelo y los parámetros que recibe para sintetizar una aproximación de la señal original. Históricamente, este método de enviar los parámetros del modelo hacia el receptor fue la primera forma de compresión de la señal vocal con pérdidas. Otras formas de compresión de la señal vocal con pérdidas que implican el envío de valores estimados de la señal original no fueron desarrollados sino hasta mucho después.

CAPÍTULO II

PRODUCCIÓN DEL HABLA EN LOS SERES HUMANOS

2.1 Modelo de Producción

Sin importar del idioma hablado, todos los seres humanos usan relativamente la misma anatomía para producir el habla. La señal producida por la anatomía de los seres humanos esta basada en las leyes de la física, En la figura 1 se muestra aproximadamente 1 segundo, de una señal vocal correspondiente a una conversación.

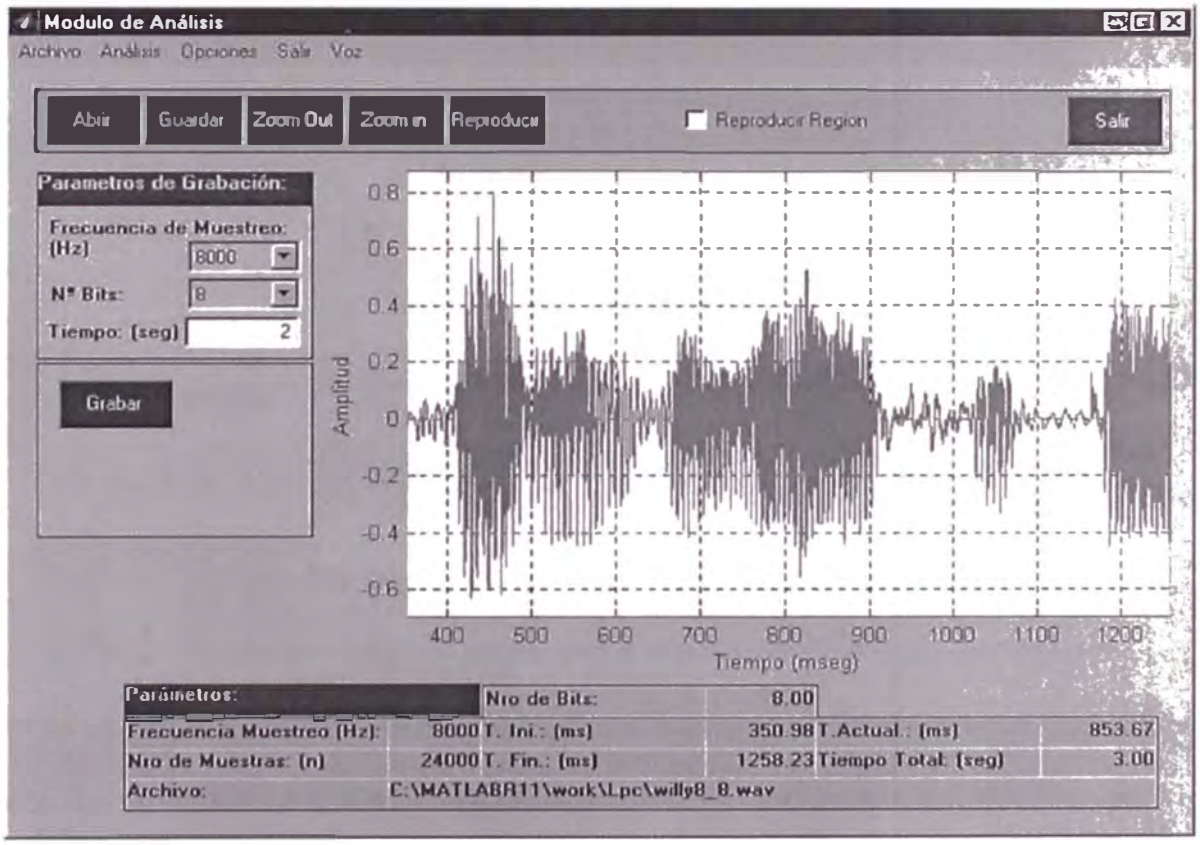

Figura 1: Muestra de una Señal Vocal

El proceso de producción del habla en los seres humanos puede explicarse como el flujo de aire que es impulsado desde los pulmones, a través del tracto vocal, y liberado a través de la boca para generar el habla, como se muestra en la figura 2. De acuerdo a esta descripción se puede considerar a los pulmones como la fuente de sonido y el tracto vocal como un filtro que produce varios tipos de sonidos que generan el habla. Esta descripción es una simplificación de cómo el sonido es producido realmente.

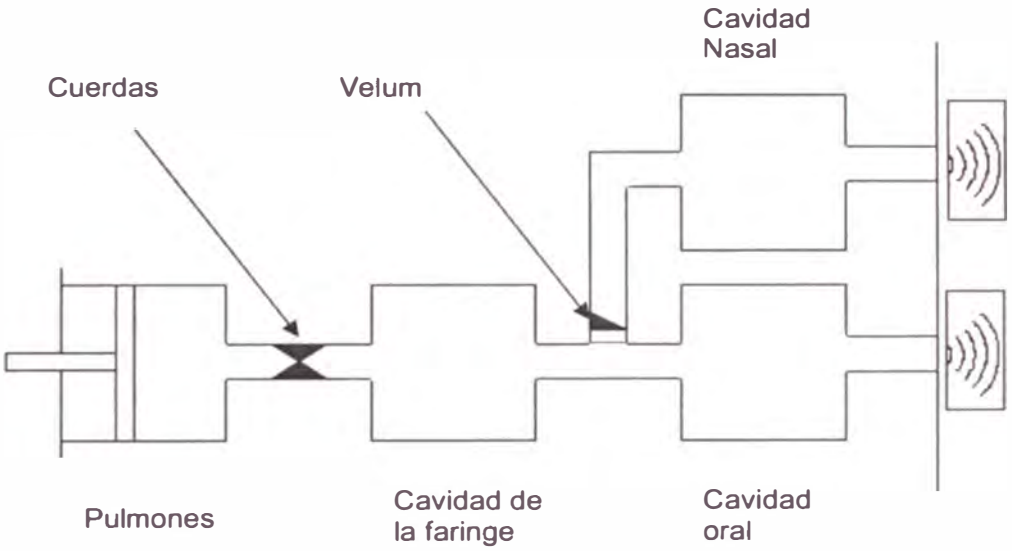

Figura 2: Modelo de producción del sonido

A fin de entender como el tracto vocal convierte el aire de los pulmones en sonido es importante entender varias definiciones claves. Los *fonemas,* se definen como un conjunto limitado de sonidos individuales. Existen dos categorías de fonemas, *los sonidos sonoros y no sonoros* que son considerados por el codificador predictivo lineal cuando se realiza el análisis y la síntesis de la señal vocal.

Los *sonidos sonoros,* son usualmente las vocales y a menudo tienen altos niveles de energía promedio y diferentes frecuencias resonantes o formantes. Estos sonidos son generados por el aire que fluye desde los pulmones, y, que atraviesan las cuerdas vocales. Como resultado las cuerdas vocales vibran de una forma ligeramente periódica produciendo una serie de pulsos de aire llamados pulsos glotales. La tasa a la cual vibran las cuerdas vocales es la que determina el pitch o frecuencia fundamental del sonido producido. Estos pulsos de aire que son creados por las vibraciones finalmente atraviesan a lo largo del tracto vocal donde se produce la resonancia de algunas frecuencias. En general se ha determinado que las mujeres y los niños tienen niveles de pitch mayores que la de los varones esto se debe a una tasa rápida de vibración durante la producción de sonidos sonoros. Por este motivo es importante incluir el periodo pitch en el análisis y síntesis del habla si se desea que la salida final represente precisamente a la señal de entrada original.

Los sonidos *no sonoros,* son usualmente algunas consonantes y generalmente tienen menos niveles de energía y mayores frecuencias que los sonidos sonoros. La producción de sonidos *no sonoros* es provocado por el aire que es forzado a través del tracto vocal en un flujo turbulento. Durante este proceso las cuerdas vocales no vibran, al contrario, ellas se mantienen abiertas hasta que el sonido es producido. En este caso el pitch no es un atributo importante ya que no hay vibración de las cuerdas vocales y tampoco pulsos glotales.

La categorización de los sonidos como *sonoros y no sonoros* es una consideración importante en el proceso de análisis y síntesis. De hecho, la vibración de

las cuerdas vocales, o la falta de vibración, es uno de los componentes claves en la producción de diferentes tipos de sonidos. Otro componente que influye en la producción del sonido es la forma del *tracto vocal* en si mismo. Diferentes formas producirán diferentes sonidos o frecuencias resonantes. En la figura 3, se muestra los órganos de producción del habla, constituidos por el tracto vocal, el cual consiste, de la garganta, la lengua, la nariz y la boca, y es el camino de la producción del habla a través de los órganos vocales, el cual le da forma a las frecuencias de vibración del aire que pasan a través de esta. A medida que una persona habla, el tracto vocal va cambiando constantemente de forma, a una tasa muy lenta para producir diferentes sonidos los cuales fluyen juntos para crear las palabras.

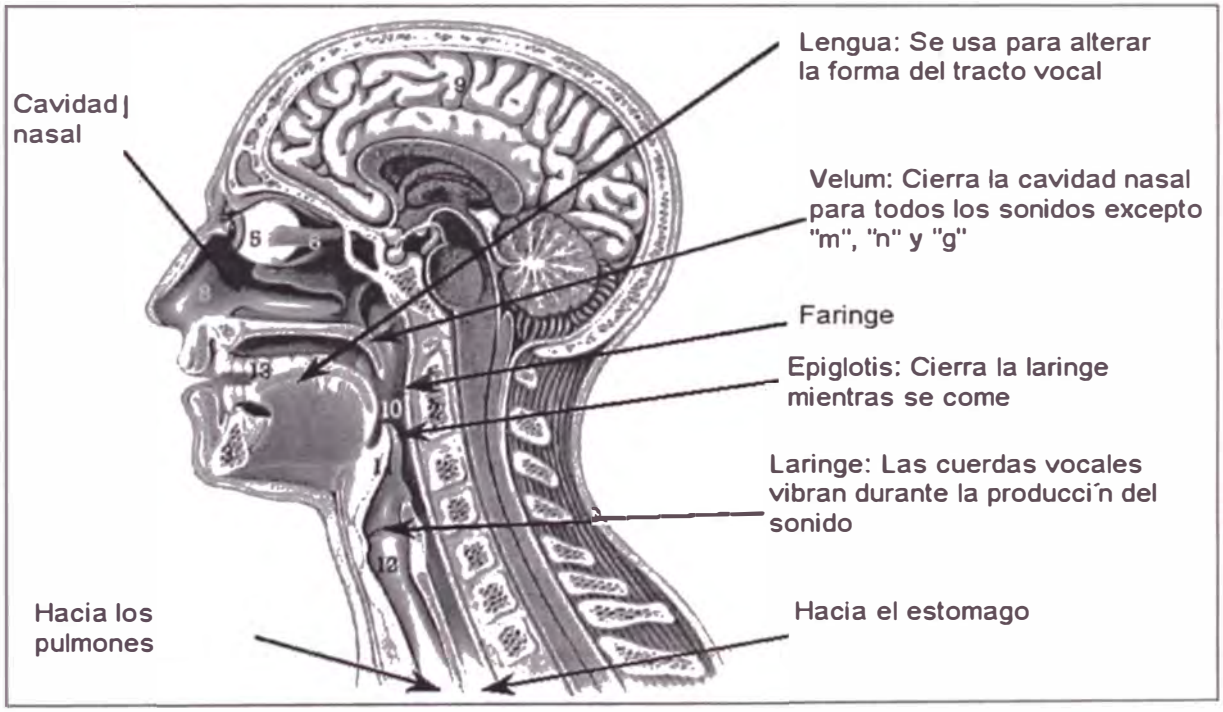

Figura 3. Organos de Producción del Habla.

Un componente final que afecta la producción del sonido en los humanos es la cantidad de aire que originan los pulmones. El aire que fluye desde los pulmones se puede considerar como la fuente para el tracto vocal el cual actúa como un filtro que toma la fuente para producir el sonido. Cuanto mayor sea el volumen de aire que va desde el tracto vocal, mayor será el volumen del sonido.

La idea del modelo de concebir el aire que va desde los pulmones, como una fuente y el tracto vocal como un filtro es llamado modelo *fuente - filtro* para la producción del sonido. Y es el modelo usado en la codificación predictiva lineal. Se basa en la idea de separar la fuente del filtro en la producción del sonido. Este modelo es usado tanto para la codificación y decodificación LPC y es derivado de una aproximación matemática del tracto vocal representado como un *tubo de diámetro variable.* La excitación del aire viajando a través del tracto vocal es la fuente. Este aire puede ser periódico, cuando se producen sonidos sonoros a través de la vibración de las cuerdas vocales, o pueden ser aleatorios y turbulentos cuando se produce sonidos no sonoros.

El proceso de codificación LPC implica la determinación de un conjunto de parámetros precisos para modelar el tracto vocal durante la producción de una señal de habla determinada. La decodificación implica el uso de los parámetros adquiridos en la codificación y análisis, para luego reconstruir una versión sintetizada de la señal de habla original. El método LPC nunca transmite ningún valor estimado de la señal al receptor, éste sólo envía el modelo para producir el habla y algunas indicaciones acerca de que tipo de sonido está siendo producido.

2.2 Modelo del Filtro del Tracto Vocal

2.2.1 Tubo sin pérdidas:

Físicamente se puede modelar el tracto vocal como un conjunto de tubos sin pérdidas conectados entre si, como se muestra en la figura 4, la función de transferencia de un tubo sin pérdidas puede describirse como un modelo *todos polos*. Esta es una aproximación razonable a la producción del habla formado por la excitación del tracto vocal mediante pulsos en la glotis (aunque estos pulsos no sean espectralmente planos).

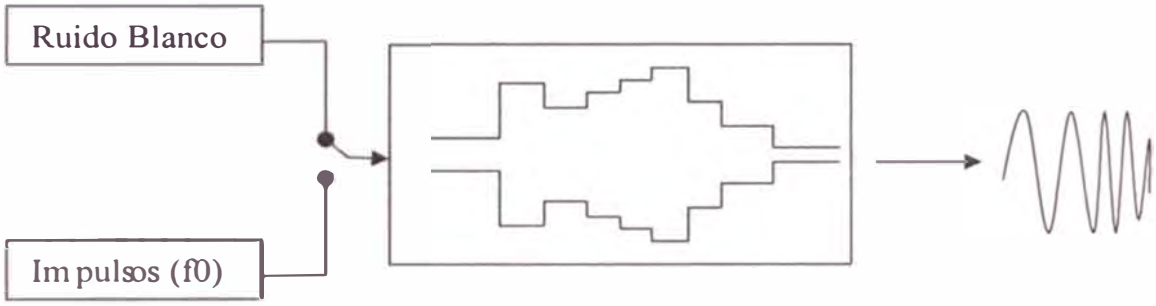

Figura 4: Modelo de tubos sin pérdidas

Pero este modelo no es preciso ya que el tracto vocal no esta formado por cilindros, y se producen pérdidas en su interior, adicionalmente el tracto vocal tiene un pasaje lateral (la cavidad nasal) que modifica su comportamiento, por otro lado los sonidos fricativos (/s/, /f/) se generan cerca de los labios. Sin embargo, con parámetros suficientes, el modelo de predicción lineal, puede conseguir una aproximación razonable a la envolvente espectral de todos los sonidos del lenguaje.

En las figuras 5 hasta la 8, se muestra el comportamiento en frecuencia de una muestra de 20 ms de una señal vocal, y sus correspondientes respuestas después del análisis LPC, de orden,12, 22, y 32

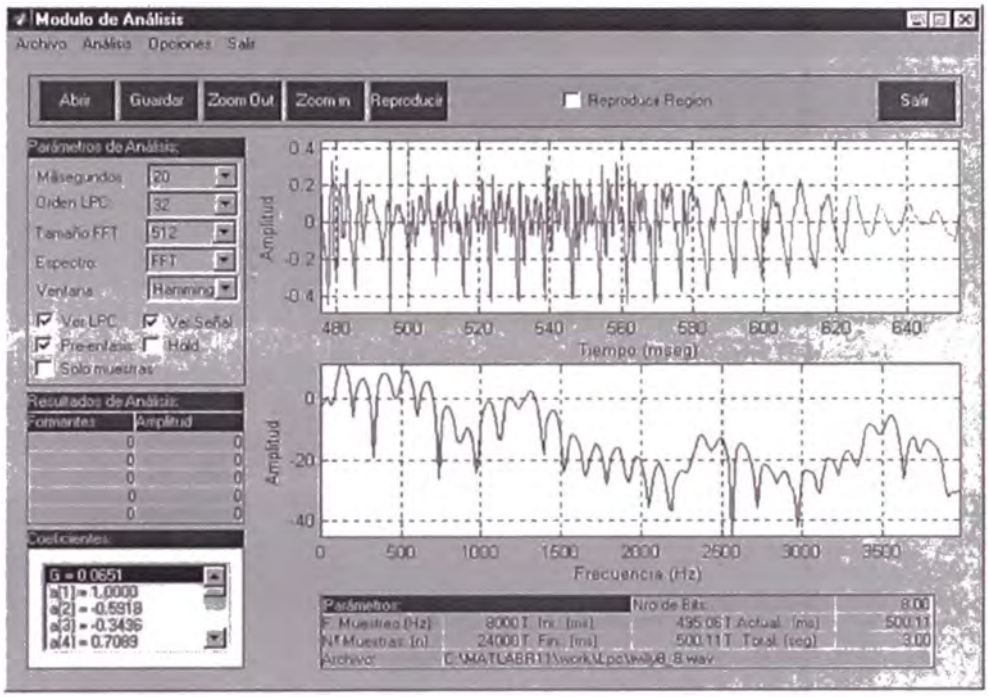

Figura 5: Espectro en frecuencia de 20 ms de la señal vocal

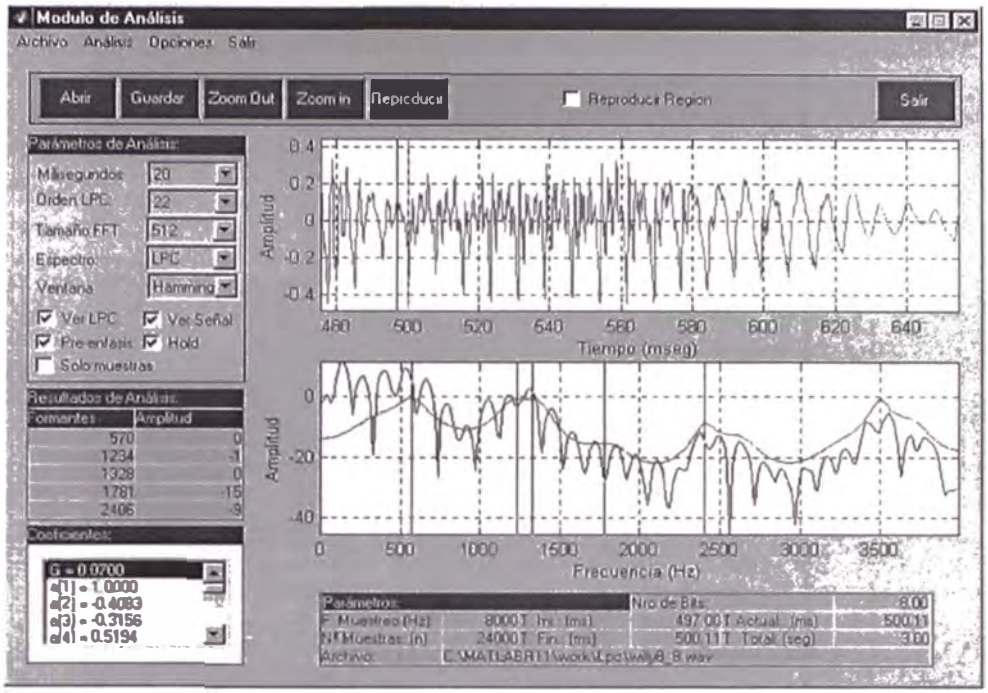

Figura 6: Diagrama LPC de orden 12

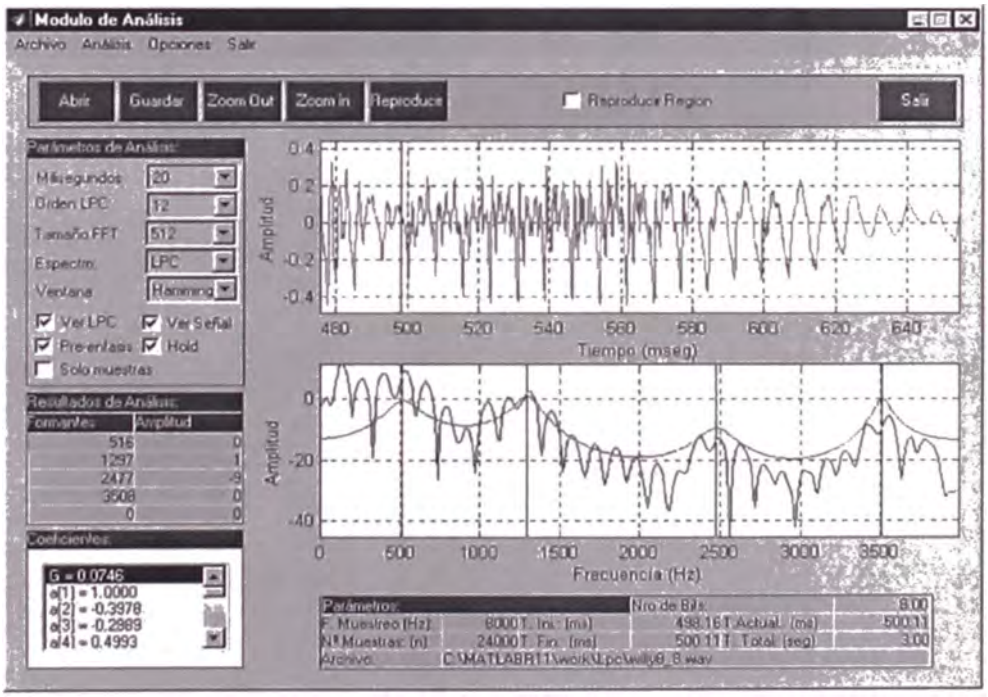

Figura 7: Diagrama LPC de orden 22

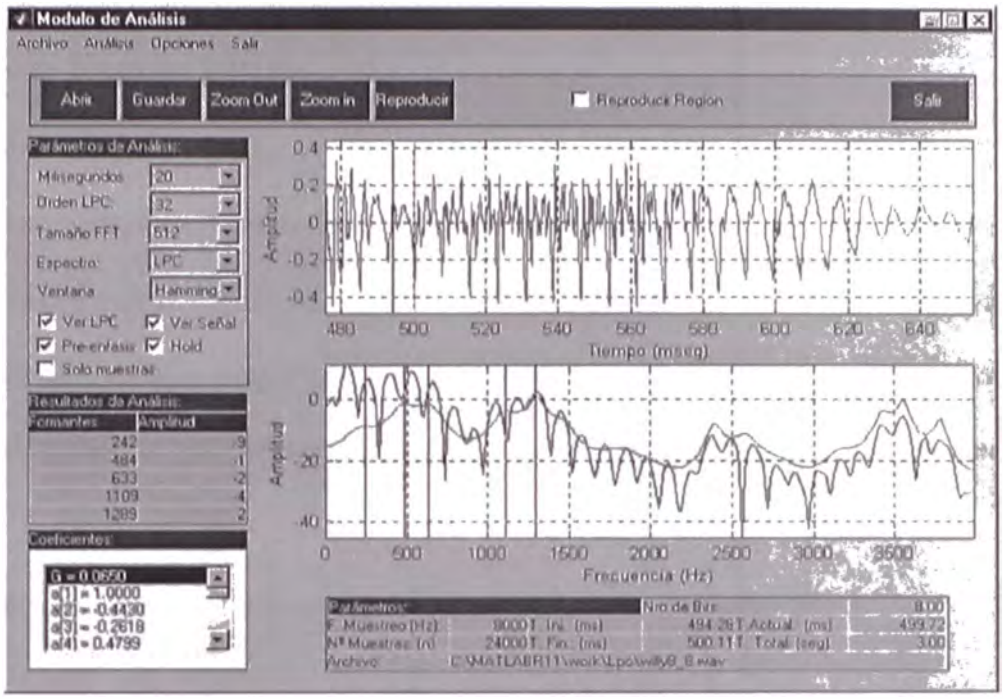

Figura 8: Diagrama LPC de orden 32

2.2.2 El modelo Fuente Filtro

El modelo fuente - filtro usado en LPC es conocido como el modelo de codificación lineal predictiva. Este tiene dos componentes claves: *análisis o codificación* y *síntesis o decodificación* . Para el análisis LPC la señal del habla es dividida en cuadros o tramas de tiempo, luego cada trama considerando un traslape adicional es analizada usando técnicas de procesamiento de señal, para contestar tres preguntas claves:

¿Es el segmento, *sonoro* o *no sonoro?*

¿Cuál es el pitch del segmento?

¿Cuáles son los parámetros necesitados para construir un filtro que modela el tracto vocal de el segmento?

El análisis LPC es usualmente llevado a cabo en el transmisor el cual se encarga de resolver las preguntas anteriores y luego transmite las respuestas de las mismas hacia el receptor. El receptor realiza la síntesis LPC usando los parámetros recibidos para construir un filtro que cuando se alimentado por una señal adecuada sea capaz de reproducir la señal del habla original. Esencialmente, la síntesis LPC trata de imitar la producción del habla humano. En la figura 9, podemos apreciar las partes del receptor que corresponden a sus equivalentes en la anatomía humana. Este diagrama es para un codificador de la señal vocal en general y no es específico para la codificación predictiva lineal. Todos los codificadores tienden a modelar dos cosas: *la excitación y la articulación.* La excitación es el tipo de sonido que es pasado a través del filtro o tracto vocal y la articulación es la transformación de la señal de excitación en voz.

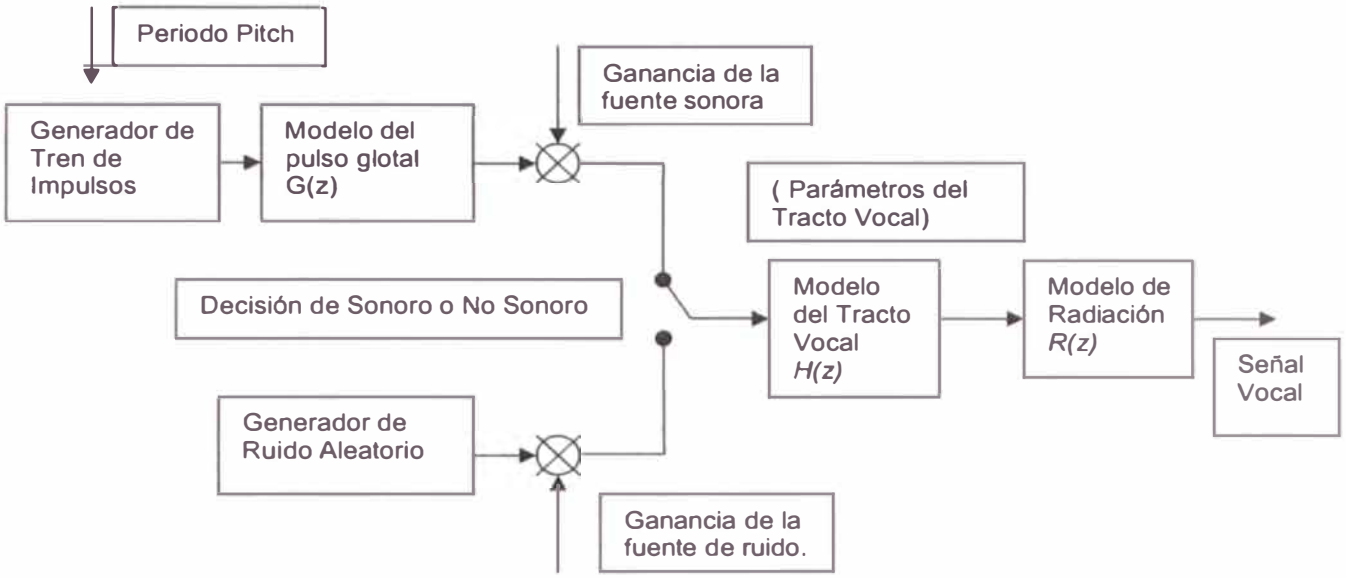

Figura 9: Diagrama de bloque del Receptor

CAPÍTULO III

EL MODELO LPC

3.1 Análisis de Predicción Lineal

La idea de un predictor lineal de orden N, es aquel en la cual la determinación del valor de la siguiente muestra de la señal se obtiene como una combinación lineal de las N muestras anteriores, cada una ponderada por un coeficiente b_i, el modelo es similar al de un filtro FIR, sin considerar el elemento b_0. Se le llama predictor lineal por que no hace uso de potencias, ni multiplicación de las muestras sino sólo usa una combinación lineal de las mismas. Este concepto de linealidad es importante ya que permite hacer el uso de la superposición de señales, como extensión de las propiedades de los sistemas lineales invariantes en el tiempo (SLIT).

El modelo básico de un predictor lineal aplicado al procesamiento de la señal vocal se muestra en la figura 10:

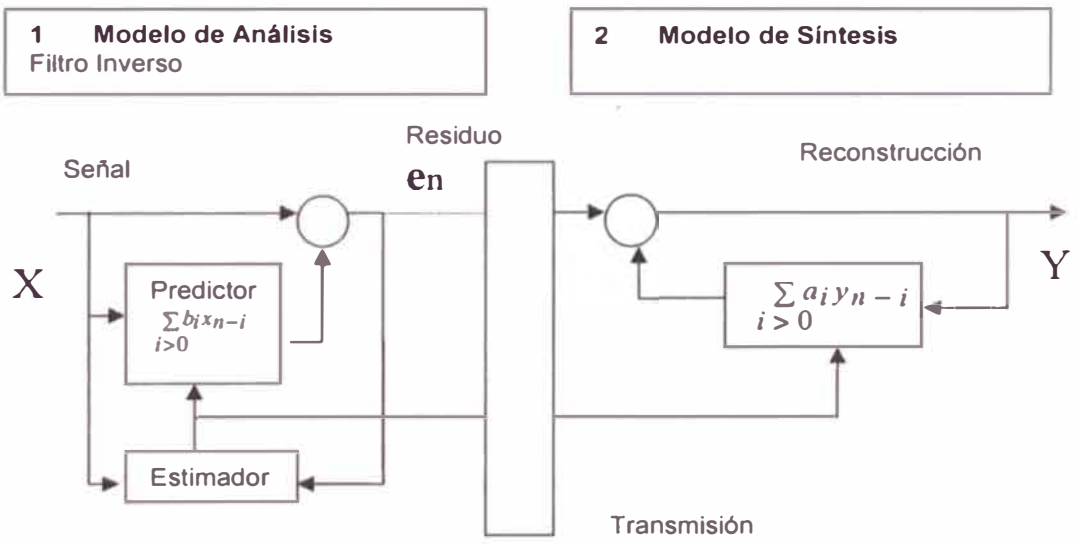

Figura 10: Aplicación del predictor lineal, al procesamiento de la señal vocal

El objetivo es encontrar los coeficientes de modo que el error de predicción sea tan pequeño como sea posible, para esto se usa el criterio de los mínimos cuadrados. Entre los métodos usados para resolver los coeficientes de predicción tenemos el de la *autocorrelación* y la *covarianza.*

El método de la covarianza, es eficiente cuando se trata de extraer el máximo de información de una cantidad reducida de datos, pero puede tener problemas de imprecisión e inestabilidad. Para obtener resultados confiables con este método es necesario contar con datos de buena calidad, sin la presencia de ruido y que representen a un sistema estacionario. No obstante de la cualidad estática del sistema vocal esta no es estacionaria ya que la glotis presenta una impedancia de carga que varía considerablemente durante el periodo glotal . Se considera que el método de la covarianza funciona mejor cuando los datos que se analizan provienen de la parte de los periodos del habla donde la glotis esta cerrada. Una propiedad inherente de este método es que la solución no necesariamente representa a un filtro estable, ya que las raíces del polinomio predictor podrían estar localizados fuera del círculo unitario.

En el caso del método de autocorrelación generalmente se usan mayores cantidades de muestras para los cálculos, para el habla típicamente se necesitan 256 muestras para cubrir más de un periodo pitch. Para reducir los efectos del truncamiento de la señal , se usa una ventana Hamming. Parte del cálculo como su nombre lo indica consiste en realizar la autocorrelación entre muestras, lo cual también incluye un proceso de promediado para reducir la influencia de ruido en la señal. Este método produce un filtro más estable.

3.2 Predicción Lineal en el procesamiento del habla

En general la señal de voz no es una señal determinística, ni estacionaria, sin embargo esta se puede considerar como localmente estacionaria. Consecuentemente, la predicción lineal se puede aplicar en muchas áreas del procesamiento del habla. Por ejemplo, se puede usar en diagnósticos clínicos para analizar los desordenes de la voz. En el área de ingeniería la predicción lineal es un método básico para el análisis, codificación y compresión del habla o para el reconocimiento del habla y del orador.

3.3 Modelo Polos – Ceros

El modelo de predicción lineal de la n-esima muestra *s(n)* es definido como una combinación lineal de la p muestras anteriores a la muestra *s(n)*. De aquí que la representación matemática para la predicción *s(n)* la cual se denota por *s(n)* se puede expresar como:

$$\tilde{s}(n) = \sum_{k=1}^{p} a_k s(n-k) \quad \ldots\ldots\ldots\ldots \quad (1)$$

donde a_k, , $1 \leq k \leq p$ denotan los coeficientes de predicción. Este modelo comúnmente usado, que consiste sólo de muestras anteriores o salidas *s(n)* de el sistema, puede ser generalizado para incluir también las entradas al sistema *u(n);* por lo tanto la representación de la señal *s(n)* de acuerdo a esto es conocido como el Modelo Autoregresivo de Medias Móviles (ARMA). Y se representa de la siguiente manera:

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + G \sum_{l=1}^{q} b_l u(n-l) \quad \ldots\ldots\ldots\ldots \quad (2)$$

donde a_k, $1 \leq k \leq p$, b_l, $1 \leq l \leq q$, y G es la ganancia. La representación de la ecuación anterior en el dominio de la frecuencia es de la siguiente manera:

$$H(z) = \frac{S(z)}{U(z)} = G \frac{1 + \sum_{l=1}^{q} b_l z^{-l}}{1 + \sum_{k=1}^{p} a_k z^{-k}} \quad \ldots\ldots\ldots\ldots \quad (3)$$

Donde *S(z)* y *U(z)* son la transformada z de *s(n)* y *u(n)*, respectivamente. *H(z)* es la función del sistema a ser modelado, por ejemplo en el modelo polos – ceros, las raíces del numerador y del denominador proporcionan los ceros y polos del sistema, respectivamente. Esto nos conduce a los casos especiales de este sistema general. Si asumimos que el numerador es fijo (Por ejemplo, $b_l = 0,\ 1 \leq k \leq q$), el modelo se reduce al modelo *todos - polos*, o Autoregresivo (AR) :

$$H(z) = \frac{G}{1 + \sum_{k=1}^{p} a_k z^{-k}} \quad \ldots\ldots\ldots\ldots \quad (4)$$

El otro caso especial obtenemos asumiendo que el denominador es fijo (Por ejemplo, $a_k = 0,\ 1 \leq k \leq p$), y el modelo se convierte en un modelo *todos - ceros*. Este es llamado modelo de medias móviles (MA). En términos de modelamiento espectral se puede decir que el modelo todos polos estima básicamente el valor máximo local y el modelo todos ceros busca principalmente el valor local mínimo del espectro. En el

modelamiento espectral del habla, es perceptualmente más importante encontrar las partes más energéticas del espectro, por ejemplo los máximos locales o formantes del habla, que los mínimos espectrales, donde hay niveles pequeños de energía. Por consiguiente, el procesamiento del habla usualmente hace uso del modelo todos polos. El modelamiento *todos polos* es el método más simple y fácil ya que no requiere grandes cantidades de calculo ni memoria, especialmente si lo comparamos con el modelo *polos - ceros*, Además, el modelo polos - ceros no pueden ser resuelto en un forma cerrada.

3.4 Modelo Todos Polos

El error entre la muestra que se predice y la muestra real, o conocido o residual se define como sigue:

$$e(n) = s(n) + \tilde{s}(n) = s(n) + \sum_{k=1}^{p} a_k s(n-k) \quad \text{..... .....} \quad (5)$$

La señal en el instante de tiempo n, de la figura 11, puede ser reconstruido de la muestra residual en el instante de tiempo n y de las muestras previas de *s(n)*:

$$s(n) = e(n) - \tilde{s}(n) = e(n) + \sum_{k=1}^{p} a_k s(n-k) \quad \text{.....} \quad (6)$$

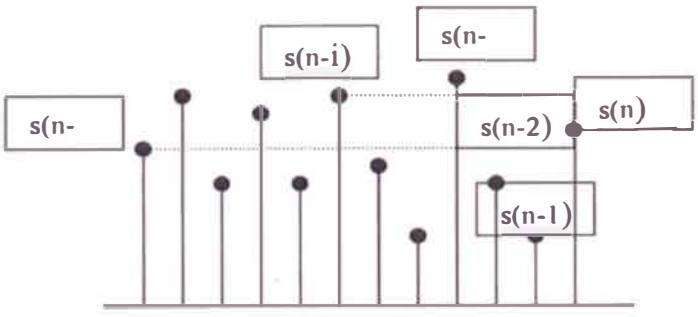

Figura 11: Filtro de Lattice de orden 2

Si el valor de p es suficientemente grande, el modelo todos polos es capaz de modelar la envolvente espectral del habla tan precisamente que el valor residual llega a tener un espectro plano. Matemáticamente este puede ser presentado como sigue:

$$E(z) = \left[1 + \sum_{k=1}^{p} a_k z^{-k}\right] S(z) = A(z)S(z) \quad \ldots\ldots\ldots\ldots \quad (7)$$

$$\Rightarrow \quad S(z) = \frac{E(z)}{A(z)} \approx \frac{G}{1 + \sum_{k=1}^{p} a_k z^{-k}} \quad \ldots\ldots\ldots\ldots \quad (8)$$

Donde *E(z)* y *S(z)* son las transformadas z de *e(n)* y *s(n)*, respectivamente. *A(z)* es llamado el filtro inverso, mientras *S(z)* es el filtro todos polos.

