# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE INGENIERÍA

## SIMULACIÓN DE UN SISTEMA DE CANCELACIÓN DE ECO EN CANALES TELEFÓNICOS

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

RODRIGO ANÍBAL VILLARREAL HERNÁNDEZ

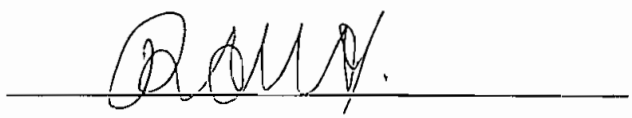
DIRECTOR: Dr. GUALBERTO HIDALGO

Quito, Marzo 2002

# DECLARACIÓN

Yo Villarreal Hernández Rodrigo Aníbal, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada por ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.



**Villarreal Hernández Rodrigo Aníbal**

# CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por el Sr. Villarreal Hernández Rodrigo Aníbal, bajo mi supervisión.

Dr. Gualberto Hidalgo
DIRECTOR DEL PRYECTO

# AGRADECIMIENTO

Mis más sinceros agradecimientos al Dr. Gualberto Hidalgo director de proyecto, por el apoyo brindado para la culminación exitosa del presente proyecto.

# DEDICATORIA

Este trabajo esta dedicado a mis padres y a mis hermanos, que me apoyaron constantemente para culminar mi carrera universitaria.

# CONTENIDO

# RESUMEN

El presente proyecto de titulación tiene como finalidad estudiar los filtros adaptivos aplicados a la cancelación de eco en señales telefónicas. El filtro adaptivo que se utiliza consta de dos partes: el filtro ajustable y el algoritmo adaptivo.

En la parte ajustable se utiliza un filtro FIR *(Finite Impulse Response)* y, como algoritmo adaptivo se utiliza el algoritmo de mínimos cuadrados medios o LMS (*Least Mean Squares*).

En el capítulo 1 se realiza un rápido estudio de los filtros digitales, las técnicas de diseño, y el algoritmo LMS mencionando sus propiedades.

En el capítulo 2 se realiza un ejemplo de simulación de un filtro digital adaptivo, por lo cual se trata sobre el filtro de Wiener y los filtros adaptivos.

En el capítulo 3 se mencionan los tipos de eco, cancelación de eco en transmisión de voz , definiciones relativas a los compensadores de eco, características de los compensadores de eco, análisis de la precisión de los coeficientes del filtro en el desempeño del algoritmo LMS para canceladores de eco G.165/G.168 y finalmente se presenta los resultados de la simulación realizada.

Adicionalmente se incluyen: anexos A y B el paquete computacional utilizado, Matlab, para realizar la simulación, en el anexo C se presenta el manual de usuario de la interfaz gráfica, en el anexo D se trata sobre las pruebas y requisitos de calidad de funcionamiento con señales de entrada aplicadas a los trayectos emisión y recepción, en el anexo E se presenta el listado del programa de la interfaz gráfica, en el anexo F se ofrece el glosario de términos.

# PRESENTACIÓN

El eco juega un papel muy grande en sistemas de telecomunicaciones, en sistemas de telefonía es generalmente indeseable. La conexión telefónica entre abonado y la central local se hace a 2 hilos, realizándose la conversión a 4 hilos en la híbrida de la central, la cual no opera perfectamente y su efecto genera muchos problemas en la fidelidad de una conversación entre abonados, por lo cual se considera conveniente minimizar este efecto con el empleo de sistemas de supresión de eco.

El presente trabajo tiene por objeto simular un sistema de cancelación de eco mediante la utilización de filtros adaptivos, para lo cual se escogió el filtro FIR (*Finite Impulse Response*) por sus estabilidad, el algoritmo adaptivo LMS por su simplicidad computacional ya que en sistemas de tiempo real se necesita un menor número de cálculos.

Para realizar la simulación se implementa una interfaz gráfica con la ayuda del MATLAB que es un programa interactivo para cálculos numéricos y visualización de datos.

# CAPÍTULO 1. INTRODUCCIÓN AL SISTEMA DE SIMULACIÓN.

## 1.1 INTRODUCCIÓN A LOS FILTROS DIGITALES

Los filtros digitales son sistemas de tiempo discreto que pueden realizar funciones de filtrado de señales. Aprovechan los avances de la tecnología digital para emular sistemas análogos. La operación de filtrado se realiza por medio de cálculos directos con las señales muestreadas.

Los filtros digitales son usados para dos propósitos generales:

- Separación de señales que están combinadas
- Reconstrucción de señales que han sido distorsionadas en alguna forma

Existen dos tipos de filtros digitales:

- Filtros FIR (Finite Impulse Response). Estos filtros tienen respuesta al impulso de duración finita, no tienen realimentación
- Filtros IIR ((Infinite Impulse Response). Estos filtros tienen respuesta al impulso de duración infinita, tienen realimentación

Los filtros analógicos (electrónicos) son sistemas lineales invariantes en el tiempo que tiene una función de transferencia con ciertas características deseadas en el dominio de la frecuencia. Un filtro procesa una señal acentuando o atenuando determinadas regiones de su espectro y/o produciendo algún desfase de acuerdo a lo deseado. Existen dos clases de filtros analógicos:

- Filtros pasivos. Estos filtros se construyen utilizando componentes pasivas (resistencias y condensadores)

- Filtros activos. Estos filtros utilizan elementos activos (amplificadores) además de los pasivos para realizar la labor del filtrado

### 1.1.1 FILTROS BÁSICOS

Los filtros digitales son una parte muy importante del Procesamiento Digital de Señales (DSP), y por su extraordinario rendimiento han llegado a ser muy populares. Como se mencionó en la introducción estos filtros tienen dos usos: separación y reconstrucción de señales. La separación de señales se emplea cuando la señal ha sido contaminada con interferencia, ruido, y otras señales. La reconstrucción de señales es usada cuando una señal ha sido distorsionada de alguna forma.

Estos problemas pueden ser atacados con filtros analógicos o digitales. ¿Cuál es mejor? Los filtros analógicos son baratos, rápidos y tienen un amplio rango dinámico en amplitud como en frecuencia. Los filtros digitales, en comparación son muy superiores en el nivel de rendimiento.

Para tener toda la información de un filtro digital se debe tener la respuesta impulsiva, la respuesta a una señal paso y la respuesta en frecuencia, cada una de las cuales nos da la información en diferente forma. Si una de las tres es especificada, las otras dos pueden ser ajustadas y directamente calculadas. Estas tres representaciones son importantes porque describen la manera en que el filtro reaccionará bajo diferentes circunstancias.

La forma más fácil de implementar un filtro digital es realizando la convolución entre la señal de entrada con la respuesta impulsiva del filtro digital, cuando un filtro es implementado por convolución, cada muestra en la salida es calculada por pesos. Si la respuesta impulsiva es usada en esta forma, los diseñadores de filtros le dan un nombre especial: Filtro Kernel.

### 1.1.2 FILTROS DIGITALES FIR

El diseño de los filtros FIR requiere la selección de la secuencia que mejor representa la respuesta al impulso de un filtro ideal. Los filtros FIR son siempre estables y capaces de tener una respuesta de fase que es lineal, lo que equivale a decir que su respuesta tiene un retraso constante. El mayor problema de los filtros FIR es que para unas especificaciones dadas requieren un filtro de orden mucho mayor que los filtros IIR.

Un filtro FIR de longitud M con entrada x[n] y salida y[n] se describe mediante la ecuación de diferencias:

$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_{M-1} x[n-M+1] = \sum_{k=0}^{M-1} b_k x[n-k] \qquad (1.1)$$

donde $b_k$ son los coeficientes del filtro.

Por otra parte se puede expresar la salida del filtro y[n] como una convolución de la entrada x[n] con la respuesta a impulso del filtro h[n]:

$$y[n] = \sum_{k=0}^{M-1} h[k] * x[n-k] \qquad (1.2)$$

ya que estas dos ecuaciones son idénticas, y por tanto, los coeficientes $b_k = h[k]$.

Se puede demostrar que la respuesta de un filtro FIR es de fase lineal si los coeficientes h[n] cumplen:

$h[n] = \pm h[M-1-n]$, para todo $n = 0, 1, \ldots\ldots M-1$.

es decir los coeficientes tienen algún tipo de simetría (par o impar).

La función de transferencia del filtro FIR, en el dominio de Z aplicando ésta condición es:

$$H(z)=\sum_{k=0}^{M-1}h[k]\cdot z^{-k}=h[0]+h[1]\cdot z^{-1}+h[2]\cdot z^{-2}+\cdots+h[M-2]\cdot z^{-(M-2)}+h[M-1]\cdot z^{-(M-1)}=$$

$$\begin{cases} z^{\frac{-(M-1)}{2}}\cdot\left\{h[\frac{M-1}{2}]+\sum_{k=0}^{\frac{(M-3)}{2}}h[k]\cdot\left[z^{\frac{(M-1-2k)}{2}}\pm z^{\frac{-(M-1-2k)}{2}}\right]\right\} M.impar \\ z^{\frac{-(M-1)}{2}}\cdot\sum_{k=0}^{\frac{M-1}{2}}h[k]\cdot\left[z^{\frac{(M-1-2k)}{2}}\pm z^{\frac{-(M-1-2k)}{2}}\right] M.par \end{cases}$$

De esta ultima expresión se deduce que:

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

lo que significa que las raíces de H(z) son las mismas que las de $H(z^{-1})$, es decir las raíces (en este caso, los ceros) ocurren en pares recíprocos. Si $z_1$ es un cero de H(z), $1/z_1$ es también un cero. Además, si $z_1$ es un cero complejo, su conjugado $z_1^*$ es también un cero complejo, así como $1/z_1^*$.

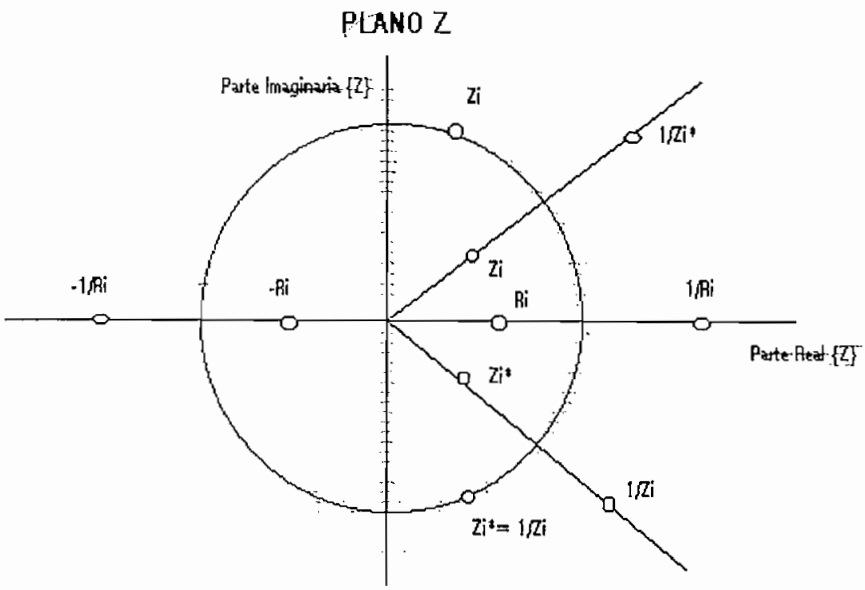



Figura 1.1 Representación de las propiedades de los filtros FIR

La longitud N de una secuencia simétrica puede ser par o impar. Esto significa que el punto medio cae en un punto de la secuencia si N es impar y entre dos puntos si N es par. Como se tiene dos tipos de simetría (par o impar), se tendrá cuatro posibles tipos de secuencia simétricas, las cuales se muestran en la tabla 1.1 junto con la DTFT de cada secuencia.

| Secuencias Simétricas | | | | | |
|---|---|---|---|---|---|
| Tipo | Simetría | N | H(F) | $\|H(0)\|$ | $\|H(1/2)\|$ |
| 1 | Par | Impar | $h[0]+2\sum_{k=1}^{L} h[k]\cos(2k\pi F)$ | $h[0]+2\sum_{k=1}^{L} h[k]$ | $h[0]+2\sum_{k=1}^{L}(-1)^k h[k]$ |
| 2 | Par | Par | $2\sum_{k=1}^{M} h[k]\cos[2\pi F(k-1/2)$ | $2\sum_{k=1}^{L} h[k]$ | 0 |
| 3 | Impar | Impar | $-j2\sum_{k=1}^{L} h[k]\sin(2k\pi F)$ | 0 | 0 |
| 4 | Impar | Par | $-j2\sum_{k=1}^{M} h[k]\sin[2\pi F(k-1/2)]$ | 0 | $-2\sum_{k=1}^{M}(-1)^k h[k]$ |

Tabla 1.1 Secuencias Simétricas

Donde:

$L=1/2(N-1)$

$M=1/2N$

F es la frecuencia digital $=f/f_s$ donde $f_s$ es la frecuencia de muestreo

De esta tabla se pueden sacar las siguientes conclusiones acerca de la aplicabilidad de cada uno de los tipos de secuencia.

- La secuencia tipo 1 puede implementar cualquier tipo de filtro. Es el único tipo capaz de realizar filtros elimina banda

- Para las secuencias de tipo 2:

  $|H(1/2)|$ =0, por lo que sólo pueden ser utilizadas para filtros paso bajo (LP) y paso banda (BP)

- Las secuencias de tipo 3:

  $|H(0)|=0=|H(1/2)|$ solo pueden ser utilizadas para filtros paso banda (BP)

- Las secuencias tipo 4:

  $|H(0)| = 0$ son apropiadas para filtros paso alta (HP) y paso banda (BP)

| Aplicaciones de las Secuencias Simétricas | | |
|---|---|---|
| Tipo | $\|H(F)\|$ | Aplicación |
| 1 | | Todo tipo de filtros |
| 2 | $\|H(1/2)\| = 0$ | Solo LP y BP |
| 3 | $\|H(0)\| = 0 = \|H(1/2)\|$ | Solo BP |
| 4 | $\|H(0)\| = 0$ | Solo HP y BP |

Tabla 1.2 Conclusiones de la aplicabilidad de las secuencias simétricas.

### 1.1.3 TÉCNICAS DE DISEÑO DE FILTROS FIR

Hay tres métodos de diseño de filtros FIR:

- Método de las Series de Fourier
- Método del Muestreo en frecuencia
- Métodos Interactivos basados en criterios de optimización

### 1.1.3.1 MÉTODO DE LAS SERIES DE FOURIER

El método se basa en seleccionar la respuesta a impulso $h_N[n]$ como una versión truncada de la respuesta al impulso $h[n]$ de un filtro ideal con respuesta de frecuencia $H(F)$.

El proceso de diseño de filtros FIR por este método es el siguiente:

- Normalización de frecuencias por la frecuencia de muestreo
- Conversión de especificaciones a las de un Prototipo de Filtro Paso Bajo
- Truncamiento de la respuesta al impulso de un filtro ideal $h[n]$ a $h_N[n]$ de longitud N. El orden del filtro es N-1
- Seleccionar una ventana $w[n]$ de N puntos para obtener $h_w[n] = h_N[n] \cdot w[n]$
- Convertir del Prototipo de Filtro Paso Bajo al Filtro deseado $h_F[n]$
- Retrasar $h_F[n]$ para asegurarse que el filtro es causal
- Se plantea realizar un filtro paso bajo ideal (digital) con una frecuencia de corte Fc., tal como se indica en la figura 1.2

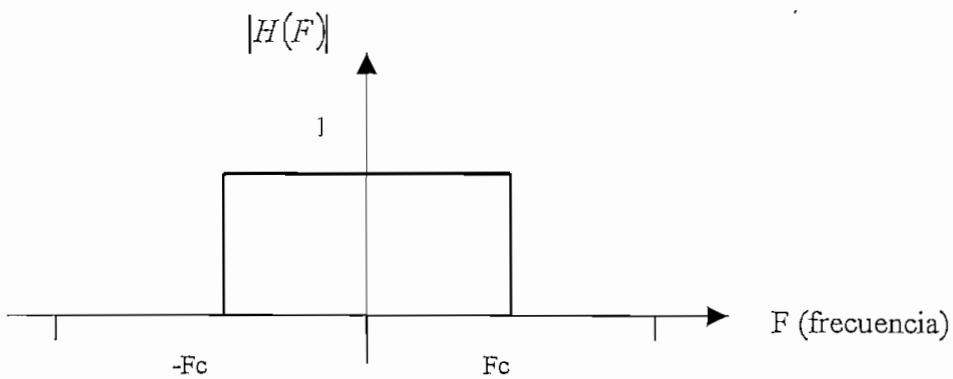



Figura 1.2 Filtro paso bajo ideal

Si se expresa matemáticamente la respuesta de frecuencia deseada:

$$H_d(F)=\begin{cases} e^{-j2\pi F(N-1)/2} \ldots\ldots\ldots\ldots |F| \le F_c \\ 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots |F| > F_c \end{cases} \qquad (1.3)$$

haciendo la Transformada inversa de Fourier discreta en el tiempo de esta función $H_d(F)$, nos queda:

$$h[n]=\frac{1}{Fs}\int_{-\frac{1}{2}}^{\frac{1}{2}} H_d(F).e^{j2\pi Fn}dF=\int_{-Fc}^{Fc} e^{-j2\left(\frac{N-1}{2}-n\right)}dF=\frac{1}{-j2\pi\left(\frac{N-1}{2}-n\right)}\left[e^{-j2\pi F\left(\frac{N-1}{2}-n\right)}\right]_{-Fc}^{Fc}=$$

$$=\frac{-2j}{-2j\pi\left(\frac{N-1}{2}-n\right)}\cdot\sin\left[2\pi F_c\left(\frac{N-1}{2}-n\right)\right]=2F_c\frac{\sin\left[2\pi F_c\left(\frac{N-1}{2}-n\right)\right]}{2\pi F_c\left(\frac{N-1}{2}-n\right)}= \qquad (1.4)$$

$$=2F_c\,\mathrm{sinc}\left[2F_c\left(\frac{N-1}{2}-n\right)\right]$$

- La función sinc(x) esta definida para todo valor de x, y además decae muy lentamente. Utilizar los valores de h[n] definidos por la ecuación (1.4) como coeficientes del filtro FIR, dará lugar a sobre impulsos en la respuesta de frecuencia del filtro. Debido a la lentitud de la función sinc(x), se necesita un filtro de elevado orden (con muchos puntos) para diseñar filtros con transiciones rápidas entre bandas

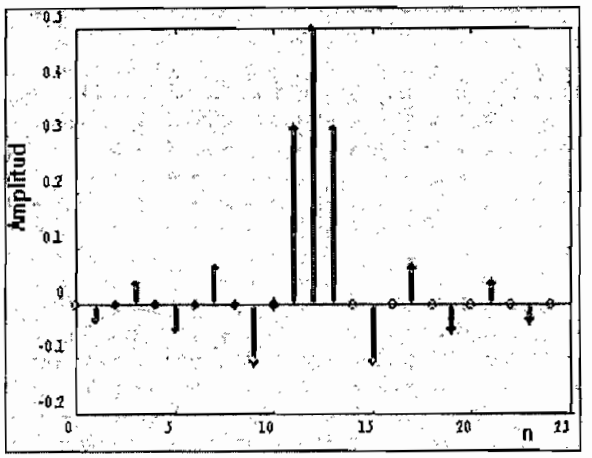



Figura 1.3 Función Sinc (x)