Durante décadas recientes la predicción lineal ha sido desarrollada y modificada, la formulación presentada aquí es la conocida como predicción lineal convencional. La predicción es optimizada minimizando el cuadrado del error de predicción:

$$E = \sum_{n} e^2(n) = \sum_{n} \left[s(n) + \sum_{k=1}^{p} a_k s(n-k) \right]^2 \quad \ldots\ldots\ldots\ldots \quad (9)$$

Alternativamente, el error de predicción en el dominio de la frecuencia se define como:

$$E = \sum_n e^2(n) = \frac{1}{M}\sum_{m=0}^{M-1}\left|E(e^{j\omega_n})\right|^2 = \frac{1}{M}\sum_{m=0}^{M-1}P(\omega_m)A(e^{j\omega_n})A(e^{-j\omega_n}) \quad \ldots\ldots (10)$$

La minimización se logra haciendo que las derivadas parciales de E con respecto a a_i sean iguales a cero,

$$\frac{\partial E}{\partial a_i} = 0, \quad 1 \le i \le p \quad \ldots\ldots\ldots\ldots\ldots (11)$$

La optimización de la predicción de acuerdo al criterio de los mínimos cuadrados, nos conducen a las siguientes ecuaciones conocidas como ecuaciones normales:

$$\sum_{k=1}^{p} a_k \sum_n s(n-k)s(n-i) = -\sum_n s(n)s(n-i) \quad 1 \le i \le p \ldots\ldots\ldots\ldots (12)$$

Si asumimos que el error debe ser minimizado sobre un intervalo infinito de la señal, $-\infty < n < \infty$, entonces la solución de las ecuaciones normales pueden obtenerse por el método de la autocorrelación. La función de autocorrelación se define como:

$$R_n(n) = \sum_{m=-\infty}^{\infty} s_n(m)s_n(m+i) \ldots\ldots\ldots\ldots\ldots (13)$$

Aplicando la función de autocorrelación las ecuaciones normales se puede reescribir en una forma matricial RA = P donde A contiene los coeficientes del filtro.

$$R = \begin{bmatrix} R_n(0) & R_n(1) & R_n(2) & R_n(p-2) & R_n(p-1) \\ R_n(1) & R_n(0) & R_n(1) & R_n(p-3) & R_n(p-2) \\ R_n(2) & R_n(1) & R_n(0) & R_n(p-4) & R_n(p-3) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ R_n(3) & R_n(2) & R_n(M-3) & R_n(1) & R_n(0) \end{bmatrix}$$

$$A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_M \end{bmatrix} \qquad P = \begin{bmatrix} R_n(1) \\ R_n(2) \\ R_n(3) \\ \vdots \\ R_n(M) \end{bmatrix}$$

Esta ecuaciones son conocidas como ecuaciones de Yule - Walker (Yule, 1927; Walker, 1931). A fin de determinar los coeficientes del filtro, mostrados en la matriz A, debemos resolver la ecuación $A = R^{-1}P$. Esta ecuación no puede ser resuelto sin antes calcular el valor de R^{-1}. Esta es una operación fácil si observamos que la matriz R es simétrica y mas aun que todas las diagonales consisten del mismo elemento. Este tipo de matriz es conocida como la matriz de Toeplitz y puede ser fácilmente invertida. De aquí , la solución para las ecuaciones matriciales, se pueden obtener rápidamente, por ejemplo usando el algoritmo recursivo de Levinson - Durbin (Markoul, 1975; Marple, 1987). La asumpción que la señal es infinita debe ser corregido, dado que en la practica la señal sólo es conocida para un intervalo finito; esto se realiza haciendo uso de una función window *w(n)*.

3.5 El algoritmo de Levinson - Durbin (L-D)

El algoritmo de Levinson - Durbin (L-D) es un algoritmo recursivo que es considerado muy eficiente computacionalmente dado que hace uso de las propiedades de R cuando determina los coeficientes del filtro. Este algoritmo se muestra mas adelante donde el filtro del orden es denotado con un superíndice $\{a_i^{(i)}\}$ para el filtro de orden j, y el error cuadrático medio de un filtro de orden j es denotado E_j en vez de $E[e_n^2]$. Cuando se aplica a un filtro de orden M, el algoritmo L-D calcula todo los filtros de orden

Algoritmo L – D

1 Set $E_0 = R_n(0), \quad i = 0$

mientras $(i < M)\{$

2. $i++$
3. *Calcular*:

$$k_i = \left[\sum_{j=1}^{i-1} a_j^{(i-1)} R_n(i-j+1).R_n(i-j+1) - R_n(i) \right] \Big/ E_{i-1}$$

4. *Fijar* $a_j^{(i)} = k_i$
5. *Calcular* $a_j^{(i)} = a_j^{(i-1)} + k_i a_{i-j}^{(i-1)}, \quad \forall\, j = 1,\ldots,i-1$
6. *Calcular* $E_i = (1 - k_i^2).E_{i-1}$

menores que M. Esto significa, que determina todo los filtros de orden N donde N = 1,...,M-1.

Durante el proceso de cálculo de los coeficientes del filtro $\{a_i\}$ también se generan un conjunto de coeficientes $\{k_i\}$, llamados coeficientes de reflexión o coeficientes de correlación parcial (PARCOR). Estos coeficientes son utilizados para resolver problemas potenciales en la transmisión de los coeficientes del filtro. La cuantificación de los coeficientes de transmisión pueden crear un problema mayor dado

que los errores de aproximación de los coeficientes del filtro pueden provocar inestabilidad en el filtro del tracto vocal y crear una señal de salida imprecisa. Este problema potencial es evitado cuantificando y transmitiendo los coeficientes de reflexión que son generados por el algoritmo de Levinson - Durbin. Estos coeficientes pueden usarse para reconstruir el conjunto de coeficientes $\{a_i\}$ y pueden garantizar un filtro estable si sus magnitudes son estrictamente menores que uno.

3.6 Cálculo de parámetros

La predicción lineal remueve la redundancia de la señal. Por lo tanto se puede usar efectivamente en la compresión de los datos a ser transmitidos. En las aplicaciones de telecomunicaciones tanto los parámetros de la predicción lineal como el valor residual necesitan ser cuantificados y transmitidos. Existe teóricamente muchas alternativas para representar los parámetros del modelo. La respuesta al impulso del filtro inverso $A(z)$, a_k, pueden ser cuantificados. Los coeficientes de la autocorrelación, coeficientes espectrales o coeficientes cepstrales así como también los polos en si podrían ser usados para la presentación del modelo, adicionalmente a los ya conocidos coeficientes de reflexión. Sin embargo a fin de cuantificar los parámetros, estos deberían ser capaces de mantener la estabilidad del filtro todos - polos después de la cuantificación y ellos deberían tener un ordenamiento natural. El ordenamiento natural significa que sobre el conjunto de parámetros, cada parámetro tendría su propia posición la cual no es intercambiable.

Una forma eficiente de presentación de la predicción lineal para fines de cuantificación es a través de los coeficientes de reflexión. Estos coeficientes pueden obtenerse a partir de los coeficientes a_i transformando la predicción en un filtro de Lattice. Esta recursión se define como sigue:

$$k_i = a_i^{(i)}, a_m^{(i-1)} = \frac{a_m^{(i)} + k_i a_{i-m}^{(i)}}{1 - k_i^2}, \quad m = 1,2,...(i-1) \ldots\ldots\ldots\ldots (14)$$

Otro punto de vista para los coeficientes de reflexión es la presentación de el error de predicción normalizado, proporcionado por el algoritmo de Durbin:

$$V_p = \prod_{i=1}^{p} (1 - k_i^2), \quad 1 \le i \le p \ldots\ldots\ldots\ldots (15)$$

El chequeo de estabilidad de los coeficientes de reflexión es fácil. Para un filtro estable las raíces del denominador, por ejemplo todos los polos, caen dentro del círculo unitario. Esto corresponde a $|k_i| < 1$ para los coeficientes de reflexión. La escala limitada de los coeficientes k_i, es beneficiosa para definir la escala de cuantificación: la robustez de los coeficientes se incrementa y se puede mantener la estabilidad después de la cuantificación. Las estructuras de Lattice que se muestran en las figuras 12 y 13, son también factibles de implementar debido a sus buenas características numéricas. Los errores en los coeficientes del filtro no se acumulan a medida que se incrementa el orden del filtro de Lattice:

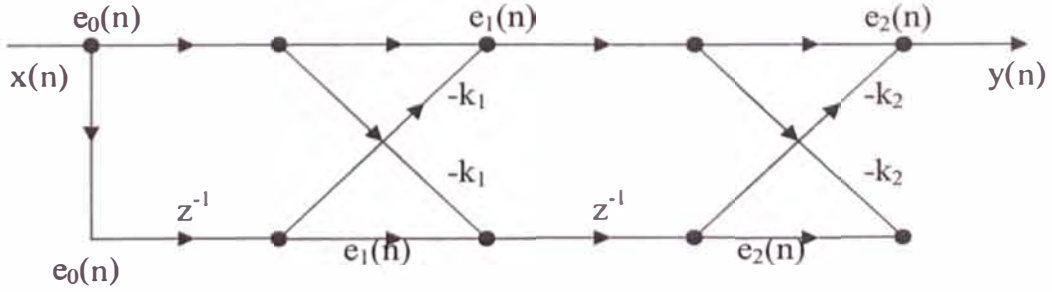

Figura 12: Filtro de Lattice de orden 2

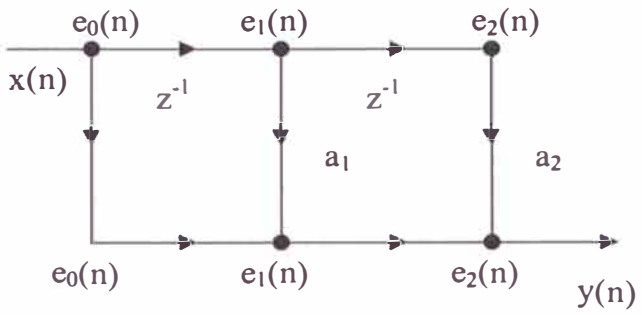

Figura 13: Forma directa del filtro de orden 2

CAPÍTULO IV

ANÁLISIS/CODIFICACIÓN LPC

4.1 Entrada del habla

Para su análisis la señal vocal será muestreada a una tasa de 8000 muestras por segundo. Esta señal luego es dividida en cuadros o tramas cada uno de los cuales son analizados y transmitidos hacia el receptor. Las 8000 muestras existentes en cada segundo de señal del habla son divididas en 160 muestras por trama, lo cual significa que cada segmento representara a 20 ms de la señal del habla a la entrada.

4.2 Pre - procesamiento de la señal vocal

Para mejorar la calidad de la señal vocal analizada se hace necesario realizar un pre procesamiento de la misma. Esto se realiza mediante el pre - énfasis de la señal y la aplicación de una ventana de Hamming.

4.3 Pre - énfasis

Un óptimo pre énfasis tiene la ventaja de garantizar que la salida espectral de la trama de datos sea tan plana posible (en el sentido de equiparar al máximo la medida espectral). Esta optimización ayuda a reducir considerablemente problemas numéricos en el análisis, pero implica realizar un poco más de cálculos. El valor óptimo del coeficiente de pre - énfasis en cada trama, está dada en función de la señal de entrada:

$$\mu = \frac{\sum_{n=0}^{N-2} S_n S_{n+1}}{\sum_{n=0}^{N-1} S_n^2} \qquad s_n = 0, n < 0 \ldots\ldots\ldots (16)$$

Para evitar estos cálculos, se escoge un valor constante pero cercano a la unidad a fin de que la estructura de los formantes mayores sea acentuada, por lo que es razonable escoger un valor entre 0.9 y 0.95. Por tanto esta etapa se realiza con el propósito de suavizar el espectro y reducir las inestabilidades de cálculo asociadas con las operaciones aritméticas de precisión finita. Además se usa para compensar la caída de -6 dB. que experimenta la señal al pasar a través del tracto vocal. Se usa un filtro digital de primer orden cuya función de transferencia es :

$$H(z) = 1 - az^{-1} \quad , a = 0.95 \ldots\ldots\ldots (17)$$

La ecuación diferencia será :

$$S(n) = s(n) - a * s(n-1) \ldots\ldots\ldots (18)$$

El diagrama de bloques se muestra en la figura 14.

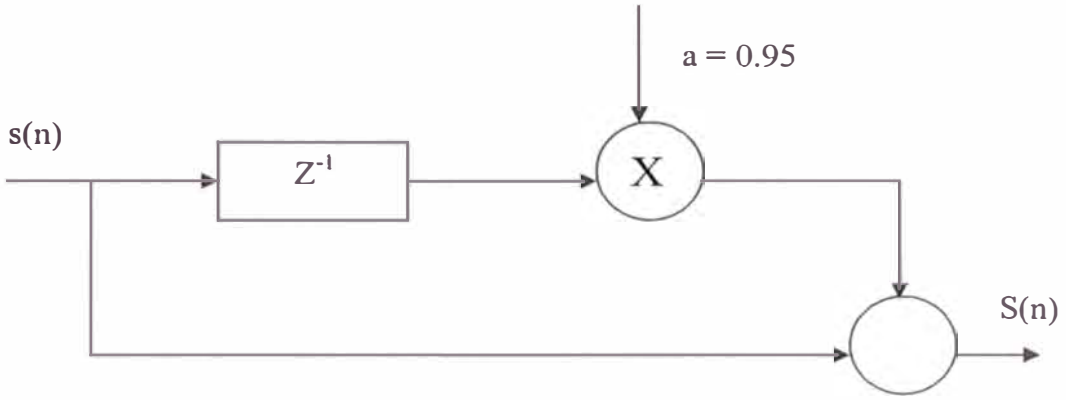

Figura 14: Diagrama de bloques del pre – énfasis.

4.4 Ventana de Hamming

La señal pre acentuada se toma cada 20ms por espacio de 30 ms. y se la somete a una ventana de Hamming, mostrada en la figura 15, con el objeto de suavizar la señal en los bordes de dicha ventana. Esta es la ventana que generalmente se usa para el análisis de señales de voz, y se define como :

$$W[nT] = 0.54 - 0.46Cos\left(\frac{2\pi n}{N}\right) \quad \text{.....} \quad (19)$$

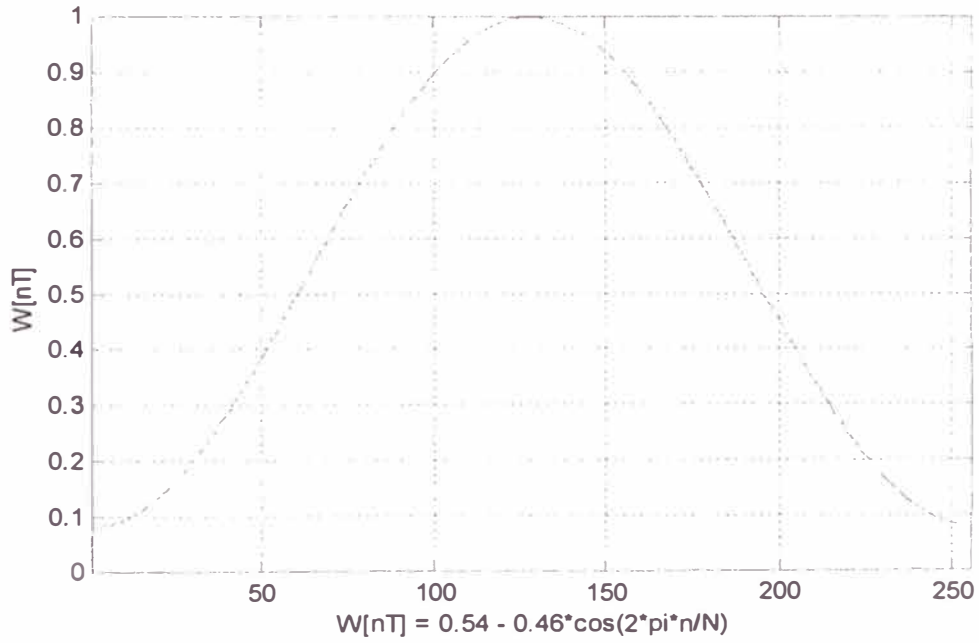

Figura 15: Ventana De Hamming

4.5 Determinación de los sonidos sonoros y no sonoros

La determinación si un segmento es sonoro o no sonoro es un parámetro importante en todos los sistemas de análisis de la señal vocal, ya que esta clasificación permitirá decidir que tipo de señal se deberá aplicar al filtro de síntesis de codificación.

Debemos recordar que los sonidos sonoros pueden considerarse como pulso periódicos. Estos sonidos contienen altos niveles de energía promedio lo cual significa que tienen amplitudes muy grandes. Además los sonidos sonoros se caracterizan por tener distintas frecuencias resonantes o formantes. Un ejemplo de un sonido sonoro se puede apreciar en la figura 16, la cual muestra la forma de onda de la vocal "e" en la palabra "test".

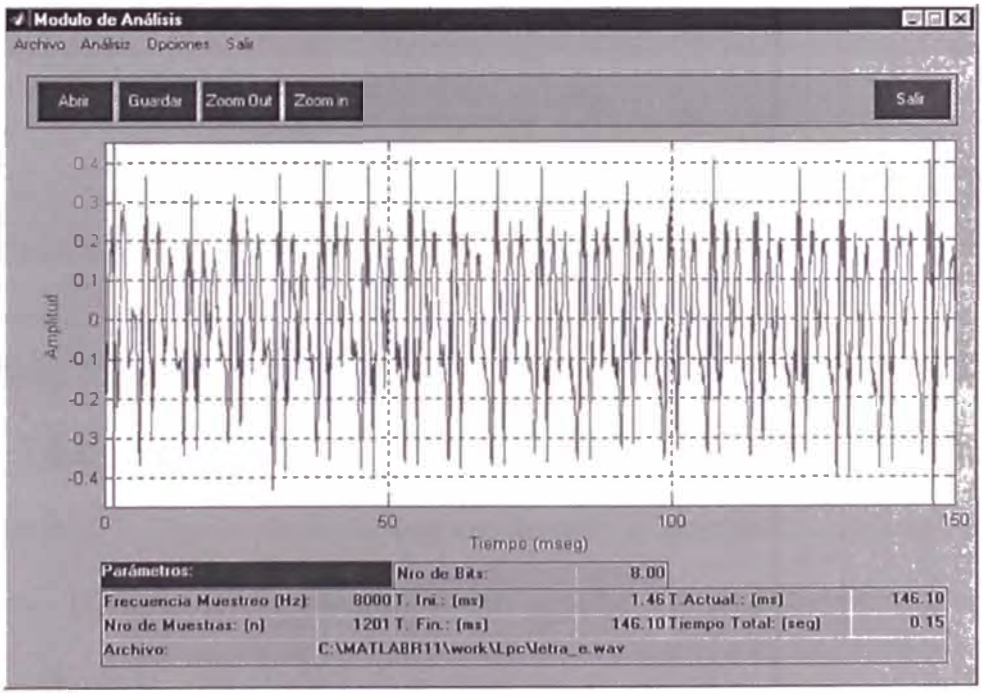

Figura 16: Señal vocal Sonora.

Los sonidos no sonoros son algunas consonantes y a menudo tienen forma de onda caótica y aleatoria. La figura 17 muestra que estos sonidos tienen menos energía y por consiguiente amplitudes más pequeñas que los sonidos sonoros. Además podemos apreciar que los sonidos no sonoros tienen mayores frecuencias que los sonidos sonoros.

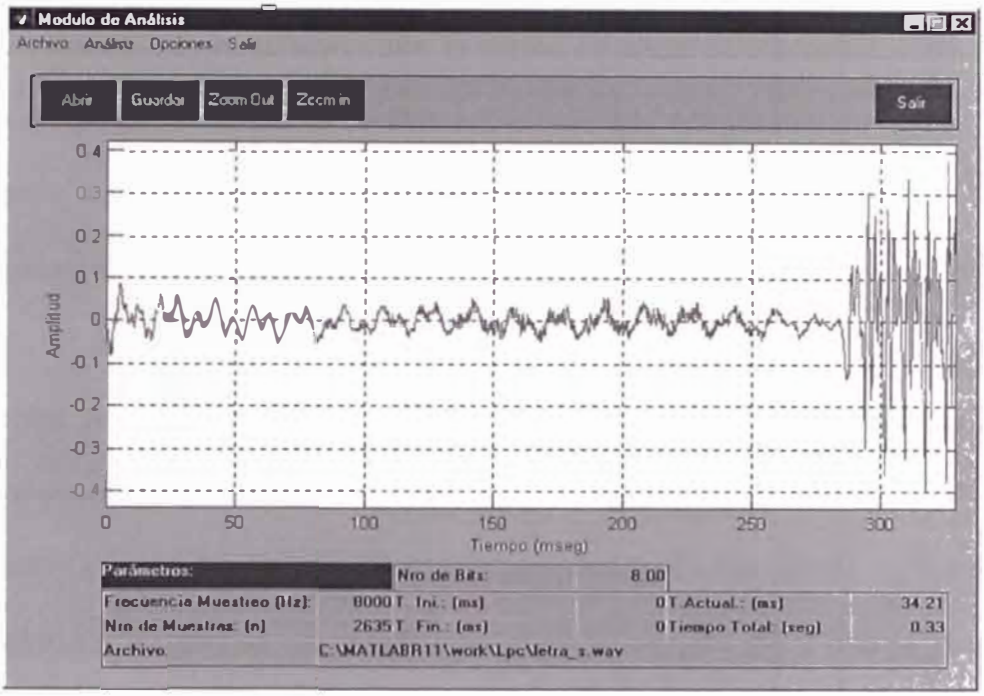

Figura 17: Señal vocal No Sonora.

Existen dos pasos en el proceso de determinación si un segmento del habla es sonoro o no sonoro. El primer paso consiste en observar la amplitud de la señal, también conocido como energía del segmento. Si los niveles de amplitud son grandes el segmento es clasificado como sonoro de lo contrario el segmento se considera como no sonoro. Esta determinación requiere de una noción pre concebida acerca del rango de valores de amplitud y niveles de energía asociados con los dos tipos de sonido.

A fin de determinar una clasificación para los sonidos que no pueden ser claramente identificados basados en el análisis de la amplitud, existe un segundo paso usado la distinción final del sonido. Este paso hace uso del hecho que los sonidos sonoros tienen grandes amplitudes, que los sonidos no sonoros tienen altas frecuencias y que los valores promedio de ambos tipos de señales son cercanos a cero. Estos tres hechos nos conducen a la conclusión que la forma de onda de los sonidos no sonoros deben cruzar el eje de coordenadas X más a menudo que las formas de ondas que los sonidos sonoros. Esto se puede apreciar claramente de las figuras mostradas. De esta manera, la determinación de las señales de vocales sonoras y no sonoras es determinada contabilizando el número de veces que una forma de onda cruza el eje X y luego comparando este valor con los rangos de valores normales para la mayoría de señales sonoras y no sonoras.

Un factor adicional que influye en la clasificación de los sonidos sonoros y no sonoros es debido a las tramas adyacentes. La clasificación de estas tramas vecinas es tomado en cuenta ya que es indeseable tener una trama no sonora en medio de tramas sonoras o viceversa.

Es importante entender que el sonido no siempre es producido de acuerdo al modelo LPC. Un ejemplo de esto ocurre cuando las tramas sonoras se mezclan con una gran cantidad de ruido y son algunas veces interpretados como sonidos no sonoros. Otro caso de un error de interpretación del modelo LPC ocurre en el caso de los sonidos conocidos como sonidos nasales. Durante la producción de estos sonidos la cavidad de la nariz destruye el concepto del tubo lineal ya que en este caso el tubo tendría una

ramificación. Este problema es a menudo ignorado en el modelo LPC pero si es tomado en cuenta en otros modelos.

Además es posible tener sonidos que son producidos de acuerdo al modelo LPC pero que no pueden ser clasificados como sonoros o no sonoros. Estos sonidos son una combinación de las formas de onda caóticas de los sonidos no sonoros y de las ondas periódicas de sonidos sonoros. Otro tipo de codificación de voz conocido como codificación de predicción lineal excitada por código (CELP) resuelve el problema de los sonidos que son combinaciones de los sonidos sonoros y no sonoros por medio de un libro de códigos estándar el cual contiene las señales problemáticas típicas. En el modelo LPC sólo se consideran dos tipos de señales por defecto usadas para excitar el filtro para producir señales sonoras o no sonoras en el decodificador. En el modelo CELP, el codificador o sintetizador compara una onda determinada con el libro de código y encuentra la entrada más parecida, esta entrada es enviada al decodificador la cual recibe el código y obtiene la señal correspondiente para excitar el filtro y producir la señal de salida en decir no usa las señales por defecto como en el caso del LPC.

4.6 Estimación del periodo PITCH

En el caso del método LPC, la determinación de un segmento si es sonoro o no sonoro no es toda la información que necesita el decodificador para reproducir exactamente la señal vocal. A fin de producir una buena señal en el decodificador también se necesita un atributo importante de la trama de la señal vocal que se esta analizando este tributo se conoce como el periodo pitch. Para las señales vocales, el pitch se debe entender como el periodo a la cual vibran las cuerdas vocales durante la producción del habla de los *sonidos sonoros*. Por lo tanto, el pitch sólo en necesario para la codificación de tramas que contengan *sonidos sonoros* más no para el caso de tramas *no sonoras* ya que estos últimos son producidos por el flujo de aire turbulento y no por la vibración de las cuerdas vocales.

La detección del pitch es un problema muy importante en el procesamiento de señales del habla. La codificación, compresión, análisis, síntesis y segmentación del habla, y el reconocimiento de palabras aisladas son muchos ejemplos de aplicaciones donde la estimación del pitch juega un papel importante.

Es computacionalmente intensivo el cálculo para determinar el pitch para una trama de la señal vocal determinada. Existen cientos de algoritmos, conocidos como PDAs (Picth Detection Algorithms) que se han propuesto para la determinación del pitch, cada uno tiene sus ventajas y desventajas, relacionados a la precisión para la determinación del picth o la dificultad de los cálculos computacionales necesarios para lograr su objetivo; a continuación se mencionan alguno de ellos.

4.6.1 Métodos de Determinación del Pitch

Entre algunos de los métodos importantes podemos mencionar los siguientes:

Análisis en el dominio del tiempo:

- El Método de Autocorrelación (1976).
- El Algoritmo de Gold Rabiner (1969).
- La Función Diferencia de Amplitud Media (AMDF).

Análisis en el dominio de la Frecuencia:

- El método Cepstrum (Noll 1964).
- Espectro del Producto Armónico. (Schroeder 1968).
- Método Heurístico de Chen. (Vhen 1998).

Otros:

- El método de Máxima Probabilidad.
- El método SIFT (Simple Inverse Filter, JD Markel 1972).
- Métodos de Redes Neuronales.

En el presente trabajo se hizo uso de los siguientes métodos:

El método de Autocorrelación.

El método Cepstrum.

El método SIFT.

Que se detallan a continuación.

a) Método de Autocorrelación

Este algoritmo hace uso de la propiedad de autocorrelación de una función periódica, $R_{xx}(k)$, la cual tendrá un valor máximo cuando k sea equivalente al periodo pitch.

La función de autocorrelación (ACF) de un segmento de voz se define como:

$$R_x = \sum_{m=0}^{N-1-k} [x(n+m)w'(n)][x(n+m+k)w'(k+m)] \quad \ldots\ldots \quad (20)$$

Una de las principales limitaciones de la autocorrelación de la señal de voz es que retiene mucha información de la señal vocal. La mayoría de los picos en el ACF pueden ser atribuidos a oscilaciones provocadas por el tracto vocal. Además si la ventana de análisis es demasiado corta comparada con el periodo pitch, podría ocurrir una falsa estimación del periodo pitch.

Así en aquellos puntos en donde los picos de la autocorrelación son mayores que aquellos debido a la periodicidad de la señal de excitación, el procedimiento simple de encontrar el pico más grande podría fallar.

b) Método de Autocorrelación Recortado y Centrado (CC-ACM)

Una mejora del método de autocorrelación (ACF) es el CC-ACM. Este pertenece al numeroso grupo de técnicas de "aplanamiento del espectro". En este caso la trama de la señal vocal antes de ser usada en el análisis, es preprocesada, pasando la señal a través de un recortador (clipper) , de esta forma se puede obtener una visión más clara de en la indicación del pitch.

El valor de recorte es determinado como el 60% del menor de los valores máximos correspondientes al primer y tercer tercio de la señal.

c) Método Cepstrum

El análisis cepstral de las señales del habla fue motivado originalmente por el deseo de separar (para los sonidos sonoros) el contenido de la señal debido a la excitación glotal de la señal debido al tracto vocal.

La excitación glotal es típicamente rica en contenido de armónicos, y su contenido es modulado por la respuesta del filtro del tracto vocal. EL análisis cepstral provee una forma de separar la excitación de la respuesta del filtro y por consiguiente facilitar la estimación de la frecuencia fundamental de excitación y las frecuencias resonantes del filtro. Se asume que las secuencias de las muestras sonoras del habla es el resultado de convolucionar la secuencia de excitación glotal *e[n]* con la respuesta discreta al impulso de tracto vocal *θ[n]*,

$$s[n] = e[n] * \theta[n] \quad \ldots\ldots \quad (21)$$

En el dominio de la frecuencia , la relación de convolución llega a ser una relación de multiplicación:

$$S(\omega) = e(\omega)\theta(\omega) \quad \ldots\ldots \quad (22)$$

Aun en el dominio de la frecuencia, no es fácil separar la señal de excitación de la respuesta del filtro. Sin embargo, de la relación conocida *logAB = logA + logB*, observamos que relación de multiplicación anterior puede convertirse en una relación de adición, entonces tomando el logaritmo.

$$\log|S(\omega)| = \log|E(\omega)| + \log|\Theta(\omega)| \quad \ldots\ldots \quad (23)$$

De la ecuación anterior, el real cepstrum de la señal s[n] se define como:

$$c[n] = \Im^{-1}_{DFTFT}\{\log|\Im\{s[n]\}|\} \quad \ldots\ldots\ldots\ldots \quad (24)$$

$$c[n] = \frac{1}{2\pi}\int_{\pi} \log|S(\omega)|e^{j\omega n}d\omega \quad \ldots\ldots\ldots\ldots \quad (25)$$

$$S(\omega) = \sum_{n=-\infty}^{\infty} s[n]e^{-jwn} \quad \ldots\ldots\ldots \quad (26)$$

d) Método SIFT

Al igual que los algoritmos de autocorrelación y el análisis cepstral, el método SIFT, tiene sus ventajas y desventajas tales como precisión, decisión de señales sonoras o no sonoras, carga computacional, etc. El algoritmo SIFT, es una técnica óptima para la estimación de la frecuencia fundamental de gran aplicación desde que fue propuesta en el año 1972.

La técnica de autocorrelación, que usa pocos periodos de pitch es adecuada y tiene un desempeño superior al seguimiento analógico el cual requiere una detección de picos sobre un rango dinámico > a 30 dB, ya que esta tiene un rango < a 10 dB, debido al hecho que la frecuencia fundamental queda dentro de la primera porción de la secuencia de autocorrelación. Pero esta técnica sufre de dos problemas.

1- Detección no trivial del pico.

2- Error debido a la señal glotal y de la onda senoidal del primer termino formante.

De otro lado el análisis cepstral usa el hecho $log\ ab = log\ a + log\ b$ resuelve muy bien el segundo problema usando el hecho que el segundo problema contribuye principalmente a los primeros pocos milisegundos de el cepstrum. Pero haciendo un reemplazo de los pocos milisegundos con ceros hace que se pierda el valioso punto de referencia o de normalización y en vista que la amplitud real de la frecuencia fundamental no es específicamente una función de la frecuencia fundamental entonces no se puede determinar si la señal es sonora o no sonora.

La técnica de filtraje inverso combina las características de los métodos de autocorrelación y el análisis cepstral. Y resuelve los problemas pero tiene una mayor carga computacional. El algoritmo SIFT además de eliminar esta desventaja, tiene algunas otras diferencias con la técnica de filtraje inverso.

1- Los picos son ensanchados.

2- Las términos de altas frecuencias son más suprimidas.

3- Los armónicos del periodo pitch son más aparentes.

4- La relación Pico de la Señal / Pico del Ruido No Deseado es mayor.

1.1

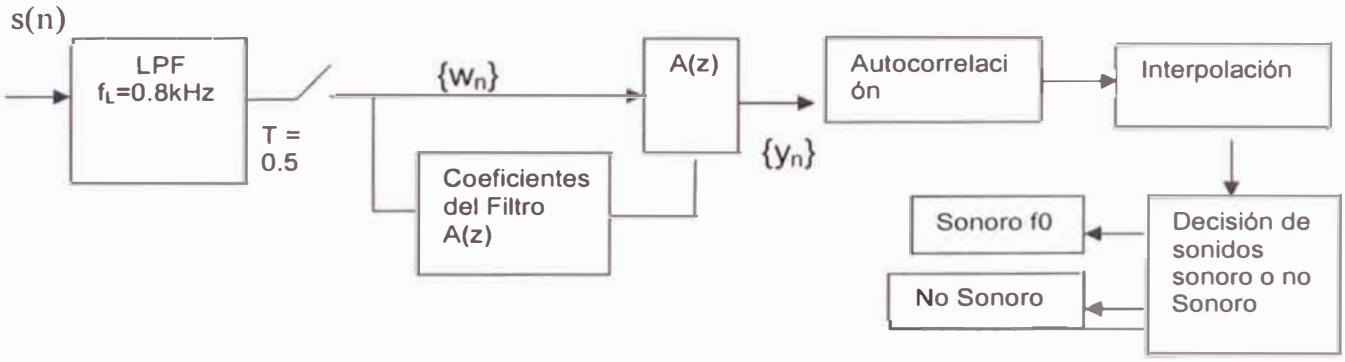

Figura 17: Diagrama de bloque del algoritmo SIFT.

A fin de reducir la cantidad de cálculos necesarios, el algoritmo realiza una decimación de la señal, como se muestra en la figura 17. Cuando se escribió el algoritmo se utilizó una frecuencia de muestreo de 10 KHz y una decimación de 5:1 lo cual disminuye la frecuencia de muestreo a 2 KHz. A fin de prevenir los efectos de aliasing se realiza un prefiltraje de la señal. Este filtro es un filtro Chebyshev de 3 polos y de 2 dB de Riple. Después de la decimación se realiza una autocorrelación para una ventana de 32 ms (N=64). Luego este número es usado para calcular los coeficientes del filtro inverso A(z) , luego el filtro inverso es aplicado al señal decimada. Lo cual nos da la señal $\{y_n\}$, luego la autocorrelación de $\{y_n\}$ nos da la señal deseada. Después de una pequeña interpolación de la región pico se realiza la determinación de si el sonido es sonoro o no sonoro. Si el segmento es no sonoro entonces f_0 es 0.

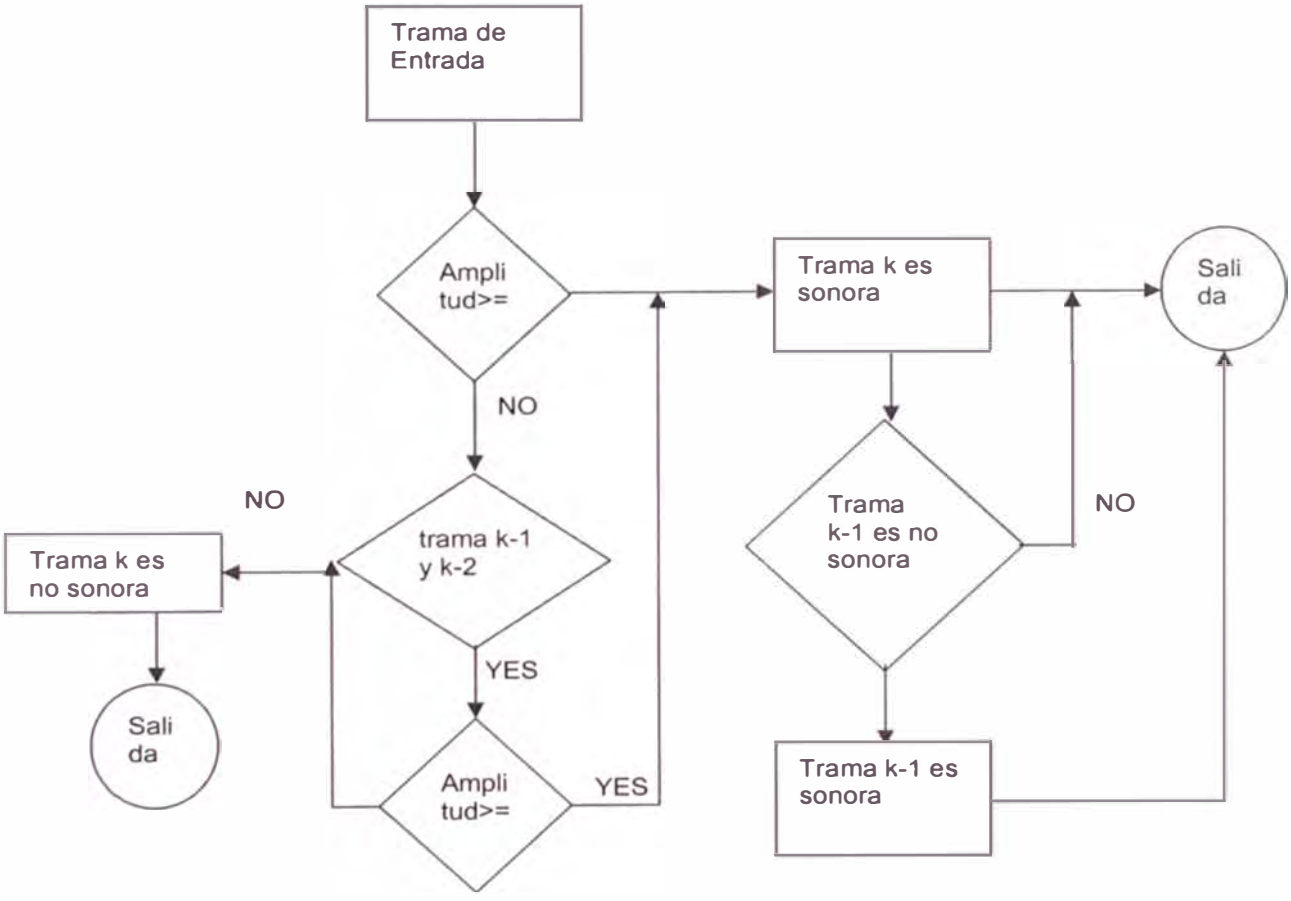

Figura 18: Algoritmo de decisión de sonido sonoro o no sonoro.

El método SIFT, no garantiza un análisis libre de error y puede tener problemas de detección la transición de un intervalo sonoro a no sonoro. No obstante de estos problemas el método SIFT es muy usado a menudo debido a las siguientes características

1- La decisión entre una señal sonora y no sonora es muy simple, como se muestra en la figura 18.

2- La implementación requiere sólo cuatro operaciones aritméticas.

3- Es computacionalmente muy eficiente.

4-No existe una diferencia perceptual respecto al análisis cepstral.

CAPÍTULO V

TRANSMISIÓN DE PARÁMETROS

Cuando la señal vocal no esta comprimida, es usualmente transmitida a una tasa de 64000 bits/segundo usando 8 bits/muestra y a una frecuencia de muestreo de 8 kHz. El método LPC reduce esta tasa a 2400 bits/segundo esto se logra dividiendo la señal del habla en tramas y luego enviando el periodo pitch, información si es una señal sonora o no sonora y los coeficientes del filtro que representa el tracto vocal de cada segmento.

La señal de entrada usada por el filtro en el extremo receptor es determinado a través de la clasificación del segmento del habla como sonoro o no sonoro, y por medio de la frecuencia pitch, como se muestra en la figura 19. El codificador envía un solo bit para indicar si la trama que se esta analizando es sonoro o no sonoro. Además el periodo pitch es cuantificado usando un cuantificador log-companded mapeado a uno de 60 valores posibles. Se usan 6 bits para representar el periodo pitch.

Si el segmento contiene una señal del habla sonora entonces se utiliza un filtro de orden 10. Esto significa que se necesitan 11 valores para representarlo: 10 coeficientes de reflección y la ganancia. Si el segmento contiene una señal no sonora entonces se usa un filtro de orden 4. Esto quiere decir que se necesitaran 5 valores: 4 coeficientes de reflexión y la ganancia. Los coeficientes de reflexión se denotan por K_n donde $1 \leq n \leq 10$ para los filtros de señales sonoras y $1 \leq n \leq 4$ para filtros de señales no sonoras.

El único problema que ocurre cuando se transmite los parámetros del filtro del tracto vocal es que los coeficientes de reflexión son muy suceptibles a errores cuando su magnitud es cercana a uno. En general es muy probable que los primeros coeficientes de

reflexión k_1 y k_2 tengan este problema. Para eliminar este problema, el método LPC usa una cuantificación no uniforme para k_1 y k_2. Para tal efecto, cada coeficiente es utilizado para generar un nuevo coeficiente, g_i de la forma:

$$g_i = \frac{1+k_i}{1-k_i}$$

Estos coeficientes nuevos, g_1 y g_2, son luego cuantificados usando un cuantificador uniforme de 5 bits.