- El truncamiento de h[n] equivale a multiplicar h[n] por una ventana rectangular w[n] de longitud N. El espectro de $h_N = h[n] \cdot w[n]$ es la convolución de H[F] y W[F]. La función W[F] produce rizados y sobre impulsos en la señal de salida de la misma forma que ocurre el efecto Gibbs al reconstruir una señal discontinua con un numero finito de coeficientes espectrales. Aquí el efecto Gibbs se da en el dominio de la frecuencia al usar un truncamiento de la respuesta a impulso. Para reducir los efectos de un truncamiento abrupto se utilizan ventanas espectrales que tienden a suavizar esos efectos

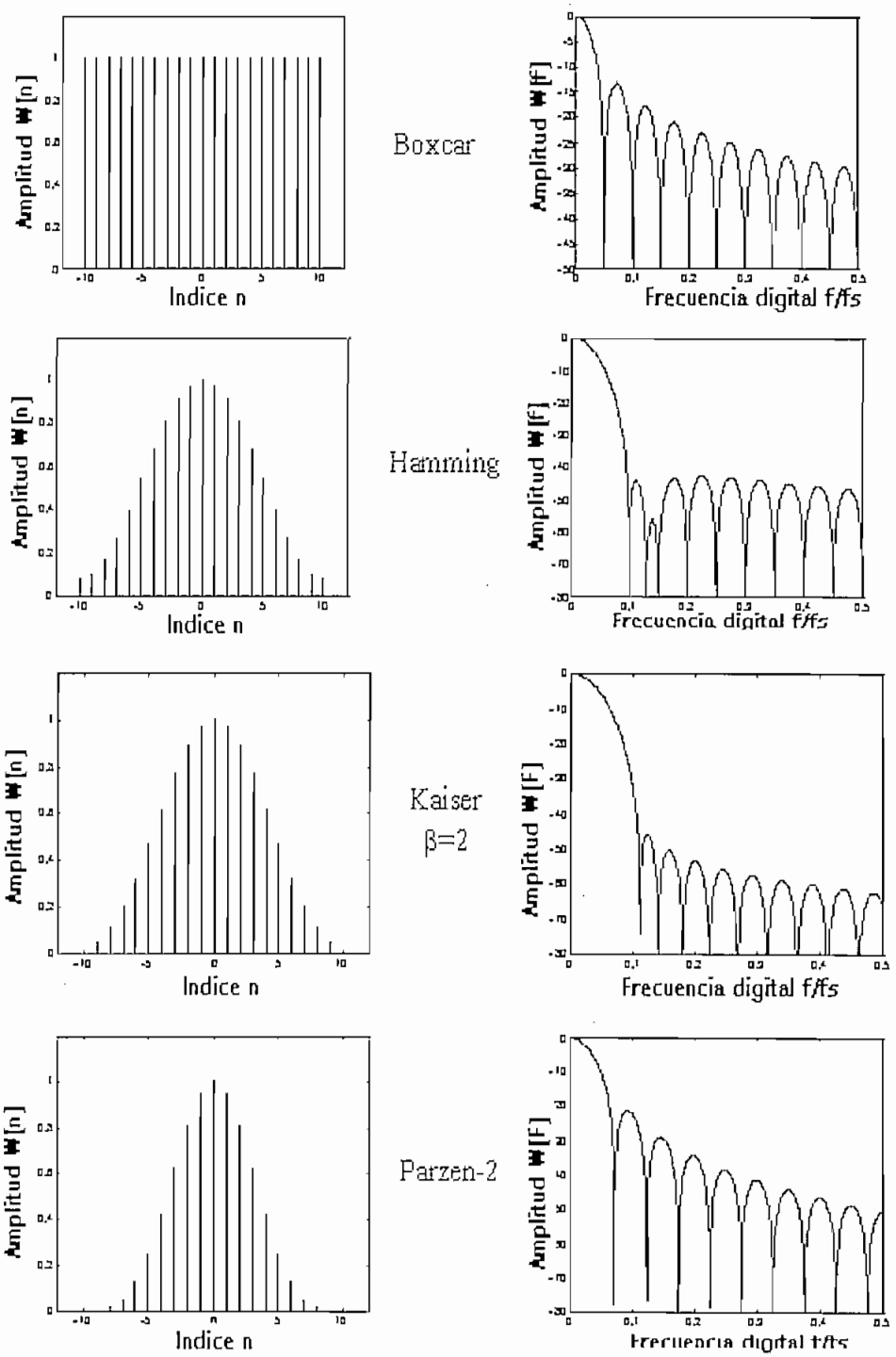
- Las ventanas más comúnmente utilizadas en el diseño de filtros FIR están listadas en la tabla 1.3. En la tabla 1.4 se especifican las principales características espectrales de la ventana de Kaiser

- De los espectros de las ventanas se hacen notar dos cosas:

    - El ancho del lóbulo principal y de transición decrece al aumentar N
    - La amplitud de los lóbulos de los lados permanece constante con N

- Idealmente el espectro de una ventana debe estar confinado en el lóbulo principal , casi sin energía en los lóbulos de los lados

- A continuación se muestran algunas ventanas y sus espectros:



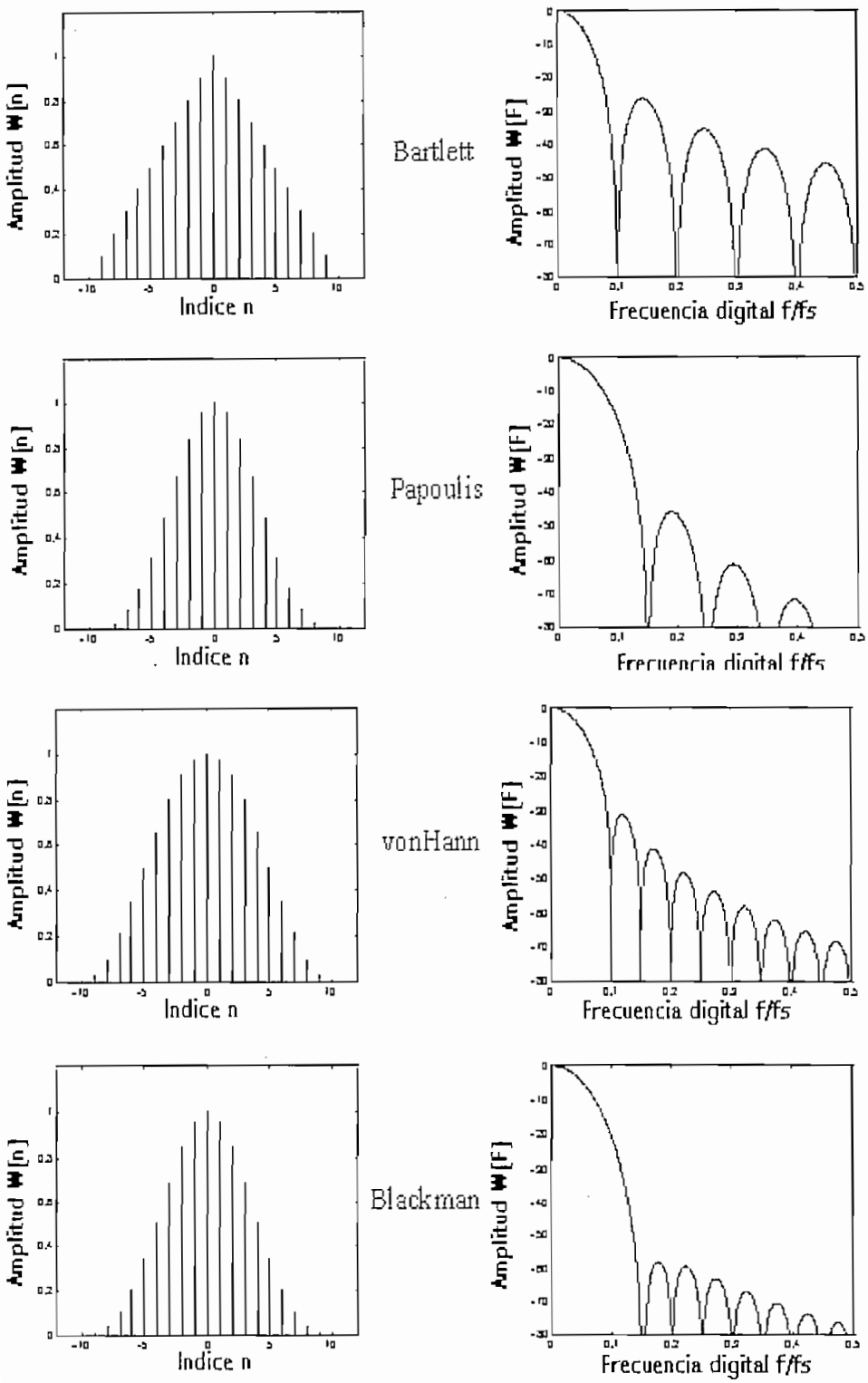



Figura 1.4 Ejemplos de ventanas espectrales y sus espectros

- Las ventanas más utilizadas son von Hann, Hamming y Kaiser

| *Ventana* | *Atenuación en lóbulo lateral (dB)* | *Ancho de Banda de la Transición* | *Máximo Rizado en elimina banda (dB)* |
|---|---|---|---|
| *Rectangular* | *-13* | $2\pi 0.9/N$ | *-21* |
| *Hanning* | *-31* | $2\pi 3.1/N$ | *-44* |
| *Hamming* | *-41* | $2\pi 3.3/N$ | *-53* |
| *Blackman* | *-57* | $2\pi 5.5/N$ | *-74* |

Tabla 1.3 Características de algunas ventanas espectrales

| *Parámetro $\beta$* | *Atenuación en lóbulo lateral (dB)* | *Ancho de banda de la Transición* | *Máximo Rizado en elimina banda (dB)* |
|---|---|---|---|
| *2.0* | *-19* | $2\pi 1.5/N$ | *-29* |
| *3.0* | *-24* | $2\pi 2.0/N$ | *-37* |
| *4.0* | *-30* | $2\pi 2.6/N$ | *-45* |
| *5.0* | *-37* | $2\pi 3.2/N$ | *-54* |
| *6.0* | *-44-* | $2\pi 3.8/N$ | *-63* |
| *7.0* | *-51* | $2\pi 4.5/N$ | *-72* |
| *8.0* | *-59* | $2\pi 5.1/N$ | *-81* |
| *9.0* | *-67* | $2\pi 5.7/N$ | *-90* |
| *10.0* | *-74* | $2\pi 6.4/N$ | *-99* |

Tabla 1.4 Características de la ventana Kaiser para distintos parámetros

- Podemos estimar la longitud del filtro (N) a partir del ancho de banda de la transición ($\Delta\omega$). Por ejemplo, si se pide un filtro con una transición entre $\omega_P$ y $\omega_S$, llamamos banda de transición a la diferencia

normalizada por la frecuencia de muestreo $F_m$, entre las frecuencias de paso banda y elimina banda (en *rad/s*),

$$\Delta\omega = \frac{(\omega_S - \omega_P)}{(F_m)} = \frac{2\pi k}{N} \Rightarrow N \approx \frac{2\pi k}{\Delta\omega} = \frac{k}{\Delta F}$$

- Otro criterio es tomar $N$ de acuerdo a la siguiente fórmula, donde $W_S$ es la mitad de la anchura del lóbulo principal (tabla 1.3), $F_P$ y $F_S$ son las frecuencias digitales de paso banda y elimina banda

$$N = \frac{W_S}{F_S - F_P}$$

### 1.1.3.2 MÉTODO DEL MUESTREO FRECUENCIAL

Se trata de reconstruir el espectro continuo *X(F)* de una señal discreta a partir de los muestreos de la función *X(F)*. El espectro reconstruido $X_N(F)$ será igual a *X(F)* sólo en las frecuencias de muestreo.

Se puede considerar la DTFT de la señal $h_N[n]$ de longitud N:

$$H_N(F) = \int_0^1 h_N[n]\exp(j2\pi nF)dF \qquad H_N[k] = \sum_{k=0}^{N-1} h_N[n]\exp(-2j\pi nk/N)$$

donde:
$F=k/N$, $k=0,1,.....,N-1$

la respuesta a impulso $h_N[n]$ se calcula con el *IDFT*,

$$h_N[n] = \frac{1}{N}\sum_{k=0}^{N-1} H_N[k]\exp(j2\pi nk/N)$$

### 1.1.3.2.1 PROCESO DE DISEÑO

- Los muestreos deben hacerse en un período (0,1) de la extensión periódica de *H(F)*
- La fase de *H(F)* es lineal y por tanto cada uno de los muestreos tiene una fase dada por $\phi(k)=-\pi k(N-1)/N$
- Para minimizar el efecto Gibbs en las discontinuidades, se permite que los valores de muestreo varíen lentamente en las discontinuidades

Ejemplo : Diseñar un filtro paso bajo dado por la respuesta de frecuencia de la figura 1.5

Tomamos 10 muestras. La fase de cada una de las muestras es

– $\pi k( N- 1)/ N$

– $H[ 0]= 1\ H[ 1]= exp(- j0.9\pi )\ H[ 2]= exp(- j1.\ 8\ \pi )\ H[ 3]= 0\ H[ 4]= 0\ H[ 5]= 0.$

– $H[ 1]=- 0.9511- j0.3090\ H[ 2]= 0.\ 8090+ j0.5878.$

– Los valores $H[ 5] \ldots H[ 9]$ se calculan teniendo en cuenta que la respuesta de frecuencia debe ser simétrica en módulo y asimétrica en fase: $H[ 6]= 0\ H[ 7]= 0\ H[ 8]= 0.8090- j0.5878\ H[ 9]=- 0.9511+ j0.\ 3090.$

– *Calculando la IDFT de estos H[k] se obtiene la secuencia h{n}, h[ n]={0.0716, - 0.0794 ,- 0.1, 0.1558, 0.452, 0.452, 0.1558, -0.1,- 0.0794, 0. 0716}*

– En la respuesta de frecuencia de este filtro se observa que efectivamente pasa por los puntos de muestreo pero a costa de un sobreimpulso. Para suavizarlo, se puede sustituir los muestreos:

*H[ 3]* por *0.5exp(- j2.7 π ).*

– Calculando el IDFT de la nueva secuencia de muestreos *H[ k]* se obtiene el filtro:

*h[n]={0.0128, 0.0157, -0.1, 0.0606, 0.5108, 0.5108, 0.0606, -0.1,0. 0157, 0.0128}*

- Se puede combinar las ventajas del diseño con ventanas estudiado anteriormente y el método del muestreo frecuencial para tener un método de diseño de filtros de respuesta de frecuencia arbitraria
- Se muestrea la respuesta de frecuencia deseada con un número alto de puntos (M =512). Se hace el *IDFT* y obtenemos la respuesta *h[n]*
    - H[n] es demasiado largo, así que se debe truncarlo a una secuencia mas pequeña con una ventana
    - Si el diseño no cumple las especificaciones se puede cambiar n, el ancho de paso banda o ajustar los muestreos en la zona de transición

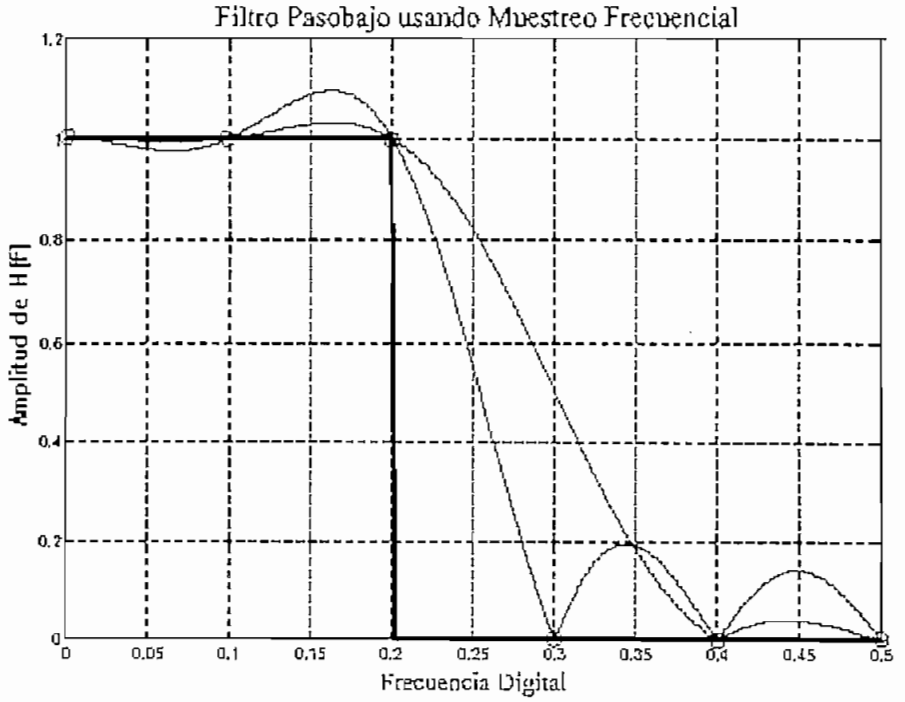



Figura 1.5 Filtro paso bajo usando muestreo frecuencial

### 1.1.3.3 MÉTODOS BASADOS EN CRITERIOS DE OPTIMIZACIÓN

Se trata de utilizar criterios para minimizar el error en la aproximación. Hay tres importantes conceptos en diseño óptimo:

- El error entre la aproximación H(F) y la respuesta deseada D(F) debe tener igual rizado
    - La respuesta frecuencial H(F) de un filtro cuya respuesta a impulso h[n] es una secuencia simétrica puede ponerse como:

$$H(F) = \sum_{k=0}^{M} \alpha_n(F)\cos(2\pi nF)$$

donde M esta relacionado con la longitud del filtro N. Esta forma es un polinomio de Chebyshev. Se debe escoger $\alpha_n$ para que el diseño sea óptimo

    - El teorema de la alternancia ofrece una pista para seleccionar $\alpha_n$

- El teorema de la Alternancia: Se aproxima D(F) por una forma polinomial de Chebyshev obteniendo H(F). Se define el error ponderado en la aproximación $\varepsilon(F)$, como $\varepsilon(F) = W(F)[D(F)-H(F)]$. El teorema dice que se pueden encontrar al menos M+2 frecuencias $F_K$, k=1,2,...,M+2 llamadas frecuencias extremas donde
    - El error varía entre dos máximos y mínimos iguales

$$\varepsilon(F_K) = -\varepsilon(F_{K+1}) \qquad k=1,2,\ldots,M+1$$

    - El error en la frecuencial $F_K$ es igual al máximo error absoluto

$$|\varepsilon(F_K)| = |\varepsilon(F)_{max}| \qquad k=1,2,\ldots,M+2$$

existe una algoritmo llamado Parks-McClellan (PM) para determinar esas frecuencias. Este algoritmo necesita los siguientes datos: las frecuencias

$F_P$ y $F_S$, la relación $\delta_1/\delta_2$ de los errores en la paso banda y en la elimina banda y la longitud N del filtro. Devuelve los coeficientes $\alpha_n$ y los valores reales de $\delta_1$ y $\delta_2$.

El filtro elimina banda diseñado anteriormente ha sido rediseñado utilizando este algoritmo. El resultado es un filtro de *N= 21*, $\delta_1$ *=0.2225dB y* $\delta_2$*=56.79 dB.*

La respuesta de frecuencia se muestra en la figura 1.6.