Todos los demás coeficientes son cuantificados usando cuantificadores uniformes, k_3 y k_4 son cuantificados usando una cuantificación uniforme de 5 bits. Para el caso de señales sonoras los coeficientes k_5 y k_8 son cuantificados usando un cuantificador uniforme de 4 bits mientras para que k_9 se usa un cuantificador uniforme de 3 bits y k_{10} usa un cuantificador de 2 bits. Para los segmentos de señales no sonoras los bits usados para almacenar los coeficientes de reflexión de la señales sonoras, son usados para protección de errores. Esto significa que los segmentos no sonoros no omiten los bits k_5 hasta k_6, por lo tanto usan la misma cantidad de espacio que los segmentos no sonoros. Esto también significa que la tasa de bits no disminuye por debajo de 2400 bits/segundo aun si se enviara sólo información de segmentos no sonoros. La variación de la tasa de bits no es buena en la transmisión del habla dado que la mayoría de señales de habla son transmitidas sobre líneas compartidas donde es importante saber cuanto de la línea se necesitará.

Una vez que los coeficientes de reflexión han sido cuantificados, la ganancia G es el único parámetro que quedaría pendiente. La ganancia es calculada usando el valor

de la raíz cuadrada del valor medio al cuadrado (root mean square), luego esta ganancia es cuantificada usando un cuantificador de 5 bit log - companded.

El número total de bits requeridos por cada trama es de 54 bits. Recordar que la señal de habla a la entrada es muestreada a una tasa de 8000 muestras por segundo y que las 8000 muestras de cada segundo de señal son divididas en segmentos de 180 muestras cada uno. Esto significa que existen aproximadamente 44.4 cuadros o tramas por segmento por segundo y por consiguiente la tasa de bit es 2400 bits/segundo.

Cantidad de Bits	Descripción
1	Sonoro/no sonoro
6	Periodo Pitch (60 valores)
10	K_1 y k_2 (5 c/u)
10	K_3 y k_4 (5 c/u)
16	K_5, k_6, k_7 y k_8 (4 c/u)
3	K_9
2	K_{10}
5	Ganancia G
1	Sincronización
54	Total de Bits por cuadro

$$Frecuencia\ de\ muestreo = 8000\ muestra/segundo$$

$$Muestras\ por\ segmento = 180\ muestras\ por\ segmento$$

$$Numero\ de\ segmentos = \frac{Frecuencia\ de\ Muestreo}{Muestras\ por\ Segmentos} = \frac{8000\ muestras/seg}{180\ muestras/seg}$$

$$Numero\ de\ segmentos = 44.4444\ segmento/segundo$$

$$Tamaño\ de\ Segmento = 54\ bits/segmento$$

$$Tasa\ de\ Bits = Tamaño\ de\ Segmento * Número\ de\ Segmentos$$

$$= (54\ bits/segmento) * (44.44\ segmentos/segundo)$$

$$= 2400\ bits/segundo$$

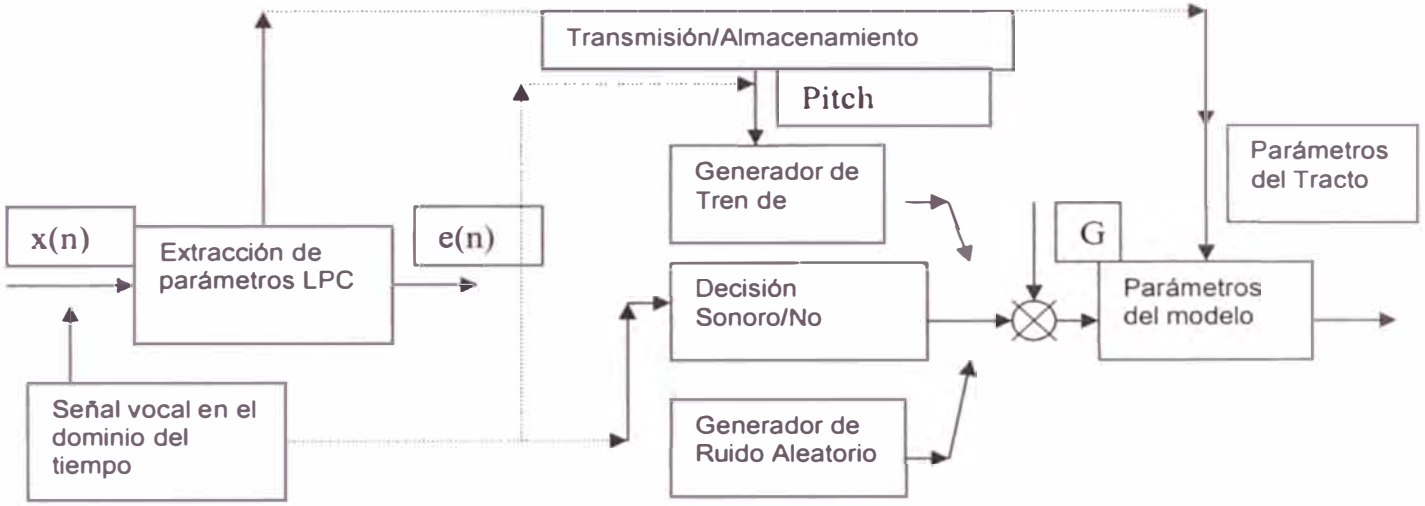

Figura 19: Diagrama de bloque del modelo de envío y recepción de los parámetros LPC.

CAPÍTULO VI

DECODIFICACIÓN / SÍNTESIS LPC

El proceso de decodificación de una secuencia de muestras de habla de un segmento es un proceso inverso al de codificación. Cada segmento es decodificado individualmente y la secuencia de segmentos de sonidos reproducidos son unidos nuevamente para representar la señal de habla completa. La decodificación o síntesis de una trama de la señal vocal es basado en los 54 bits de información que son transmitidos desde el codificador.

La señal de habla es clasificada como sonora o no sonora basado en el bit de enviado por el codificador. El decodificador necesita saber que tipo de señal contiene el segmento de señal a fin de determinar que tipo de señal de excitación le proporcionará al filtro LPC. A diferencia de otros algoritmos de compresión del habla como el CELP el cual tiene un libro de códigos (codebook) de posibles señales de excitación, el método LPC sólo contiene dos tipo de señales posibles. Para el caso de segmentos sonoros, se usa un pulso como señal de excitación. Este pulso consta de 40 muestras y esta localmente almacenado en el decodificador. Para el caso de segmentos no sonoros se utiliza ruido blanco proporcionado por un generador de números pseudo aleatorio.

Para el caso de los segmentos sonoros el periodo pitch es usado para determina4r si las 40 muestras necesitan ser truncadas o extendidas. Si el pulso necesita ser extendido esta es completado con ceros. Esta combinación de determinación del sonido sonoro o

no sonoro y el periodo pitch son los únicos parámetros necesarios para producir la señal de excitación.

Cada trama de la señal vocal tiene un filtro LPC diferente que es producido eventualmente usando los coeficientes de reflexión y la ganancia que son recibidas desde el codificador. Para el caso del filtro de señales sonoras se usan 10 coeficientes de reflexión y 4 para el caso de señales no sonoras. Estos coeficientes de reflexión son usados para generar los coeficientes del tracto vocal los cuales son usados para crear el filtro.

El paso final el proceso de decodificación es pasar la señal de excitación a través del filtro producido para sintetizar las señal de habla.

CAPÍTULO VII

APLICACIONES DEL LPC

En general, el uso más común de la compresión del habla es en los sistemas de telefonía. De hecho, gran cantidad de la tecnología de compresión del habla fue desarrollado por las compañías de teléfono. La codificación predictiva lineal sólo tiene aplicación en el área de telefonía segura, debido a su baja tasa de bits. Los sistemas de telefonía segura requieren bajas tasas de bits ya que primero son digitalizadas, luego encriptadas y transmitidas. Estos sistemas tienen como primer objetivo reducir la tasa de bits tanto como sea posible, mientras mantienen un nivel de calidad del habla que es aún entendible. Otros estándares como el de los celulares digitales y el de las redes de telefonía internacional tiene mayores niveles de calidad y por ende una mayor tasa de bits, en estos casos, la comprensión del mensaje no es suficiente, sino también se debe tener la capacidad de reconocer al interlocutor.

Un segundo campo de aplicación de LPC es el de la conversión de texto a voz. En este tipo de síntesis la señal de voz es generado directamente desde un texto. Dado que la síntesis LPC implica la generación de voz a partir del modelo del tracto vocal, entonces provee un método perfecto para genera voz a partir de un texto.

Futuras aplicaciones del LPC y otros esquemas de compresión serán los sistemas de correo por voz, máquinas de respuesta automática, y aplicaciones multimedia. La mayoría de aplicaciones multimedia, a diferencia de las aplicaciones telefónicas, implican comunicación en un solo sentido y almacenamiento de datos. Un ejemplo de

una aplicación multimedia que podría implicar voz, es que las anotaciones de voz acerca de un documento de texto sean salvados en el mismo documento. El método de compresión usado dependerá del nivel de calidad deseado y de la cantidad de almacenamiento disponible . La codificación predictiva lineal provee un método favorable de compresión para aplicaciones multimedia dado que necesita el menor espacio de almacenamiento como resultado de su baja tasa de bits.

CONCLUSIONES Y RECOMENDACIONES

La codificación predictiva lineal es una técnica de análisis y síntesis para la compresión de la señal vocal con pérdidas, que intenta modelar el mecanismo de producción de la señal vocal humana, en vez de transmitir un estimado de la misma. La codificación predictiva lineal logra una tasa de bits de 2400 bits/segundo lo cual lo hace ideal para el uso en sistemas de telefonía segura. Los sistemas de telefonía segura son aquellos en donde lo más importante es la comprensión del mensaje transmitido, en vez de la calidad del mismo. EL método LPC logra una baja tasa de bits, a condición de producir un mensaje con sonido sintético, e incluso tiene algunas dificultades con ciertos sonidos.

Los codificadores LPC, dividen la señal en cuadros o tramas para su respectivo análisis, y luego se extraen parámetros para cada trama analizada los cuales son enviados al decodificador. Entre los parámetros enviados por el decodificador tenemos información si la trama es sonora o no sonora, lo cual se usa para decidir que tipo de señal de excitación se debe usar en el decodificador, el periodo pitch de las tramas sonoras, y también se envía los coeficientes que modelan el tracto vocal, los cuales son usados en el decodificador para reconstruir el filtro al cual se le aplicará la señal de excitación para sintetizar la señal vocal transmitida.

Se ha desarrollado un pequeño sistema para realizar un estudio más exahustivo de la señal vocal, la cual quedará como una herramienta para los investigadores que deseen continuar con el estudio de la misma.

Se ha podido observar, que el algoritmo de predicción lineal utilizado permite obtener una señal sintetizada similar a la señal original, donde apreciamos que la naturalidad de la señal vocal está en función del cálculo de la frecuencia fundamental de cada trama, por lo que como trabajos futuros recomiendo profundizar más la investigación sobre el uso de ventanas adaptivas, y del estudio de la tasa de variación del pitch para cada trama analizada, otros de los puntos que pude notar es que los métodos del cálculo del pitch están basados en la comparación de umbrales, y sería interesante analizar el efecto de usar umbrales adaptivos en el cálculo del pitch. Por último sería recomendable migrar el desarrollo actual hecho en matlab hacia nuevos lenguajes de programación visuales, y mejor aún lograr su implementación en hardware a través del uso de DSP's y de esta forma avanzar en el desarrollo de un sistema integral para el análisis de la señal vocal, la cual además de ser utilizada como una herramienta para la enseñanza del procesamiento del habla, podría servir a muchos profesionales del área.

ANEXO I

GRÁFICOS DE LOS RESULTADOS DEL ANÁLISIS DE LA SEÑAL.

A continuación se muestra los resultados obtenidos del Análisis de la señal vocal correspondiente al texto, "Codificación de Voz usando Predicción Lineal" usando el algoritmo de predicción lineal desarrollado en Matlab.

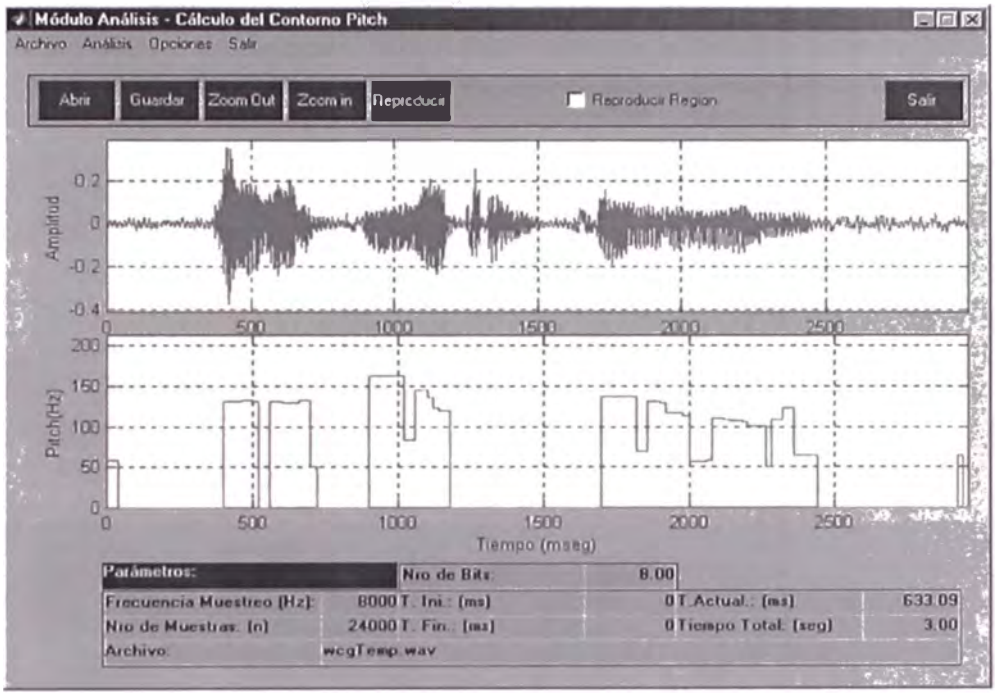

Figura 20: Muestra el contorno F0 usando el método de autocorrelación.

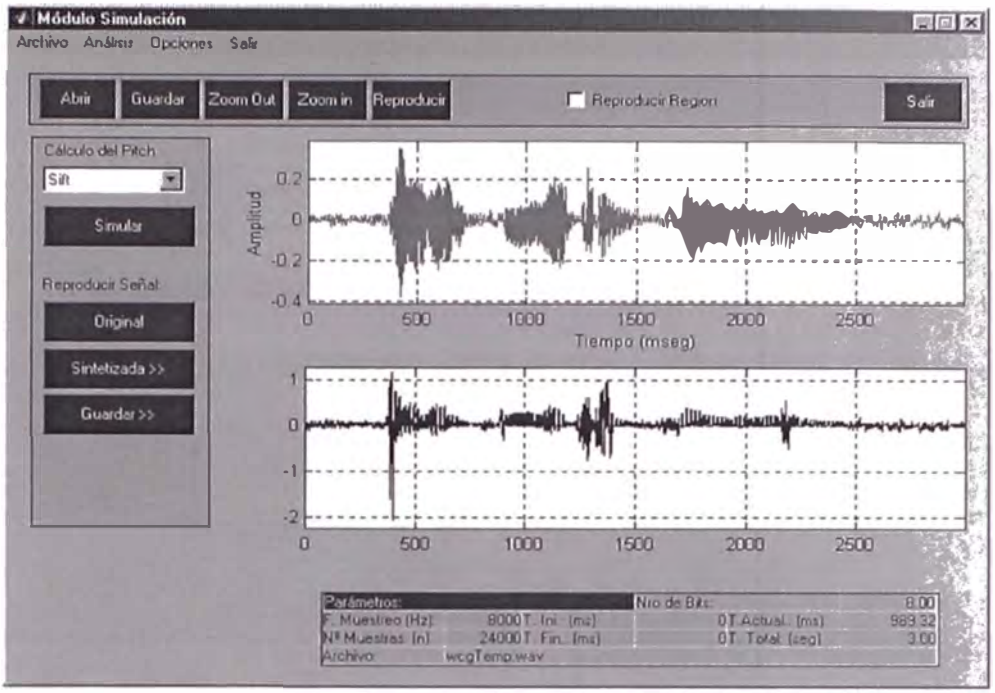

Figura 21: Muestra el contorno F0 usando el método de cepstrum.

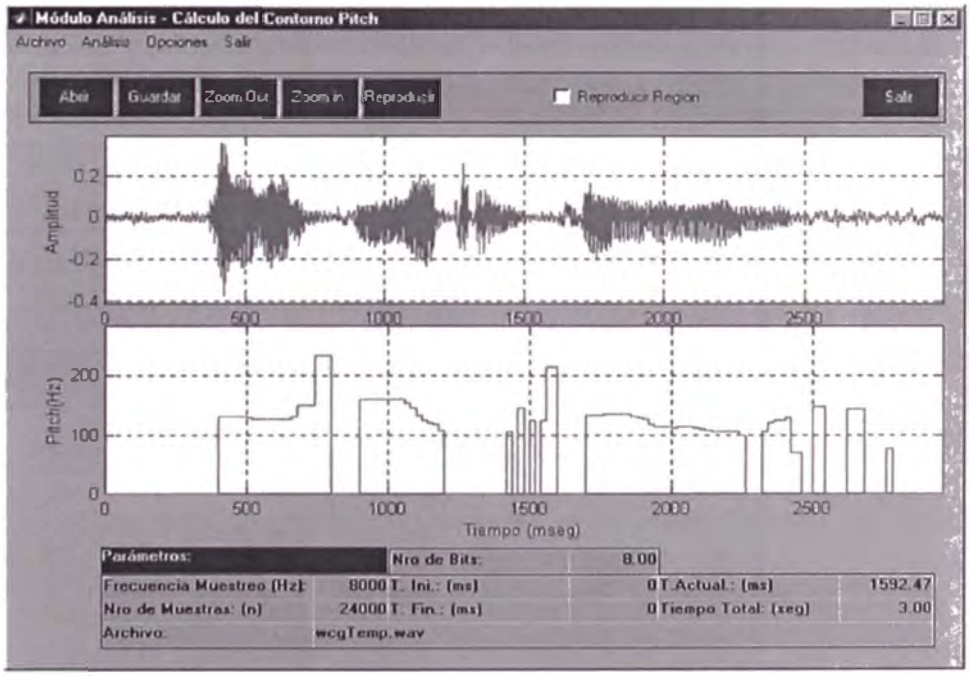

Figura 22: Muestra el contorno F0 usando el método de SIFT.

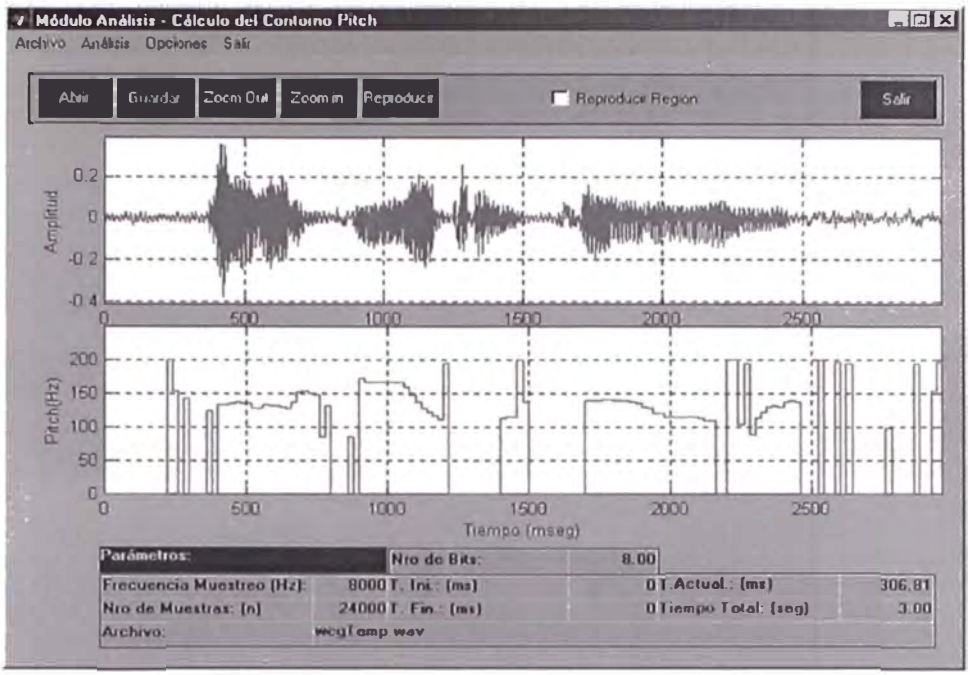

Figura 23: Muestra la señal original y sintetizada.

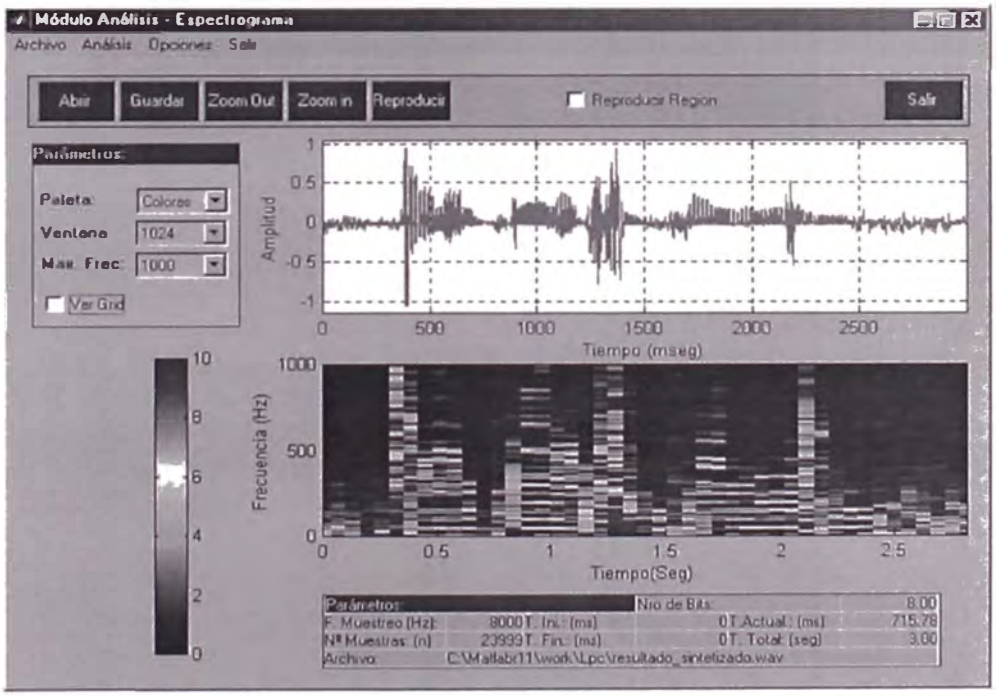

Figura 24: Muestra el espectrograma de la señal sintetizada

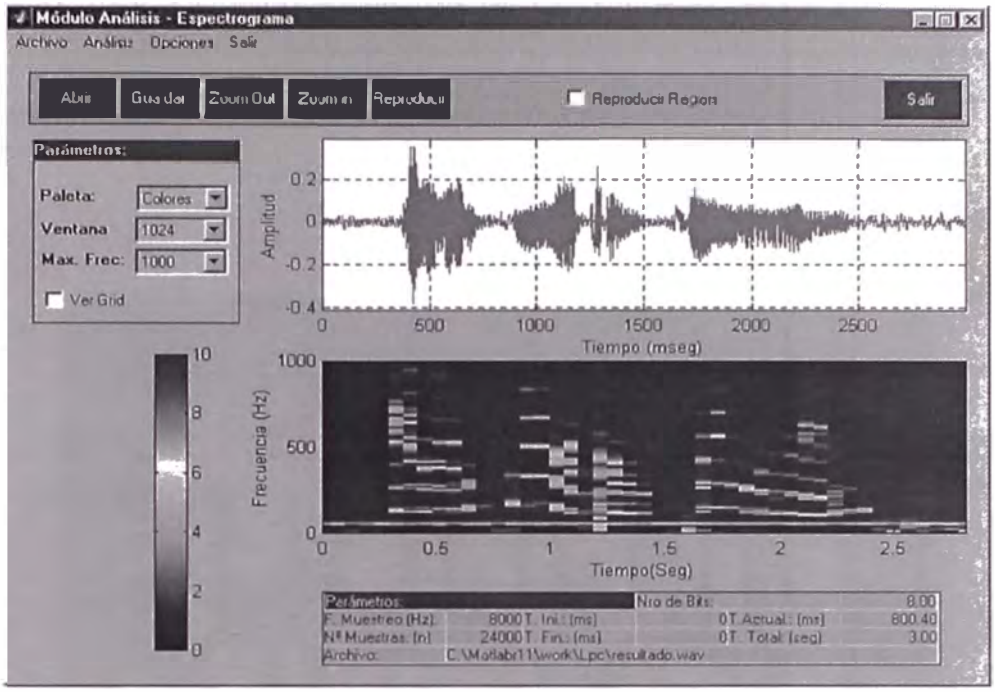

Figura 25: Muestra el espectrograma de la señal real.

A continuación mostramos en las figuras 26 hasta 32, el proceso del cálculo del pitch de una trama de 20 ms, usando el método SIFT:

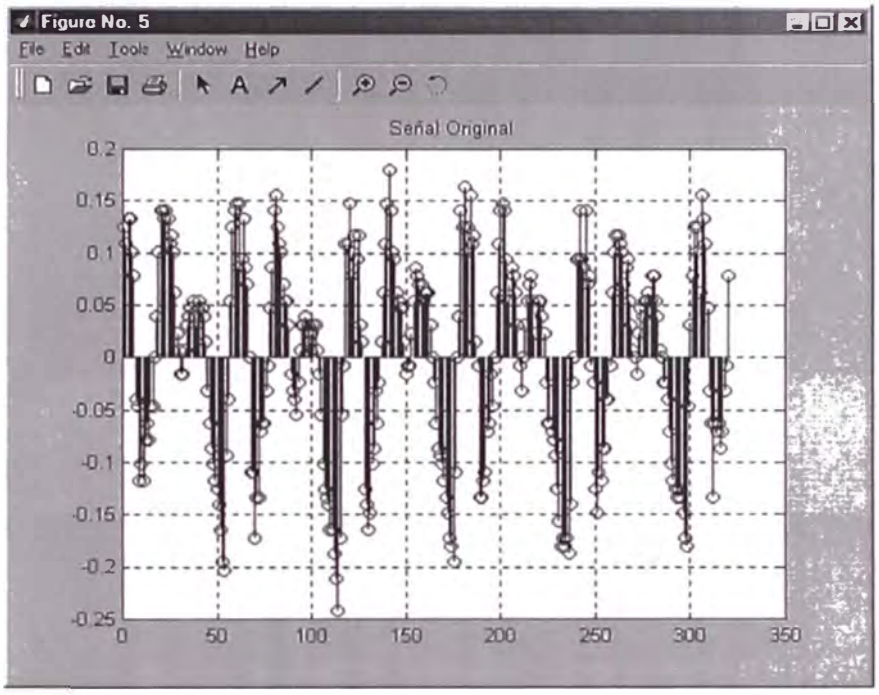

Figura 26: Señal Original

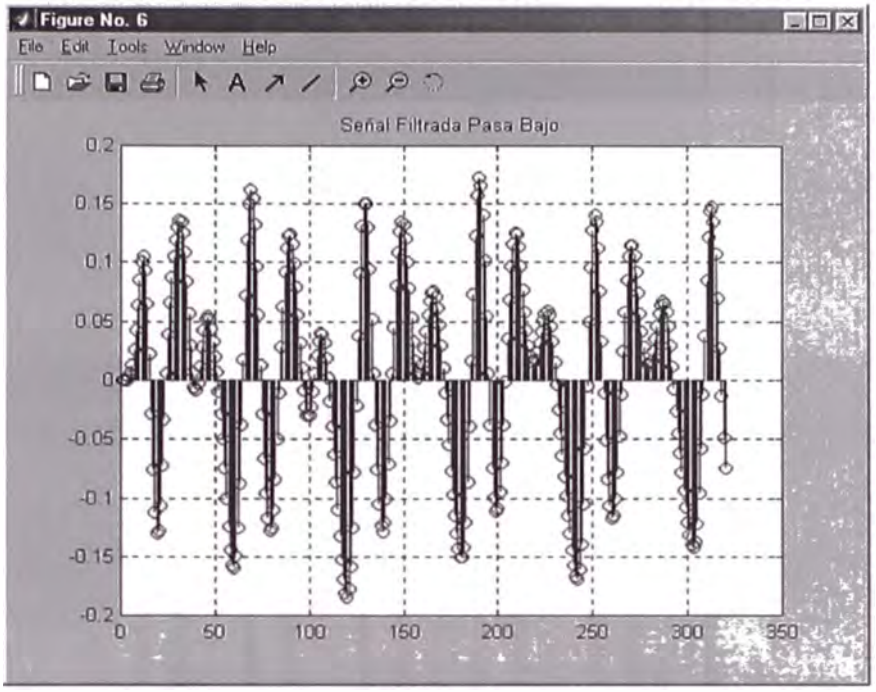

Figura 27: Señal Filtrada.

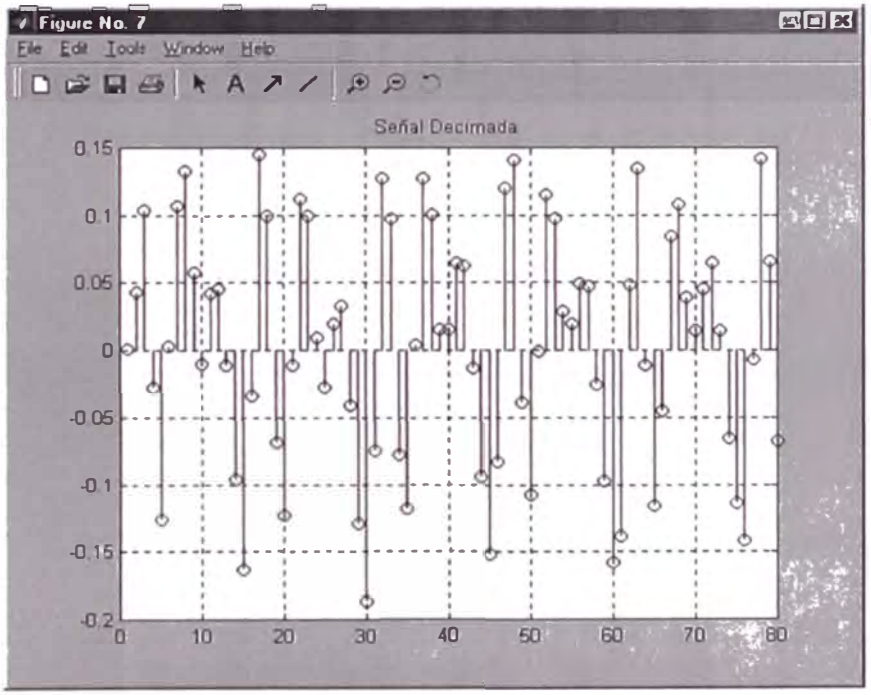

Figura 28: Señal Decimada.

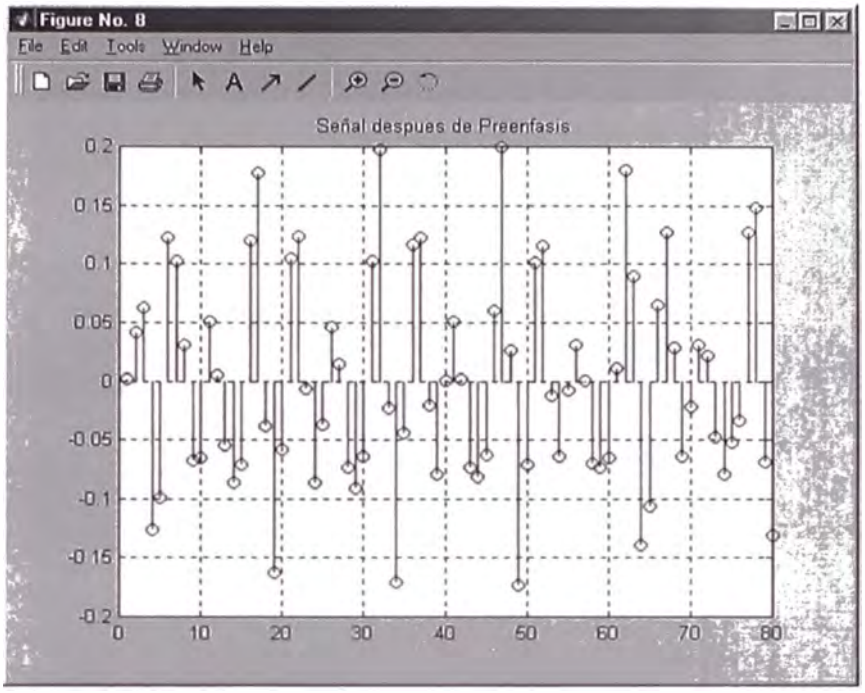

Figura 29: Señal después del preénfasis.

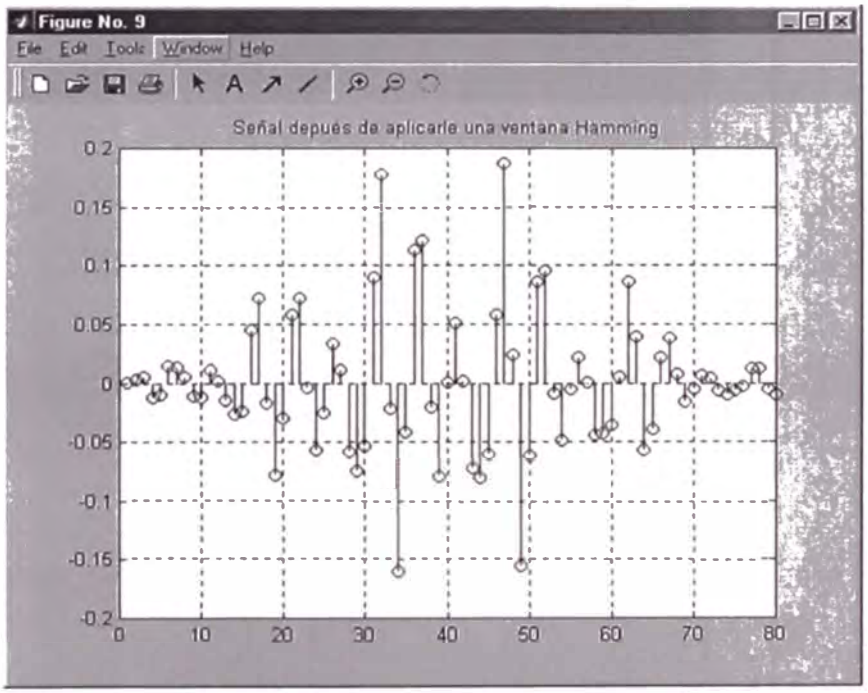

Figura 30: Señal después de la ventana Hamming.

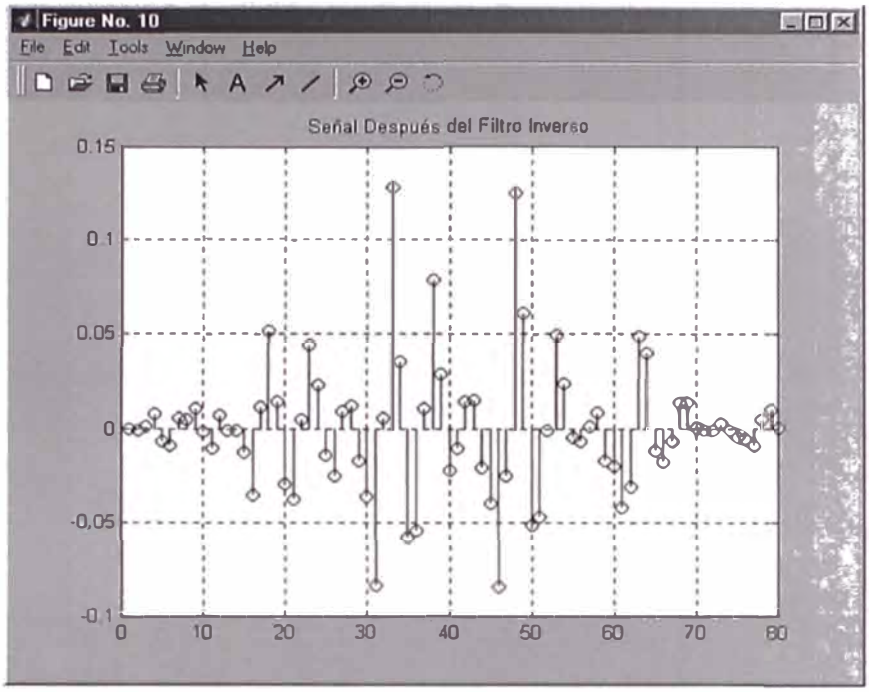

Figura 31: Señal después de ser pasada por el filtro inverso.

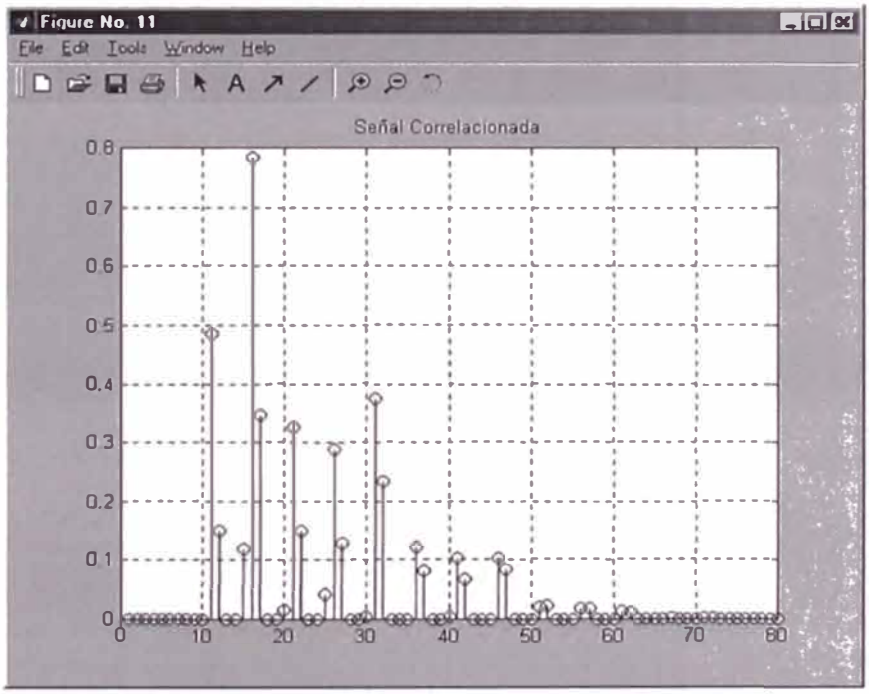

Figura 31: Señal Correlacionada.