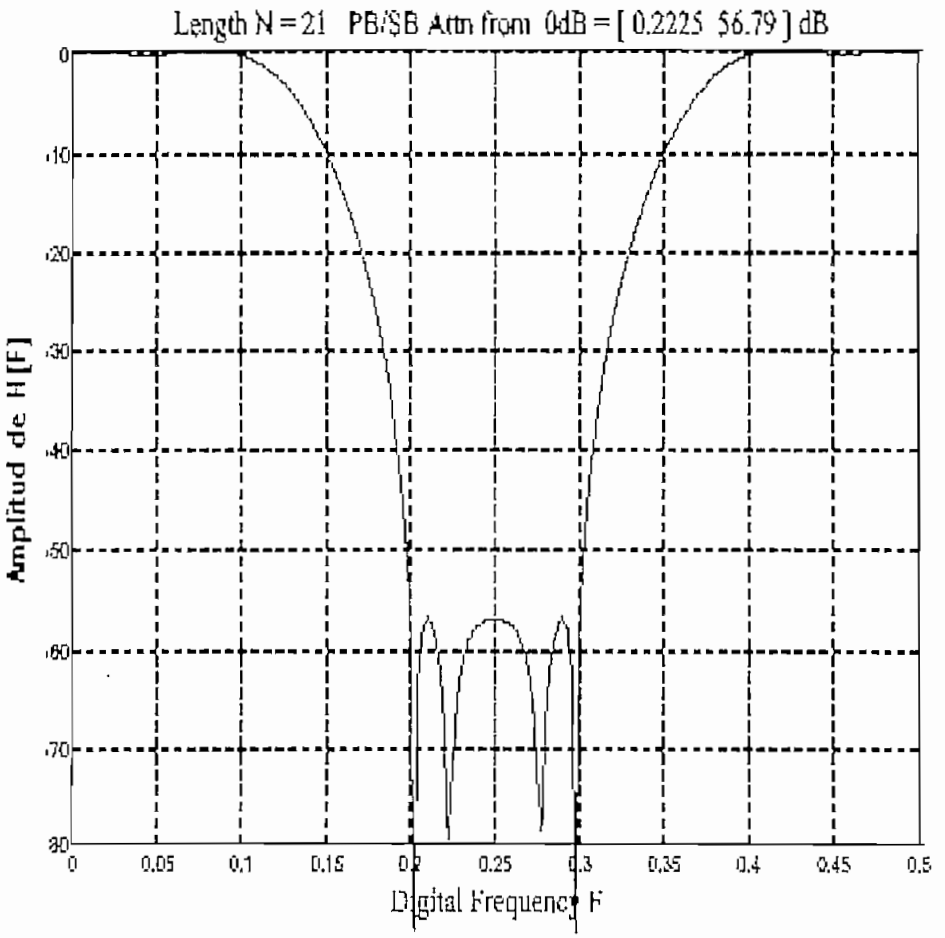



Figura 1.6 Respuesta de frecuencia de un filtro elimina banda

### 1.1.4 FUNCIONES DE MATLAB PARA REALIZAR FILTROS FIR:

- Función FIR1

>> B = fir1(N,Wn,type,window);

Diseña un filtro FIR paso bajo de orden N (longitud N +1) y frecuencia de corte Wn (normalizada con respecto a la frecuencia de Nyquist, ($0 \leq Wn \leq 1$). Se pueden especificar otro tipo de filtros mediante el parámetro *type*. Por ejemplo, para un filtro elimina banda:

>> B = fir1(N,[W1 W2],'stop');

Por defecto la función FIR usa la ventana de Hamming. Otro tipo de ventanas como las que se dan a continuación puede también especificarse

>> B = fir1(N,Wn,bartlett(N+1)):

>> B = fir1(N,Wn,'high',chebwin(n+1,R));

- Función FIR2

>> B = fir2(N,F,M,window);

Diseña un filtro FIR utilizando el método del muestreo frecuencial. Los parámetros de entrada son el orden del filtro N (longitud N+1) y dos vectores F y M que especifican la frecuencia y la magnitud, de forma que "plot(F,M)" es la respuesta deseada del filtro.
Se pueden indicar saltos bruscos en la respuesta frecuencial duplicando el valor de la frecuencia de corte.

F debe estar entre 0 y 1, en orden creciente, siendo el primer elemento igual a 0 y el último a 1.
El parámetro window indica el tipo de ventana a utilizar . Por defecto, usa la ventana de Hamming.

```
>>B = fir2( N, F, M);
```

Se pueden especificar más parámetros en esta función,

```
>> B = fir2( N, F, M, npt, lap, window);
```

La función fir2 interpola la respuesta frecuencial deseada (F,M) con npt puntos (por defecto, npt=512). Si dos valores sucesivos de F son iguales, se crea una región de lap puntos alrededor de este punto (por defecto, lap=25).

- Funcion FIRLS

```
>> B = firls(N,F,M);
```

Diseño de filtros FIR usando la minimización del error por mínimos cuadrados. Los argumentos de entrada son el orden del filtro N, y dos vectores F y M cuyo formato difiere de los análogos en la función fir2.
El filtro obtenido es la mejor aproximación a (F,M) por mínimos cuadrados.
F es un vector que indica los límites de las bandas de frecuencia en parejas (por tanto el tamaño de F debe ser par), y en orden ascendente entre 0 y 1. M es un vector del mismo tamaño que F que indica la magnitud deseada para cada banda de frecuencias. La respuesta deseada es la línea que conecta los puntos (F(k), M(k)) y (F(K+1), M(k+1) para k impar.

Las bandas de frecuencia entre F(k+1) y F(k+2) para k impar son tratadas por firls como bandas de transición. También existe un argumento opcional que consiste en un vector W cuyo tamaño es la mitad de F. W es un factor de ponderación del error para cada banda de frecuencias.

>> B = firls( N, F, M, W);

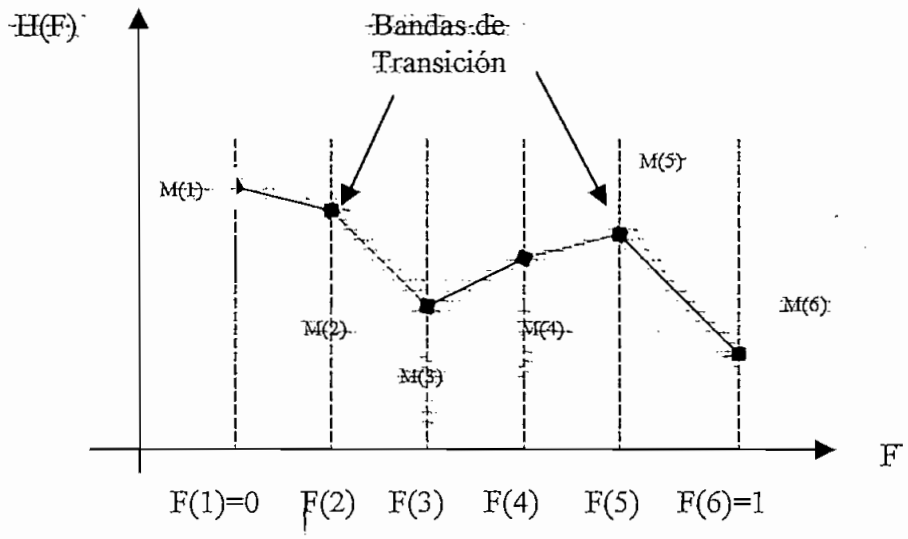



Figura 1.7 Bandas de transición

- Algoritmo de Parks-McClean

Hay dos funciones en MATLAB para realizar este algoritmo: remezord y remez.

>> [N, Fo, Mo, W] = remezord( F, M, DEV, Fs)

Calcula el orden N, las bandas de frecuencia normalizadas Fo las magnitudes en esas bandas Mo y los factores de ponderación W que luego serán utilizados como argumentos de entrada de la función remez. Estos valores cumplen las especificaciones dadas por F, M, DEV. F es un vector de frecuencias de corte en Hz, en orden

ascendente entre 0 y Fs/2. Si no se especifica Fs, Fs =2 por defecto. El primer elemento de F es siempre 0 y el último es siempre Fs/2, pero no deben ser especificados en el vector F. El vector M indica la respuesta deseada en cada banda. Por tanto, el vector M tiene un tamaño igual a (length(F)+2)/2. DEV es un vector que indica el máximo rizado permitido en cada banda

>> b = remez( N, Fo, Mo, W);

Con los valores obtenidos en la función remezord, se puede implementar el algoritmo de Parks- McClellan. Fo y Mo son dos vectores de igual magnitud.

Fo(k) y Fo(k+1) k impar especifica bandas de frecuencia y Mo(k) y Mo(k+1) la correspondiente magnitud para cada frecuencia. El filtro obtenido es la mejor aproximación por minimax.

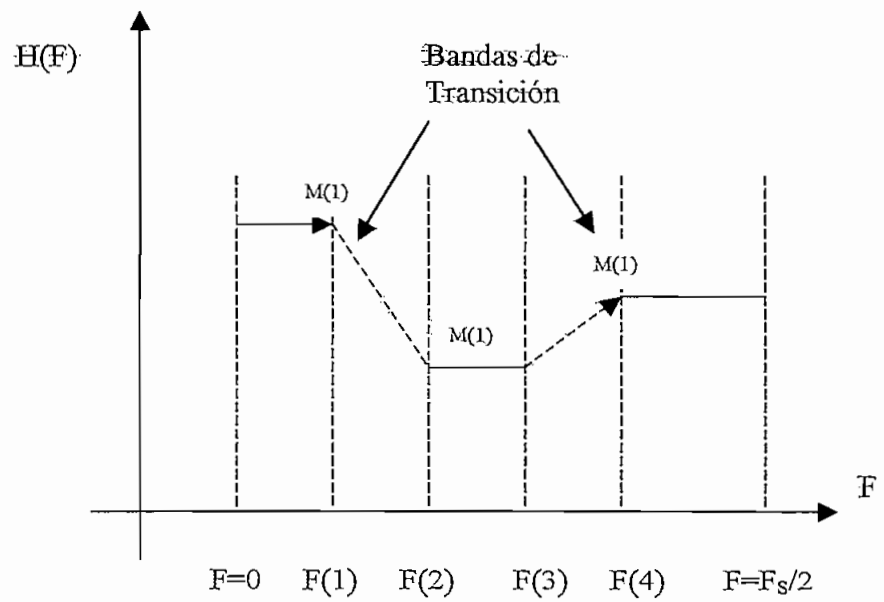



Figura 1.8 Bandas de transición

## 1.2 ALGORITMO LMS (*Least-Mean-Square*)

El algoritmo LMS ha ganado considerable popularidad por su simplicidad y fácil implementación, lo cual lo hace atractivo para muchas aplicaciones en las que interesa bajos requerimientos computacionales.

El algoritmo LMS pertenece a la familia de los algoritmos de gradiente estocástico. Con el término "estocástico" se pretende distinguir este algoritmo del Steepest Descent, que utiliza un gradiente determinista para el cálculo de los coeficientes del filtro.

El LMS comprende dos procesos básicos:

- Un proceso de filtrado, implica el cálculo de la salida generada por un filtro transversal[1], y la generación de una estimación del error comparando esta salida con la respuesta deseada.
- Un proceso adaptivo, que realiza el ajuste automático de los coeficientes del filtro de acuerdo con la estimación del error.

El concepto de estimación óptima es de fundamental importancia en el tratamiento de los filtros adaptivos.

El combinador lineal adaptivo constituye un estimador del tipo no recursivo; su estructura se muestra en la figura 1.9, donde el estimado $y_n$ (señal de salida del combinador lineal ) se define en términos de una combinación lineal de la señal de entrada $X_n$:

$$y_n = w_1 x_{1n} + w_2 x_{2n} + .... + w_{M-1} x_{M-1,n} \qquad (1.5)$$

El vector de la señal de entrada se define como:

[1] Filtro Transversal.- En esta clase de filtros es la transversal puesto que existe una relación simple y analíticamente manejable (lineal) entre la función de transferencia y los coeficientes del filtro.

$$X_n = \begin{bmatrix} x_{1n} \\ x_{2n} \\ \cdot \\ \cdot \\ \cdot \\ x_{M-1,n} \end{bmatrix}$$

Se asume que todas las componentes de la señal de entrada, aparecen simultáneamente sobre todas las líneas de entrada, a instantes discretos de tiempo indexados por el subíndice j. Los coeficientes de ponderación o factores de multiplicación $w_1, w_2, \ldots w_n$ son ajustables, el vector de coeficientes de ponderación de define de la siguiente manera:

$$W = \begin{bmatrix} W_1 \\ W_2 \\ \cdot \\ \cdot \\ \cdot \\ W_n \end{bmatrix}$$

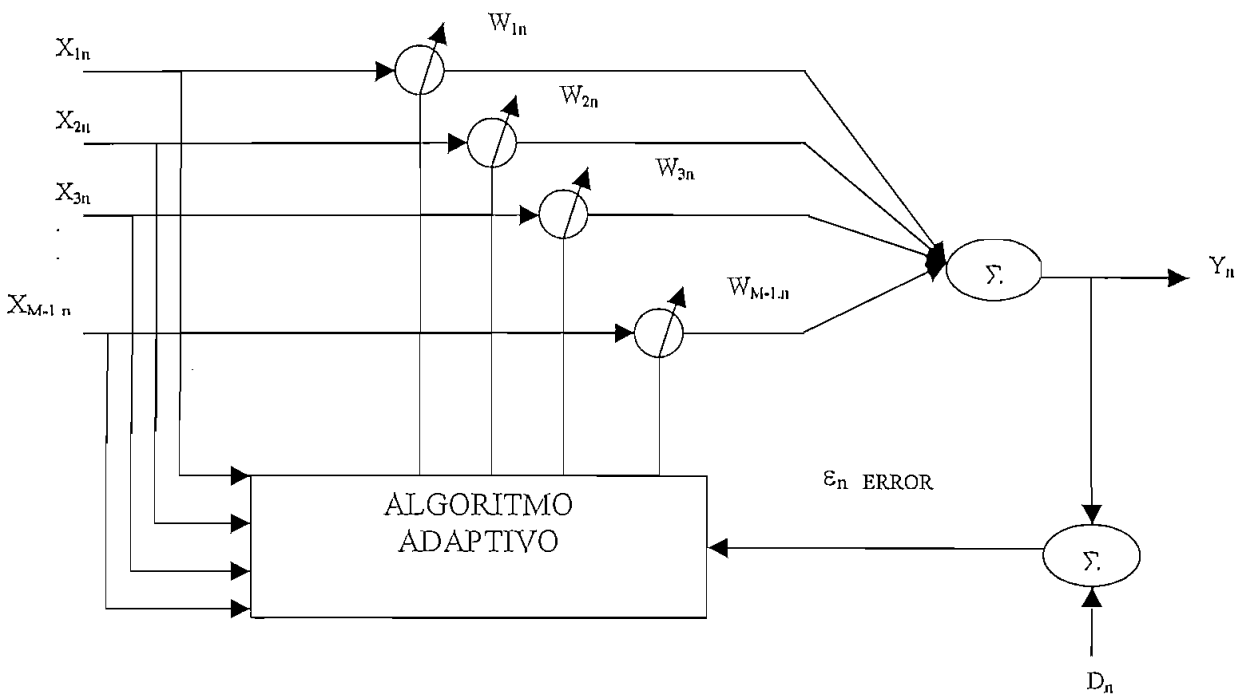



Fig. 1.9 Diagrama esquemático del combinador lineal adaptivo.

De acuerdo a la ecuación (1.5), la señal de salida $y_j$ es igual al producto interno de $X_n^T$ y W:

$$Y(n) = X(n)^T W = W^T X(n) \tag{1.6}$$

donde $W^T$ representa la matriz transpuesta de W.

Se define la señal de error $\varepsilon_n$, como la diferencia entre la señal $d_n$ (la cual se provee externamente) y la salida del combinador lineal $y_n$.

$$\begin{aligned} \varepsilon(n) &= d(n) - y(n) \\ \varepsilon(n) &= d(n) - X(n)^T W = d(n) - W^T X(n) \end{aligned} \tag{1.7}$$

El algoritmo adaptivo tiene como finalidad ajustar los valores de los coeficientes de ponderación del combinador lineal a fin que se minimice el error cuadrático medio (valor esperado del cuadrado de la señal de error); se trata entonces de encontrar un conjunto de coeficientes de ponderación, tales que produzcan una señal a la salida del sistema que sea igual o tan cercana como sea posible a la respuesta deseada, de modo que el error que se genere sea el mínimo.

En el diseño de un filtro adaptivo (variable en el tiempo) hay que tomar en cuenta que para cada instante de tiempo n, se ha de encontrar un conjunto de coeficientes óptimos $w_k(n)$ para k=0,1,.....M-1. Pero se puede simplificar el problema considerando una actualización de los pesos del filtro de la forma:

$$w(n+1) = w(n) + \Delta w_n \tag{1.8}$$

donde $\Delta w_n$ es un factor de corrección que se aplica a los coeficientes w(n) en el instante n para obtener el nuevo conjunto de coeficientes w(n+1) en el instante n+1. Esta ecuación de actualización es la base de los algoritmos adaptivos, y el diseño de cada filtro adaptivo requiere definir este factor $\Delta w_n$.

Elevando al cuadrado la ecuación (1.7) se obtiene una expresión para el error cuadrático dado por:

$$\varepsilon_n^{\,2} = d_n^{\,2} - 2d_n X_n^{\,T} W + W^T X_n X_n^{\,T} W \qquad (1.9)$$

El error cuadrático medio será entonces el valor esperado del error al cuadrado.

$$E[\varepsilon_n^{\,2}] = E[d_n^{\,2}] - 2E[d_n X_n^{\,T}]W + W^T E[X_n X_n^{\,T}]W \qquad (1.10)$$

Definiendo el vector $R_{xd}$ como la correlación cruzada entre la respuesta deseada $d_j$ (un escalar) y el vector de datos de entrada $X_j$, y la matriz $R_{xx}$ como la autocorrelación de los datos de entrada, se tiene:

$$R_{xd} = E[d_n X_n] = E\begin{bmatrix} d_n x_{1n} \\ d_n x_{2n} \\ \cdot \\ \cdot \\ \cdot \\ d_n x_{n,M-1} \end{bmatrix}$$

$$R_{xx} = E[X_n X_n^{\,T}] = E\begin{bmatrix} x_{1n}x_{1n} & x_{1n}x_{2n} & x_{1n}x_{3n} & \cdots \\ x_{2n}x_{1n} & x_{2n}x_{2n} & x_{2n}x_{3n} & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \cdots & x_{M-1,n}x_{M-1,n} \end{bmatrix}$$

La matriz $R_{xx}$ es simétrica[2], definida positiva y en raras excepciones semidefinida positiva. Se puede entonces expresar el error cuadrático medio como:

$$E[\varepsilon_n^{\,2}] = E[d_n^{\,2}] - 2R_{xd}^{\,T} W + W^T R_{xx} W \qquad (1.11)$$

[2] Matriz simétrica.- Una matriz cuadrada A es simétrica si $A = A^t$, es decir, si $a_{ij} = a_{ji}\ \forall$ i, j.

El error cuadrático medio es por lo tanto una función cuadrática del vector de coeficientes de ponderación en cualquier tiempo particular, cuya representación tendrá la forma de una superficie hiperparaboidal cóncava (Figura 1.10). El hecho de ajustar los valores de los coeficientes de ponderación para minimizar el error involucra descender a lo largo de dicha superficie con la finalidad de obtener el punto mínimo de la misma, generalmente usando métodos del gradiente.