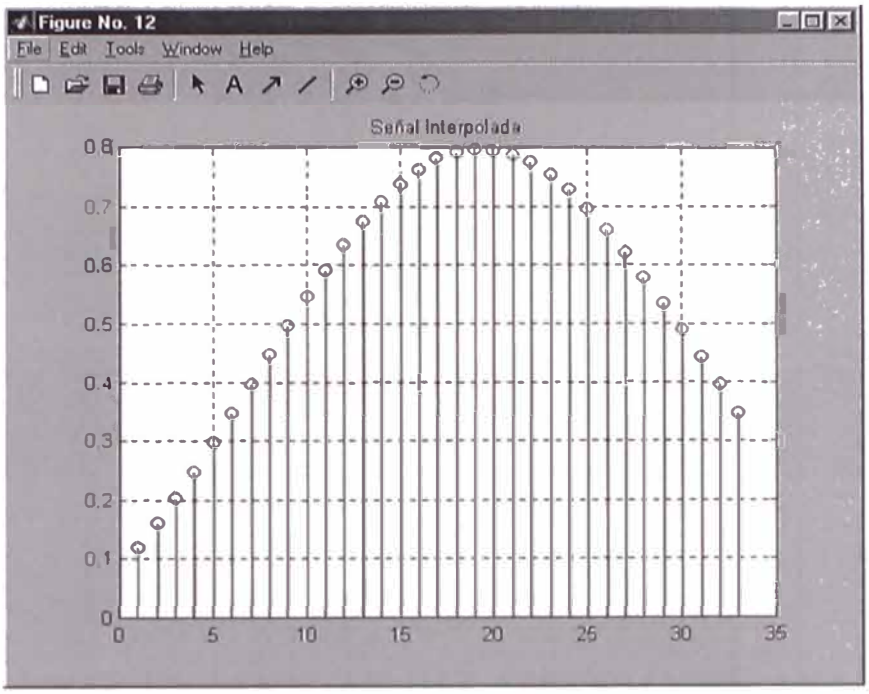

Figura 32: Señal Interpolada.

ANEXO II

CÓDIGO FUENTE EN MATLAB DEL SISTEMA DE ANÁLISIS DE VOZ.

```
%*************************************************************************
**
%                          wcgAdquisicionDeVoz.m
%*************************************************************************
**

%*************************************************************************
**
%        Tesis:       Análisis LPC
%        Autor:       William Castro Grijalva
%        Función:     Esta función controla las funciones de
%                     adquisición (reproducción y grabación) del habla
%                     desde la  tarjeta de sonido.
%*************************************************************************
**
function wcgAdquisicionDeVoz(hFig, sAction);

   switch sAction
   case 'reproducir_todo'
      ReproducirTodo(hFig)
   case 'reproducir_region'
      ReproducirRegion(hFig)
   case 'grabar'
      GrabarSonido(hFig)
   end

%*************************************************************************
**
%        Función: Reproducir la señal completa
%*************************************************************************
**
function ReproducirTodo(hFig);

   %Obtiene los datos que se van a analizar
   hAxes = findobj(hFig, 'Tag','axeAnalisis');
   hAxesLine = findobj(hAxes, 'Tag','axeLinea')
   XData = get(hAxesLine, 'XData')
   YData = get(hAxesLine, 'YData');

   if length(YData) == 0
      return;
   end

   %Obtiene => [iFrecMuestreo, iNroMuestras ,dNroSegundos]
   Parametros = get(hAxes,'UserData');

   %Reproduce el archivo
   sound(YData, Parametros(1))
```

```
%**************************************************************************
**
%        Función: Reproducir solo la region que se ha seleccionado
%**************************************************************************
**
function ReproducirRegion(hFig);

   %Obtiene los limites de las barras  , que se muestran en los labels
   hTime = findobj(hFig, 'Tag', 'txtTimIni');
   iTimeIni = get(hTime, 'UserData');

   hTime = findobj(hFig, 'Tag', 'txtTimFin');
   iTimeFin = get(hTime, 'UserData');

   %Si no se ha seleccionado un rango no hace nada...
   if iTimeIni == iTimeFin
      return
   end

   %Obtiene los datos que se van a analizar ...
   hAxes = findobj(hFig, 'Tag','axeAnalisis');
   hAxesLine = findobj(hAxes, 'Tag','axeLinea');
   XData = get(hAxesLine, 'XData');
   YData = get(hAxesLine, 'YData');

   if length(YData) == 0
      return;
   end

   Parametros = get(hAxes,'UserData'); %[iFrecMuestreo, iNroMuestras
   ,dNroSegundos]

   iFrecMuestreo = Parametros(1);

   iMuestrasIni = round(iTimeIni/1000*iFrecMuestreo);
   iMuestrasFin = round(iTimeFin/1000*iFrecMuestreo);

   YRegion = YData(iMuestrasIni:iMuestrasFin);

   %Reproduce el archivo
   sound(YRegion, iFrecMuestreo);

%**************************************************************************
**
%        Función: Grabar la voz usando la tarjeta de sonido
%**************************************************************************
**
function GrabarSonido(hFig);

   [iFs, iNroBits, iSeg] = ObtenerParametrosDeGrabacion(hFig);

   if iFs == -1
      msgbox 'Falta asignar la frecuencia de muestreo ...';
```

```
      return;
   end
   if iNroBits == -1
      msgbox 'Falta asignar el número de bits ...';
      return;
   end
   if iSeg == -1
      msgbox 'Falta asignar el tiempo de grabación ...';
      return;
   end

   %Graba la señal
   YData = wavrecord(iSeg*iFs, iFs, 1);

   %Guarda la grabacion a una rchivo temporal ...
   wavwrite(YData, iFs, iNroBits, 'wcgTemp.wav');

   %Recupera el archivo temporal
   aParametro.hFig = hFig;
   aParametro.sArchivo = 'wcgTemp.wav';
   wcgArchivo('LeerDeRuta', aParametro);

%************************************************************************
**
%        Función: Obtener los parametros de grabación
%        elegidos por el usuario
%************************************************************************
**
function [iFs, iNroBits, iSeg] = ObtenerParametrosDeGrabacion(hFig);

   iFs = -1;
   iNroBits = -1;
   iSeg = -1;

   hFs = findobj(hFig, 'Tag','cmbFs');
   iIndex = get(hFs, 'Value');
   sString = cellstr(get(hFs, 'String'));
   iFs = str2num(char(sString(iIndex)));
   if isempty(iFs)
      iFs = -1;
      return;
   end

   hNroBits = findobj(hFig, 'Tag','cmbNroBits');
   iIndex = get(hNroBits, 'Value');
   sString = cellstr(get(hNroBits, 'String'));
   iNroBits = str2num(char(sString(iIndex)));
   if isempty(iNroBits)
      iNroBits = -1;
      return;
   end

   hSeg = findobj(hFig, 'Tag', 'txtSeg');
```

```
sSeg = get(hSeg, 'String');
iSeg = str2num(sSeg);
if isempty(iSeg)
   iSeg = -1;
   return;
end
```

```
%*************************************************************
%                        wcgAnalisisLPC.m
%*************************************************************

%*************************************************************
%       Tesis:      Analisis LPC
%       Autor:      William Castro Grijalva
%       Funcion:    Realizar el analisis LPC de la señal de voz
%*************************************************************
function wcgAnalisisLPC(hFig, iMuestraActual, iSegundo, iFrecMuestreo);

      hAxe = findobj(hFig, 'Tag','axeLPC');
      if isempty(hAxe)
         return;
      end

      %Almacena en el eje los parametros de la señal a analizar
      set(hAxe, 'UserData', [iMuestraActual, iSegundo, iFrecMuestreo]);

      sSenal = AnalizaFrameDeSenal(hFig, iMuestraActual, iSegundo, ...
               iFrecMuestreo);   %Obtiene la señal

%*************************************************************
%     Función: Analizar la ventana de tiempo configurada
%*************************************************************
function [XData, YData] = AnalizaFrameDeSenal(hFig, iMuestraActual, ...
                                        iSegundo, iFrecMuestreo);


      %Obtiene los parametros para el análisis
      [iDuracion, iOrdenLPC, iFFTSize, sTipoEspectro, sVentana, ...
      iPreenfasis, iSuperponer] = ObtieneParametrosLPC(hFig);

      %Obtiene los datos que se van a analizar ...
      hAxes = findobj(hFig, 'Tag','axeAnalisis');
      hAxesLine = findobj(hFig, 'Tag','axeLinea');
      XData = get(hAxesLine, 'XData');
      YData = get(hAxesLine, 'YData');

      iNumeroTotalDeMuestras = length(YData);
      iNroDeMuestrasPorFrame = round(iDuracion*iFrecMuestreo/1000);

      %Valida que el tamaño del frame sea adecuado ...
      if (iNumeroTotalDeMuestras - iMuestraActual) <
      iNroDeMuestrasPorFrame
      msgbox('La cantidad de muestras es menor que lo que se necesita
      para hacer el analisis');
         return;
      end

      %Obtiene el frame de la señal que se va a analizar ...
      sSenal= YData(iMuestraActual:1:(iMuestraActual +
      iNroDeMuestrasPorFrame - 1))';
```

```
%Verifica si el numero de muestras es mayor que el FFT Size
%Por defecto el FFTSize debe estar en 512
if iNroDeMuestrasPorFrame > iFFTSize
   msgbox('El numero de muestras del frame es mayor que el tamaño
   del FFT','Aviso','warn');
   return;
end

%************************************************
%        DIBUJA EL ESPECTRO LPC
%************************************************
if strcmp(sTipoEspectro ,'LPC') == 1

   %************************************************
   %        %PRE-ENFASIS
   %************************************************
   if iPreenfasis == 1
      sSenalLP = filter([1 -0.95], [1], sSenal);
   else
      sSenalLP = sSenal;
   end

   %************************************************
   %        ES HAMMING?
   %************************************************
   if strcmp(sVentana,'Hamming') == 1
      sSenalLP = hamming(iNroDeMuestrasPorFrame).*sSenalLP;
   end

   %************************************************
   %        ANALISIS LPC
   %************************************************
   [a,rxx,k] = AnalisisLPC(sSenalLP , iOrdenLPC);
   %Ganancia LPC
   Ganancia = a*rxx;

   %************************************************
   %        Muestra lo coeficientes
   %************************************************
   cargarCoeficientes(hFig, a, k, Ganancia);

   [H,F]= freqz(sqrt(Ganancia), a, iFFTSize, iFrecMuestreo);
   fftFreqs = 0.5*(0:iFFTSize-1)/iFFTSize*iFrecMuestreo;
   fftMag = abs(H);
   dBMag = 20*log10(fftMag);
else
   %************************************************
   %        DIBUJA EL ESPECTRO FFT
   %************************************************
   sSenalLP = sSenal;
```

```
        iFFTSize2 = iFFTSize/2;

        fftFreqs = (0:iFFTSize2-1)/iFFTSize*iFrecMuestreo;
           fd = zeros(iFFTSize,1);

        %************************************************
        %        ES HAMMING?
        %************************************************
        if strcmp(sVentana,'Hamming') == 1
           fd(1:iNroDeMuestrasPorFrame) =
           hamming(iNroDeMuestrasPorFrame).*sSenal;
        else
           fd(1:iNroDeMuestrasPorFrame) = sSenal;
        end

        fd2 = abs(fft((fd)));
        fftMag = fd2(1:iFFTSize2);

        if fftMag(1) < 1E-8
           fftMag(1) = fftMag(2)/10;
        end

        dBMag = 20*log10(fftMag);
     end

     GraficaUnaSenal(hFig, fftFreqs, dBMag, iMuestraActual, iSegundo,
     iFrecMuestreo, iSuperponer, sTipoEspectro, iFFTSize);
     CalculaPicosDeLaSenal(hFig, dBMag, sTipoEspectro, iFrecMuestreo,
     iFFTSize);

%**********************************************************************
**%         Función: Obtiene los parámetros para el análisis LPC
%**********************************************************************
**function [iDuracion, iOrdenLPC, iFFTSize, sTipoEspectro, sVentana,
iPreenfasis, iSuperponer] = ObtieneParametrosLPC(hFig);

     hDuracion = findobj(hFig, 'Tag','cmbDuracion');
     iIndex = get(hDuracion, 'Value');
     sString = cellstr(get(hDuracion, 'String'));
     iDuracion = str2num(char(sString(iIndex)));

     hOrdenLPC = findobj(hFig, 'Tag','cmbOrdenLPC');
     iIndex = get(hOrdenLPC, 'Value');
     sString = cellstr(get(hOrdenLPC, 'String'));
     iOrdenLPC = str2num(char(sString(iIndex)));

     hFFTSize = findobj(hFig, 'Tag','cmbFFTSize');
     iIndex = get(hFFTSize, 'Value');
     sString = cellstr(get(hFFTSize, 'String'));
     iFFTSize = str2num(char(sString(iIndex)));

     hTipo = findobj(hFig, 'Tag','cmbTipoEspectro');
     iIndex = get(hTipo, 'Value');
```

```
sString = cellstr(get(hTipo, 'String'));
sTipoEspectro = char(sString(iIndex));

hTipo = findobj(hFig, 'Tag','cmbVentana');
iIndex = get(hTipo, 'Value');
sString = cellstr(get(hTipo, 'String'));
sVentana = char(sString(iIndex));

hUnidad = findobj(hFig, 'Tag','chkPreenfasis');
iCheck = get(hUnidad, 'Value');
if iCheck == 1
   iPreenfasis = 1;
else
   iPreenfasis = 0;
end

hUnidad = findobj(hFig, 'Tag','chkOverlay');
iCheck = get(hUnidad, 'Value');
if iCheck == 1
   iSuperponer = 1;
else
   iSuperponer = 0;
end

%************************************************************************
**%            Función: Realizar el gráfica la señal LPC
%************************************************************************
**function GraficaUnaSenal(hFig, fftFreqs, dBMag, iMuestraActual,
iSegundo, iFrecMuestreo, iSuperponer, sTipoEspectro, iFFTSize);

hAxe = findobj(hFig, 'Tag', 'axeLPC');
axes(hAxe)

   if iSuperponer == 1

   hLineaF = findobj(hAxe, 'Tag', 'LineaF'); %Busca todas las
   lineas formantes y las borra
   for i=1:length(hLineaF)
      delete(hLineaF(i));
   end

      hold on;
   end

if strcmp(sTipoEspectro ,'LPC') == 1
   hLine = plot(fftFreqs,dBMag);
else
   iModoStem = ModoStem(hFig);
      if iModoStem == 0
      hLine = plot(fftFreqs,dBMag);
   else
         len=length(dBMag);
      SenalRetrasada = zeros(len,1);
```

```
            SenalAdelantada = zeros(len,1);
            SenalRetrasada(1) = dBMag(1);
            SenalRetrasada(2:len) = dBMag(1:len-1);
            SenalAdelantada(1:len-1) = dBMag(2:len);
            SenalAdelantada(len) = dBMag(len);
            PicosDeLaSenal = find((dBMag > SenalRetrasada) & (dBMag >
            SenalAdelantada));
            Param = iFrecMuestreo/iFFTSize;
            FrecuenciasPico = (PicosDeLaSenal-1)*Param;
               hLine = stem(FrecuenciasPico, dBMag(PicosDeLaSenal)); %-
               - plotea solo los picos
         end
      end
      grid on;

      if iSuperponer == 1
         hold off;
      end

         ConfiguraColoresEjeLPC(hFig, hAxe, hLine, dBMag, iSuperponer,
         fftFreqs);
         aAzulMarino = [0.12156862745098 0.149019607843137
         0.396078431372549];


function ConfiguraColoresEjeLPC(hFig, hAxes, hLine, dBMag, iSuperponer,
fftFreqs);

      axes(hAxes);

      aBlanco = [1 1 1];
      aAzulMarino = [0.12156862745098 0.149019607843137
      0.396078431372549];

      [sSimbolo, aColor, YMin, YMax] = getSimboloYColor(dBMag,
      iSuperponer);

      XMax = max(fftFreqs);
      set(hAxes, 'Color', aBlanco);          %Cambia color de fondo
      set(hAxes, 'XColor', aAzulMarino);        %Cambia color de fondo
      set(hAxes, 'YColor', aAzulMarino);        %Cambia color de fondo
      set(hAxes, 'LineStyleOrder', sSimbolo);
      set(hAxes, 'YLim', [YMin YMax]);
      set(hAxes, 'XLim', [0 XMax]);
      set(hAxes, 'Tag', 'axeLPC');

      set(hLine, 'Color',aColor);
      set(hLine, 'Tag', 'axeLineaLPC');


      xlabel('Frecuencia (Hz)', 'Color', aAzulMarino);
      ylabel('Amplitud', 'Color', aAzulMarino);
```

```
%************************************************************************
**%         Obtiene los el simbolo y el color con el que va a graficar
%************************************************************************
**function [sSimbolo, aColor, Ymin, Ymax] = getSimboloYColor(dBMag,
iSuperponer);

   global iContadorDeColor iContadorDeSimbolo iContador Ymin Ymax;

      aPaletaDeColores='ymcrgbw';  % define all  colors 'y'=yellow,
      'm'=magenta ,etc
      sTipoDeLineas = '-:-.--';
      aPaleta(1,1:3) = [ 0.470588235294118 0.0274509803921569
      0.423529411764706 ];
      aPaleta(2,1:3) = [ 0.215686274509804 0.215686274509804
      0.658823529411765 ];
      aPaleta(3,1:3) = [ 0.682352941176471 0 0 ];
      aPaleta(4,1:3) = [ 1 0 0.501960784313725 ];
      aPaleta(5,1:3) = [ 0.949019607843137 0.407843137254902
      0.0509803921568627 ];
      aPaleta(6,1:3) = [ 0.258823529411765 0.627450980392157
      0.203921568627451 ];
      aPaleta(7,1:3) = [ 0.12156862745098 0.149019607843137
      0.396078431372549 ];

      %Si se desea que la curvas se superpongan ...
      if iSuperponer == 1

         dMin = min(dBMag);
         dMax = max(dBMag); % -- LPC plot

         if (dMin < Ymin), Ymin = dMin; end;
         if (dMax > Ymax), Ymax = dMax; end;

         iContadorDeColor = iContadorDeColor + 1;
         iContadorDeSimbolo = iContadorDeSimbolo + 1;
         iContador = iContador + 1;

         if iContador > 2
               sSimbolo = [sTipoDeLineas(iContador:iContador+1)];
               iContador = iContador + 1;
            else
               sSimbolo = [sTipoDeLineas(iContador)];
            end

         if (iContadorDeColor==8), iContadorDeColor=1; end;
         if (iContadorDeSimbolo==5), iContadorDeSimbolo=1; end;
         if (iContador==6),  iContador=0; end;

      else
         Ymin = min(dBMag);
         Ymax = max(dBMag);
         iContadorDeColor = 3;
```

```
            iContadorDeSimbolo = 1;
            iContador=1;
            sSimbolo = '-';
        end

        aColor = aPaleta(iContadorDeColor,1:3);

function iModoStem = ModoStem(hFig);

        hUnidad = findobj(hFig, 'Tag','chkFFTStem');
        iCheck = get(hUnidad, 'Value');
        if iCheck == 1
            iModoStem = 1;
        else
            iModoStem = 0;
        end

function CalculaPicosDeLaSenal(hFig, dBMag, sTipoEspectro,
iFrecMuestreo, iFFTSize);
global Ymin Ymax;

        hAxes = findobj(hFig, 'Tag', 'axeLPC');

        dMin = min(dBMag);
        dMax = max(dBMag);

        if (dMin < Ymin), Ymin = dMin; end;
        if (dMax > Ymax), Ymax = dMax; end;

            aAzulMarino = [0.12156862745098 0.149019607843137
            0.396078431372549];


            hF01 = findobj(hFig, 'Tag', 'txtFormante01');
            hF02 = findobj(hFig, 'Tag', 'txtFormante02');
            hF03 = findobj(hFig, 'Tag', 'txtFormante03');
            hF04 = findobj(hFig, 'Tag', 'txtFormante04');
            hF05 = findobj(hFig, 'Tag', 'txtFormante05');

            hA01 = findobj(hFig, 'Tag', 'txtAmplitud01');
            hA02 = findobj(hFig, 'Tag', 'txtAmplitud02');
            hA03 = findobj(hFig, 'Tag', 'txtAmplitud03');
            hA04 = findobj(hFig, 'Tag', 'txtAmplitud04');
            hA05 = findobj(hFig, 'Tag', 'txtAmplitud05');

        set(hF01,'String','0');
        set(hF02,'String','0');
        set(hF03,'String','0');
        set(hF04,'String','0');
        set(hF05,'String','0');

        set(hA01,'String','0');
        set(hA02,'String','0');
```

```
    set(hA03,'String','0');
    set(hA04,'String','0');
    set(hA05,'String','0');

    %Busca los picos de la señal
    if strcmp(sTipoEspectro ,'LPC') == 1

          len = length(dBMag);
          SenalRetrasada = zeros(len,1);
          SenalAdelantada = zeros(len,1);
          SenalRetrasada(1) = dBMag(1);
          SenalRetrasada(2:len) = dBMag(1:len-1);
          SenalAdelantada(1:len-1) = dBMag(2:len);
          SenalAdelantada(len) = dBMag(len);
          PicosDeLaSenal = find((dBMag > SenalRetrasada) & (dBMag >
          SenalAdelantada));
          Param = 0.5*iFrecMuestreo/iFFTSize;
          FrecuenciasPico = (PicosDeLaSenal-1)*Param;
          NroDePicos = length(PicosDeLaSenal);

          if NroDePicos >= 1

set(hF01,'String',sprintf('%d',round(FrecuenciasPico(1))));

set(hA01,'String',sprintf('%d',round(dBMag(PicosDeLaSenal(1)))));

             h2 = line('Parent',hAxes, ...
             'Color',aAzulMarino, ...
             'EraseMode','xor', ...
             'Tag','LineaF', ...
             'UserData',0, ...
             'XData',[FrecuenciasPico(1) FrecuenciasPico(1)], ...
             'YData',[Ymin Ymax]);
          end;

          if NroDePicos >= 2

set(hF02,'String',sprintf('%d',round(FrecuenciasPico(2))));

set(hA02,'String',sprintf('%d',round(dBMag(PicosDeLaSenal(2)))));

             h2 = line('Parent',hAxes, ...
             'Color',aAzulMarino, ...
             'EraseMode','xor', ...
             'Tag','LineaF', ...
             'UserData',0, ...
             'XData',[FrecuenciasPico(2) FrecuenciasPico(2)], ...
             'YData',[Ymin Ymax]);

          end;
          if NroDePicos >= 3

set(hF03,'String',sprintf('%d',round(FrecuenciasPico(3))));
```

```
set(hA03,'String',sprintf('%d',round(dBMag(PicosDeLaSenal(3)))));

                h2 = line('Parent',hAxes, ...
                'Color',aAzulMarino, ...
                'EraseMode','xor', ...
                'Tag','LineaF', ...
                'UserData',0, ...
                'XData',[FrecuenciasPico(3) FrecuenciasPico(3)], ...
                'YData',[Ymin Ymax]);

            end;
            if NroDePicos >= 4

   set(hF04,'String',sprintf('%d',round(FrecuenciasPico(4))));

set(hA04,'String',sprintf('%d',round(dBMag(PicosDeLaSenal(4)))));

                h2 = line('Parent',hAxes, ...
                'Color',aAzulMarino, ...
                'EraseMode','xor', ...
                'Tag','LineaF', ...
                'UserData',0, ...
                'XData',[FrecuenciasPico(4) FrecuenciasPico(4)], ...
                'YData',[Ymin Ymax]);

            end;
            if NroDePicos >= 5

   set(hF05,'String',sprintf('%d',round(FrecuenciasPico(5))));

set(hA05,'String',sprintf('%d',round(dBMag(PicosDeLaSenal(5)))));

                h2 = line('Parent',hAxes, ...
                'Color',aAzulMarino, ...
                'EraseMode','xor', ...
                'Tag','LineaF', ...
                'UserData',0, ...
                'XData',[FrecuenciasPico(5) FrecuenciasPico(5)], ...
                'YData',[Ymin Ymax]);

            end;
      end;


%************************************************************************
**%         Función: Realizar el calculo de los coeficiente LPC
%        ,de los coeficientes de reflexión y de la autocorrelación.
%************************************************************************
**function [a, rxx, k] = AnalisisLPC(sSenalLP,iOrdenLPC)

   %Realiza la autocorrelacion de la señal
   rxx = xcorr(sSenalLP,iOrdenLPC);
```

```
   %Ya que la función xcorr devuelve valores adicionale, hay que
   limitarla
   rxx(1:iOrdenLPC)=[];

   %Realiza el analisis LPC usando el metodo de levinson-durbin
   %************************************************************
   %  La función aryule es similar a la funcion lpc()
   %  ademas de los coeficientes a devuelve los
   %  coeficiente de relefexion k
   %************************************************************
   %a = lpc(sSenalLP, iOrdenLPC);
   [a e k] = aryule(sSenalLP, iOrdenLPC);


function cargarCoeficientes(h0, a, k, Ganancia);

   hLst = findobj(h0, 'Tag', 'lstCoef');
   set(hLst, 'String', '');

   aCoef(1) = cellstr(sprintf('G = %6.4f', Ganancia));

   for i = 1:length(a)
      sItem = cellstr(sprintf('a[%d] = %6.4f', i, a(i)));
      aCoef(i+1) = sItem;
   end

   for j = 1:length(k)
      sItem = cellstr(sprintf('k[%d] = %6.4f', j, k(j)));
      aCoef(i+j) = sItem;
   end

   hLst = findobj(h0, 'Tag', 'lstCoef');
   set(hLst, 'String', aCoef);


%************************************************************************
**
%                          wcgAnalisisLPCTrama.m
%************************************************************************
**
function [a, k, G] = wcgAnalisisLPCTrama(aSenalA, iMuestrasPorVentana,
f0)


   if f0 == 0
      ORDEN_LPC = 4;
   else
      ORDEN_LPC = 10;
   end

   P = -0.95;
```

```
%*********************************************************
%  Realiza un Preenfasis de la señal
%*********************************************************
aSenal = filter([1 P], [1], aSenalA);

%************************************************
%  Se aplica la ventana Hamming a la señal
%************************************************
aSenal = hamming(iMuestrasPorVentana).*aSenal;

%*********************************************************
%  Cálculo de los parametros LPC (ai, ki Y G)
%*********************************************************
[ai ,r, ki] = AnalisisLPC(aSenal , ORDEN_LPC);

G =  ai*r;  %(Esta es la ganacia al cuadrado)
k = ki';
a = ai;

%************************************************************************
**%         Función: Realizar el calculo de los coeficiente LPC
%        ,de los coeficientes de reflexión y de la autocorrelación.
%************************************************************************
**function [a, rxx, k] = AnalisisLPC(sSenalLP,iOrdenLPC)

%Realiza la autocorrelacion de la señal
rxx = xcorr(sSenalLP,iOrdenLPC);

%Ya que la función xcorr devuelve valores adicionale, hay que
limitarla
rxx(1:iOrdenLPC)=[];

%Realiza el analisis LPC usando el metodo de levinson-durbin
%*************************************************************
%  La función aryule es similar a la funcion lpc()
%  ademas de los coeficientes a devuelve los
%  coeficiente de relefexion k
%*************************************************************
%a = lpc(sSenalLP, iOrdenLPC);
[a e k] = aryule(sSenalLP, iOrdenLPC);
```

```
%*************************************************************************
**
%                        wcgArchivo.m
%*************************************************************************
**

%*************************************************************************
**%         Tesis:      Analisis LPC
%         Autor:      William Castro Grijalva
%         Funcion:    Realizar el manejo de almacenamiento y
          recuperacion de la señal de voz
%*************************************************************************
**function wcgArchivo(sAction, aParametro)

   switch sAction
   case 'LeerDeRuta'
      figure(aParametro.hFig);
      RecargaArchivo(aParametro.hFig, aParametro.sArchivo);

   case 'LeerArchivo'
      figure(aParametro);
      CargarDatos(aParametro);

   case 'GuardarTodoAArchivo' %Guarda en un archivo especifico

      if ~wcgConfiguracion(aParametro.hFig, 'Hay_Datos_Msg')
         return;
      end

      GuardarSenal(aParametro.hFig, 'Todo', aParametro.sArchivo);

   case 'GuardarTodoArcDefecto'  %guarda en el archivo de donde se
   cargo la señal

      if ~wcgConfiguracion(aParametro, 'Hay_Datos_Msg')
         return;
      end

      GuardarTodoArcDefecto(aParametro, 'Todo');

   case 'GuardarTodoComo'  %Pide el archivo donde se debe guardar

      if ~wcgConfiguracion(aParametro, 'Hay_Datos_Msg')
         return;
      end

      figure(aParametro);
      GuardarTodoComo(aParametro, 'Todo');

   case 'GuardarRegionComo'

      if ~wcgConfiguracion(aParametro, 'Hay_Datos_Msg')
         return;
```

```
      end

      figure(aParametro);
         GuardarRegionComo(aParametro, 'Region');

   case 'ConfiguraGrafico'

      figure(aParametro.hFig);
      ConfiguraGrafico(aParametro.hFig,aParametro.Fs,
aParametro.Muestras, aParametro.Seg, aParametro.Senal,
aParametro.NroBits);

   end

%************************************************************************
**%   Función que muestra un dialogo para obtener un archivo
%************************************************************************
**function [sRuta,sArchivo] = ShowDialogo(sComando,sFileExt)

if (strcmp(sComando, 'abrir'))
   [sArchivo, sRuta] = uigetfile(sFileExt, 'Seleccione el archivo que
desea abrir:');
   if ((~isstr(sArchivo)) | ~min(size(sArchivo))), return; end
elseif (strcmp(sComando, 'guardar'))
   [sArchivo, sRuta]  = uiputfile(sFileExt, 'Seleccione el archivo
donde desea guardar');
   if ((~isstr(sArchivo)) | ~min(size(sArchivo))), return; end
else
   msgbox([' Comando desconocido ', sComando, '.']);
   sArchivo = '';
   sRuta = '';
end

%************************************************************************
**%   Función de lectura de archivos de formato WAV
%************************************************************************
**function [iFrecMuestreo, iNroMuestras, dNroSegundos, aSenalOrigen,
iNroBits] = LeerArchivoWavDefault(sRuta, bRestarMedia);

[aSenalOrigen, iFrecMuestreo, iNroBits] = wavread(sRuta);
iNroMuestras = length(aSenalOrigen);
dNroSegundos = iNroMuestras/iFrecMuestreo;

if bRestarMedia
   aSenalOrigen = aSenalOrigen - mean(aSenalOrigen);
end

%*********************************************************************
%          Carga los parámetros de la señal
%*********************************************************************
function ConfiguraGrafico(hFig, iFrecMuestreo, iNroMuestras,
dNroSegundos, aSenalOrigen, iNroBits, sArchivo, bActualizarVista);
```

```
sTemp = sprintf('%d',iFrecMuestreo);
hParam = findobj(hFig, 'Tag', 'txtFs');   %Obtiene el handle del eje
set(hParam,'String', sTemp);
sTemp = sprintf('%d',iNroMuestras);
hParam = findobj(hFig, 'Tag', 'txtNroMue');  %Obtiene el handle del eje
set(hParam,'String', sTemp);
sTemp = sprintf('%4.2f', dNroSegundos);
hParam = findobj(hFig, 'Tag', 'txtTimTot');  %Obtiene el handle del eje
set(hParam,'String', sTemp);
sTemp = sprintf('%4.2f', fix(iNroBits));
hParam = findobj(hFig, 'Tag', 'txtNroBit');  %Obtiene el handle del eje
set(hParam,'String', sTemp);
hParam = findobj(hFig, 'Tag', 'txtTimIni');  %Obtiene el handle del eje
set(hParam,'String', '0');
set(hParam,'UserData', '0');
hParam = findobj(hFig, 'Tag', 'txtTimFin');  %Obtiene el handle del eje
set(hParam,'String', '0');
set(hParam,'UserData', '0');
hParam = findobj(hFig, 'Tag','lblRutaArchivo'); %Obtiene el handle del
eje
set(hParam,'String', sArchivo);

%Toma los parametros el eje de muestra
hAxes = findobj(hFig , 'Tag', 'axeAnalisis');
aPosition = get(hAxes, 'Position');

Et = 1000*(iNroMuestras-1)/iFrecMuestreo;
t = (0:(1000/iFrecMuestreo):Et);

%Crea el eje
%hAxes = subplot(1,1,1);
axes(hAxes);
set(hAxes, 'Units', 'Pixels' );
set(hAxes, 'Position', aPosition );

hMnu = findobj(hFig , 'Tag', 'mnuStem');
sCheck = get(hMnu, 'Check');

%if strcmp(sCheck, 'off')
   hLine = plot(t, aSenalOrigen, 'b');
%else
%  hLine = stem(t, aSenalOrigen, 'b');
%end

grid on;

aBlanco = [ 1 1 1 ];
aCeleste = [ 0.874509803921569 0.901960784313726 0.949019607843137 ];
aAzulMarino = [ 0.12156862745098 0.149019607843137 0.396078431372549 ];
aVerdeLight = [ 0.258823529411765 0.627450980392157 0.203921568627451
];

Ymin = min(aSenalOrigen);
```

```
Ymax = max(aSenalOrigen);

set(hAxes, 'YLim',    [1.1*Ymin 1.1*Ymax]);

set(hAxes, 'XLim',    [0 Et]);
set(hAxes,'Color',aBlanco);
set(hAxes,'XColor',aAzulMarino);
set(hAxes,'YColor',aAzulMarino);

%!!!!!!!!!!!!IMPORTANTE GUARDAR ESTO PARAMETROS EN EL USER DATA ....
set(hAxes,'UserData', [iFrecMuestreo, iNroMuestras ,dNroSegundos]);

set(hAxes, 'Tag', 'axeAnalisis');          %Esto se adiciona por que al
setear los valores anteriores se borra el tag

set(hLine,'Color',aVerdeLight);  %Cambia color de fondo
set(hLine,'Tag','axeLinea');  %Cambia color de fondo

xlabel('Tiempo (mseg)', 'Color', aAzulMarino);
ylabel('Amplitud', 'Color', aAzulMarino);

h2 = line('Parent',hAxes, ...
   'Color',[0.501960784313725 0 0.250980392156863], ...
   'EraseMode','xor', ...
   'Tag','LineaY', ...
   'UserData',0, ...
   'XData',[0 0], ...
   'YData',[-1 0.8]);
h2 = line('Parent',hAxes, ...
   'Color',[0.501960784313725 0 0.250980392156863], ...
   'EraseMode','xor', ...
   'Tag','LineaY2', ...
   'UserData',0, ...
   'XData',[0 0], ...
   'YData',[-1 0.8]);

if bActualizarVista == 1
   %Almacena la informacion de la primera vista esto es para hacer un
seguimiento de los zooms
   hVista = findobj(hFig, 'Tag', 'txtVistas');
   set(hVista, 'UserData', [1 get(hAxes, 'XLim')] );
end

%*************************************************************************
**%   Función que muestra un dialogo para obtener un archivo
%*************************************************************************
**function CargarDatos(hFig);

[sRuta,sNombreArchivo] = ShowDialogo('abrir','*.wav');

if (~ischar(sRuta)), return; end

sArchivo = [sRuta,sNombreArchivo];
```

```
fFile = fopen(sArchivo,'r');
if fFile <=0
   msgbox('El archivo especificado no existe ...','Error','warn');
   return;
end

%Busca la extensión del archivo
iPosPunto = find(sArchivo == '.');
sExtension = lower(sArchivo(iPosPunto + 1:length(sArchivo)));

if strcmp(sExtension, 'wav') == 1
   [iFrecMuestreo, iNroMuestras, dNroSegundos, aSenalOrigen, iNroBits]
= LeerArchivoWavDefault(sArchivo, 1);
   ConfiguraGrafico(hFig, iFrecMuestreo, iNroMuestras, dNroSegundos,
aSenalOrigen, iNroBits, sArchivo, 1);
end

function RecargaArchivo(hFig, sArchivo);

fFile = fopen(sArchivo,'r');
if fFile <=0
   msgbox('El archivo especificado no existe ...','Error','warn');
   return;
end

%Busca la extensión del archivo
iPosPunto = find(sArchivo == '.');
sExtension = lower(sArchivo(iPosPunto + 1:length(sArchivo)));

if strcmp(sExtension, 'wav') == 1
   [iFrecMuestreo, iNroMuestras, dNroSegundos, aSenalOrigen, iNroBits]
= LeerArchivoWavDefault(sArchivo, 1);
   ConfiguraGrafico(hFig, iFrecMuestreo, iNroMuestras, dNroSegundos,
aSenalOrigen, iNroBits, sArchivo, 1);
end

%************************************************************************
**%   Función que se usa para guardar la señal
%************************************************************************
**function GuardarTodoArcDefecto(hFig, sOpcion);

      hFile = findobj(hFig, 'Tag','lblRutaArchivo');
      sArchivo = get(hFile, 'string');

      if isempty(sArchivo)
         msgbox "No existe un archivo por defecto ..."
         return;
      end
      figure(hFig);

      GuardarSenal(hFig, 'Todo', sArchivo);

function GuardarTodoComo(hFig, sOpcion);
```

```
    [sRuta, sNombreArchivo] = ShowDialogo('guardar','*.wav');

    if sRuta == 0
          return;
    end

    sArchivo = [sRuta, sNombreArchivo];
    %Devuelve la extensión del archivo
    iPosPunto = find(sArchivo == '.');
    sExtension = lower(sArchivo(iPosPunto + 1:length(sArchivo)));

    %Le agrega la extension wav si es que no tiene ...
    if isempty(sExtension)  == 1
          sArchivo = [sArchivo, '.wav'];
    end

    GuardarSenal(hFig, 'Todo', sArchivo);

function GuardarRegionComo(hFig, sOpcion);

    [sRuta, sNombreArchivo] = ShowDialogo('guardar','*.wav');

    if sRuta == 0
          return;
    end

    sArchivo = [sRuta, sNombreArchivo];
    %Devuelve la extension del archivo
    iPosPunto = find(sArchivo == '.');
    sExtension = lower(sArchivo(iPosPunto + 1:length(sArchivo)));

    %Le agrega la extension wav si es que no tiene ...
    if isempty(sExtension)  == 1
          sArchivo = [sArchivo, '.wav'];
    end

    GuardarSenal(hFig, 'Region', sArchivo);

function GuardarSenal(hFig, sOpcion, sArchivo);

    hFs = findobj(hFig, 'Tag','txtFs');
    sString = get(hFs, 'String');
    iFs = str2num(sString);
    if isempty(iFs)
          iFs = -1;
          return;
    end

    hNroBits = findobj(hFig, 'Tag','txtNroBit');
    sString = get(hNroBits , 'String');
    iNroBits = str2num(sString);
    if isempty(iNroBits)
```

```
        iNroBits = -1;
        return;
    end

    %Obtiene los datos que se van a analizar ...
    hAxes = findobj(hFig, 'Tag','axeAnalisis');
    hAxesLine = findobj(hFig, 'Tag','axeLinea');
    YData = get(hAxesLine, 'YData');

    if strcmp(sOpcion, 'Todo') == 1
        wavwrite(YData, iFs, iNroBits, sArchivo);
    else
        %Obtiene los limites de las barras  , que se muestran en los
        labels
        hTime = findobj(hFig, 'Tag', 'txtTimIni');
        iTimeIni = get(hTime, 'UserData');

        hTime = findobj(hFig, 'Tag', 'txtTimFin');
        iTimeFin = get(hTime, 'UserData');

        %Si no se ha seleccionado un rango no hace nada...
        if iTimeIni == iTimeFin
        return
    end

    Parametros = get(hAxes,'UserData'); %[iFrecMuestreo, iNroMuestras
,dNroSegundos]

    iFrecMuestreo = Parametros(1);

    iMuestrasIni = round(iTimeIni/1000*iFrecMuestreo);
    iMuestrasFin = round(iTimeFin/1000*iFrecMuestreo);

    YRegion = YData(iMuestrasIni:iMuestrasFin);

    wavwrite(YRegion, iFs, iNroBits, sArchivo);
end
```

```
%*************************************************************************
**
%                        wcgAdquisicionDeVoz.m
%*************************************************************************
**

%*************************************************************************
**%         Tesis:      Analisis LPC
%         Autor:      William Castro Grijalva
%         Funcion:    Interface Grafica de Usuario para el módulo de
          adquisición (Ejes)
%*************************************************************************
**function wcgAxesAdquisicion(h0, sAction)

switch sAction
case 'crear'
   CrearGUI(h0);
case 'eliminar'
   EliminarGUI(h0);
end;

function CrearGUI(h0);
load wcgAxesAdquisicion;

h1 = axes('Parent',h0, ...
   'Units','pixels', ...
   'Box','on', ...
   'CameraUpVector',[0 1 0], ...
   'CameraUpVectorMode','manual', ...
   'Color',[1 1 1], ...
   'ColorOrder',mat1, ...
   'Position',[236 140 453 288], ...
   'Tag','axeAnalisis', ...
   'UserData',[22050 110241 4.999591836734694], ...
   'XColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'XGrid','on', ...
   'XLim',[0 4999.546485260771], ...
   'XLimMode','manual', ...
   'YColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'YGrid','on', ...
   'ZColor',[0 0 0], ...
   'ZGrid','on');
h2 = line('Parent',h1, ...
   'Color',[0.258823529411765 0.627450980392157 0.203921568627451], ...
   'Tag','axeLinea', ...
   'XData',[], ...
   'YData',[]);
h2 = text('Parent',h1, ...
   'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'HandleVisibility','off', ...
   'HorizontalAlignment','center', ...
   'Position',[2488.712299078924 -1.150522648083624 17.32050807568877],
...
```

```
    'String','Tiempo (mseg)', ...
    'Tag','axeXTag', ...
    'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[-365.0111371982421 -0.1094076655052265
17.32050807568877], ...
    'Rotation',90, ...
    'String','Amplitud', ...
    'Tag','axeYTag', ...
    'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','right', ...
    'Position',mat2, ...
    'Tag','axeZTag', ...
    'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[2488.712299078924 0.8439024390243901 17.32050807568877],
...
    'Tag','axeTitle', ...
    'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);

h2 = line('Parent',h1, ...
    'Color',[0.501960784313725 0 0.250980392156863], ...
    'EraseMode','xor', ...
    'Tag','LineaY', ...
    'UserData',0, ...
    'XData',[0 0], ...
    'YData',[-1 0.8]);
h2 = line('Parent',h1, ...
    'Color',[0.501960784313725 0 0.250980392156863], ...
    'EraseMode','xor', ...
    'Tag','LineaY2', ...
    'UserData',0, ...
    'XData',[0 0], ...
    'YData',[-1 0.8]);

function EliminarGUI(h0);

    hAxes = findobj(h0,'Tag','axeAnalisis');
    delete(hAxes);
```

```
%**************************************************************************
**
%                        wcgAxesAnalisis.m
%**************************************************************************
**

%**************************************************************************
**%         Tesis:      Analisis LPC
%         Autor:      William Castro Grijalva
%         Funcion:    Interface Grafica de Usuario para el módulo de
Analisis (Ejes)
%**************************************************************************
**function wcgAxesAnalisis(h0, sAction);

switch sAction
case 'crear'
   CrearGUI(h0);
case 'eliminar'
   EliminarGUI(h0);
end;

function CrearGUI(h0);

load wcgAxeAnalisis

h1 = axes('Parent',h0, ...
   'Units','pixels', ...
   'Box','on', ...
   'CameraUpVector',[0 1 0], ...
   'CameraUpVectorMode','manual', ...
   'Color',[1 1 1], ...
   'ColorOrder',mat1, ...
   'Position',[72 140 616 288], ...
   'Tag','axeAnalisis', ...
   'UserData',[22050 110241 4.999591836734694], ...
   'XColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'XGrid','on', ...
   'XLim',[0 4999.546485260771], ...
   'XLimMode','manual', ...
   'YColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'YGrid','on', ...
   'ZColor',[0 0 0], ...
   'ZGrid','on');
h2 = line('Parent',h1, ...
   'Color',[0.258823529411765 0.627450980392157 0.203921568627451], ...
   'Tag','axeLinea', ...
   'XData',[], ...
   'YData',[]);
h2 = text('Parent',h1, ...
   'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'HandleVisibility','off', ...
   'HorizontalAlignment','center', ...
```

```
    'Position',[2491.577264785696 -1.150522648083624 17.32050807568877],
...
    'String','Tiempo (mseg)', ...
    'Tag','axeXTag', ...
    'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',mat2, ...
    'Rotation',90, ...
    'String','Amplitud', ...
    'Tag','axeYTag', ...
    'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','right', ...
    'Position',[-590.1104048176649 1.213937282229965 17.32050807568877],
...
    'Tag','axeZTag', ...
    'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[2491.577264785696 0.8439024390243899 17.32050807568877],
...
    'Tag','axeTitle', ...
    'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h2 = line('Parent',h1, ...
    'Color',[0.501960784313725 0 0.250980392156863], ...
    'EraseMode','xor', ...
    'Tag','LineaY', ...
    'UserData',0, ...
    'XData',[0 0], ...
    'YData',[-1 0.8]);
h2 = line('Parent',h1, ...
    'Color',[0.501960784313725 0 0.250980392156863], ...
    'EraseMode','xor', ...
    'Tag','LineaY2', ...
    'UserData',0, ...
    'XData',[0 0], ...
    'YData',[-1 0.8]);

function EliminarGUI(h0);
    hAxes = findobj(h0,'Tag','axeAnalisis');
    delete(hAxes);
```

```
%*************************************************************************
**
%                          wcgAxesDEP.m
%*************************************************************************
**

%*************************************************************************
**
%        Tesis:      Analisis LPC
%        Autor:      William Castro Grijalva
%        Funcion:    Interface Grafica de Usuario para la DEP (Ejes)
%*************************************************************************
**
function wcgAxesDEP(h0, sAction);

switch sAction
case 'crear'
   CrearGUI(h0);
case 'eliminar'
   EliminarGUI(h0);
end;

function CrearGUI(h0);

load wcgAxeAnalisis

h1 = axes('Parent',h0, ...
   'Units','pixels', ...
   'Box','on', ...
   'CameraUpVector',[0 1 0], ...
   'CameraUpVectorMode','manual', ...
   'Color',[1 1 1], ...
   'ColorOrder',mat1, ...
   'Position',[72 140 616 136], ...
   'Tag','axeDEP', ...
   'UserData',[22050 110241 4.999591836734694], ...
   'XColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'XGrid','on', ...
   'XLim',[0 4999.546485260771], ...
   'XLimMode','manual', ...
   'YColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'YGrid','on', ...
   'ZColor',[0 0 0], ...
   'ZGrid','on');
h2 = line('Parent',h1, ...
   'Color',[ 0.258823529411765 0.627450980392157 0.203921568627451],
...
   'Tag','axeLineaDEP', ...
   'XData',[], ...
   'YData',[]);
h2 = text('Parent',h1, ...
   'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'HandleVisibility','off', ...
```

```
   'HorizontalAlignment','center', ...
   'Position',[2491.577264785696 -1.150522648083624 17.32050807568877],
...
   'String','Frecuencia (Hz)', ...
   'Tag','axeXDEPTag', ...
   'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
   'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'HandleVisibility','off', ...
   'HorizontalAlignment','center', ...
   'Position',mat2, ...
   'Rotation',90, ...
   'String','Energia', ...
   'Tag','axeYDEPTag', ...
   'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
   'Color',[0 0 0], ...
   'HandleVisibility','off', ...
   'HorizontalAlignment','right', ...
   'Position',[-590.1104048176649 1.213937282229965 17.32050807568877],
...
   'Tag','axeZDEPTag', ...
   'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
   'Color',[0 0 0], ...
   'HandleVisibility','off', ...
   'HorizontalAlignment','center', ...
   'Position',[2491.577264785696 0.8439024390243899 17.32050807568877],
...
   'Tag','axeDEPTitle', ...
   'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);

function EliminarGUI(h0);
   hAxes = findobj(h0,'Tag','axeDEP');
   delete(hAxes);
```

```
%**************************************************************************
**
%                          wcgAxesEspectrograma.m
%**************************************************************************
**

%**************************************************************************
**
%        Tesis:       Analisis LPC
%        Autor:       William Castro Grijalva
%        Funcion:     Interface Grafica de Usuario para el Espectrograma
(Ejes)
%**************************************************************************
***
function wcgAxesEspectrograma(h0, sAction);

switch sAction
case 'crear'
   CrearGUI(h0);
case 'eliminar'
   EliminarGUI(h0);
end;

function CrearGUI(h0);
load wcgAxesEspectrograma;

h1 = axes('Parent',h0, ...
   'Units','pixels', ...
   'Box','on', ...
   'CameraUpVector',[0 1 0], ...
   'CameraUpVectorMode','manual', ...
   'Color',[1 1 1], ...
   'ColorOrder',mat3, ...
   'Position',[226 118 461 136], ...
   'Tag','axeEspectrograma', ...
   'UserData',[22050 110241 4.999591836734694], ...
   'XColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'XGrid','on', ...
   'XLim',[0 4999.546485260771], ...
   'XLimMode','manual', ...
   'YColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'YGrid','on', ...
   'ZColor',[0 0 0], ...
   'ZGrid','on');

h2 = line('Parent',h1, ...
   'Color',[0.258823529411765 0.627450980392157 0.203921568627451], ...
   'Tag','axeLineaEspectro', ...
   'XData',[], ...
   'YData',[]);

h2 = text('Parent',h1, ...
   'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
```

```
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',mat4, ...
    'String','Frecuencia (Hz)', ...
    'Tag','axeXTag', ...
    'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);

h2 = text('Parent',h1, ...
    'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',mat5, ...
    'Rotation',90, ...
    'String','Amplitud', ...
    'Tag','axeYTag', ...
    'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);

h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','right', ...
    'Position',mat6, ...
    'Tag','axeZTag', ...
    'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);


function EliminarGUI(h0);

    hAxes = findobj(h0,'Tag','axeEspectrograma');
    delete(hAxes);

    hAxes = findobj(h0,'Tag','Colorbar');
    delete(hAxes);


%**************************************************************************
**
%                        wcgAxesLPC.m
%**************************************************************************
**

%**************************************************************************
**%         Tesis:      Analisis LPC
%         Autor:      William Castro Grijalva
%         Funcion:    Interface Grafica de Usuario para el módulo
Analisis LPC (Ejes)
%**************************************************************************
**
function fig = wcgAxesLPC(h0, sAction);
```

```
switch sAction
case 'crear'
   CrearGUI(h0);
case 'eliminar'
   EliminarGUI(h0);
case 'ver_ambos'
   VerAmbos(h0);
case 'ver_lpc'
   VerLPC(h0);
case 'ver_analisis'
   VerAnalisis(h0);
end;

function CrearGUI(h0);
load wcgAxesLPC;

%************************************************************************
**%               Ejes para ver la senal
%************************************************************************
**h1 = axes('Parent',h0, ...
   'Units','pixels', ...
   'Box','on', ...
   'CameraUpVector',[0 1 0], ...
   'CameraUpVectorMode','manual', ...
   'Color',[1 1 1], ...
   'Position',[226 293 461 136], ...
   'Tag','axeAnalisis', ...
   'UserData',[22050 110241 4.999591836734694], ...
   'XColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'XGrid','on', ...
   'XLim',[0 4999.546485260771], ...
   'XLimMode','manual', ...
   'YColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'YGrid','on', ...
   'ZColor',[0 0 0], ...
   'ZGrid','on');
h2 = line('Parent',h1, ...
   'Color',[0.258823529411765 0.627450980392157 0.203921568627451], ...
   'Tag','axeLinea', ...
   'XData',[], ...
   'YData',[]);
h2 = text('Parent',h1, ...
   'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'HandleVisibility','off', ...
   'HorizontalAlignment','center', ...
   'Position',mat2, ...
   'String','Tiempo (mseg)', ...
   'Tag','axeXTag', ...
   'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
   'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'HandleVisibility','off', ...
```

```
    'HorizontalAlignment','center', ...
    'Position',[-358.6631174208815 -0.02222222222222237
17.32050807568877], ...
    'Rotation',90, ...
    'String','Amplitud', ...
    'Tag','axeYTag', ...
    'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','right', ...
    'Position',[-2456.298925367249 1.948148148148148 17.32050807568877],
...
    'Tag','axeZTag', ...
    'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',mat3, ...
    'Tag','axeTitle', ...
    'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h2 = line('Parent',h1, ...
    'Color',[0.501960784313725 0 0.250980392156863], ...
    'EraseMode','xor', ...
    'Tag','LineaY', ...
    'UserData',0, ...
    'XData',[0 0], ...
    'YData',[-1 0.8]);
h2 = line('Parent',h1, ...
    'Color',[0.501960784313725 0 0.250980392156863], ...
    'EraseMode','xor', ...
    'Tag','LineaY2', ...
    'UserData',0, ...
    'XData',[0 0], ...
    'YData',[-1 0.8]);


%**************************************************************************
**%               Ejes para ver el grafico LPC
%**************************************************************************
**h1 = axes('Parent',h0, ...
    'Units','pixels', ...
    'Box','on', ...
    'CameraUpVector',[0 1 0], ...
    'CameraUpVectorMode','manual', ...
    'Color',[1 1 1], ...
    'ColorOrder',mat4, ...
    'Position',[226 118 461 136], ...
    'Tag','axeLPC', ...
```

```
    'UserData',[22050 110241 4.999591836734694], ...
    'XColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
    'XGrid','on', ...
    'XLim',[0 4999.546485260771], ...
    'XLimMode','manual', ...
    'YColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
    'YGrid','on', ...
    'ZColor',[0 0 0], ...
    'ZGrid','on');
h2 = line('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0.258823529411765 0.627450980392157 0.203921568627451], ...
    'Interruptible','off', ...
    'Tag','axeLineaLPC', ...
    'XData',[], ...
    'YData',[]);
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Interruptible','off', ...
    'Position',[2488.904663314601 -1.355555555555555 17.32050807568877],
...
    'String','Tiempo (mseg)', ...
    'Tag','axeXTag', ...
    'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Interruptible','off', ...
    'Position',[-358.6631174208815 -0.02222222222222214
17.32050807568877], ...
    'Rotation',90, ...
    'String','Amplitud', ...
    'Tag','axeYTag', ...
    'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','right', ...
    'Interruptible','off', ...
    'Position',[-2456.298925367249 4.540740740740741 17.32050807568877],
...
    'Tag','axeZTag', ...
    'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
```

```
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Interruptible','off', ...
    'Position',[2488.904663314601 1.103703703703704 17.32050807568877],
...
    'Tag','axeTitle', ...
    'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h2 = line('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0.501960784313725 0 0.250980392156863], ...
    'EraseMode','xor', ...
    'Interruptible','off', ...
    'Tag','LineaYLPC', ...
    'UserData',0, ...
    'XData',[0 0], ...
    'YData',[-1 0.8]);
h2 = line('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0.501960784313725 0 0.250980392156863], ...
    'EraseMode','xor', ...
    'Interruptible','off', ...
    'Tag','LineaY2LPC', ...
    'UserData',0, ...
    'XData',[0 0], ...
    'YData',[-1 0.8]);

function EliminarGUI(h0);

    hAxes = findobj(h0,'Tag','axeAnalisis');
    delete(hAxes);

    hAxes = findobj(h0,'Tag','axeLPC');
    delete(hAxes);

function VerAmbos(h0);

    hAxes = findobj(h0,'Tag','axeAnalisis');
    set(hAxes, 'Position',[226 293 461 136]);
    set(hAxes, 'Visible', 'on');

    hLinea = findobj(hAxes,'Tag','axeLinea');
    set(hLinea, 'Visible', 'on');

    hLinea = findobj(hAxes,'Tag','LineaY');
    set(hLinea, 'Visible', 'on');

    hLinea = findobj(hAxes,'Tag','LineaY2');
    set(hLinea, 'Visible', 'on');

    hAxes = findobj(h0,'Tag','axeLPC');
```

```
    set(hAxes, 'Position',[226 118 461 136]);
    set(hAxes, 'Visible', 'on');

    hLinea = findobj(hAxes,'Tag','axeLineaLPC');
    set(hLinea, 'Visible', 'on');

    hLinea = findobj(hAxes,'Tag','LineaYLPC');
    set(hLinea, 'Visible', 'on');

    hLinea = findobj(hAxes,'Tag','LineaY2LPC');
    set(hLinea, 'Visible', 'on');

function VerAnalisis(h0);

    hAxes = findobj(h0,'Tag','axeAnalisis');
    set(hAxes, 'Position', [226 118  461 288]);
    set(hAxes, 'Visible', 'on');

    hLinea = findobj(hAxes,'Tag','axeLinea');
    set(hLinea, 'Visible', 'on');

    hLinea = findobj(hAxes,'Tag','LineaY');
    set(hLinea, 'Visible', 'on');

    hLinea = findobj(hAxes,'Tag','LineaY2');
    set(hLinea, 'Visible', 'on');


    %Hace invisible el eje de LPC
    hAxes = findobj(h0,'Tag','axeLPC');
    set(hAxes, 'Visible', 'off');

    hLinea = findobj(hAxes,'Tag','axeLineaLPC');
    set(hLinea, 'Visible', 'off');

    hLinea = findobj(hAxes,'Tag','LineaYLPC');
    set(hLinea, 'Visible', 'off');

    hLinea = findobj(hAxes,'Tag','LineaY2LPC');
    set(hLinea, 'Visible', 'off');

    hLineaF = findobj(hAxes, 'Tag', 'LineaF');   %Busca todas las lineas
    formantes y las borra
    for i=1:length(hLineaF)
       set(hLineaF(i), 'Visible', 'off');
    end


function VerLPC(h0);

    hAxes = findobj(h0,'Tag','axeLPC');
    set(hAxes, 'Position', [226 118 461 288]);
```

```
set(hAxes, 'Visible', 'on');

hLinea = findobj(hAxes,'Tag','axeLineaLPC');
set(hLinea, 'Visible', 'on');

hLinea = findobj(hAxes,'Tag','LineaYLPC');
set(hLinea, 'Visible', 'on');

hLinea = findobj(hAxes,'Tag','LineaY2LPC');
set(hLinea, 'Visible', 'on');

hLineaF = findobj(hAxes, 'Tag', 'LineaF');   %Busca todas las lineas
formantes y las borra
for i=1:length(hLineaF)
   set(hLineaF(i), 'Visible', 'on');
end


%Hace invisible el eje de la Señal

hAxes = findobj(h0,'Tag','axeAnalisis');
set(hAxes, 'Visible', 'off');

hLinea = findobj(hAxes,'Tag','axeLinea');
set(hLinea, 'Visible', 'off');

hLinea = findobj(hAxes,'Tag','LineaY');
set(hLinea, 'Visible', 'off');

hLinea = findobj(hAxes,'Tag','LineaY2');
set(hLinea, 'Visible', 'off');
```

```
%**************************************************************************
**
%                          wcgAxesPitch.m
%**************************************************************************
**

%**************************************************************************
**%          Tesis:       Analisis LPC
%         Autor:       William Castro Grijalva
%         Funcion:     Interface Grafica de Usuario para el módulo de
Calculo del Pitch
%**************************************************************************
**function wcgAxesPitch(h0, sAction);

switch sAction
case 'crear'
   CrearGUI(h0);
case 'eliminar'
   EliminarGUI(h0);
end;

function CrearGUI(h0);

load wcgAxeAnalisis

h1 = axes('Parent',h0, ...
   'Units','pixels', ...
   'Box','on', ...
   'CameraUpVector',[0 1 0], ...
   'CameraUpVectorMode','manual', ...
   'Color',[1 1 1], ...
   'ColorOrder',mat1, ...
   'Position',[72 140 616 136], ...
   'Tag','axePitch', ...
   'UserData',[22050 110241 4.999591836734694], ...
   'XColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'XGrid','on', ...
   'XLim',[0 4999.546485260771], ...
   'XLimMode','manual', ...
   'YColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'YGrid','on', ...
   'ZColor',[0 0 0], ...
   'ZGrid','on');
h2 = line('Parent',h1, ...
   'Color',[0.258823529411765 0.627450980392157 0.203921568627451], ...
   'Tag','axeLineaPitch', ...
   'XData',[], ...
   'YData',[]);
h2 = text('Parent',h1, ...
   'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'HandleVisibility','off', ...
   'HorizontalAlignment','center', ...
```

```
   'Position',[2491.577264785696 -1.150522648083624 17.32050807568877],
...
   'String','Frecuencia (Hz)', ...
   'Tag','axeXPitchTag', ...
   'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
   'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'HandleVisibility','off', ...
   'HorizontalAlignment','center', ...
   'Position',mat2, ...
   'Rotation',90, ...
   'String','Energia', ...
   'Tag','axeYPitchTag', ...
   'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
   'Color',[0 0 0], ...
   'HandleVisibility','off', ...
   'HorizontalAlignment','right', ...
   'Position',[-590.1104048176649 1.213937282229965 17.32050807568877],
...
   'Tag','axeZPitchTag', ...
   'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
   'Color',[0 0 0], ...
   'HandleVisibility','off', ...
   'HorizontalAlignment','center', ...
   'Position',[2491.577264785696 0.8439024390243899 17.32050807568877],
...
   'Tag','axePitchTitle', ...
   'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);

function EliminarGUI(h0);
   hAxes = findobj(h0,'Tag','axePitch');
   delete(hAxes);
```

```
%**************************************************************************
**
%                        wcgAxesSimulacion.m
%**************************************************************************
**

%**************************************************************************
**%         Tesis:      Analisis LPC
%         Autor:      William Castro Grijalva
%         Funcion:    Interface Grafica de Usuario para el módulo de
Analisis (Ejes)
%**************************************************************************
**function wcgAxesSimulacion(h0, sAction)
switch sAction
case 'crear'
   CrearGUI(h0);
case 'eliminar'
   EliminarGUI(h0);
end;

function CrearGUI(h0);

load wcgAxesSimulacion;

%**************************************************************************
**%               Ejes para ver la senal
%**************************************************************************
**h1 = axes('Parent',h0, ...
   'Units','pixels', ...
   'Box','on', ...
   'CameraUpVector',[0 1 0], ...
   'CameraUpVectorMode','manual', ...
   'Color',[1 1 1], ...
   'ColorOrder',mat1, ...
   'Position',[216 299 470 129], ...
   'Tag','axeAnalisis', ...
   'UserData',[22050 110241 4.999591836734694], ...
   'XColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'XGrid','on', ...
   'XLim',[0 4999.546485260771], ...
   'XLimMode','manual', ...
   'YColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'YGrid','on', ...
   'ZColor',[0 0 0], ...
   'ZGrid','on');
 h2 = line('Parent',h1, ...
   'Color',[0.258823529411765 0.627450980392157 0.203921568627451], ...
   'Tag','axeLinea', ...
   'XData',[], ...
   'YData',[]);
 h2 = text('Parent',h1, ...
   'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
   'HandleVisibility','off', ...
```

```
    'HorizontalAlignment','center', ...
    'Position',[2494.443235716461 -1.375 17.32050807568877], ...
    'String','Tiempo (mseg)', ...
    'Tag','axeXTag', ...
    'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[-351.7804563189882 -0.015625 17.32050807568877], ...
    'Rotation',90, ...
    'String','Amplitud', ...
    'Tag','axeYTag', ...
    'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','right', ...
    'Position',[-2302.562986815195 2.015625 17.32050807568877], ...
    'Tag','axeZTag', ...
    'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[2494.443235716461 1.109375 17.32050807568877], ...
    'Tag','axeTitle', ...
    'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h2 = line('Parent',h1, ...
    'Color',[0.501960784313725 0 0.250980392156863], ...
    'EraseMode','xor', ...
    'Tag','LineaY', ...
    'UserData',0, ...
    'XData',[0 0], ...
    'YData',[-1 0.8]);
h2 = line('Parent',h1, ...
    'Color',[0.501960784313725 0 0.250980392156863], ...
    'EraseMode','xor', ...
    'Tag','LineaY2', ...
    'UserData',0, ...
    'XData',[0 0], ...
    'YData',[-1 0.8]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%              ejes para la simulacion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%h1 = axes('Parent',h0, ...
    'Units','pixels', ...
    'Box','on', ...
```

```
    'CameraUpVector',[0 1 0], ...
    'CameraUpVectorMode','manual', ...
    'Color',[1 1 1], ...
    'ColorOrder',mat2, ...
    'Position',[214 122 473 128], ...
    'Tag','axeSim02', ...
    'UserData',[22050 110241 4.999591836734694], ...
    'XColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
    'XGrid','on', ...
    'XLim',[0 4999.546485260771], ...
    'XLimMode','manual', ...
    'YColor',[0.12156862745098 0.149019607843137 0.396078431372549], ...
    'YGrid','on', ...
    'ZColor',[0 0 0], ...
    'ZGrid','on');
h2 = line('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0.258823529411765 0.627450980392157 0.203921568627451], ...
    'Interruptible','off', ...
    'Tag','axeLineaSim', ...
    'XData',[], ...
    'YData',[]);
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Interruptible','off', ...
    'Position',[2489.180983127715 -1.377952755905511 17.32050807568877],
...
    'String','Tiempo (mseg)', ...
    'Tag','axeXTag', ...
    'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0.12156862745098 0.149019607843137 0.396078431372549], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Interruptible','off', ...
    'Position',[-349.5445635881474 -0.02362204724409445
17.32050807568877], ...
    'Rotation',90, ...
    'String','Amplitud', ...
    'Tag','axeYTag', ...
    'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','right', ...
    'Interruptible','off', ...
```

```
    'Position',[-2266.743533571621 4.826771653543307 17.32050807568877],
...
    'Tag','axeZTag', ...
    'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Interruptible','off', ...
    'Position',[2489.180983127715 1.110236220472441 17.32050807568877],
...
    'Tag','axeTitle', ...
    'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h2 = line('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0.501960784313725 0 0.250980392156863], ...
    'EraseMode','xor', ...
    'Interruptible','off', ...
    'Tag','LineaY1_S2', ...
    'UserData',0, ...
    'XData',[0 0], ...
    'YData',[-1 0.8]);
h2 = line('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0.501960784313725 0 0.250980392156863], ...
    'EraseMode','xor', ...
    'Interruptible','off', ...
    'Tag','LineaY2_S2', ...
    'UserData',0, ...
    'XData',[0 0], ...
    'YData',[-1 0.8]);

function EliminarGUI(h0);

   hAxes = findobj(h0,'Tag','axeAnalisis');
   delete(hAxes);

   hAxes = findobj(h0,'Tag','axeSim02');
   delete(hAxes);
```

```
%*************************************************************************
**
%                        wcgConfiguracion.m
%*************************************************************************