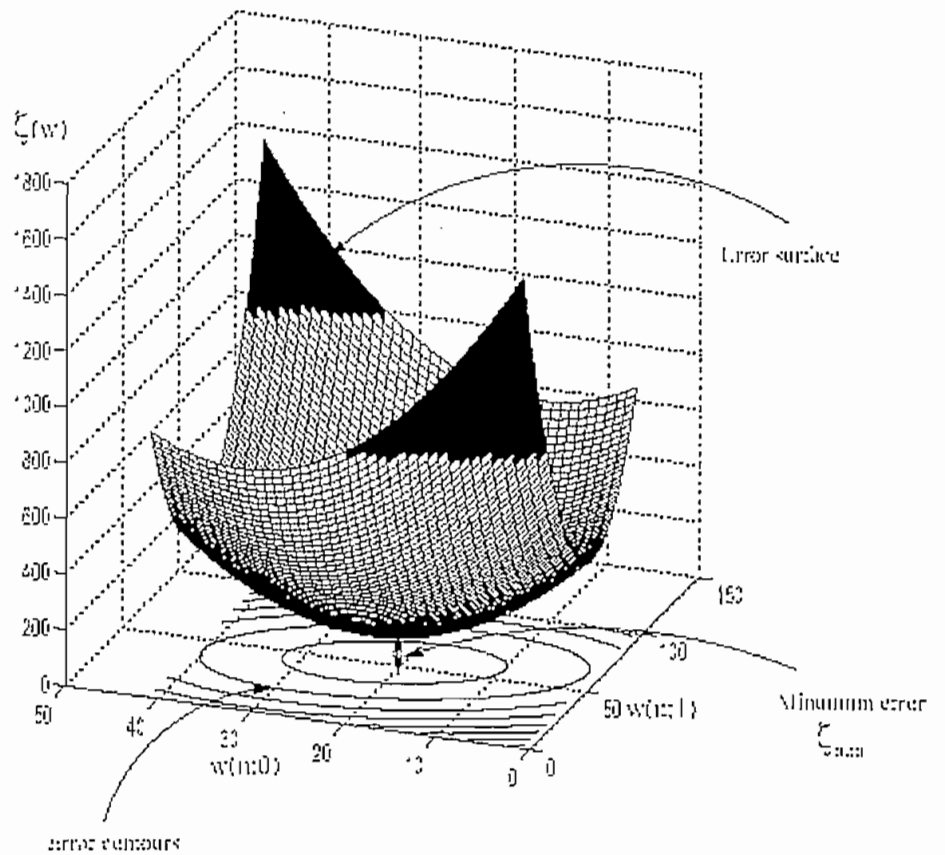



Figura 1.10 Error cuadrático medio

El gradiente en cualquier punto de la superficie de comportamiento se puede obtener diferenciando la función de error cuadrático medio mse (*mean square error*), con respecto al vector de coeficientes de ponderación.

$$\nabla_n = \frac{\partial E[\varepsilon_n^{\,2}]}{\partial W} \qquad (1.12)$$

sustituyendo en esta expresión, la fórmula para el error cuadrático medio, se tiene:

$$\nabla_n = \frac{\partial}{\partial W}\left[E[d_n{}^2] - (2R_{xd}{}^T W) + (W^T R_{xx} W)\right]$$

$$\nabla_n = -2\frac{\partial}{\partial W}(R_{xd}{}^T W) + \frac{\partial}{\partial W}(W^T R_{xx} W)$$

entonces se tiene lo siguiente:

$$\frac{\partial}{\partial W}(R_{xd}{}^T W) = R_{xd}$$

$$\frac{\partial}{\partial W}(W^T R_{xx} W) = 2R_{xx} W$$

Por lo tanto el gradiente del error cuadrático medio es igual a:

$$\nabla_j = -2R_{xd} + 2R_{xx} W \qquad (1.13)$$

Entonces el vector de coeficientes de ponderación óptimo $W^*$ se lo obtiene igualando a cero el gradiente de la función error cuadrático medio:

$$-2R_{xd} + 2R_{xx} W^* = 0$$

entonces

$$W^* = R_{xx}{}^{-1} R_{xd} \qquad (1.14)$$

El vector de coeficientes de ponderación óptimo $W^*$ permite obtener el error cuadrático medio mínimo.

El algoritmo LMS es una implementación del método de la pendiente más pronunciada. Un procedimiento interactivo en el cual la actualización del vector de coeficientes, $W_{j+1}$, se realiza en base al vector de coeficientes de ponderación anterior , $W_j$, más un término proporcional al gradiente negativo, esto es:

$$W_{n+1} = W_n - \mu_n \nabla_n \qquad (1.15)$$

siendo $\mu_{(n)}$ el FACTOR DE CONVERGENCIA, que determina cuan grande o pequeño es el paso, y cuyo valor será muy importante para la velocidad de convergencia y la estabilidad del algoritmo LMS.

Se puede afirmar que dicho algoritmo conduce a la convergencia de H(n) hacia $H_{opt}$ en el límite cuando $n \to \infty$, supuesto que la secuencia de tamaños de paso $\mu(n)$ es absolutamente sumable y tal que $\mu_{(n)} \to 0$ si $n \to \infty$.

Pero se ha empezado suponiendo que $R_{xx}$ y $R_{dx}$ eran matrices conocidas, y, sin embargo, tal como se ha expuesto anteriormente, ese no es el caso de las aplicaciones que requieren de un filtrado adaptivo (en general), lo que obliga a utilizar estimaciones del vector de dirección en lugar del vector real. Recordando el principio de ortogonalidad:

$$E[e(n)x(n-k)] = 0 \qquad k=0,1...M-1 \qquad (1.16)$$

si MSE mínimo.

Se puede expresar el primer miembro, que es precisamente $r_{EX}$ (k) (correlación cruzada del error y los datos) vectorialmente como:

$$E[e(n)X(n)con.X(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ \cdots \\ x(n-(M-1)) \end{bmatrix}$$

Dado que $E[\varepsilon(n)X(n)] = R_d - R_x H(n)$:

$$\nabla j = -\nabla_c MSE(n) = -2[R_x * H(n) - R_d] = 2E[\varepsilon(n)\, X(n)] \qquad (1.17)$$

Se observa que, efectivamente, cuando se ha alcanzado el mínimo del MSE, como ε(n) y X (n) serán no correlacionados, resulta nulo el gradiente del MSE respecto de los coeficientes, como ya se ha expuesto anteriormente al hablar de cómo llegar al mínimo.

Widrow hace una estimación del gradiente, con vistas a no tener que calcular valores esperados, que consiste en considerar que el valor del gradiente en la iteración n-ésima es:

$$\nabla_c MSE(n) = 2[R_x * H(n) - R_d] = -2E[\varepsilon(n)\, X(n)] = -2\varepsilon(n) X(n) \qquad (1.18)$$

aproximando el valor esperado por el valor instantáneo, de modo que queda:

$$H(n+1) = H(n) + 2\mu(n)\varepsilon(n) X(n) \qquad n=0,1..... \qquad (1.19)$$

Esta aproximación se basa en que si el valor esperado es el más probable, el valor actual es muy probable que coincida con el esperado.

Como se ve en la expresión del algoritmo, el valor del paso es variable. Es una práctica común en filtrado adaptativo el usar un tamaño de paso fijo, porque simplifica la implementación del algoritmo, y permite la adaptación a señales de características estadísticas variables con el tiempo, objetivo que con un paso variable que tendiera a cero a medida que $n \to \infty$ no es posible alcanzar. Así el algoritmo resultante es:

$$H(n+1) = H(n) + 2\mu\, \varepsilon(n) X(n) \qquad n=0,1.....$$

que se denomina ALGORITMO L.M.S. (least-mean-square) o ALGORITMO WIDROW-HOFF.

### 1.2.1 PROPIEDADES DEL ALGORITMO LMS

Se cita algunas propiedades importantes del algoritmo LMS. Se va a centrar en la convergencia, estabilidad y del error generado como resultado de usar estimaciones para el gradiente.

#### 1.2.1.1 CONVERGENCIA

El algoritmo converge siempre que:

$0<2\mu < 2/\lambda_K \qquad k=0,1...M-1$

siendo $\lambda_K$ los autovalores de la matriz de autocorrelación de la señal de entrada, $R_{XX}$, de elementos $r_{LK} = r_{XX}(l-k)$ con $l=0,1..M-1$ y $k=0,1...M-1$, y siendo $r_{XX}(l-k) = E[x(n)x(n-l+k)] = E[x(n-k)x(n-l)]$. Por tanto, el rango de valores que asegura la convergencia del algoritmo es $0<2\mu < 2/\lambda_{MAX}$ con $\lambda_{MAX}$ el mayor de los autovalores de la matriz de autocorrelación.

Como $R_X$ es una matriz de autocorrelaciones, los autovalores son siempre positivos, luego una cota superior de $\lambda_{MAX}$ será:

$$\lambda_{MAX} < \sum_{K=0}^{M-1} \lambda_K = TRAZA(R_X) = M*r_{XX}(0) = M*P_X \qquad (1.20)$$

siendo $P_X$ la potencia de la señal de entrada que es fácilmente estimable a partir de la señal de entrada, esto es, del conjunto de muestras de que se dispone.

Así, adoptando el intervalo de variación: $0<2\mu < 2/ M*r_{XX}(0)$, se tiene que mientras más grande sea $2\mu$ dentro del rango , más rápida será la convergencia, pero

menor será la estabilidad alrededor del valor mínimo (se pasa de largo el mínimo), y a menor $2\mu$ dentro del rango, más lenta es la convergencia, pero más estable será alrededor del valor óptimo (es posible que si $2\mu$ es muy grande, el algoritmo permanezca indefinidamente sin llegar al MSE mínimo, si no mas bien oscilando en torno a él , con lo que no llega nunca a alcanzarlo).