**

%*************************************************************************
**%         Tesis:      Analisis LPC
%         Autor:      William Castro Grijalva
%         Funcion:    Archivo de configuracion
%*************************************************************************
**function result = wcgConfiguracion(h0, sAction);

   result = 0;

   switch(sAction)
   case 'crear_archivo_param'
      CreaArchivoDeParametros(h0);
   case 'cargar_archivo_param'
      CargarParametros(h0);
   case 'restablecer_vista'
      RestablecerVista(h0);
   case 'IsStem'
      IsStem(h0);
   case 'Hay_Datos'
      result = HayDatos(h0, 0);
   case 'Hay_Datos_Msg'
      result = HayDatos(h0, 1);
   end

function CreaArchivoDeParametros(h0);

   hFile = fopen('lpccfg.wcg', 'w');
   hObj = findobj(h0, 'Tag','lblRutaArchivo');

   if HayDatos(h0,0)
      sArchivo = get(hObj, 'String');
      fprintf(hFile, '%s\n', sArchivo);
   end

   fclose(hFile);

function CargarParametros(h0);

   i=0;
   hFile = fopen('lpccfg.wcg', 'r');

   %detecta si el archivo esta vacio
   iVacio = fgetl(hFile);
   if iVacio == -1
      %no hay datos
      fclose(hFile);
      return
```

```
   else
      %Si no esta vacio pone el puntero al inicio del file
      %para comenzar a leer ...
      fseek(hFile,0,-1);
      while feof(hFile)==0
         i = i+1;
         sLinea = fgetl(hFile);
         aParametro(i)  = cellstr(sLinea);
      end

      aParam.hFig = h0;
      aParam.sArchivo = char(aParametro(1));

      wcgArchivo('LeerDeRuta', aParam);
      RestablecerVista(h0);
   end
   fclose(hFile);

function RestablecerVista(hFig);

   %Obtiene los limites de las barras  , que se muestran en los labels
      hVista = findobj(hFig, 'Tag', 'txtVistas');
   oVista = get(hVista, 'UserData');
   iVista = oVista(1);  %Toma la vista actual

   if iVista == 0
      return;
   end

   iTimeIni = oVista(2*iVista);
   iTimeFin = oVista(2*iVista+1);

      hAxes = findobj(hFig, 'Tag', 'axeAnalisis');
   aPosition = get(hAxes, 'Position');
   set(hAxes, 'XLim',   [iTimeIni iTimeFin]);

function bHayDatos = HayDatos(h0, bMostrarMsg);

   bHayDatos = 1;

   %Valida que haya datos en el eje ...
   hAxes = findobj(h0, 'Tag', 'axeAnalisis');
   hLinea = findobj(h0, 'Tag', 'axeLinea');
   YData = get(hLinea, 'YData');

   if isempty(YData)
      bHayDatos = 0;
      if bMostrarMsg
               msgbox('No existen datos ...');
      end
         return;
   end
```

```
function IsStem(hFig);
   hMnu = findobj(hFig , 'Tag', 'mnuStem');
   sEstado = get(hMnu, 'Check');
   if strcmp(sEstado, 'on');
      set(hMnu, 'Check', 'off');
   else
      set(hMnu, 'Check', 'on');
   end

%**************************************************************************
**
%                          wcgDensidadEP.m
%**************************************************************************
**

%**************************************************************************
**%         Tesis:       Analisis LPC
%         Autor:       William Castro Grijalva
%         Funcion:     Proceso de Cálculo de la DEP
%**************************************************************************
**
function wcgDensidadEP(h0, sAction)

   switch sAction
   case 'mostrar_dep'
      DensidadEspectralDeLaSenal(h0);
   end

function DensidadEspectralDeLaSenal(hFig);

      %Obtiene los datos que se van a analizar ...
      hAxes = findobj(hFig, 'Tag','axeAnalisis');
      hAxesLine = findobj(hFig, 'Tag','axeLinea');
      XData = get(hAxesLine, 'XData');
      YData = get(hAxesLine, 'YData');

      aParametro = get(hAxes, 'UserData');
      iFrecMuestreo = aParametro(1);

      %*****************************************
      %     OBTIENE LA VENTANA VISIBLE
      %*****************************************
      aTiempo = get(hAxes, 'XLim');
      iTimeIni = aTiempo(1);
      iTimeFin = aTiempo(2);
      sSenalVisible =
YData(1+round(iTimeIni/1000*iFrecMuestreo):1:round(iTimeFin/1000*iFrecM
uestreo));

      %*****************************************
      %     DENSIDAD ESPECTRAL DE POTENCIA
      %*****************************************
```

```
hAxes = findobj(hFig, 'Tag','axeDEP');
axes(hAxes);
pwelch(sSenalVisible, 512, iFrecMuestreo, [ ], 256);
grid on;

aBlanco = [ 1 1 1 ];
aCeleste = [ 0.874509803921569 0.901960784313726
0.949019607843137 ];
aAzulMarino = [ 0.12156862745098 0.149019607843137
0.396078431372549 ];
aVerdeLight = [ 0.258823529411765 0.627450980392157
0.203921568627451 ];

set(hAxes,'Color',aBlanco);
set(hAxes,'XColor',aAzulMarino);
set(hAxes,'YColor',aAzulMarino);

set(hAxes, 'Xlim', [ 0 iFrecMuestreo/2]);
set(hAxes, 'Tag','axeDEP');
title('');

xlabel('Frecuencia (Hz)');
ylabel('Magnitud del Espectro de Potencia (dB)');

hLine = findobj('Tag','axeLineaDEP');
set(hLine,'Color',aVerdeLight);  %Cambia color de fondo
set(hLine,'Tag','axeLineaDEP');  %Cambia color de fondo
```

```
%*************************************************************************
**
%                        wcgEspectrograma.m
%*************************************************************************
**

%*************************************************************************
**%         Tesis:      Analisis LPC
%        Autor:      William Castro Grijalva
%        Funcion:    Proceso de Cálculo del Espectrograma
%*************************************************************************
**function wcgEspectrograma(h0, sAction);

   switch sAction
   case 'mostrar_espectro'
      muestraEspectrograma(h0);
   end

function muestraEspectrograma(h0);

      %Obtiene los datos que se van a analizar ...
      hAxes = findobj(h0, 'Tag','axeAnalisis');
      hAxesLine = findobj(h0, 'Tag','axeLinea');
      XData = get(hAxesLine, 'XData');
      YData = get(hAxesLine, 'YData');

      aParametro = get(hAxes, 'UserData');
      iFrecMuestreo = aParametro(1);

      %Obtiene los parametros para el análisis
      [sColor, iWinSize, iFecuenciaMax] = getParametros(iFrecMuestreo);

      aTiempo = get(hAxes, 'XLim');

      iTimeIni = aTiempo(1);
      iTimeFin = aTiempo(2);

      sSenalVisible =
YData(1+round(iTimeIni/1000*iFrecMuestreo):1:round(iTimeFin/1000*iFrecM
uestreo));


      %*********************************
      % ESPECTROGRAMA
      %*********************************
      hAxes = findobj(h0, 'Tag','axeEspectrograma');
      axes(hAxes);

      wha=hamming(iWinSize);

      [b,f,t] = specgram(sSenalVisible, iWinSize, iFrecMuestreo, wha);
      imagesc(t,f, abs(b)), axis xy;
```

```
      if ~(max(t)==0)

         if strcmp(sColor,'Colores')
            axis ([0 max(t), 0 max(f)]), colormap(jet);
            axis ([0 max(t), 0 iFecuenciaMax]), colormap(jet);
         else
            axis ([0 max(t), 0 max(f)]),
colormap(flipud(gray(length(colormap))));
            axis ([0 max(t), 0 iFecuenciaMax]),
colormap(flipud(gray(length(colormap))));
         end
      end

      caxis([0 10]);
      hBar = colorbar('vert');
      set(hBar, 'Position',[0.15 0.0509 0.0342 0.4726]);
      set(hBar, 'Tag', 'Colorbar');


      set(hAxes, 'Position',[226 118 461 136]);
      xlabel('Tiempo(Seg)');
      ylabel('Frecuencia (Hz)');
      set(hAxes, 'Tag', 'axeEspectrograma');



%************************************************************************
**%         Función: Obtiene los parámetros para el análisis LPC
%************************************************************************
**function [sColor, iWindowSize, iFecuenciaMax] = getParametros(Fs);

      hColor = findobj('Tag','cmbColor');
      iIndex= get(hColor, 'Value');
      sString = cellstr(get(hColor, 'String'));
      sColor = char(sString(iIndex));

      hWindowSize = findobj('Tag','cmbWinSize');
      iIndex = get(hWindowSize, 'Value');
      sString = cellstr(get(hWindowSize, 'String'));
      iWindowSize = str2num(char(sString(iIndex)));

      hWindowSize = findobj('Tag','cmbMaxFre');
      iIndex = get(hWindowSize, 'Value');
      sString = cellstr(get(hWindowSize, 'String'));

      switch char(sString(iIndex))
      case 'Fs'
            iFecuenciaMax = Fs;
      case 'Fs/2'
            iFecuenciaMax = round(Fs/2);
      otherwise
            iFecuenciaMax = str2num(char(sString(iIndex)));
      end
```

```
%*************************************************************************
**
%                          wcgInterpolacion.m
%*************************************************************************
**

%*************************************************************************
***%          Tesis:       Analisis LPC
%         Autor:       William Castro Grijalva
%         Funcion:     Proceso de Cálculo del Espectrograma
%*************************************************************************
**
function rr = wcgInterpolacion(x,r,K)
%x = [10 11 12];
%r = [0.7 0.8 -0.17];

M = 400;
%K = 16;

Mp = K*M;
step = (x(2) - x(1))/K;
xi = [x(1):step:x(3)];

for i = 1:(length(xi))

   if (xi(i)-fix(xi(i))) ~= 0

      m = xi(i);
      a = m;

      s = 0;
      for j = 1:3

         L = x(j);
         alpha = 2*pi*(L+a)/M;
         beta = 2*pi*(L-a)/M;

         s = s + r(j)*((sin(alpha*(M-1)/2))/(sin(alpha/2)) + ...
                       (sin(beta*(M-1)/2))/(sin(beta/2)));
      end

      rr(i) = (1/M)*s;
   else
      rr(i) = r(1+(i-1)/K);
   end

end
```

```
%*************************************************************************
**
%                          wcgLPC.m
%*************************************************************************
**

%*************************************************************************
**%          Tesis:       Analisis LPC
%         Autor:       William Castro Grijalva
%         Funcion:     Interface Grafica de Usuario para el módulo de
Analisis LPC
%*************************************************************************
**function fig = wcgLPC()
load wcgLPC;
h0 = figure('Color',[0.745098039215686 0.776470588235294
0.894117647058823], ...
    'Colormap',mat0, ...
    'FileName','C:\MATLABR11\work\Lpc01\wcg_CreaGUI_Analisis.m', ...
    'HandleVisibility','callback', ...
    'MenuBar','none', ...
    'Name','Modulo de Análisis', ...
    'NumberTitle','off', ...
    'PaperPosition',[18 180 576 432], ...
    'PaperUnits','points', ...
    'Position',[39 67 700 493], ...
    'Resize','off', ...
    'Tag','wcgAnalisis', ...
    'ToolBar','none', ...
    'UserData','[ ]');

%Crea las llamadas ...
sCallback = sprintf('wcgMouse(''OnMouseDown'', %d)',h0);
set(h0, 'WindowButtonDownFcn',sCallback);
sCallback = sprintf('wcgMouse(''OnMouseMove'', %d)',h0);
set(h0, 'WindowButtonMotionFcn',sCallback);
sCallback = sprintf('wcgMouse(''OnMouseUp'', %d)',h0);
set(h0, 'WindowButtonUpFcn',sCallback);

wcgMenus(h0);
wcgToolbarAnalisis(h0, 'crear');
wcgAxesAnalisis(h0, 'crear');
wcgStatusAnalisisLarge(h0, 'crear');
```

```
%*************************************************************************
**
%                        wcgArchivo.m
%*************************************************************************
**

%*************************************************************************
**%         Tesis:      Analisis LPC
%        Autor:      William Castro Grijalva
%        Funcion:    Menús del módulo de adquisición
%*************************************************************************
**function wcgMenuAdquisicion(h0, sAction);
switch sAction
case 'crear'
   CrearGUI(h0);
case 'eliminar'
   EliminarGUI(h0);
end;

function CrearGUI(h0);
h1 = uimenu('Parent',h0, ...
   'Label','Voz', ...
   'Tag','mnuVoz');
h2 = uimenu('Parent',h1, ...
   'Label','Reproducir', ...
   'Tag','mnuPlay');

   sCallback = sprintf('wcgMouse(''cmdPlayADQ_OnClick'', %3.4f)',h0);
   set(h2, 'Callback',sCallback);


h2 = uimenu('Parent',h1, ...
   'Callback','lpc03_mouse(''mnuGrabar_OnClick'')', ...
   'Label','Grabar', ...
   'Tag','mnuGrabar');

   sCallback = sprintf('wcgMouse(''cmdGrabarADQ_OnClick'', %3.4f)',h0);
   set(h2, 'Callback',sCallback);

function EliminarGUI(h0);

   hAxes = findobj(h0,'Tag','mnuVoz');
   delete(hAxes);

   hAxes = findobj(h0, 'Tag','mnuPlay');
   delete(hAxes);

   hAxes = findobj(h0, 'Tag','mnuGrabar');
   delete(hAxes);
```

```
%*************************************************************************
**
%                          wcgMenus.m
%*************************************************************************
**

%*************************************************************************
**%          Tesis:       Analisis LPC
%         Autor:       William Castro Grijalva
%         Funcion:     Menus del Sistema
%*************************************************************************
**function wcgMenus(h0)
h1 = uimenu('Parent',h0, ...
   'Label','Archivo', ...
   'Tag','mnuArchivo');
h2 = uimenu('Parent',h1, ...
   'Label','&Abrir Archivo ... ', ...
   'Tag','mnuAbrir');

   sCallback = sprintf('wcgArchivo(''LeerArchivo'', %d)',h0);
   set(h2, 'Callback',sCallback);

h2 = uimenu('Parent',h1, ...
   'Label','&Guardar Archivo ...', ...
   'Tag','mnuGuardar');

   sCallback = sprintf('wcgArchivo(''GuardarTodoArcDefecto'', %d)',h0);
   set(h2, 'Callback',sCallback);

h2 = uimenu('Parent',h1, ...
   'Label','&Guardar Archivo Como ...', ...
   'Tag','mnuGuardar');

   sCallback = sprintf('wcgArchivo(''GuardarTodoComo'', %d)',h0);
   set(h2, 'Callback',sCallback);



h2 = uimenu('Parent',h1, ...
   'Label','Guardar &Selección ...', ...
   'Tag','mnuGuardarRegion');

   sCallback = sprintf('wcgArchivo(''GuardarRegionComo'', %d)',h0);
   set(h2, 'Callback',sCallback);

h2 = uimenu('Parent',h1, ...
   'Label','&Salir', ...
   'Tag','mnusalir');
h1 = uimenu('Parent',h0, ...
   'Label','Análisis', ...
   'Tag','mnuAnalisis');
h2 = uimenu('Parent',h1, ...
```

```
    'Label','Calculo LPC', ...
    'Checked', 'off', ...
    'Tag','mnuLPC');

    sCallback = sprintf('wcgVerGUILPC(%d)',h0);
    set(h2, 'Callback',sCallback);

h2 = uimenu('Parent',h1, ...
    'Label','Cálculo del Pitch', ...
    'Tag','mnuPitch');

h3 = uimenu('Parent',h2, ...
    'Label','Autocorrelación', ...
    'Tag','mnuPitchAutocorrelacion');

    sCallback = sprintf('wcgVerGUIPitch(%d, ''auto'')',h0);
    set(h3, 'Callback',sCallback);

h3 = uimenu('Parent',h2, ...
    'Label','Cepstrum', ...
    'Tag','mnuPitchCepstrum');

    sCallback = sprintf('wcgVerGUIPitch(%d, ''ceps'')',h0);
    set(h3, 'Callback',sCallback);

h3 = uimenu('Parent',h2, ...
    'Label','SIFT', ...
    'Tag','mnuSIFT');

    sCallback = sprintf('wcgVerGUIPitch(%d, ''sift'')',h0);
    set(h3, 'Callback',sCallback);

h2 = uimenu('Parent',h1, ...
    'Label','Espéctograma', ...
    'Tag','mnuEspectograma');

    sCallback = sprintf('wcgVerGUIEspectrograma(%d)', h0);
    set(h2, 'Callback',sCallback);

h2 = uimenu('Parent',h1, ...
    'Label','Densidad Espectral de Potencia', ...
    'Tag','mnuDensidadEspectral');

    sCallback = sprintf('wcgVerGUIDEP(%d)',h0);
    set(h2, 'Callback',sCallback);

h1 = uimenu('Parent',h0, ...
    'Label','Opciones', ...
    'Tag','mnuOpciones');

h2 = uimenu('Parent',h1, ...
    'Label','Análisis', ...
    'Tag','mnuAnalisis');
```

```
    sCallback = sprintf('wcgVerGUIAnalisis(%d)', h0);
    set(h2, 'Callback',sCallback);

h2 = uimenu('Parent',h1, ...
    'Label','Adquisición', ...
    'Tag','mnuAdquisicion');

    sCallback = sprintf('wcgVerGUIAdquisicion(%d)',h0);
    set(h2, 'Callback',sCallback);

h2 = uimenu('Parent',h1, ...
    'Label','Simulación', ...
    'Tag','mnuSimulacion');

    sCallback = sprintf('wcgVerGUISimulacion(%d)', h0);
    set(h2, 'Callback',sCallback);

h1 = uimenu('Parent',h0, ...
    'Callback','close(gcf)', ...
    'Label','Salir', ...
    'Tag','mnuSalir');
```

```
%*************************************************************************
**
%                          wcgMouse.m
%*************************************************************************
**

%*************************************************************************
***         Tesis:      Analisis LPC
%         Autor:      William Castro Grijalva
%         Funcion:    Funcion que maneja los eventos del mouse
%*************************************************************************
**
function wcgMouse(sAction, hFig);

   if nargin < 1,
      return;
   end;

   %Fuerza que el gcf sea el que tenga el tag sTag ...
   %figure(hFig)
   sTag = get(hFig, 'Tag');

   %Valida que haya datos en el eje ...
   hAxes = findobj(hFig, 'Tag', 'axeAnalisis');
   hLinea = findobj(hFig, 'Tag', 'axeLinea');
   YData = get(hLinea, 'YData');

      if isempty(YData) & strcmp(sAction, 'cmdGrabarADQ_OnClick')==1 &
      strcmp(sAction, 'mnuGrabarADQ_OnClick') == 1
      hMsg  = findobj(hFig, 'Tag', 'lblRutaArchivo');
      set(hMsg, 'String', '');
      set(hMsg, 'String', 'No existen datos para realizar el análisis
...');
      return;
   end


   switch (sAction)
   case 'OnMouseDown'
      OnMouseDown(hFig);
      case 'OnMouseMove',

      %if strcmp(sTag, 'wcgLPC')
      %  bMostraLineaY2 = 1;
      %else4
      %  bMostraLineaY2 = 1;
      %end

      bMostraLineaY2 = 1;
         OnMouseMove(hFig, bMostraLineaY2);

      case 'OnMouseUp',
```

```
    if strcmp(sTag, 'wcgLPC')
       bHacerAnalisisLPC = 1;
    else
       bHacerAnalisisLPC = 0;
    end
       OnMouseUp(hFig, bHacerAnalisisLPC);

case 'chkVerSenalLPC_OnClick'
    chkVerSenalLPC_OnClick(hFig);

case 'chkVerLPCLPC_OnClick'
    chkVerLPCLPC_OnClick(hFig);

case 'ActualizarLPC',
    ActualizarLPC(hFig)

case 'cmdPlayADQ_OnClick'
    cmdPlayADQ_OnClick(hFig);

case 'cmdGrabarADQ_OnClick'
    cmdGrabarADQ_OnClick(hFig);

case 'txtSegADQ_LostFocus'
    txtSegADQ_LostFocus(hFig);

case 'mnuPlayADQ_OnClick'
    mnuPlayADQ_OnClick(hFig);

case 'mnuGrabarADQ_OnClick',
    mnuGrabarADQ_OnClick(hFig)

case 'cmbColorEspectro_OnClick'
    wcgEspectrograma(hFig, 'mostrar_espectro');

case 'cmbWinSizeEspectro_OnClick'
    wcgEspectrograma(hFig, 'mostrar_espectro');

case 'cmbMaxFreEspectro_OnClick'
    wcgEspectrograma(hFig, 'mostrar_espectro');

case 'chkVerGriEspectro_OnClick'

    hColor = findobj(hFig, 'Tag','chkVerGri');
    iIndex= get(hColor, 'Value');
    if iIndex == 1
       grid on;
    else
       grid off;
    end

end
```

```
%*************************************************************************
**
%        Función: se ejecuta cuando se hace el click sobre la señal
%*************************************************************************
**
function OnMouseDown(hFig);

   ID_boton = get(hFig,'SelectionType');   % Indica que boton de mouse
   se presionó

   %PARA EL BOTON IZQUIERDO DEL MOUSE
   if (strcmp(ID_boton,'normal')==1)

      [iMuestraIni, iSegundo, iFrecMuestreo] = getSegundos(hFig);

      if iMuestraIni == -1
         return   %si hay error
      end

      hTime = findobj(hFig, 'Tag', 'txtTimIni');
      set(hTime, 'UserData', iSegundo);

      sSegundos = sprintf('%5.2f', iSegundo);
      set(hTime, 'String', sSegundos);
      % ... Coloca el numero de segundos en el label

      %Se usa la propiedad UserData de linea Y para almacenar el estado
      del mouse
      hLinea = findobj(hFig, 'Tag', 'LineaY');
      if ~isempty(hLinea)        %Lo pone a cero para indicar que el
      boton del mouse esta Up
         set(hLinea,'UserData',[1]);
      end;

      DrawLine (iSegundo, iFrecMuestreo, 'LineaY', hFig);

   end

   %PARA BOTON DERECHO DEL MOUSE
   if (strcmp(ID_boton,'alt')==1)

   end

%*************************************************************************
**
%        Función: se ejecuta cuando se hace mueve el mouse sobre la
señal
%*************************************************************************
**
function OnMouseMove(hFig, bMostrarLineaY2);

   ID_boton = get(hFig,'SelectionType');   % Indica que boton de mouse
   se presionó
```

```
%PARA EL BOTON IZQUIERDO DEL MOUSE
if (strcmp(ID_boton,'normal')==1)

   [iMuestra, iSegundo, iFrecMuestreo] = getSegundos(hFig);

   if iMuestra == -1
      return   %si hay error
   end

   hTime = findobj(hFig, 'Tag', 'txtTime');
   sSegundos = sprintf('%5.2f', iSegundo);
   set(hTime, 'String', sSegundos);
   % ... Coloca el numeo de segundos en el label

   %Si esta en modo LPC no se muestra el efecto de
   %desplazamiento de la linea ...
   if bMostrarLineaY2 == 0
         %Dibuja la linea en cero ...
         DrawLine (0, iFrecMuestreo, 'LineaY2', hFig);
         return;
   end

   %Si esta en modo Analisis, dibuja la linea que muestra
   %el corrimiento
   hLinea = findobj(hFig, 'Tag', 'LineaY');
   if ~isempty(hLinea)
      bMouseEst = get(hLinea,'UserData');
      if bMouseEst(1) == 1    %Esto indica que el mouse esta
         presionado
         DrawLine (iSegundo, iFrecMuestreo, 'LineaY2',hFig);
      end;
   end;
end


function OnMouseUp(hFig, bHacerAnalisisLPC);

   hLinea = findobj(hFig, 'Tag', 'LineaY');
   if ~isempty(hLinea)  %Lo pone a cero para indicar que el boton del
      mouse esta Up
      set(hLinea,'UserData', [0]);
   end;

   hTime = findobj(hFig, 'Tag', 'txtTimIni');
   iTimeIni = get(hTime, 'UserData');

   [iMuestraFin, iSegundo, iFrecMuestreo] = getSegundos(hFig);

   if iMuestraFin == -1
      return   %si hay error
   end
```

```
   ID_boton = get(hFig,'SelectionType');   % Indica que boton de mouse
   se presionó

   %PARA EL BOTON IZQUIERDO DEL MOUSE
   if (strcmp(ID_boton,'normal')==1)

         if iSegundo >= iTimeIni
            hTime = findobj(hFig, 'Tag', 'txtTimFin');
            set(hTime, 'UserData', iSegundo);
            sSegundos = sprintf('%5.2f', iSegundo);
            set(hTime, 'String', sSegundos);
         else
            hTime = findobj(hFig, 'Tag', 'txtTimIni');
            set(hTime, 'UserData', iSegundo);
            sSegundos = sprintf('%5.2f', iSegundo);
            set(hTime, 'String', sSegundos);

            hTime = findobj(hFig, 'Tag', 'txtTimFin');
            set(hTime, 'UserData', iTimeIni);
            sSegundos = sprintf('%5.2f', iTimeIni);
            set(hTime, 'String', sSegundos);
         end

         %Muestra en forma interactiva los valores en el eje de
         analisis LPC
         if bHacerAnalisisLPC == 1
            wcgAnalisisLPC(hFig, iMuestraFin, iSegundo, iFrecMuestreo);
         end

   elseif (strcmp(ID_boton,'tag')==1)


   end

function [iMuestra, iSegundo, iFrecMuestreo] = getSegundos(hFig);

   Coord = get(hFig,'CurrentPoint');  %Obtiene las coordenadas en
pixels
   Xpixel = Coord(1);
   Ypixel = Coord(2);

   %Coloca el numeo de segundos en el label ...
   hAxes = findobj(hFig, 'Tag', 'axeAnalisis');

   %Si aun no se ha creado el objeto Axes1
   if isempty(hAxes)
      iMuestra = -1;
      iSegundo = -1;
      iFrecMuestreo  = -1;
      return;
   end

   aPosition = get(hAxes, 'Position'); %[left bottom width height]
```

```
   %Verifica que el click se haya hecho dentro de los ejes
   if Xpixel < aPosition(1) | Xpixel > (aPosition(1)+aPosition(3))
      iMuestra = -1;
      iSegundo = -1;
      iFrecMuestreo  = -1;
      return;
   end

   %Los párametros se inicializaron con la carga de datos ...
   aParametros = get(hAxes, 'UserData');
   XLim = get(hAxes, 'XLim');

   iSegundo = XLim(1) + (Xpixel - aPosition(1))*(XLim(2)-
XLim(1))/aPosition(3);
   iMuestra = round(aParametros(2)*(iSegundo/1000)/aParametros(3));
   iFrecMuestreo  = aParametros(1);

function DrawLine(iSegundo, iFrecMuestreo, sLineName,hFig)

         %Dibuja la linea en la que se ubica el cursor ...
         hold on;
         %iLinea = iMuestra*1000/iFrecMuestreo;
         YLim = get(gca, 'YLim');
         XLim = [iSegundo iSegundo];

         hLinea = findobj(hFig, 'Tag', sLineName);
         if isempty(hLinea)   %Si aun no existe la linea lo crea (la
            primera vez) ...
            hLinea = plot(XLim, YLim);
            set(hLinea,'UserData',[0]);
         else              %Si ya existe la linea entonces olo lo
            cambia de lugar ...
            set(hLinea,'Xdata',XLim,'Ydata',Ylim);
         end;

         aRojo = [ 0.501960784313725 0 0.250980392156863 ];
         set(hLinea, 'Erasemode','xor', 'color',aRojo , 'Tag',
         sLineName);

         hold off;
         %...termina de dibujar la linea

%************************************************************************
***                  FUNCIONES SOBRE LAS OPCIONES DE LPC ...
%************************************************************************
**function  chkVerSenalLPC_OnClick(h0);

      EvaluaChecksVisuales(h0)

function chkVerLPCLPC_OnClick(h0);

      EvaluaChecksVisuales(h0)
```

```
function EvaluaChecksVisuales(h0)

      hCheckSe = findobj(h0, 'Tag','chkVerSenal');
      iValueSe = get(hCheckSe, 'Value');

      hCheckLPC = findobj(h0, 'Tag','chkVerLPC');
      iValueLPC = get(hCheckLPC, 'Value');

      if iValueSe == 0
         if iValueLPC == 0
            set(hCheckSe, 'Value', 1);
            set(hCheckLPC, 'Value', 1);
            wcgAxesLPC(h0, 'ver_ambos');
         else
            wcgAxesLPC(h0, 'ver_lpc');
         end
      else
         if iValueLPC == 0
            wcgAxesLPC(h0, 'ver_analisis');
         else
            wcgAxesLPC(h0, 'ver_ambos');
         end
      end

function ActualizarLPC(hFig);

      %Toma los parametros el eje de muestra ...
         hAxes = findobj(hFig, 'Tag', 'axeLPC');
      aParametros = get(hAxes, 'UserData');

      if isempty(aParametros)
         return;
      end

      iMuestraActual = aParametros(1);
      iSegundo =  aParametros(2);
      iFrecMuestreo  = aParametros(3);

      wcgAnalisisLPC(hFig, iMuestraActual, iSegundo, iFrecMuestreo);

%************************************************************************
**%                   FUNCIONES PARA ADQUISICION ...
%************************************************************************
**
function cmdPlayADQ_OnClick(hFig);

   hUnidad = findobj(hFig, 'Tag','chkRegion');
   iCheck = get(hUnidad, 'Value');
   if iCheck == 0
      wcgAdquisicionDeVoz(hFig, 'reproducir_todo');
   else
      wcgAdquisicionDeVoz(hFig, 'reproducir_region');
```

```
   end

function cmdGrabarADQ_OnClick(hFig);

      wcgAdquisicionDeVoz(hFig, 'grabar');

function txtSegADQ_LostFocus(hFig);
   hSeg = findobj(hFig, 'Tag', 'txtSeg');
   sSeg = deblank(get(hSeg, 'String'));

   ind1 = find(abs(sSeg)<46);
   ind2 = find(abs(sSeg)>57);

   if (size(ind1) > 0 )
      msgbox 'El tiempo ingresada no es válido ...';
      set(hSeg, 'String', '') ;
      set(hSeg, 'CurrentObject');
      return;
   elseif (size(ind2) > 0)
      msgbox 'El tiempo ingresada no es válido ...';
      set(hSeg, 'String', '') ;
      set(hSeg, 'CurrentObject');
      return;
   end

   sSeg = get(hSeg, 'String');
   iSeg = str2num(sSeg);
   if iSeg > 600
      msgbox('El tiempo de grabación debe ser menor a 10 minutos ');
      set(hSeg, 'String', '') ;
      set(hSeg, 'CurrentObject');
      return;
   end

function mnuPlayADQ_OnClick(hFig);
   cmdPlayADQ_OnClick(hFig)

function mnuGrabarADQ_OnClick(hFig);
   cmdGrabarADQ_OnClick(hFig);
```

```
%*************************************************************************
**
%                          wcgOpcionesAdquisicion.m
%*************************************************************************
**

%*************************************************************************
**%          Tesis:       Analisis LPC
%         Autor:       William Castro Grijalva
%         Funcion:     Interface Grafica de Usuario para el módulo de
          adquisición (Parametros)
%*************************************************************************
**function wcgOpcionesAdquisicion(h0, sAction);

switch sAction
case 'crear'
   CrearGUI(h0);
case 'eliminar'
   EliminarGUI(h0);
end;

function CrearGUI(h0);

load wcgOpcionesAdquisicion;

h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.752941176470588 0.815686274509804
0.886274509803922], ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'ListboxTop',0, ...
   'Position',[12.75 143.25 122.25 84], ...
   'String','Parámetros', ...
   'Style','frame', ...
   'Tag','fraAdquisicion01');

h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'FontWeight','normal', ...
   'ForegroundColor',[0.760784313725490l 0.8 0.894117647058823], ...
   'ListboxTop',0, ...
   'Position',[22.5 195.75 50 23.25], ...
   'String','Grabar', ...
   'Tag','cmdGrabar', ...
   'TooltipString','Grabar ...', ...
   'UserData','[ ]');

   sCallback = sprintf('wcgMouse(''cmdGrabarADQ_OnClick'', %3.4f)',h0);
   set(h1, 'Callback',sCallback);
```

```
h1 = uicontrol('Parent',h0, ...
   'BackgroundColor',[0.752941176470588 0.815686274509804
0.886274509803922], ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'ListboxTop',0, ...
   'Position',[18 306 163 121], ...
   'String','Parámetros', ...
   'Style','frame', ...
   'Tag','fraAdquisicion02');

h1 = uicontrol('Parent',h0, ...
   'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'FontWeight','bold', ...
   'ForegroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[18 405 162 22], ...
   'String','Parametros de Grabación:', ...
   'Style','text', ...
   'Tag','lblParGra');

h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.752941176470588 0.815686274509804
0.886274509803922], ...
   'FontWeight','bold', ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[18 270.75 109.5 27], ...
   'String','Frecuencia de Muestreo: (Hz)', ...
   'Style','text', ...
   'Tag','lblFreMue');

h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.752941176470588 0.815686274509804
0.886274509803922], ...
   'ListboxTop',0, ...
   'Position',[79.5 271.5 50.25 15], ...
   'String',mat3, ...
   'Style','popupmenu', ...
   'Tag','cmbFs', ...
   'Value',1);

h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.752941176470588 0.815686274509804
0.886274509803922], ...
```

```
    'FontWeight','bold', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[18.75 246.75 89.25 18.75], ...
    'String','N° Bits:', ...
    'Style','text', ...
    'Tag','lblNroBit2');

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.815686274509804
0.886274509803922], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ListboxTop',0, ...
    'Position',[79.5 253.5 50.25 15], ...
    'String',mat4, ...
    'Style','popupmenu', ...
    'Tag','cmbNroBits', ...
    'Value',1);

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.815686274509804
0.886274509803922], ...
    'FontWeight','bold', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[17.25 233.25 89.25 14.25], ...
    'String','Tiempo: (seg)', ...
    'Style','text', ...
    'Tag','lblTie');

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[78 234.75 51 15], ...
    'String','2', ...
    'Style','edit', ...
    'Tag','txtSeg');

    sCallback = sprintf('wcgMouse(''txtSegADQ_LostFocus'', %3.4f)',h0);
    set(h1, 'Callback',sCallback);


function EliminarGUI(h0);
```

```
hObj = findobj(h0, 'Tag','fraAdquisicion01');
delete(hObj);

hObj = findobj(h0, 'Tag','cmdPlay');
delete(hObj);

hObj = findobj(h0, 'Tag','cmdGrabar');
delete(hObj);

hObj = findobj(h0, 'Tag','fraAdquisicion02');
delete(hObj);

hObj = findobj(h0, 'Tag','lblParGra');
delete(hObj);

hObj = findobj(h0, 'Tag','lblFreMue');
delete(hObj);

hObj = findobj(h0, 'Tag','cmbFs');
delete(hObj);

hObj = findobj(h0, 'Tag','lblNroBit2');
delete(hObj);

hObj = findobj(h0, 'Tag','cmbNroBits');
delete(hObj);

hObj = findobj(h0, 'Tag','lblTie');
delete(hObj);

hObj = findobj(h0, 'Tag','txtSeg');
delete(hObj);
```

```
%**************************************************************************
**
%                          wcgOpcionesEspectrograma.m
%**************************************************************************
**

%**************************************************************************
**%         Tesis:      Analisis LPC
%        Autor:      William Castro Grijalva
%        Funcion:    Interface Grafica de Usuario para el Espectrograma
%**************************************************************************
**function wcgOpcionesEspectrograma(h0, sAction);

switch sAction
case 'crear'
   CrearGUI(h0);
case 'eliminar'
   EliminarGUI(h0);
end;

function CrearGUI(h0);
load wcgOpcionesEspectrograma;

h1 = uicontrol('Parent',h0, ...
   'BackgroundColor',[0.752941176470588 0.815686274509804
0.886274509803922], ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'ListboxTop',0, ...
   'Position',[19 283 148 144], ...
   'String','Parámetros', ...
   'Style','frame', ...
   'Tag','fraPar');

h1 = uicontrol('Parent',h0, ...
   'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'FontWeight','bold', ...
   'ForegroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[19 410 148 16], ...
   'String','Parámetros:', ...
   'Style','text', ...
   'Tag','lblPar1');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.752941176470588 0.815686274509804
0.886274509803922], ...
   'FontWeight','bold', ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
```

```
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[17.25 273 89.25 18.75], ...
    'String','Paleta:', ...
    'Style','text', ...
    'Tag','lblPaleta');

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.815686274509804
0.886274509803922], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ListboxTop',0, ...
    'Position',[69 278.25 50.25 15], ...
    'String',mat1, ...
    'Style','popupmenu', ...
    'Tag','cmbColor', ...
    'Value',1);

    sCallback = sprintf('wcgMouse(''cmbColorEspectro_OnClick'',
%d)',h0);
    set(h1, 'Callback',sCallback);


h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.815686274509804
0.886274509803922], ...
    'Callback','lpc04_OnLoad('''')', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ListboxTop',0, ...
    'Position',[68.25 259.5 50.25 15], ...
    'String',['64  ';'128 ';'256 ';'512 ';'1024'], ...
    'Style','popupmenu', ...
    'Tag','cmbWinSize', ...
    'Value',5);

    sCallback = sprintf('wcgMouse(''cmbWinSizeEspectro_OnClick'',
%d)',h0);
    set(h1, 'Callback',sCallback);


h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.815686274509804
0.886274509803922], ...
    'FontWeight','bold', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
```

```
    'Position',[18 253.5 47.25 18.75], ...
    'String','Ventana', ...
    'Style','text', ...
    'Tag','lblVentana');

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.815686274509804
0.886274509803922], ...
    'Callback','lpc04_OnLoad(''')', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ListboxTop',0, ...
    'Position',[68.25 240.75 50.25 15], ...
    'String',mat2, ...
    'Style','popupmenu', ...
    'Tag','cmbMaxFre', ...
    'Value',1);

    sCallback = sprintf('wcgMouse(''cmbMaxFreEspectro_OnClick'',
%d)',h0);
    set(h1, 'Callback',sCallback);

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.815686274509804
0.886274509803922], ...
    'FontWeight','bold', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[18 237 47.25 17.25], ...
    'String','Max. Frec:', ...
    'Style','text', ...
    'Tag','lblMaxFre');

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.815686274509804
0.886274509803922], ...
    'Callback','lpc04_OnLoad(''Grid'')', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ListboxTop',0, ...
    'Position',[20.25 219 89.25 12], ...
    'String','Ver Grid', ...
    'Style','checkbox', ...
    'Tag','chkVerGri');

    sCallback = sprintf('wcgMouse(''chkVerGriEspectro_OnClick'',
%d)',h0);
    set(h1, 'Callback',sCallback);
```

```
function EliminarGUI(h0);

   hObj = findobj(h0,'Tag','fraPar');
   delete(hObj);

   hObj = findobj(h0,'Tag','lblPar1');
   delete(hObj);

   hObj = findobj(h0,'Tag','lblPaleta');
   delete(hObj);

   hObj = findobj(h0,'Tag','cmbColor');
   delete(hObj);

   hObj = findobj(h0,'Tag','cmbWinSize');
   delete(hObj);

   hObj = findobj(h0,'Tag','lblVentana');
   delete(hObj);

   hObj = findobj(h0,'Tag','cmbMaxFre');
   delete(hObj);

   hObj = findobj(h0,'Tag','lblMaxFre');
   delete(hObj);

   hObj = findobj(h0,'Tag','chkVerGri');
   delete(hObj);
```

```
%*************************************************************************
**
%                          wcgOpcionesLPC.m
%*************************************************************************
**

%*************************************************************************
**%          Tesis:       Analisis LPC
%         Autor:       William Castro Grijalva
%         Funcion:     Interface Grafica de Usuario para el módulo de
Analisis LPC
%*************************************************************************
**
function fig = wcgOpcionesLPC(h0, sAction);

switch sAction
case 'crear'
   CrearGUI(h0);
case 'eliminar'
   EliminarGUI(h0);
end;

function CrearGUI(h0);

load wcgOpcionesLPC;

h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.7607843137254901 0.8 0.894117647058823], ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'ListboxTop',0, ...
   'Position',[10.5 177 117.75 145.5], ...
   'String','Parámetros', ...
   'Style','frame', ...
   'Tag','fraPar01');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.8901960784313731 0.909803921568627
0.949019607843137], ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'ListboxTop',0, ...
   'Position',[10.5 96.25 117 75], ...
   'String','Parámetros', ...
   'Style','frame', ...
   'Tag','fraPar02');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'ForegroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
```

```
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[10.5 310.5 117.75 12], ...
    'String','Parámetros de Análisis:', ...
    'Style','text', ...
    'Tag','lblParAna');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.7607843137254901 0.8 0.894117647058823], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[15.75 293.25 66 11.25], ...
    'String','Milisegundos', ...
    'Style','text', ...
    'Tag','lblMs');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.7607843137254901 0.8 0.894117647058823], ...
    'Callback','wcgMouse(''chkVerSenalLPC_OnClick'', 1.0000)', ...
    'ListboxTop',0, ...
    'Position',[75 292.5 50.25 15], ...
    'String',['10';'20';'30';'40';'50'], ...
    'Style','popupmenu', ...
    'Tag','cmbDuracion', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.7607843137254901 0.8 0.894117647058823], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[15 270.75 89.25 18.75], ...
    'String','Orden LPC:', ...
    'Style','text', ...
    'Tag','lblOrden');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.7607843137254901 0.8 0.894117647058823], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ListboxTop',0, ...
    'Position',[75 276 50.25 15], ...
    'String',mat2, ...
    'Style','popupmenu', ...
    'Tag','cmbOrdenLPC', ...
    'Value',2);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.7607843137254901 0.8 0.894117647058823], ...
```

```
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[14.25 252.75 89.25 18.75], ...
    'String','Tamaño FFT', ...
    'Style','text', ...
    'Tag','lblFFTSize');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.760784313725490 0.8 0.894117647058823], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ListboxTop',0, ...
    'Position',[74.25 259.5 50.25 15], ...
    'String',['64  ';'128 ';'256 ';'512 ';'1024'], ...
    'Style','popupmenu', ...
    'Tag','cmbFFTSize', ...
    'Value',4);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.760784313725490 0.8 0.894117647058823], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[15 235.5 89.25 18.75], ...
    'String','Espectro:', ...
    'Style','text', ...
    'Tag','lblEspec');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.760784313725490 0.8 0.894117647058823], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ListboxTop',0, ...
    'Position',[74.25 243 50.25 15], ...
    'String',['LPC';'FFT'], ...
    'Style','popupmenu', ...
    'Tag','cmbTipoEspectro', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ForegroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[10.5 162 117 11.25], ...
    'String','Resultados de Análisis:', ...
    'Style','text', ...
    'Tag','lblAna');
```

```
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'ForegroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[10.5 150 57.75 11.25], ...
   'String','Formantes', ...
   'Style','text', ...
   'Tag','lblForma');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'ForegroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[69 150 58.5 11.25], ...
   'String','Amplitud', ...
   'Style','text', ...
   'Tag','lblAmp');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','right', ...
   'ListboxTop',0, ...
   'Position',[69 139.5 57 10], ...
   'String','17', ...
   'Style','text', ...
   'Tag','txtAmplitud01');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','right', ...
   'ListboxTop',0, ...
   'Position',[11.25 139.5 57 9.75], ...
   'String','445', ...
   'Style','text', ...
   'Tag','txtFormante01');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
```

```
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[12 129 56.25 9.75], ...
    'String','1828', ...
    'Style','text', ...
    'Tag','txtFormante02');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[69 129 57 9.75], ...
    'String','3', ...
    'Style','text', ...
    'Tag','txtAmplitud02');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[12 118.5 56.25 9.75], ...
    'String','2359', ...
    'Style','text', ...
    'Tag','txtFormante03');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[69 118.5 57 9.75], ...
    'String','-6', ...
    'Style','text', ...
    'Tag','txtAmplitud03');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[12 108 57 9.75], ...
```

```
    'String','3531', ...
    'Style','text', ...
    'Tag','txtFormante04');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[69 108 57 9.75], ...
    'String','3', ...
    'Style','text', ...
    'Tag','txtAmplitud04');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[12 97.5 56.25 9.75], ...
    'String','0', ...
    'Style','text', ...
    'Tag','txtFormante05');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[69 97.5 57 9.75], ...
    'String','0', ...
    'Style','text', ...
    'Tag','txtAmplitud05');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.760784313725490 0.8 0.894117647058823], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ListboxTop',0, ...
    'Position',[72.75 192.75 54.75 15], ...
    'String','Hold', ...
    'Style','checkbox', ...
    'Tag','chkOverlay');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.760784313725490 0.8 0.894117647058823], ...
```

```
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ListboxTop',0, ...
    'Position',[75 226.5 50.25 15], ...
    'String',mat3, ...
    'Style','popupmenu', ...
    'Tag','cmbVentana', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.7607843137254901 0.8 0.894117647058823], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[15.75 219 47.25 18.75], ...
    'String','Ventana', ...
    'Style','text', ...
    'Tag','lblVentana');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.7607843137254901 0.8 0.894117647058823], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ListboxTop',0, ...
    'Position',[15 192.75 53.25 15], ...
    'String','Pre-enfasis', ...
    'Style','checkbox', ...
    'Tag','chkPreenfasis', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.7607843137254901 0.8 0.894117647058823], ...
    'Callback','wcgMouse(''chkVerLPCLPC_OnClick'', 1)', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ListboxTop',0, ...
    'Position',[15.75 207 110.25 15], ...
    'String','Ver LPC', ...
    'Style','checkbox', ...
    'Tag','chkVerLPC', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.7607843137254901 0.8 0.894117647058823], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ListboxTop',0, ...
    'Position',[15 179.25 108 15], ...
    'String','Solo muestras', ...
    'Style','checkbox', ...
    'Tag','chkFFTStem');
h1 = uicontrol('Parent',h0, ...
```

```
    'Units','points', ...
    'BackgroundColor',[0.7607843137254901 0.8 0.894117647058823], ...
    'Callback','wcgMouse(''chkVerSenalLPC_OnClick'', 1.0000)', ...
    'FontWeight','light', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ListboxTop',0, ...
    'Position',[72.75 209.25 51.75 9.75], ...
    'String','Ver Señal', ...
    'Style','checkbox', ...
    'Tag','chkVerSenal', ...
    'UserData','[ ]', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.8901960784313731 0.909803921568627
0.949019607843137], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ListboxTop',0, ...
    'Position',[10.5 11.25 117 79.5], ...
    'String','Parámetros', ...
    'Style','frame', ...
    'Tag','fraPar03');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ForegroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[10.5 81.75 117 11.25], ...
    'String','Coeficientes:', ...
    'Style','text', ...
    'Tag','lblCoef');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.8901960784313731 0.909803921568627
0.949019607843137], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'Position',[18.75 21 101.25 52.5], ...
    'String',mat4, ...
    'Style','listbox', ...
    'Tag','lstCoef', ...
    'Value',1);


function EliminarGUI(h0);

    hObj = findobj(h0, 'Tag','fraPar01');
    delete(hObj);
```

```
hObj = findobj(h0, 'Tag','fraPar02');
delete(hObj);
hObj = findobj(h0, 'Tag','lblParAna');
delete(hObj);
hObj = findobj(h0, 'Tag','lblDurFra');
delete(hObj);
hObj = findobj(h0, 'Tag','lblMs');
delete(hObj);
hObj = findobj(h0, 'Tag','optMS');
delete(hObj);
hObj = findobj(h0, 'Tag','optMuestras');
delete(hObj);
hObj = findobj(h0, 'Tag','cmbDuracion');
delete(hObj);
hObj = findobj(h0, 'Tag','lblOrden');
delete(hObj);
hObj = findobj(h0, 'Tag','cmbOrdenLPC');
delete(hObj);
hObj = findobj(h0, 'Tag','lblFFTSize');
delete(hObj);
hObj = findobj(h0, 'Tag','cmbFFTSize');
delete(hObj);
hObj = findobj(h0, 'Tag','lblEspec');
delete(hObj);
hObj = findobj(h0, 'Tag','cmbTipoEspectro');
delete(hObj);
hObj = findobj(h0, 'Tag','lblAna');
delete(hObj);
hObj = findobj(h0, 'Tag','lblForma');
delete(hObj);
hObj = findobj(h0, 'Tag','lblAmp');
delete(hObj);
hObj = findobj(h0, 'Tag','txtAmplitud01');
delete(hObj);
hObj = findobj(h0, 'Tag','txtFormante01');
delete(hObj);
hObj = findobj(h0, 'Tag','txtFormante02');
delete(hObj);
hObj = findobj(h0, 'Tag','txtAmplitud02');
delete(hObj);
hObj = findobj(h0, 'Tag','txtFormante03');
delete(hObj);
hObj = findobj(h0, 'Tag','txtAmplitud03');
delete(hObj);
hObj = findobj(h0, 'Tag','txtFormante04');
delete(hObj);
hObj = findobj(h0, 'Tag','txtAmplitud04');
delete(hObj);
hObj = findobj(h0, 'Tag','txtFormante05');
delete(hObj);
hObj = findobj(h0, 'Tag','txtAmplitud05');
delete(hObj);
hObj = findobj(h0, 'Tag','chkOverlay');
```

```
delete(hObj);
hObj = findobj(h0, 'Tag','cmbVentana');
delete(hObj);
hObj = findobj(h0, 'Tag','lblVentana');
delete(hObj);
hObj = findobj(h0, 'Tag','chkPreenfasis');
delete(hObj);
hObj = findobj(h0, 'Tag','chkFFTStem');
delete(hObj);
hObj = findobj(h0, 'Tag','chkVerLPC');
delete(hObj);
hObj = findobj(h0, 'Tag','chkVerSenal');
delete(hObj);
hObj = findobj(h0, 'Tag','fraPar03');
delete(hObj);
hObj = findobj(h0, 'Tag','lblCoef');
delete(hObj);
hObj = findobj(h0, 'Tag','lstCoef');
delete(hObj);
```

```
%*************************************************************************
**
%                          wcgOpcionesSimulacion.m
%*************************************************************************
**

%*************************************************************************
**%          Tesis:       Analisis LPC
%         Autor:       William Castro Grijalva
%         Funcion:     Interface Grafica de Usuario para el módulo de
simulacion (Parametros)
%*************************************************************************
**function wcgOpcionesSimulacion(h0, sAction);

switch sAction
case 'crear'
   CrearGUI(h0);
case 'eliminar'
   EliminarGUI(h0);
end;

function CrearGUI(h0);
load wcgOpcionesSimulacion;

h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'ListboxTop',0, ...
   'Position',[12.75 91.5 96 230.25], ...
   'Style','frame', ...
   'Tag','fraPar');

h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'ForegroundColor',[0.760784313725490 0.8 0.894117647058823], ...
   'ListboxTop',0, ...
   'Position',[19.5 173.25 81 24], ...
   'String','Sintetizada >>', ...
   'Tag','cmdSint', ...
   'TooltipString','Reproduce Señal Sintetizada', ...
   'UserData','[ ]');

   sCallback = sprintf('wcgSimulacion(%3.4f, ''reproducir_sin'');',h0);
   set(h1, 'Callback',sCallback);

h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'ForegroundColor',[0.760784313725490 0.8 0.894117647058823], ...
```

```
    'ListboxTop',0, ...
    'Position',[19.5 201 81 24], ...
    'String','Original', ...
    'Tag','cmdOri', ...
    'TooltipString','Reproduce Señal Original', ...
    'UserData','[ ]');

    sCallback = sprintf('wcgAdquisicionDeVoz(%3.4f,
''reproducir_todo'');',h0);
    set(h1, 'Callback',sCallback);

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ForegroundColor',[0.760784313725490 0.8 0.894117647058823], ...
    'ListboxTop',0, ...
    'Position',[19.5 146 81 24], ...
    'String','Guardar >>', ...
    'Tag','cmdGuardarSint', ...
    'TooltipString','Guardar la Señal Sintetizada', ...
    'UserData','[ ]');

    sCallback = sprintf('wcgSimulacion(%3.4f, ''guardar_sint'');',h0);
    set(h1, 'Callback',sCallback);

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ListboxTop',0, ...
    'Position',[18.75 290.25 76.5 14.25], ...
    'String',mat1, ...
    'Style','popupmenu', ...
    'Tag','cmbTipPit', ...
    'Value',1);

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[19.5 306.75 67.5 12], ...
    'String','Cálculo del Pitch', ...
    'Style','text', ...
    'Tag','lblCalPit');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
```

```
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[18.75 228 67.5 12], ...
   'String','Reproducir Señal:', ...
   'Style','text', ...
   'Tag','lblRepSen');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'Callback','wcgArchivo(''LeerArchivo'', 1)', ...
   'ForegroundColor',[0.7607843137254901 0.8 0.894117647058823], ...
   'ListboxTop',0, ...
   'Position',[19.5 257.25 81 24], ...
   'String','Simular', ...
   'Tag','cmdSimular', ...
   'TooltipString','Simular', ...
   'UserData','[ ]');

   sCallback = sprintf('wcgSimulacion(%3.4f, ''sintetizar'');',h0);
   set(h1, 'Callback',sCallback);


function EliminarGUI(h0);

   hObj = findobj(h0,'Tag','fraPar');
   delete(hObj);

   hObj = findobj(h0,'Tag','cmdSint');
   delete(hObj);

   hObj = findobj(h0,'Tag','cmdOri');
   delete(hObj);

   hObj = findobj(h0,'Tag','cmbTipPit');
   delete(hObj);

   hObj = findobj(h0,'Tag','lblRepSen');
   delete(hObj);

   hObj = findobj(h0,'Tag','lblCalPit');
   delete(hObj);

   hObj = findobj(h0,'Tag','cmdSimular');
   delete(hObj);

   hObj = findobj(h0,'Tag','cmdGuardarSint');
   delete(hObj);
```

```
%************************************************************************
**
%                          wcgPitch.m
%************************************************************************
**

%************************************************************************
**%          Tesis:       Analisis LPC
%         Autor:       William Castro Grijalva
%         Funcion:     Procesos para el calculo del Pitch
%************************************************************************
**function [aTrama, iNroDeTramas, iTamanoDeTrama, iNroMuestras] =
wcgPitch(h0, sAction);

   %Obtiene los datos que se van a analizar ...
   hAxes = findobj(h0, 'Tag','axeAnalisis');
   hAxesLine = findobj(h0, 'Tag','axeLinea');
   XData = get(hAxesLine, 'XData');
   YData = get(hAxesLine, 'YData');

   aParametro = get(hAxes, 'UserData');
   iFrecMuestreo = aParametro(1);

   aTiempo = get(hAxes, 'XLim');
   iTimeIni = aTiempo(1);
   iTimeFin = aTiempo(2);
   sSenalVisible =
YData(1+round(iTimeIni/1000*iFrecMuestreo):1:round(iTimeFin/1000*iFrecM
uestreo));

   switch sAction
   case 'auto'
      [aTrama, iNroDeTramas, iTamanoDeTrama, iNroMuestras] =
wcgPitchAutocorrelacion(h0, sSenalVisible, iFrecMuestreo);
   case 'ceps'
      [aTrama, iNroDeTramas, iTamanoDeTrama, iNroMuestras] =
wcgPitchCepstrum(h0, sSenalVisible, iFrecMuestreo);
   case 'sift'
      [aTrama, iNroDeTramas, iTamanoDeTrama, iNroMuestras] =
wcgPitchSift(h0, sSenalVisible, iFrecMuestreo);
   end

   %*********************************************************
   %              VISUALIZACION DE LA SEÑAL
   %*********************************************************
   hAxes = findobj(h0, 'Tag','axePitch');
   axes(hAxes);
   t = 0:iNroDeTramas;
   t = iTamanoDeTrama*t;

   %hLine = stairs(t, [aTrama.f0]);
   %hold on;
   hLine = stairs(t, [aTrama.f02]);
```

```
   axis([t(1) t(iNroDeTramas) 0 max([aTrama.f02])+50]);
   grid on;

   aBlanco = [ 1 1 1 ];
   aAzulMarino = [ 0.12156862745098 0.149019607843137 0.396078431372549
];
   aVerdeLight = [ 0.258823529411765 0.627450980392157
0.203921568627451 ];

   %Et = 1000*(iNroMuestras-1)/iFrecMuestreo;
   %set(hAxes, 'XLim',  [0 Et]);
   set(hAxes,'Color',aBlanco);            %Cambia color de fondo
   set(hAxes,'XColor',aAzulMarino);    %Cambia color de fondo
   set(hAxes,'YColor',aAzulMarino);    %Cambia color de fondo
   xlabel('Tiempo (mseg)', 'Color', aAzulMarino);
      ylabel('Pitch(Hz)', 'Color', aAzulMarino);
   set(hAxes, 'UserData', sAction);
   set(hAxes, 'Tag','axePitch'); %Esto se adiciona por que al setear
   los valores anteriores se borra el tag
   set(hLine,'Color',aVerdeLight);     %Cambia color de fondo
   set(hLine,'Tag','axeLineaPitch');      %Cambia color de fondo
```

```
%*************************************************************************
**
%                        wcgPitchAutocorrelacion.m
%*************************************************************************
**

%*************************************************************************
**%         Tesis:      Analisis LPC
%        Autor:      William Castro Grijalva
%        Funcion:    Proceso del Calculo del pitch por Autocorrelación
%*************************************************************************
**function [aTrama, iNroDeTramas, iTamanoDeTrama, iMuestrasPorTrama]  =
wcgPitchAutocorrelacion(h0, aSenalOrigen, iFrecMuestreo);

      %*****************************************
      %  CALCULO DEL PITCH POR AUTOCORRELACION
      %*****************************************
      iNroMuestras = length(aSenalOrigen);

      %*****************************************
      %  PARAMETROS
      %*****************************************
      iTamanoDeTrama = 20;          %son 20 ms
      iTraslape = 10;               %son 10 ms de traslape
      iVentanaDeAnalisis = iTamanoDeTrama + iTraslape;
      UMBRAL = 0.38;
      Ts = 1000/iFrecMuestreo;

      %*****************************************
      %  CREA UN FILTRO BUTTERWORTH DE 900HZ
      %*****************************************
      n = 4;                        %Orden del filtro
      Wn = 900/(iFrecMuestreo/2);      %Frecuencia del filtro
normalizada
      [bf0,af0] = butter(n,Wn);

      %*****************************************
      %              MUESTRAS
      %*****************************************
      iMuestrasPorTrama = floor((iTamanoDeTrama/1000)*iFrecMuestreo);
      iMuestrasPorVentana =
floor((iVentanaDeAnalisis/1000)*iFrecMuestreo);
      iNroDeTramas = floor(iNroMuestras/iMuestrasPorTrama);

      %Completa la ultima trama ...
      iUltMuestra = iMuestrasPorTrama*iNroDeTramas +
iMuestrasPorVentana;
      iNroCeros = iUltMuestra-length(aSenalOrigen);
      aSenalOrigen((iUltMuestra - iNroCeros + 1):iUltMuestra)     =
zeros(1,iNroCeros);

      for t = 1:(iNroDeTramas+1)
```

```
        %*************************************
        %           INICIALIZA
        %*************************************
        aTrama(t).f0 = 0;     %Valor de la frecuencia fundamental.
        aTrama(t).f01 = 0;       %Valor de la frecuencia fundamental
(median).

        %************************************************
        %           TRAMA A ANALIZAR
        %************************************************
        iMuestraIni =(t-1)*iMuestrasPorTrama + 1;
        aSenal = aSenalOrigen(iMuestraIni : iMuestraIni +
        iMuestrasPorVentana - 1)';


        %************************************************
        %           ESTIMACION DEL PITCH
        %************************************************
        [f0, A0] = wcgPitchAutocorrelacionTrama(bf0, af0, ...
        iMuestrasPorVentana, iFrecMuestreo, aSenal, Ts);

        aTrama(t).f0 = f0;
        aTrama(t).A0 = A0;

        %************************************************
        %     REALIZA ANALISIS LPC DE LA TRAMA
        %************************************************
        [a, k, G] = wcgAnalisisLPCTrama(aSenal, iMuestrasPorVentana,
        f0);
        aTrama(t).a = a;
        aTrama(t).k = k;
        aTrama(t).G = G;


        %*****************************************
        %     DECISION DE VOICE O UNVOICE
        %*****************************************
        if t >= 3          %A partir de la tercera trama
           if aTrama(t).A0 > UMBRAL
              if (aTrama(t-1).A0 < UMBRAL) & (aTrama(t-2).A0 > UMBRAL)
                 aTrama(t-1).A0 == UMBRAL + 0.01;
              end
           else
              if (aTrama(t-1).A0 > UMBRAL) & (aTrama(t-2).A0 > UMBRAL)
                 if aTrama(t).A0 >= 0.3
                    aTrama(t).A0 = UMBRAL + 0.01;
                       if (aTrama(t-1).A0 < UMBRAL) & (aTrama(t-2).A0
                       > UMBRAL)
                       aTrama(t-1).A0 = UMBRAL + 0.01;
                    end
                 end
              end
           end
```

```
         end

      end

      for t = 1:(iNroDeTramas+1)
         aTrama(t).f01 = (aTrama(t).f0)*(aTrama(t).A0 > UMBRAL);
      end

      f = medfilt1([aTrama.f01], 3);
      for i =1:length(f)
         aTrama(i).f02 = f(i);
      end
%**************************************************************************
**
%                        wcgPitchAutocorrelacionTrama.m
%**************************************************************************
**
function [f0, A0] = wcgPitchAutocorrelacionTrama(bf0, af0,
iMuestrasPorVentana ...
               , iFrecMuestreo, aSenal, Ts)

      %**********************************
      %     FILTRA LA SEÑAL
      %**********************************
      aSenal = filter(bf0,af0,aSenal);


   %**************************************************************************
**
      %
      %  BUSCA EL NIVEL DE RECORTE (center clipping autocorrelation
         method)
      %
      %  Señal = (----iTrama0a33-----)(----------)(----iTrama66a99----)
      %  iTrama0a33  = Max(primer tercio de la señal)
      %  iTrama66a99  = Max(tercer tercio de la señal)
      %
      %  Nivel de clipping = 68%(Maximo Entre iTrama0a33 e iTrama66a99)

   %**************************************************************************
**
      iPorcentajeClipping = 0.68;

      iTrama0a33 = floor(iMuestrasPorVentana/3);
      iMax0a33 = max(abs(aSenal(1:iTrama0a33)));

      iTrama66a99 = floor(2*iMuestrasPorVentana/3);
      iMax66a99 = max(abs(aSenal(iTrama66a99:iMuestrasPorVentana)));

      CL = iPorcentajeClipping * min(iMax0a33, iMax66a99);
```

```
    %*************************************************************************
*
        %      CENTRA LA ONDA RECORTADA Y CALCULA LA AUTOCORRELACION

    %*************************************************************************
*
        slope = 1;
        aSenalClip = (aSenal - CL*sign(aSenal)).*((aSenal > CL | aSenal <
-CL)...
                *slope);


        %****************************************
        %      AUTOCORRELACION DE LA SEÑAL
        %****************************************
        aSenalCorr = xcorr(aSenalClip,length(aSenalClip));
        aSenalCorr(1:length(aSenalClip))=[];
        aSenalCorr = aSenalCorr(1:length(aSenalClip));

        %****************************************
        %      NORMALIZANDO LA SEÑAL
        %****************************************
        aSenalCorr = aSenalCorr/aSenalCorr(1);

        iInicio = find(aSenalCorr < 0.3);
        start = max([20 iInicio(1)]);
        [A0, iIndex] = max(aSenalCorr(start:length(aSenalCorr)));

        Tpitch = (iIndex + start - 1)*Ts;          %Periodo Pitch
        f0 = 1000/Tpitch;                        %Por 1000 para que este en
Hz
```

```
%*************************************************************************
**
%                        wcgPitchCepstrum.m
%*************************************************************************
**

      function [aTrama, iNroDeTramas, iTamanoDeTrama,
      iMuestrasPorTrama] = wcgPitchCepstrum(h0, aSenalOrigen,
      iFrecMuestreo)

      %*****************************************
      %  CALCULO DEL PITCH POR CEPSTRUM
      %*****************************************
      iNroMuestras = length(aSenalOrigen);

      %*****************************************
      %  PARAMETROS
      %*****************************************
      iTamanoDeTrama = 20;          %son 20 ms
      iTraslape = 10;               %son 10 ms de traslape
      iVentanaDeAnalisis = iTamanoDeTrama + iTraslape;

      %*****************************************
      %              MUESTRAS
      %*****************************************
      iMuestrasPorTrama = floor((iTamanoDeTrama/1000)*iFrecMuestreo);
      iMuestrasPorVentana =
      floor((iVentanaDeAnalisis/1000)*iFrecMuestreo);
      iNroDeTramas = floor(iNroMuestras/iMuestrasPorTrama);

      %Completa la ultima trama ...
      iUltMuestra = iMuestrasPorTrama*iNroDeTramas +
iMuestrasPorVentana;
      iNroCeros = iUltMuestra-length(aSenalOrigen);
      aSenalOrigen((iUltMuestra - iNroCeros + 1):iUltMuestra)     =
      zeros(1,iNroCeros);

      for t = 1:(iNroDeTramas+1)

         %*************************************
         %           INICIALIZA
         %*************************************
         aTrama(t).f0 = 0;    %Valor de la frecuencia fundamental.

         %*********************************************
         %           TRAMA A ANALIZAR
         %*********************************************
         iMuestraIni =(t-1)*iMuestrasPorTrama + 1;
         aSenal = aSenalOrigen(iMuestraIni : iMuestraIni +
         iMuestrasPorVentana - 1)';

         %*********************************************
         %           ESTIMACION DEL PITCH
```

```
    %************************************************
    f0 = wcgPitchCepstrumTrama(iMuestrasPorVentana, iFrecMuestreo,
    aSenal); % Use the autocorr. method
    aTrama(t).f0 = f0;

    %************************************************
    %     REALIZA ANALISIS LPC DE LA TRAMA
    %************************************************
    [a, k, G] = wcgAnalisisLPCTrama(aSenal, iMuestrasPorVentana,
    f0);
    aTrama(t).a = a;
    aTrama(t).k = k;
    aTrama(t).G = G;


end

f = medfilt1([aTrama.f0], 3);
for i =1:length(f)
   aTrama(i).f02 = f(i);
end
```

```
%*************************************************************************
**
%                          wcgPitchCepstrumTrama.m
%*************************************************************************
**

function f0   = wcgPitchCepstrumTrama(iMuestrasPorVentana,
iFrecMuestreo, sSenal)

   sSenal = hamming(iMuestrasPorVentana).*sSenal;
   cn1 = rceps(sSenal);

   LF = floor(iFrecMuestreo/500);
   HF = floor(iFrecMuestreo/70);

   cn = cn1(LF:HF);
   [mx_cep ind]=max(cn);

   if mx_cep > 0.09 & ind >LF
      f0 = iFrecMuestreo/(LF+ind);
   else
      f0 = 0;
   end
```

```
%*************************************************************************
**
%                          wcgPitchSift.m
%*************************************************************************
**

%*************************************************************************
**%          Tesis:       Analisis LPC
%         Autor:       William Castro Grijalva
%         Funcion:     Proceso del Calculo del pitch por el método sift
%*************************************************************************
**function [aTrama, iNroDeTramas, iTamanoDeTrama, iMuestrasPorTrama] =
wcgPitchSift(h0, aSenalOrigen, iFrecMuestreo);

      iNroMuestras = length(aSenalOrigen);

      %*****************************************
      %  CREA UN FILTRO PASA BAJO CHEBYCHEV DE 800HZ
      %*****************************************
      Wn = 800/(iFrecMuestreo/2);      %Frecuencia del filtro
normalizada
      n = 6;                           %Orden del filtro
      [bf0,af0] = cheby1(n,0.5,Wn);

      %*****************************************
      %  PARAMETROS DE ANALISIS (20,10) (10,10)
      %*****************************************
      iTamanoDeTrama =  20;                                    %son 20 ms
      iTraslape = 10;                                  %son 10 ms de
      traslape
      iVentanaDeAnalisis = iTamanoDeTrama + iTraslape;

      iMuestrasPorTrama = floor((iTamanoDeTrama/1000)*iFrecMuestreo);
      iMuestrasPorVentana =
      floor((iVentanaDeAnalisis/1000)*iFrecMuestreo);

      D = 4;   %Orden de la decimacion
      Ts = 1/iFrecMuestreo;
      Td = D*Ts;
      k1 = [0:Ts:(iMuestrasPorVentana-1)*Ts];
      k2 = [0:Td:(iMuestrasPorVentana-1)*Ts];
      UMBRAL = 0.38;

      iNroDeTramas = floor(iNroMuestras/iMuestrasPorTrama);

      %*****************************************
      %     Completa la ultima trama ...
      %*****************************************
      iUltMuestra = iMuestrasPorTrama*iNroDeTramas +
iMuestrasPorVentana;
      iNroCeros = iUltMuestra-length(aSenalOrigen);
      aSenalOrigen((iUltMuestra - iNroCeros + 1):iUltMuestra)  =
      zeros(1,iNroCeros);
```

```
for t = 1:(iNroDeTramas+1)

   aTrama(t).f0 = 0;     %Valor de la frecuencia fundamental.
   aTrama(t).f01 = 0;       %Valor de la frecuencia fundamental
   (median).

   iMuestraIni =(t-1)*iMuestrasPorTrama + 1;
   aSenal = aSenalOrigen(iMuestraIni : iMuestraIni +
   iMuestrasPorVentana - 1)';

   %****************************************
   %     HACE EL CALCULO DEL PITCH
   %****************************************

   if t == 10000  %Indice que se va a mostrar los datos de la
      trama 30
      verGraficos = 1;
   else
      verGraficos = 0;
   end

   [f0, A0]= wcgPitchSiftTrama(bf0, af0, D, aSenal, Td,
   verGraficos);

   aTrama(t).A0 = A0;            %Amplitud Maxima del Pitch
   aTrama(t).f0 = f0;

   %*********************************************
   %     REALIZA ANALISIS LPC DE LA TRAMA
   %*********************************************
   [a, k, G] = wcgAnalisisLPCTrama(aSenal, iMuestrasPorVentana,
   f0);

   aTrama(t).a = a;
   aTrama(t).k = k;
   aTrama(t).G = G;

   %****************************************
   %     DECISION DE VOICE O UNVOICE
   %****************************************
   if t >= 3         %A partir de la tercera trama
      if aTrama(t).A0 > UMBRAL
         if (aTrama(t-1).A0 < UMBRAL) & (aTrama(t-2).A0 > UMBRAL)
            aTrama(t-1).A0 == UMBRAL + 0.01;
         end
      else
         if (aTrama(t-1).A0 > UMBRAL) & (aTrama(t-2).A0 > UMBRAL)
            if aTrama(t).A0 >= 0.3
               aTrama(t).A0 = UMBRAL + 0.01;
                  if (aTrama(t-1).A0 < UMBRAL) & (aTrama(t-2).A0
                  > UMBRAL)
                  aTrama(t-1).A0 = UMBRAL + 0.01;
```

```
                end
            end
        end
    end
  end
end

for t = 1:(iNroDeTramas+1)
   aTrama(t).f01 = (aTrama(t).f0)*(aTrama(t).A0 > UMBRAL);
end

for t = 1:(iNroDeTramas+1)
   aTrama(t).f02 = aTrama(t).f01 ;
end

return
```

```
%*************************************************************************
**
%                        wcgPitchSiftTrama.m
%*************************************************************************
**

function [f0, A0]= wcgPitchSiftTrama(bf0, af0, D ...
        , aSenal, Td, verGraficos);

        %**********************************
        %     FILTRA LA SEÑAL
        %**********************************
        aSenalIn = filter(bf0,af0,aSenal);

        %*********************************************************
        %     REALIZA UNA DECIMACIÓN DE LA SEÑAL DE 8 A 2 KHZ
        %*********************************************************
        aSenalW = decimate(aSenalIn, D);

        %****************************************
        %     PREENFASIS DE LA SEÑAL
        %****************************************
        aSenalPE = filter([1 -0.95], [1], aSenalW);

        %****************************************
        %     APLICANDO UNA VENTANA DE HAMMING
        %****************************************
        aSenalH = aSenalPE.*hamming(length(aSenalPE));

        %****************************************
        %     CALCULA LOS COEFICIENTES Ai
        %****************************************
        iOrdenLPC = 4;
        [ai e ki] = aryule(aSenalH, iOrdenLPC);

        %****************************************
        %     FILTRA LA SEÑAL CON EL FILTRO INVERSO
        %****************************************
           aSenalInv = filter(ki, 1, aSenalH);

        %****************************************
        %     AUTOCORRELACION DE LA SEÑAL
        %****************************************
        aSenalCorr = xcorr(aSenalInv,length(aSenalInv));
        aSenalCorr(1:length(aSenalInv))=[];
        aSenalCorr = aSenalCorr(1:length(aSenalInv));

        %****************************************
        %     NORMALIZANDO LA SEÑAL
        %****************************************
        aSenalCorr = aSenalCorr/aSenalCorr(1);
        aSenalCorr(1:10) = zeros(1,10);  %pone acero los primeros 2ms
        de señal
```

```
%*****************************************
%     CONSIDERANDO LOS VALORES > O
%*****************************************
aSenalCorr = aSenalCorr.*(aSenalCorr>0);
[aSenalSort, Posicion]  = sort(-aSenalCorr);

%*****************************************
%     HALLANDO LOS DOS VALORES MAXIMOS
%*****************************************
Amax1 = abs(aSenalSort(1));
Pos1 = Posicion(1);

Amax2 = abs(aSenalSort(2));
Pos2 = Posicion(2);

%*****************************************
%     INTERPOLANDO EL PRIMER MAXIMO
%*****************************************
K = 16;           %Constante de interpolacion
x = [Pos1-1 Pos1 Pos1+1];
y = aSenalCorr(Pos1-1:Pos1+1);
aSenalInter1 = wcgInterpolacion(x,y,K);

[aSenalSort, PosTemp]  = sort(-aSenalInter1);
%encuentra el maximo de la señal interpolada
A01  = abs(aSenalSort(1));
%Amplitud Maxima del Pitch
Tpitch1 = (Pos1-2)*Td + (PosTemp(1)-1)*(Td/K)  ;
%Periodo Pitch

%*****************************************
%     INTERPOLANDO EL SEGUNDO MAXIMO
%*****************************************
x = [Pos2-1 Pos2 Pos2+1];
y = aSenalCorr(Pos2-1:Pos2+1);
aSenalInter2 = wcgInterpolacion(x,y,K);

[aSenalSort, PosTemp]  = sort(-aSenalInter2);
%encuentra el maximo de la señal interpolada
A02  = abs(aSenalSort(1));
%Amplitud Maxima del Pitch
Tpitch2 = (Pos2-2)*Td + (PosTemp(1)-1)*(Td/K)  ;
%Periodo Pitch

if A01 > A02
   if abs(Tpitch1/2-Tpitch2) > 0.14
      Tpitch = Tpitch2;
      A0 = A02;
   else
      Tpitch = Tpitch1;
      A0 = A01;
   end
```

```
else
   if abs(Tpitch2/2-Tpitch1) > 0.14
      Tpitch = Tpitch1 ;
      A0 = A01;
   else
      Tpitch = Tpitch2;
      A0 = A02;
   end
end

f0 = 1/Tpitch;
%Por 1000 para que este en Hz
f0 = (f0 > 80).*(f0 < 320).*f0;


if (verGraficos == 1)

   figure(5)
   stem(aSenal);
   grid on;
   title('Señal Original');

   figure(6)
   stem(aSenalIn);
   grid on;
   title('Señal Filtrada Pasa Bajo');

   figure(7)
   stem(aSenalW);
   grid on;
   title('Señal Decimada');

   figure(8)
   stem(aSenalPE);
   grid on;
   title('Señal despues de Preenfasis');

   figure(9)
   stem(aSenalH);
   grid on;
   title('Señal depués de aplicarle una ventana Hamming');

   figure(10)
   stem(aSenalInv);
   grid on;
   title('Señal Después del Filtro Inverso');

   figure(11)
   stem(aSenalCorr);
   grid on;
   title('Señal Correlacionada');

   figure(12)
```

```
    stem(aSenalInter1);
    grid on;
    title('Señal Interpolada');

    figure(13)
    stem(aSenalInter2);
    grid on;
end
```

```
%*************************************************************************
**
%                          wcgSimulacion.m
%*************************************************************************
**

%*************************************************************************
**
%        Tesis:      Analisis LPC
%        Autor:      William Castro Grijalva
%        Función: Esta funcion controla las funciones de
%                 simulacion
%*************************************************************************
**
function wcgSimulacion(hFig, sAction);

   switch sAction
   case 'reproducir_sin'
      ReproducirSenalSintetizada(hFig);
   case 'sintetizar'
      wcgSintetizar(hFig);
   case 'mostrar_sim'
      ZoomSim(hFig);
   case 'guardar_sint'
      GuardarSenalSintetizada(hFig);
   end

%*************************************************************************
**
%        Función: Reproducir la señal completa
%*************************************************************************
**
function ReproducirSenalSintetizada(hFig);

   %Obtiene los datos que se van a analizar ...
   hAxes = findobj(hFig, 'Tag','axeSim02');
   hAxesLine = findobj(hAxes, 'Tag','axeLineaSim');
   XData = get(hAxesLine, 'XData');
   YData = get(hAxesLine, 'YData');

   if length(YData) == 0
      return;
   end

   %Obtiene los datos que se van a analizar ...
   hAxes = findobj(hFig, 'Tag','axeAnalisis');
   Parametros = get(hAxes,'UserData');

   %Reproduce el archivo
   sound(YData, Parametros(1));


function wcgSintetizar(hFig    )
```

```
      hTipo = findobj(hFig, 'Tag','cmbTipPit');
      iIndex = get(hTipo, 'Value');
      sString = cellstr(get(hTipo, 'String'));
      sTipPitch = char(sString(iIndex));

      %*********************************************************
      %        ANALISIS DE LA SEÑAL USANDO LPC
      %*********************************************************
      hAxes = findobj(hFig, 'Tag','axeAnalisis');
      hAxesLine = findobj(hFig, 'Tag','axeLinea');
      XData = get(hAxesLine, 'XData');
      YData = get(hAxesLine, 'YData');

      aParametro = get(hAxes, 'UserData');
      iFrecMuestreo = aParametro(1);

      aTiempo = get(hAxes, 'XLim');
      iTimeIni = aTiempo(1);
      iTimeFin = aTiempo(2);
      sSenalVisible =
YData(1+round(iTimeIni/1000*iFrecMuestreo):1:round(iTimeFin/1000*iFrecM
uestreo));


      switch sTipPitch
      case 'Autocorrelación'
         [aTrama, iNroDeTramas, iTamanoDeTrama, iMuestrasPorTrama] =
wcgPitchAutocorrelacion(hFig, sSenalVisible, iFrecMuestreo);
      case 'Sift'
         [aTrama, iNroDeTramas, iTamanoDeTrama, iMuestrasPorTrama] =
wcgPitchSift(hFig, sSenalVisible, iFrecMuestreo);
      case 'Cepstrum'
         [aTrama, iNroDeTramas, iTamanoDeTrama, iMuestrasPorTrama] =
wcgPitchCepstrum(hFig, sSenalVisible, iFrecMuestreo);
      end

      %*********************************************************
      %        SINTESIS DE LA SEÑAL USANDO LPC
      %*********************************************************
      aNuevaSenal = [];
      for w = 1:(iNroDeTramas+1)

         if aTrama(w).f02 == 0
               aExcitacion = randn(1, iMuestrasPorTrama);
               aExcitacion = aExcitacion/max(aExcitacion);
         else
               iMuestrasPitch = floor((1/aTrama(w).f02)*iFrecMuestreo);
               aExcitacion = zeros(1, iMuestrasPorTrama);

               iIndex = 1:iMuestrasPorTrama;
               aExcitacion(mod(iIndex,iMuestrasPitch)==0) = 1;
         end
```

```
        ai = [aTrama(w).a];

        %Filtro Inverso
        aSenalOut = sqrt(aTrama(w).G)*filter(1, ai, aExcitacion);

        %Deenfasis
        aSenalOut = filter([1], [1 -0.95], aSenalOut);

        aNuevaSenal = [aNuevaSenal aSenalOut];
    end

    i = 0;

    %************************************************************
    %        GRAFICA LA SEÑAL RECONSTRUIDA
    %************************************************************
    hAxes = findobj(hFig, 'Tag','axeSim02');
    axes(hAxes);

    aNuevaSenal = aNuevaSenal(1:length(aNuevaSenal)-1);
    aNuevaSenal = aNuevaSenal + mean(aNuevaSenal);

    Et = 1000*(length(aNuevaSenal))/iFrecMuestreo;
    t = (0:1000/iFrecMuestreo:Et-1/iFrecMuestreo);

    hLine = plot(t, aNuevaSenal);
    grid on;

    Ymin = min(aNuevaSenal);
    Ymax = max(aNuevaSenal);

    set(hAxes, 'YLim',    [1.1*Ymin 1.1*Ymax]);
    set(hAxes, 'XLim',    [iTimeIni iTimeFin]);

    set(hAxes, 'Tag', 'axeSim02');      %Esto se adiciona por que al
    setear los valores anteriores se borra el tag
    set(hLine,'Tag','axeLineaSim');     %Cambia color de fondo

function ZoomSim(hFig)

    hAxes = findobj(hFig, 'Tag', 'axeAnalisis');
  aTime = get(hAxes, 'XLim');

  hAxes = findobj(hFig, 'Tag', 'axeSim02');
  set(hAxes, 'XLim',    [aTime(1) aTime(2)]);

function GuardarSenalSintetizada(hFig);

  [sRuta,sNombreArchivo] = ShowDialogo('guardar','*.wav');
  sArchivo = [sRuta, sNombreArchivo];

  hFs = findobj(hFig, 'Tag','txtFs');
```

```
    sString = get(hFs, 'String');
    iFs = str2num(sString);
    if isempty(iFs)
          iFs = -1;
          return;
    end

    hNroBits = findobj(hFig, 'Tag','txtNroBit');
    sString = get(hNroBits , 'String');
    iNroBits = str2num(sString);
    if isempty(iNroBits)
          iNroBits = -1;
          return;
    end

    %Obtiene los datos que se van a analizar ...
    hAxes = findobj(hFig, 'Tag','axeSim02');
    hAxesLine = findobj(hFig, 'Tag','axeLineaSim');
    YData = get(hAxesLine, 'YData');

    if ~isempty(YData)
       wavwrite(YData, iFs, iNroBits, sArchivo);
    else
       msgbox('No hay datos simulados ...');
    end


%************************************************************************
**%   Función que muestra un dialogo para obtener un archivo
%************************************************************************
**function [sRuta,sArchivo] = ShowDialogo(sComando,sFileExt)

if (strcmp(sComando, 'abrir'))
   [sArchivo, sRuta] = uigetfile(sFileExt, 'Seleccione el archivo que
desea abrir:');
   if ((~isstr(sArchivo)) | ~min(size(sArchivo))), return; end
elseif (strcmp(sComando, 'guardar'))
   [sArchivo, sRuta]  = uiputfile(sFileExt, 'Seleccione el archivo
donde desea guardar');
   if ((~isstr(sArchivo)) | ~min(size(sArchivo))), return; end
else
   msgbox([' Comando desconocido ', sComando, '.']);
   sArchivo = '';
   sRuta = '';
end
```

```
%**************************************************************************
**
%                        wcgStatusAnalisisLarge.m
%**************************************************************************
**

%**************************************************************************
**%         Tesis:      Analisis LPC
%        Autor:      William Castro Grijalva
%        Funcion:    Interface Grafica de Usuario para los parametros
%**************************************************************************
**function wcgStatusAnalisisLarge(h0, sAction);

switch sAction
case 'crear'
   CrearGUI(h0);
case 'eliminar'
   EliminarGUI(h0);
end;

function CrearGUI(h0);

h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.8901960784313731 0.909803921568627
0.949019607843137], ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'ListboxTop',0, ...
   'Position',[51 11.25 459.75 45], ...
   'Style','frame', ...
   'Tag','FraPar01');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontWeight','bold', ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[52.5 41.25 115.5 13.5], ...
   'String','Frecuencia Muestreo (Hz):', ...
   'Style','text', ...
   'Tag','lblFs', ...
   'UserData',1737.49928396072);
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontWeight','bold', ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
```

```
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[52.5 27 115.5 13.5], ...
    'String','Nro de Muestras: (n)', ...
    'Style','text', ...
    'Tag','lblNroMue', ...
    'UserData',1737.49928396072);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'FontWeight','bold', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[210 27 99 13.5], ...
    'String','T. Fin.: (ms)', ...
    'Style','text', ...
    'Tag','lblTFin', ...
    'UserData',1737.49928396072);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'FontWeight','bold', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[210 41.25 99 13.5], ...
    'String','T. Ini.: (ms)', ...
    'Style','text', ...
    'Tag','lblTIni', ...
    'UserData',1737.49928396072);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'FontWeight','bold', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[309.75 41.25 48 13.5], ...
    'String','0', ...
    'Style','text', ...
    'Tag','txtTimIni', ...
    'UserData',0);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
```

```
    'FontWeight','bold', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[309.75 27 48 13.5], ...
    'Style','text', ...
    'Tag','txtTimFin', ...
    'String','0', ...
    'UserData',3559.415255463888);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'FontWeight','bold', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[168.75 41.25 40.5 13.5], ...
    'String','0', ...
    'Style','text', ...
    'Tag','txtFs', ...
    'UserData','1214.61548690671');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'FontWeight','bold', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[168.75 27 40.5 13.5], ...
    'String','0', ...
    'Style','text', ...
    'Tag','txtNroMue', ...
    'UserData',1214.61548690671);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'FontWeight','bold', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[358.5 41.25 99 13.5], ...
    'String','T.Actual.: (ms)', ...
    'Style','text', ...
    'Tag','lblTAct', ...
    'UserData',1737.49928396072);
h1 = uicontrol('Parent',h0, ...
```

```
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.878431372549021, ...
    'FontWeight','bold', ...
    'ForegroundColor',[0.121568627450981 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[458.25 41.25 51 13.5], ...
    'String','0', ...
    'Style','text', ...
    'Tag','txtTime');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.878431372549021, ...
    'FontWeight','bold', ...
    'ForegroundColor',[0.121568627450981 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[458.25 27 51 13.5], ...
    'String','0', ...
    'Style','text', ...
    'Tag','txtTimTot');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.878431372549021, ...
    'FontWeight','bold', ...
    'ForegroundColor',[0.121568627450981 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[358.5 27 99 13.5], ...
    'String','Tiempo Total: (seg)', ...
    'Style','text', ...
    'Tag','lblTTot');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.121568627450981 0.149019607843137
0.396078431372549], ...
    'FontWeight','bold', ...
    'ForegroundColor',[0.760784313725491 0.8 0.894117647058823], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[51 57 158.0 16], ...
    'String','Parámetros:', ...
    'Style','text', ...
    'Tag','lblPar', ...
    'UserData',1737.49928396072);
h1 = uicontrol('Parent',h0, ...
```

```
   'BackgroundColor',[0.8901960784313731 0.909803921568627
0.949019607843137], ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'ListboxTop',0, ...
   'Position',[281 75 199 22], ...
   'String','Parámetros', ...
   'Style','frame', ...
   'Tag','fraPar');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontWeight','bold', ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[211.5 57 97.5 13.5], ...
   'String','Nro de Bits:', ...
   'Style','text', ...
   'Tag','lblNroBit');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontWeight','bold', ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','right', ...
   'ListboxTop',0, ...
   'Position',[309.75 57 48 13.5], ...
   'String','0', ...
   'Style','text', ...
   'Tag','txtNroBit');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontWeight','bold', ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[52.5 12.75 115.5 13.5], ...
   'String','Archivo:', ...
   'Style','text', ...
   'Tag','lblArchivo', ...
   'UserData',1737.49928396072);
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
```

```
    'FontWeight','bold', ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[168.75 12.75 340.5 13.5], ...
    'Style','text', ...
    'Tag','lblRutaArchivo', ...
    'UserData',1737.49928396072);

function EliminarGUI(h0);
    hObj = findobj(h0, 'Tag','fraPar');
    delete(hObj);
    hObj = findobj(h0, 'Tag','FraPar01');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblFs');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblNroMue');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblTFin');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblTIni');
    delete(hObj);
    hObj = findobj(h0, 'Tag','txtTimIni');
    delete(hObj);
    hObj = findobj(h0, 'Tag','txtTimFin');
    delete(hObj);
    hObj = findobj(h0, 'Tag','txtFs');
    delete(hObj);
    hObj = findobj(h0, 'Tag','txtNroMue');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblTAct');
    delete(hObj);
    hObj = findobj(h0, 'Tag','txtTime');
    delete(hObj);
    hObj = findobj(h0, 'Tag','txtTimTot');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblTTot');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblPar');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblNroBit');
    delete(hObj);
    hObj = findobj(h0, 'Tag','txtNroBit');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblArchivo');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblRutaArchivo');
    delete(hObj);
```

```
%************************************************************************
**
%                        wcgStatusAnalisisSmall.m
%************************************************************************
**

%************************************************************************
**%         Tesis:      Analisis LPC
%         Autor:      William Castro Grijalva
%         Funcion:    Interface Grafica de Usuario para losparametros de
          la señal
%************************************************************************
**function fig = wcgStatusAnalisisSmall(h0, sAction);

switch sAction
case 'crear'
   CrearGUI(h0);
case 'eliminar'
   EliminarGUI(h0);
end;

function CrearGUI(h0);

h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.8901960784313731 0.909803921568627
0.949019607843137], ...
   'FontSize',5, ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'ListboxTop',0, ...
   'Position',[168.75 10.5 331.5 43.5], ...
   'Style','frame', ...
   'Tag','FraPar01');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontSize',5, ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[169.5 33 65.25 9.75], ...
   'String','F. Muestreo (Hz):', ...
   'Style','text', ...
   'Tag','lblFs', ...
   'UserData',1737.49928396072);
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontSize',5, ...
```

```
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[169.5 22.5 65.25 9.75], ...
   'String','N° Muestras: (n)', ...
   'Style','text', ...
   'Tag','lblNroMue', ...
   'UserData',1737.49928396072);
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontSize',5, ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[276.75 22.5 60 9.75], ...
   'String','T. Fin.: (ms)', ...
   'Style','text', ...
   'Tag','lblTFin', ...
   'UserData',1737.49928396072);
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontSize',5, ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[276.75 33 60 9.75], ...
   'String','T. Ini.: (ms)', ...
   'Style','text', ...
   'Tag','lblTIni', ...
   'UserData',1737.49928396072);
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontSize',5, ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','right', ...
   'ListboxTop',0, ...
   'Position',[337.5 33 48 9.75], ...
   'String','0', ...
   'Style','text', ...
   'Tag','txtTimIni', ...
   'UserData',0);
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
```

```
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontSize',5, ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','right', ...
   'ListboxTop',0, ...
   'Position',[337.5 22.5 48 9.75], ...
   'String','0', ...
   'Style','text', ...
   'Tag','txtTimFin', ...
   'UserData',3559.415255463888);
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontSize',5, ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','right', ...
   'ListboxTop',0, ...
   'Position',[235.5 33 40.5 9.75], ...
   'String','0', ...
   'Style','text', ...
   'Tag','txtFs', ...
   'UserData','1214.61548690671');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontSize',5, ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','right', ...
   'ListboxTop',0, ...
   'Position',[235.5 22.5 40.5 9.75], ...
   'String','0', ...
   'Style','text', ...
   'Tag','txtNroMue', ...
   'UserData',1214.61548690671);
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontSize',5, ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[386.25 33 60 9.75], ...
   'String','T.Actual.: (ms)', ...
   'Style','text', ...
   'Tag','lblTAct', ...
```

```
    'UserData',1737.49928396072);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'FontSize',5, ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[447 33 51 9.75], ...
    'String','1659.29', ...
    'Style','text', ...
    'Tag','txtTime');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'FontSize',5, ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[447 22.5 51 9.75], ...
    'String','0', ...
    'Style','text', ...
    'Tag','txtTimTot');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
    'FontSize',5, ...
    'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[386.25 22.5 60 9.75], ...
    'String','T. Total: (seg)', ...
    'Style','text', ...
    'Tag','lblTTot');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'FontSize',5, ...
    'ForegroundColor',[0.7607843137254901 0.8 0.894117647058823], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[170.25 43.5 166.5 9.75], ...
    'String','Parámetros:', ...
    'Style','text', ...
    'Tag','lblPar', ...
    'UserData',1737.49928396072);
```

```
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontSize',5, ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[337.5 43.5 108.75 9.75], ...
   'String','Nro de Bits:', ...
   'Style','text', ...
   'Tag','lblNroBit');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontSize',5, ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','right', ...
   'ListboxTop',0, ...
   'Position',[447 43.5 50.25 9.75], ...
   'String','16.00', ...
   'Style','text', ...
   'Tag','txtNroBit');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontSize',5, ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[169.5 12 65.25 9.75], ...
   'String','Archivo:', ...
   'Style','text', ...
   'Tag','lblArchivo', ...
   'UserData',1737.49928396072);
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.694117647058824 0.737254901960784
0.87843137254902], ...
   'FontSize',5, ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'HorizontalAlignment','left', ...
   'ListboxTop',0, ...
   'Position',[235.5 12 262.5 9.75], ...
   'String','No existen datos para realizar el análisis ...', ...
   'Style','text', ...
   'Tag','lblRutaArchivo', ...
```

```
    'UserData',1737.49928396072);

function EliminarGUI(h0);
    hObj = findobj(h0, 'Tag','FraPar01');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblFs');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblNroMue');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblTFin');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblTIni');
    delete(hObj);
    hObj = findobj(h0, 'Tag','txtTimIni');
    delete(hObj);
    hObj = findobj(h0, 'Tag','txtTimFin');
    delete(hObj);
    hObj = findobj(h0, 'Tag','txtFs');
    delete(hObj);
    hObj = findobj(h0, 'Tag','txtNroMue');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblTAct');
    delete(hObj);
    hObj = findobj(h0, 'Tag','txtTime');
    delete(hObj);
    hObj = findobj(h0, 'Tag','txtTimTot');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblTTot');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblPar');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblNroBit');
    delete(hObj);
    hObj = findobj(h0, 'Tag','txtNroBit');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblArchivo');
    delete(hObj);
    hObj = findobj(h0, 'Tag','lblRutaArchivo');
    delete(hObj);
```

```
%*********************************************************************
**
%                        wcgToolbarAnalisis.m
%*********************************************************************
**

%*********************************************************************
**%         Tesis:      Analisis LPC
%        Autor:      William Castro Grijalva
%        Funcion:    Interface Grafica de Usuario de la Barra de
Herramientas
%*********************************************************************
**function wcgToolbarAnalisis(h0, sAction)

switch sAction
case 'crear'
   CrearGUI(h0);
case 'eliminar'
   EliminarGUI(h0);
end;

function CrearGUI(h0);

h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.647058823529412 0.705882352941176
0.854901960784314], ...
   'ListboxTop',0, ...
   'Position',[11.25 328.5 504.75 30.75], ...
   'Style','frame', ...
   'Tag','fraBot');
h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'Callback','close(gcf)', ...
   'ForegroundColor',[0.8117647058823531 0.843137254901961
0.913725490196078], ...
   'ListboxTop',0, ...
   'Position',[469.25 332.25 42.75 24], ...
   'String','Salir', ...
   'Tag','cmdSalir', ...
   'TooltipString','Haga click para salir');

h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'Callback','wcgZoom(''In'', 1)', ...
   'ForegroundColor',[0.8117647058823531 0.843137254901961
0.913725490196078], ...
   'ListboxTop',0, ...
   'Position',[150 332.25 42.75 24], ...
```

```
    'String','Zoom in', ...
    'Tag','cmdZoomIn', ...
    'TooltipString','Haga click para hacer zoom');

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'Callback','wcgZoom(''Out'', 1)', ...
    'ForegroundColor',[0.8117647058823531 0.843137254901961
0.913725490196078], ...
    'ListboxTop',0, ...
    'Position',[105 332.25 42.75 24], ...
    'String','Zoom Out', ...
    'Tag','cmdZoomOut', ...
    'TooltipString','Haga click para hacer zoom');

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ForegroundColor',[0.7607843137254901 0.8 0.894117647058823], ...
    'ListboxTop',0, ...
    'Position',[60 332.25 42.75 24], ...
    'String','Guardar', ...
    'Tag','cmdGuardar', ...
    'TooltipString','Salir', ...
    'UserData','[ ]');

    sCallback = sprintf('wcgSimulacion(''Guardar'', %d)',h0);
    set(h1, 'Callback',sCallback);

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
    'ForegroundColor',[0.7607843137254901 0.8 0.894117647058823], ...
    'ListboxTop',0, ...
    'Position',[16.5 332.25 41.25 24], ...
    'String','Abrir', ...
    'Tag','cmdAbrir', ...
    'TooltipString','Salir', ...
    'UserData','[ ]');

    sCallback = sprintf('wcgArchivo(''LeerArchivo'', %d)',h0);
    set(h1, 'Callback',sCallback);

    hObj = findobj(h0,'Tag','txtVistas');
    if isempty(hObj)     %la primera vez crea el txtVista
       h1 = uicontrol('Parent',h0, ...
       'Units','points', ...
       'BackgroundColor',[0.745098039215686 0.776470588235294
0.894117647058823], ...
```

```
      'ForegroundColor',[0.07843137254901961 0.207843137254902
0.729411764705882], ...
      'HorizontalAlignment','left', ...
      'ListboxTop',0, ...
      'Position',[178.5 371.25 434.25 18.75], ...
      'String','[]', ...
      'Style','text', ...
      'Tag','txtVistas', ...
      'UserData',[1 0 4999.546485260771], ...
      'Visible','off');
   end

h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'ForegroundColor',[0.8117647058823531 0.843137254901961
0.913725490196078], ...
   'ListboxTop',0, ...
   'Position',[195 332.25 42.75 24], ...
   'String','Reproducir', ...
   'Tag','cmdReproducir', ...
   'TooltipString','Haga click para reproducir');

   sCallback = sprintf('wcgMouse(''cmdPlayADQ_OnClick'', %3.4f)',h0);
   set(h1, 'Callback',sCallback);


h1 = uicontrol('Parent',h0, ...
   'Units','points', ...
   'BackgroundColor',[0.647058823529412 0.705882352941176
0.854901960784314], ...
   'ForegroundColor',[0.12156862745098 0.149019607843137
0.396078431372549], ...
   'ListboxTop',0, ...
   'Position',[300 332.25 100 24], ...
   'String','Reproducir Region', ...
   'Style','checkbox', ...
   'Tag','chkRegion');


function    EliminarGUI(h0);

   hObj = findobj(h0,'Tag','fraBot');
   delete(hObj);

   hObj = findobj(h0,'Tag','cmdSalir');
   delete(hObj);

   hObj = findobj(h0,'Tag','cmdZoomIn');
   delete(hObj);
```

```
   hObj = findobj(h0,'Tag','cmdZoomOut');
   delete(hObj);

   hObj = findobj(h0,'Tag','cmdGuardar');
   delete(hObj);

   hObj = findobj(h0,'Tag','cmdAbrir');
   delete(hObj);

   hObj = findobj(h0,'Tag','cmdReproducir');
   delete(hObj);

   hObj = findobj(h0,'Tag','chkRegion');
   delete(hObj);
%**************************************************************************
**
%                        wcgVerGUIAdquisicion.m
%**************************************************************************
**

%**************************************************************************
**%         Tesis:      Analisis LPC
%        Autor:      William Castro Grijalva
%        Funcion:    Crear Interface Grafica de Usuario para el módulo
         de adquisición
%**************************************************************************
**function wcgVerGUIAdquisicion(h0);

%Detecta de donde viene ...
sTag = get(h0, 'Tag');
wcgConfiguracion(h0, 'crear_archivo_param');

switch sTag
case 'wcgLPC'

   wcgOpcionesLPC(h0, 'eliminar');           %crea opciones de LPC
   wcgAxesLPC(h0, 'eliminar')                %crea los nuevos ejes
   wcgStatusAnalisisSmall(h0, 'eliminar');      %crear el status bar

   wcgAxesAdquisicion(h0, 'crear');
   wcgOpcionesAdquisicion(h0, 'crear');
   wcgMenuAdquisicion(h0, 'crear');
   wcgStatusAnalisisLarge(h0, 'crear');

case 'wcgAnalisis'

   wcgAxesAnalisis(h0, 'eliminar');       %elimina el eje de analisis
   wcgAxesAdquisicion(h0, 'crear');
   wcgOpcionesAdquisicion(h0, 'crear');
   wcgMenuAdquisicion(h0, 'crear');

case 'wcgEspectrograma'
```

```
    wcgAxesAnalisis(h0, 'eliminar');
    wcgAxesEspectrograma(h0, 'eliminar');
    wcgOpcionesEspectrograma(h0, 'eliminar');
    wcgStatusAnalisisSmall(h0, 'eliminar');      %crear el status bar

    wcgAxesAdquisicion(h0, 'crear');
    wcgOpcionesAdquisicion(h0, 'crear');
    wcgMenuAdquisicion(h0, 'crear');
    wcgStatusAnalisisLarge(h0, 'crear');

case 'wcgAdquisicion'
    return;

case 'wcgDEP'

    wcgAxesAnalisis(h0, 'eliminar');
    wcgAxesDEP(h0, 'eliminar');

    wcgAxesAdquisicion(h0, 'crear');
    wcgOpcionesAdquisicion(h0, 'crear');
    wcgMenuAdquisicion(h0, 'crear');

case 'wcgPitch'

    wcgAxesAnalisis(h0, 'eliminar');
    wcgAxesPitch(h0, 'eliminar');

    wcgAxesAdquisicion(h0, 'crear');
    wcgOpcionesAdquisicion(h0, 'crear');
    wcgMenuAdquisicion(h0, 'crear');

case 'wcgSimulacion'

    wcgAxesSimulacion(h0, 'eliminar');
    wcgOpcionesSimulacion(h0, 'eliminar');
    wcgStatusAnalisisSmall(h0, 'eliminar');

    wcgAxesAdquisicion(h0, 'crear');
    wcgOpcionesAdquisicion(h0, 'crear');
    wcgMenuAdquisicion(h0, 'crear');
    wcgStatusAnalisisLarge(h0, 'crear');

end

wcgConfiguracion(h0, 'cargar_archivo_param');
set(h0, 'Tag', 'wcgAdquisicion');
set(h0, 'Name', 'Módulo de Adquisición');


%**************************************************************************
**
%                        wcgVerGUIAnalisis.m
```

```
%**************************************************************************
**

%**************************************************************************
**%         Tesis:      Analisis LPC
%         Autor:      William Castro Grijalva
%         Funcion:    Crear Interface Grafica de Usuario para el módulo
          de Analisis
%**************************************************************************
**function wcgVerGUIAnalisis(h0);

%Detecta de donde viene ...
sTag = get(h0, 'Tag');
wcgConfiguracion(h0, 'crear_archivo_param');

switch sTag
case 'wcgLPC'

   wcgOpcionesLPC(h0, 'eliminar');           %crea opciones de LPC
   wcgAxesLPC(h0, 'eliminar')                %crea los nuevos ejes
   wcgStatusAnalisisSmall(h0, 'eliminar');      %crear el status bar
   wcgAxesAnalisis(h0, 'crear');       %elimina el eje de analisis
   wcgStatusAnalisisLarge(h0, 'crear');         %elimina el status
large

case 'wcgAnalisis'

   return;

case 'wcgEspectrograma'

   wcgAxesAnalisis(h0, 'eliminar');
   wcgAxesEspectrograma(h0, 'eliminar');
   wcgOpcionesEspectrograma(h0, 'eliminar');
   wcgStatusAnalisisSmall(h0, 'eliminar');      %crear el status bar

   wcgAxesAnalisis(h0, 'crear');       %elimina el eje de analisis
   wcgStatusAnalisisLarge(h0, 'crear');         %elimina el status
large


case 'wcgAdquisicion'

   wcgAxesAdquisicion(h0, 'eliminar');
   wcgOpcionesAdquisicion(h0, 'eliminar');
   wcgMenuAdquisicion(h0, 'eliminar');
   wcgAxesAnalisis(h0, 'crear');

case 'wcgPitch'

   wcgAxesAnalisis(h0, 'eliminar');
   wcgAxesPitch(h0, 'eliminar');
```

```
   wcgAxesAnalisis(h0, 'crear');

case 'wcgDEP'

   wcgAxesAnalisis(h0, 'eliminar');
   wcgAxesDEP(h0, 'eliminar');
   wcgAxesAnalisis(h0, 'crear');

case 'wcgSimulacion'

   wcgAxesSimulacion(h0, 'eliminar');
   wcgOpcionesSimulacion(h0, 'eliminar');
   wcgStatusAnalisisSmall(h0, 'eliminar');

   wcgAxesAnalisis(h0, 'crear');
   wcgStatusAnalisisLarge(h0, 'crear');          %elimina el status
large

end

wcgConfiguracion(h0, 'cargar_archivo_param');
set(h0, 'Tag', 'wcgAnalisis');
set(h0, 'Name', 'Módulo de Análisis');
```

```
%*************************************************************************
**
%                        wcgVerGUIDEP.m
%*************************************************************************
**

%*************************************************************************
**%         Tesis:      Analisis LPC
%         Autor:      William Castro Grijalva
%         Funcion:    Crear Interface Grafica de Usuario para el módulo
          de DEP
%*************************************************************************
**function wcgVerGUIDEP(h0);

   if ~wcgConfiguracion(h0, 'Hay_Datos_Msg')
      return;
   end

%Detecta de donde viene ...
sTag = get(h0, 'Tag');
wcgConfiguracion(h0, 'crear_archivo_param');

switch sTag
case 'wcgLPC'

   wcgOpcionesLPC(h0, 'eliminar');           %crea opciones de LPC
   wcgAxesLPC(h0, 'eliminar')                %crea los nuevos ejes
   wcgStatusAnalisisSmall(h0, 'eliminar');      %crear el status bar

   wcgAxesAnalisis(h0, 'crear');
   wcgAxesDEP(h0, 'crear');
   wcgStatusAnalisisLarge(h0, 'crear');      %crear el status bar

case 'wcgAnalisis'

   wcgAxesAnalisis(h0, 'eliminar');       %elimina el eje de analisis

   wcgAxesAnalisis(h0, 'crear');
   wcgAxesDEP(h0, 'crear');

case 'wcgEspectrograma'

   wcgAxesAnalisis(h0, 'eliminar');
   wcgAxesEspectrograma(h0, 'eliminar');
   wcgOpcionesEspectrograma(h0, 'eliminar');
   wcgStatusAnalisisSmall(h0, 'eliminar');      %crear el status bar

   wcgAxesAnalisis(h0, 'crear');
   wcgAxesDEP(h0, 'crear');
   wcgStatusAnalisisLarge(h0, 'crear');      %crear el status bar

case 'wcgAdquisicion'
```

```
    wcgAxesAdquisicion(h0, 'eliminar');
    wcgOpcionesAdquisicion(h0, 'eliminar');
    wcgMenuAdquisicion(h0, 'eliminar');

    wcgAxesAnalisis(h0, 'crear');
    wcgAxesDEP(h0, 'crear');

case 'wcgPitch'

    wcgAxesAnalisis(h0, 'eliminar');
    wcgAxesPitch(h0, 'eliminar');

    wcgAxesAnalisis(h0, 'crear');
    wcgAxesDEP(h0, 'crear');

case 'wcgDEP'
    return

case 'wcgSimulacion'

    wcgAxesSimulacion(h0, 'eliminar');
    wcgOpcionesSimulacion(h0, 'eliminar');
    wcgStatusAnalisisSmall(h0, 'eliminar');

    wcgAxesAnalisis(h0, 'crear');
    wcgAxesDEP(h0, 'crear');
    wcgStatusAnalisisLarge(h0, 'crear');

end

wcgConfiguracion(h0, 'cargar_archivo_param');
set(h0, 'Tag', 'wcgDEP');
set(h0, 'Name', 'Módulo Análisis - Densidad Espectral de Energia');

cambiarPosicionDeEje(h0);
wcgDensidadEP(h0, 'mostrar_dep');


function hacerInvisibleAxeAnalisis(h0);

    hAxes = findobj(h0,'Tag','axeAnalisis');
    set(hAxes, 'Visible', 'off');

    hLinea = findobj(hAxes,'Tag','axeLinea');
    set(hLinea, 'Visible', 'off');

    hLinea = findobj(hAxes,'Tag','LineaY');
    set(hLinea, 'Visible', 'off');

    hLinea = findobj(hAxes,'Tag','LineaY2');
    set(hLinea, 'Visible', 'off');
```

```
function cambiarPosicionDeEje(h0);

   hAxes = findobj(h0,'Tag','axeAnalisis');
   set(hAxes, 'Position',[72 293 616 136]);
```

```
%*************************************************************************
**
%                        wcgVerGUIEspectrograma.m
%*************************************************************************
**

%*************************************************************************
**%         Tesis:      Analisis LPC
%         Autor:      William Castro Grijalva
%         Funcion:    Crear Interface Grafica de Usuario para el módulo
          del espectrograma
%*************************************************************************
**function wcgVerGUIEspectrograma(h0);

   if ~wcgConfiguracion(h0, 'Hay_Datos_Msg')
      return;
   end

sTag = get(h0, 'Tag');
wcgConfiguracion(h0, 'crear_archivo_param');

switch sTag
case 'wcgLPC'

   wcgOpcionesLPC(h0, 'eliminar');           %crea opciones de LPC
   wcgAxesLPC(h0, 'eliminar')                %crea los nuevos ejes

   wcgAxesAnalisis(h0, 'crear');
   wcgAxesEspectrograma(h0, 'crear');
   wcgOpcionesEspectrograma(h0, 'crear');

case 'wcgAnalisis'

   wcgAxesAnalisis(h0, 'eliminar');       %elimina el eje de analisis
   wcgStatusAnalisisLarge(h0, 'eliminar');

   wcgAxesAnalisis(h0, 'crear');
   wcgAxesEspectrograma(h0, 'crear');
   wcgOpcionesEspectrograma(h0, 'crear');
   wcgStatusAnalisisSmall(h0, 'crear');      %crear el status bar

case 'wcgEspectrograma'
   return;

case 'wcgAdquisicion'

   wcgAxesAdquisicion(h0, 'eliminar');
   wcgOpcionesAdquisicion(h0, 'eliminar');
   wcgMenuAdquisicion(h0, 'eliminar');
   wcgStatusAnalisisLarge(h0, 'eliminar');      %crear el status bar


   wcgAxesAnalisis(h0, 'crear');
```

```
   wcgAxesEspectrograma(h0, 'crear');
   wcgOpcionesEspectrograma(h0, 'crear')
   wcgStatusAnalisisSmall(h0, 'crear');      %crear el status bar

case 'wcgDEP'

   wcgAxesAnalisis(h0, 'eliminar');
   wcgAxesDEP(h0, 'eliminar');
   wcgStatusAnalisisLarge(h0, 'eliminar')       %crear el status bar

   wcgAxesAnalisis(h0, 'crear');
   wcgAxesEspectrograma(h0, 'crear')
   wcgOpcionesEspectrograma(h0, 'crear');
   wcgStatusAnalisisSmall(h0, 'crear');      %crear el status bar

case 'wcgPitch'
   wcgAxesAnalisis(h0, 'eliminar');
   wcgAxesPitch(h0, 'eliminar');
   wcgStatusAnalisisLarge(h0, 'eliminar');      %crear el status bar

   wcgAxesAnalisis(h0, 'crear');
   wcgAxesEspectrograma(h0, 'crear');
   wcgOpcionesEspectrograma(h0, 'crear');
   wcgStatusAnalisisSmall(h0, 'crear');      %crear el status bar

case 'wcgSimulacion'

   wcgAxesSimulacion(h0, 'eliminar');
   wcgOpcionesSimulacion(h0, 'eliminar');
   wcgStatusAnalisisSmall(h0, 'eliminar');

   wcgAxesAnalisis(h0, 'crear');
   wcgAxesEspectrograma(h0, 'crear');
   wcgOpcionesEspectrograma(h0, 'crear');
   wcgStatusAnalisisSmall(h0, 'crear');      %crear el status bar

end

   wcgConfiguracion(h0, 'cargar_archivo_param');
   set(h0, 'Tag', 'wcgEspectrograma');
   set(h0, 'Name', 'Módulo Análisis - Espectrograma');
   cambiarPosicionDeEje(h0);
   wcgEspectrograma(h0, 'mostrar_espectro');

function hacerInvisibleAxeAnalisis(h0);



   hAxes = findobj(h0,'Tag','axeAnalisis')
   set(hAxes, 'Visible', 'off');

   hLinea = findobj(hAxes,'Tag','axeLinea');
   set(hLinea, 'Visible', 'off');
```

```
   hLinea = findobj(hAxes,'Tag','LineaY');
   set(hLinea, 'Visible', 'off');

   hLinea = findobj(hAxes,'Tag','LineaY2');
   set(hLinea, 'Visible', 'off');

function cambiarPosicionDeEje(h0);

   hAxes = findobj(h0,'Tag','axeAnalisis');
   %set(hAxes, 'Position',[221 290 443 142]);
   set(hAxes, 'Position',[226 293 461 136]);

%**************************************************************************
**
%                        wcgVerGUILPC.m
%**************************************************************************
**

%**************************************************************************
***         Tesis:      Analisis LPC
%         Autor:      William Castro Grijalva
%         Funcion:    Crear Interface Grafica de Usuario para el módulo
          de Analisis LPC
%**************************************************************************
**function wcgVerGUILPC(h0)

%Detecta de donde viene ...
sTag = get(h0, 'Tag');
wcgConfiguracion(h0, 'crear_archivo_param');

switch sTag
case 'wcgLPC'
   return;
case 'wcgAnalisis'

   wcgAxesAnalisis(h0, 'eliminar');       %elimina el eje de analisis
   wcgStatusAnalisisLarge(h0, 'eliminar');         %elimina el status
large

   wcgOpcionesLPC(h0, 'crear');           %crea opciones de LPC
   wcgAxesLPC(h0, 'crear')                %crea los nuevos ejes
   wcgStatusAnalisisSmall(h0, 'crear');      %crear el status bar

case 'wcgEspectrograma'

   wcgAxesAnalisis(h0, 'eliminar');
   wcgAxesEspectrograma(h0, 'eliminar');
   wcgOpcionesEspectrograma(h0, 'eliminar');

   wcgOpcionesLPC(h0, 'crear');           %crea opciones de LPC
   wcgAxesLPC(h0, 'crear')                %crea los nuevos ejes
   %wcgStatusAnalisisSmall(h0, 'crear');     %crear el status bar
```

```
case 'wcgAdquisicion'

   wcgAxesAdquisicion(h0, 'eliminar');
   wcgOpcionesAdquisicion(h0, 'eliminar');
   wcgMenuAdquisicion(h0, 'eliminar');
   wcgStatusAnalisisLarge(h0, 'eliminar');          %elimina el status
large

   wcgOpcionesLPC(h0, 'crear');              %crea opciones de LPC
   wcgAxesLPC(h0, 'crear')                   %crea los nuevos ejes
   wcgStatusAnalisisSmall(h0, 'crear');         %crear el status bar

case 'wcgDEP'

   wcgAxesAnalisis(h0, 'eliminar');
   wcgAxesDEP(h0, 'eliminar');
   wcgStatusAnalisisLarge(h0, 'eliminar');          %elimina el status
large

   wcgOpcionesLPC(h0, 'crear');              %crea opciones de LPC
   wcgAxesLPC(h0, 'crear')                   %crea los nuevos ejes
   wcgStatusAnalisisSmall(h0, 'crear');         %crear el status bar

case 'wcgPitch'

   wcgAxesAnalisis(h0, 'eliminar');
   wcgAxesPitch(h0, 'eliminar');
   wcgStatusAnalisisLarge(h0, 'eliminar');          %elimina el status
large

   wcgOpcionesLPC(h0, 'crear');              %crea opciones de LPC
   wcgAxesLPC(h0, 'crear')                   %crea los nuevos ejes
   wcgStatusAnalisisSmall(h0, 'crear');         %crear el status bar

case 'wcgSimulacion'

   wcgAxesSimulacion(h0, 'eliminar');
   wcgOpcionesSimulacion(h0, 'eliminar');
   wcgStatusAnalisisSmall(h0, 'eliminar');

   wcgOpcionesLPC(h0, 'crear');              %crea opciones de LPC
   wcgAxesLPC(h0, 'crear')                   %crea los nuevos ejes
   wcgStatusAnalisisSmall(h0, 'crear');         %crear el status bar

end

wcgConfiguracion(h0, 'cargar_archivo_param');
set(h0, 'Tag', 'wcgLPC');
set(h0, 'Name', 'Módulo Análisis - Análisis Predictivo Lineal');
```

```
%*************************************************************************
**
%                        wcgVerGUIPitch.m
%*************************************************************************
**

%*************************************************************************
**%         Tesis:      Analisis LPC
%        Autor:      William Castro Grijalva
%        Funcion:    Crear Interface Grafica de Usuario para el módulo
del calculo del pitch
%*************************************************************************
**function wcgVerGUIPitch(h0, sTipo);

   if ~wcgConfiguracion(h0, 'Hay_Datos_Msg')
      return;
   end

%Detecta de donde viene ...
sTag = get(h0, 'Tag');
wcgConfiguracion(h0, 'crear_archivo_param');

switch sTag
case 'wcgLPC'

   wcgOpcionesLPC(h0, 'eliminar');           %crea opciones de LPC
   wcgAxesLPC(h0, 'eliminar')                %crea los nuevos ejes
   wcgStatusAnalisisSmall(h0, 'eliminar');      %crear el status bar

   wcgAxesAnalisis(h0, 'crear');
   wcgAxesPitch(h0, 'crear');
   wcgStatusAnalisisLarge(h0, 'crear');      %crear el status bar

case 'wcgAnalisis'

   wcgAxesAnalisis(h0, 'eliminar');       %elimina el eje de analisis

   wcgAxesAnalisis(h0, 'crear');
   wcgAxesPitch(h0, 'crear');

case 'wcgEspectrograma'

   wcgAxesAnalisis(h0, 'eliminar');
   wcgAxesEspectrograma(h0, 'eliminar');
   wcgOpcionesEspectrograma(h0, 'eliminar');
   wcgStatusAnalisisSmall(h0, 'eliminar');      %crear el status bar

   wcgAxesAnalisis(h0, 'crear');
   wcgAxesPitch(h0, 'crear');
   wcgStatusAnalisisLarge(h0, 'crear');      %crear el status bar

case 'wcgAdquisicion'
```

```
    wcgAxesAdquisicion(h0, 'eliminar');
    wcgOpcionesAdquisicion(h0, 'eliminar');
    wcgMenuAdquisicion(h0, 'eliminar');

    wcgAxesAnalisis(h0, 'crear');
    wcgAxesPitch(h0, 'crear');

case 'wcgPitch'

case 'wcgDEP'

    wcgAxesAnalisis(h0, 'eliminar');
    wcgAxesDEP(h0, 'eliminar');

    wcgAxesAnalisis(h0, 'crear');
    wcgAxesPitch(h0, 'crear');

case 'wcgSimulacion'

    wcgAxesSimulacion(h0, 'eliminar');
    wcgOpcionesSimulacion(h0, 'eliminar');
    wcgStatusAnalisisSmall(h0, 'eliminar');

    wcgAxesAnalisis(h0, 'crear');
    wcgAxesPitch(h0, 'crear');
    wcgStatusAnalisisLarge(h0, 'crear');

end

    wcgConfiguracion(h0, 'cargar_archivo_param');
    set(h0, 'Tag', 'wcgPitch');
    set(h0, 'Name', 'Módulo Análisis - Cálculo del Contorno Pitch');
    cambiarPosicionDeEje(h0);
    wcgPitch(h0, sTipo);

function hacerInvisibleAxeAnalisis(h0);

    hAxes = findobj(h0,'Tag','axeAnalisis');
    set(hAxes, 'Visible', 'off');

    hLinea = findobj(hAxes,'Tag','axeLinea');
    set(hLinea, 'Visible', 'off');

    hLinea = findobj(hAxes,'Tag','LineaY');
    set(hLinea, 'Visible', 'off');

    hLinea = findobj(hAxes,'Tag','LineaY2');
    set(hLinea, 'Visible', 'off');

function cambiarPosicionDeEje(h0);

    hAxes = findobj(h0,'Tag','axeAnalisis');
    set(hAxes, 'Position',[72 293 616 136]);
```

```
%*************************************************************************
**
%                        wcgVerGUISimulacion.m
%*************************************************************************
**

%*************************************************************************
**%         Tesis:      Analisis LPC
%        Autor:      William Castro Grijalva
%        Funcion:    Crear Interface Grafica de Usuario para el módulo
de Analisis LPC
%*************************************************************************
**function wcgVerGUISimulacion(h0);

%Detecta de donde viene ...
sTag = get(h0, 'Tag');
wcgConfiguracion(h0, 'crear_archivo_param');

switch sTag
case 'wcgLPC'

   wcgOpcionesLPC(h0, 'eliminar');           %crea opciones de LPC
   wcgAxesLPC(h0, 'eliminar')                %crea los nuevos ejes
   wcgStatusAnalisisSmall(h0, 'eliminar');      %crear el status bar

   wcgAxesSimulacion(h0, 'crear');
   wcgOpcionesSimulacion(h0, 'crear');
   wcgStatusAnalisisSmall(h0, 'crear');
%  wcgToolbarAnalisis(h0, 'crear');

case 'wcgAnalisis'

   wcgAxesAnalisis(h0, 'eliminar');       %elimina el eje de analisis
   wcgStatusAnalisisLarge(h0, 'eliminar');         %elimina el status
large

   wcgAxesSimulacion(h0, 'crear');
   wcgOpcionesSimulacion(h0, 'crear');
   wcgStatusAnalisisSmall(h0, 'crear');
%  wcgToolbarAnalisis(h0, 'crear');

case 'wcgEspectrograma'

   wcgAxesAnalisis(h0, 'eliminar');
   wcgAxesEspectrograma(h0, 'eliminar');
   wcgOpcionesEspectrograma(h0, 'eliminar');

   wcgAxesSimulacion(h0, 'crear');
   wcgOpcionesSimulacion(h0, 'crear');

case 'wcgAdquisicion'

   wcgAxesAdquisicion(h0, 'eliminar');
```

```
   wcgOpcionesAdquisicion(h0, 'eliminar');
   wcgMenuAdquisicion(h0, 'eliminar');
   wcgStatusAnalisisLarge(h0, 'eliminar');          %elimina el status
large

   wcgAxesSimulacion(h0, 'crear');
   wcgOpcionesSimulacion(h0, 'crear');
   wcgStatusAnalisisSmall(h0, 'crear');
%  wcgToolbarAnalisis(h0, 'crear');

case 'wcgDEP'

   wcgAxesAnalisis(h0, 'eliminar');
   wcgAxesDEP(h0, 'eliminar');
   wcgStatusAnalisisLarge(h0, 'eliminar');          %elimina el status
large

   wcgAxesSimulacion(h0, 'crear');
   wcgOpcionesSimulacion(h0, 'crear');
   wcgStatusAnalisisSmall(h0, 'crear');
%  wcgToolbarAnalisis(h0, 'crear');

case 'wcgPitch'

   wcgAxesAnalisis(h0, 'eliminar');
   wcgAxesPitch(h0, 'eliminar');
   wcgStatusAnalisisLarge(h0, 'eliminar');          %elimina el status
large

   wcgAxesSimulacion(h0, 'crear');
   wcgOpcionesSimulacion(h0, 'crear');
   wcgStatusAnalisisSmall(h0, 'crear');
%  wcgToolbarAnalisis(h0, 'crear');

case 'wcgSimulacion'

   return;

end

wcgConfiguracion(h0, 'cargar_archivo_param');
set(h0, 'Tag', 'wcgSimulacion');
set(h0, 'Name', 'Módulo Simulación');
```

```
%**************************************************************************
**
%                          wcgZoom.m
%**************************************************************************
**

%**************************************************************************
**%          Tesis:       Analisis LPC
%         Autor:       William Castro Grijalva
%         Funcion:     Maneja las opciones de acercamiento y alejamiento
         de la señal
%**************************************************************************
**function wcgZoom(sAction, hFig);
   %Estas variable se definen para almacenar las vistas de los
diferentes Zooms

   if nargin < 1,
      return;
   end;

   %Valida que haya datos en el eje ...
   hAxes = findobj(hFig, 'Tag', 'axeAnalisis');
   hLinea = findobj(hFig, 'Tag', 'axeLinea');
   YData = get(hLinea, 'YData');

   if isempty(YData)
      hMsg  = findobj(hFig, 'Tag', 'lblRutaArchivo');
      set(hMsg, 'String', '');
      set(hMsg, 'String', 'No existen datos para realizar el análisis
...');
      return;
   end

   %Este codigo valida que solo se ejecute si la ventana activa es
"lpc01"
   %esto se hace para evitar errores
   figure(hFig);

   switch (sAction)
   case 'In'
      OnZoomIn(hFig);
      case 'Out',
         OnZoomOut(hFig);
   end

%**************************************************************************
**
%        Función: se ejecuta cuando se hace el click sobre el zoom in
%**************************************************************************
**
function OnZoomIn(hFig);

   %Obtiene los limites de las barras  , que se muestran en los labels
```

```
   hTime = findobj(hFig, 'Tag', 'txtTimIni');
   iTimeIni = get(hTime, 'UserData');

   hTime = findobj(hFig, 'Tag', 'txtTimFin');
   iTimeFin = get(hTime, 'UserData');

   %Si no se ha seleccionado un rango no hace nada...
   if iTimeIni == iTimeFin
      return
   end

      hAxes = findobj(hFig, 'Tag', 'axeAnalisis');
   aPosition = get(hAxes, 'Position');


      hVista = findobj(hFig, 'Tag', 'txtVistas');
   oVista = get(hVista, 'UserData');

   %Esto es para que el zoom se realize si el rango actual
   %es diferente al rango anterior
   iTimeIniAnt = oVista(2*oVista(1));
   iTimeFinAnt = oVista(2*oVista(1)+1);

   if iTimeIniAnt == iTimeIni & iTimeFinAnt == iTimeFin
      return;
   end

   set(hAxes, 'XLim',    [iTimeIni iTimeFin]);
   oVista(1) = oVista(1) + 1;
   set(hVista, 'UserData', [oVista get(hAxes, 'XLim')] );


   %Escala el eje Y
   aParametro = get(hAxes, 'UserData');
   hAxesLine = findobj(hAxes, 'Tag','axeLinea');
   YData = get(hAxesLine, 'YData');

   if iTimeIni == 0
      iMuestraIni  = 1;
   else
      iMuestraIni = floor(iTimeIni/1000*aParametro(1));
   end

   iMuestraFin = floor(iTimeFin/1000*aParametro(1));

   %sSenalVisible = YData(1 + round(iTimeIni*aParametro(1)/1000):1:
round(iTimeFin*aParametro(1)/1000))';

   sSenalVisible = YData(iMuestraIni:iMuestraFin)';

   Ymin = min(sSenalVisible);
   Ymax = max(sSenalVisible);
```

```
   %set(hAxes, 'YLim',   [1.1*Ymin 1.1*Ymax]);

   hAxesLine = findobj(hAxes, 'Tag','LineaY');
   set(hAxes, 'YLim',    [1.1*Ymin 1.1*Ymax]);

   hAxesLine = findobj(hAxes, 'Tag','LineaY2');
   set(hAxes, 'YLim',    [1.1*Ymin 1.1*Ymax]);

   actualizacionDeDatos(hFig);



%**************************************************************************
**
%        Función: se ejecuta cuando se hace el click sobre el zoom in
%**************************************************************************
**
function OnZoomOut(hFig);

   %Obtiene los limites de las barras  , que se muestran en los labels
      hVista = findobj(hFig, 'Tag', 'txtVistas');
   oVista = get(hVista, 'UserData');
   iVista = oVista(1)-1;    %Toma la vista anterior

   if iVista == 0
      return;
   end

   iTimeIni = oVista(2*iVista);
   iTimeFin = oVista(2*iVista+1);

      hAxes = findobj(hFig, 'Tag', 'axeAnalisis');
   aPosition = get(hAxes, 'Position');
   set(hAxes, 'XLim',    [iTimeIni iTimeFin]);

   oVista(1) = oVista(1) - 1;
   set(hVista, 'UserData', [ oVista(1:length(oVista)-2)] );

   %Escala el eje Y
   aParametro = get(hAxes, 'UserData');
   hAxesLine = findobj(hAxes, 'Tag','axeLinea');
   YData = get(hAxesLine, 'YData');


   if iTimeIni == 0
      iMuestraIni  = 1;
   else
      iMuestraIni = floor(iTimeIni/1000*aParametro(1));
   end

   iMuestraFin = floor(iTimeFin/1000*aParametro(1));

   sSenalVisible = YData(iMuestraIni:iMuestraFin)';
```

```
   %sSenalVisible =
YData(1+round(iTimeIni*aParametro(1)/1000):1:round(iTimeFin*aParametro(
1)/1000))';

   Ymin = min(sSenalVisible);
   Ymax = max(sSenalVisible);

   set(hAxes, 'YLim',    [1.1*Ymin 1.1*Ymax]);

   hAxesLine = findobj(hAxes, 'Tag','LineaY');
   set(hAxes, 'YLim',    [1.1*Ymin 1.1*Ymax]);

   hAxesLine = findobj(hAxes, 'Tag','LineaY2');
   set(hAxes, 'YLim',    [1.1*Ymin 1.1*Ymax]);

   actualizacionDeDatos(hFig);

function actualizacionDeDatos(h0);
   %Realiza congiuraciones adicionales en la ventana actual
   %despues de un ZoomIn o ZoomOut

sTag = get(h0, 'Tag');

switch sTag
case 'wcgLPC'
case 'wcgAnalisis'
case 'wcgEspectrograma'
   wcgEspectrograma(h0, 'mostrar_espectro');
case 'wcgAdquisicion'
case 'wcgDEP'
   wcgDensidadEP(h0, 'mostrar_dep');
case 'wcgPitch'

   hAxes = findobj(h0, 'Tag','axePitch');
   sTipo = get(hAxes, 'UserData');
   wcgPitch(h0, sTipo);
case 'wcgSimulacion'
   wcgSimulacion(h0, 'mostrar_sim');
end
```

BIBLIOGRAFÍA

1. John D. Markel, The Sift Algoritm for Fundamental Frequency Estimation., IEEE Trans. Audio Electroacustic, Vol- Au-20, pp 367-377, (1972).
2. J.-S. Roger Jang, Query By Singing (CBMR Content-based Music Retrieval) , CS Dept, Tsing-Hua Univ, Taiwan. http://www.cs.nthu.edu.tw/~jang
3. Woojay Jeon. Pitch Detection of Speech Using a Modified SIFT Algorithm. Transaction on digital processing of speech signals, vol. 01, N° 1, Abril 2001.
4. Royal Institute of Technology, Stockholm , Speech Signal Processing ,Speech Communication and music acoustic
5. C. Britton, Rorabaugh, DSP Primer, 1995.
6. Jeremy Bradburry, Linear Predictive Coding , Dic 2000.
7. Goangshiuan S. Ying, Leah H. Jamieson, and Carl D Mitchell, A Probabilistic approach to AMDF Detection. http://purcell.ecn.purdue.edu.~speechg
8. Mehmet Ugur Dogan, TTS Sistem For Turkish Pitch Detection
9. Sussana Varho, Helsinsky University of Technologie, New Linear Predictive Methods for Digital Speech Proccesing, 20 Abril 2001.
10. Violeta Gambiroza, Alexander Kuzmanovic, Yongue Liu, Liang Sun, Yuanbin Guo, Espectral Analisis and LPC Vocoder.
11. Jean Francois Frigon and Vladislav Teplitsky, Implementation of Linear Predictive Coding (LPC) of Speech