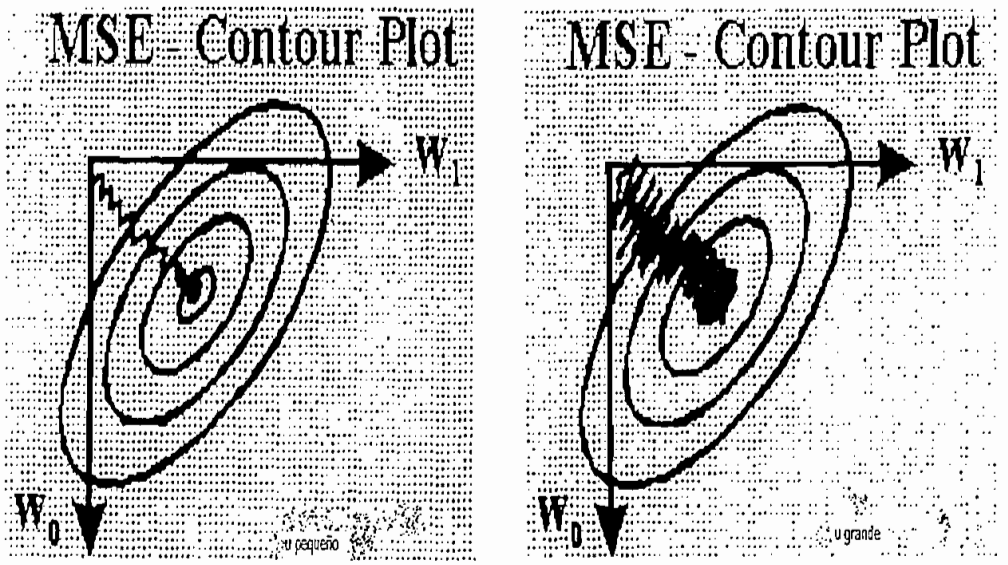



Figura 1.11 Convergencia del MSE

### 1.2.1.2 VELOCIDAD DE CONVERGENCIA

La velocidad de convergencia es máxima cuando $|1-2\mu\lambda_K|$ es pequeño. Entonces, si se toma el límite $2\mu = 2/\lambda_{MAX}$ :

$$|1-2\mu\,\lambda_K|_{minimo} = |1-2\lambda_K/\lambda_{MAX}|_{minimo} = |1-2\lambda_{MIN}/\lambda_{MAX}|$$

Así, el valor más pequeño para $|1-2\mu\lambda_K|$ viene determinado por la relación $\lambda_{MIN}/\lambda_{MAX}$, luego es $\lambda_{MIN}/\lambda_{MAX}$ quien fija la velocidad de convergencia. Si $\lambda_{MIN}/\lambda_{MAX}$ es pequeño, la convergencia será lenta, porque $|1-2\mu\lambda_K|$ es grande , pero si es

grande, la convergencia será rápida, permitiendo usar el filtrado adaptivo en aplicaciones en tiempo real.

### 1.2.1.3 ERROR SUPLEMENTARIO

Otra importante característica del algoritmo LMS es el "ruido" (error) resultante de la estimación del gradiente en lugar de usar el gradiente verdadero.

Este "ruido" causa fluctuaciones aleatorias de los coeficientes del filtro alrededor de sus valores óptimos , lo cual provoca un aumento en el MSE mínimo a la salida del filtro adaptivo.

Así, el MSE total es el $MSE_{min}$ ya visto, más un $E_A$ que se denomina ERROR CUADRÁTICO MEDIO EN EXCESO. Dicho error en exceso puede aproximarse a :

$$E_A \approx 2\mu\,(M * r_{XX}(0) / 2) * MSE_{MIN} \qquad (1.21)$$

con $r_{XX}(0)$ la potencia de la señal de entrada.

Así, se observa que $E_A$ es proporcional al tamaño del paso, por lo que el tamaño del paso $2\mu$ se ha de elegir en base a un compromiso entre convergencia del algoritmo rápida, con valores del paso altos (siempre menores que $2/M*r_{XX}(0)$), y un $E_A$ pequeño (que implica valores del paso pequeños). En la práctica es deseable obtener un $E_A < E_{MIN}$:

$$E_A / E_{MIN} \approx (\mu * M * r_{XX}(0) * E_{MIN} / 2 * E_{MIN}) < 1 \Rightarrow \mu < 1 / M * r_{XX}(0) \qquad (1.22)$$

que es precisamente el criterio anteriormente elegido para garantizar la convergencia del algoritmo, luego con un $\mu<1/M*r_{XX}(0)$ se solucionan ambos problemas. Normalmente suele elegirse el valor de $\mu$ igual a:

$$\mu = 1 / 2M*r_{XX}(0) \qquad (1.23)$$

### 1.2.1.4 ERROR DE RETRASO

En la introducción se ha indicado que el filtrado adaptivo se utiliza cuando las características de la señal son desconocidas a priori, o cuando sus estadísticos son variables con el tiempo.

En el segundo caso, es preciso tener en cuenta que el algoritmo LMS únicamente es apropiado para seguir señales cuyas características estadísticas varían lentamente con el tiempo.

Si los estadísticos de la señal son variables con el tiempo, la superficie que representa el MSE cambiará con el tiempo, y así lo hará también el MSE mínimo. En otras palabras, el $MSE_{min}$ es una función del tiempo, $MSE_{min}(n)$, y la superficie M-dimensional de error MSE se mueve con el índice n. Los coeficientes del filtro, fijados en $H_{opt}= R_{XX}^{-1}*R_{dx}$ ya no serán los óptimos, y tienen que ser reajustados.

(NOTA: precisamente el algoritmo LMS al no utilizar las matrices $R_{Xx}$ y $R_{DX}$ para actualizar los coeficientes, debido a que no se utiliza el gradiente verdadero sino una estimación del mismo, reduce el peso computacional del proceso y permite el uso del filtrado adaptativo en aplicaciones en TIEMPO REAL, cosa que un algoritmo que calculase ambas matrices para cada actualización de los coeficientes no permitiría).

El algoritmo LMS trata de seguir al $MSE_{min}(n)$ de la superficie M-dimensional, pero siempre se retrasa, debido al uso de gradientes estimados. Así, el algoritmo LMS incurre en otro tipo de error llamado ERROR DE

RETRASO, $E_L$ , cuyo valor cuadrático medio decrece con incrementos del paso $\mu$ ($E_L \approx 1/2\mu$ ). Así, el MSE total se puede expresar como:

$$MSE = MSE_{min}(n) + E_A + E_L \qquad (1.24)$$

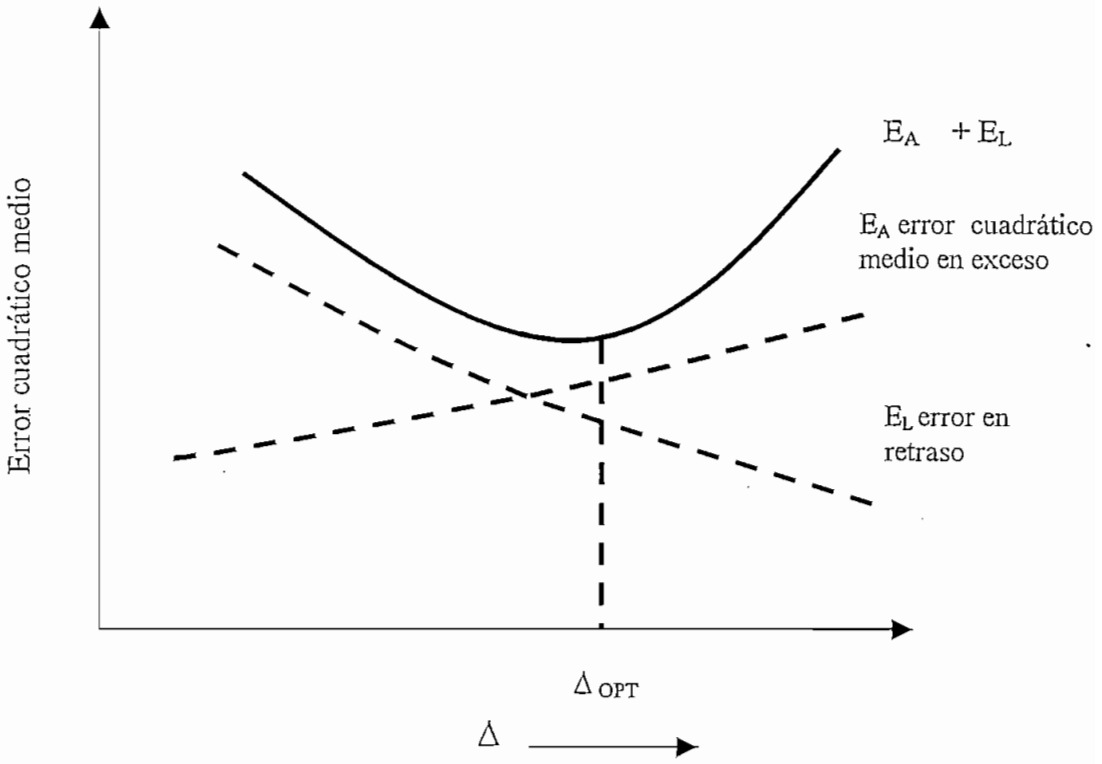



Figura 1.12 MSE total

Al aumentar el tamaño del paso aumenta $E_A$, pero disminuye $E_L$ . El MSE total exhibe un mínimo que determina la opción del paso óptimo.

Cuando las variaciones estadísticas de la señal son muy rápidas, el algoritmo no puede seguir estas variaciones, y será $E_L$ el que domine la actualización del filtro, por muy grande que se elija el paso. Por ello, el algoritmo resulta inapropiado en estos casos.

# CAPITULO 2. SIMULACIÓN DE UN FILTRO DIGITAL ADAPTIVO

## 2.1INTRODUCCIÓN

### 2.1.1 FILTROS WIENER

El filtro de Wiener es un filtro FIR estático, es decir, que sus coeficientes no varían con el tiempo, por lo que funciona solo para procesos estacionarios.

Una de las aplicaciones de un filtro de Wiener es la estimación de una señal a partir de otra conociendo la correlación cruzada entre ellas de antemano.

El filtro se utiliza en el siguiente esquema:

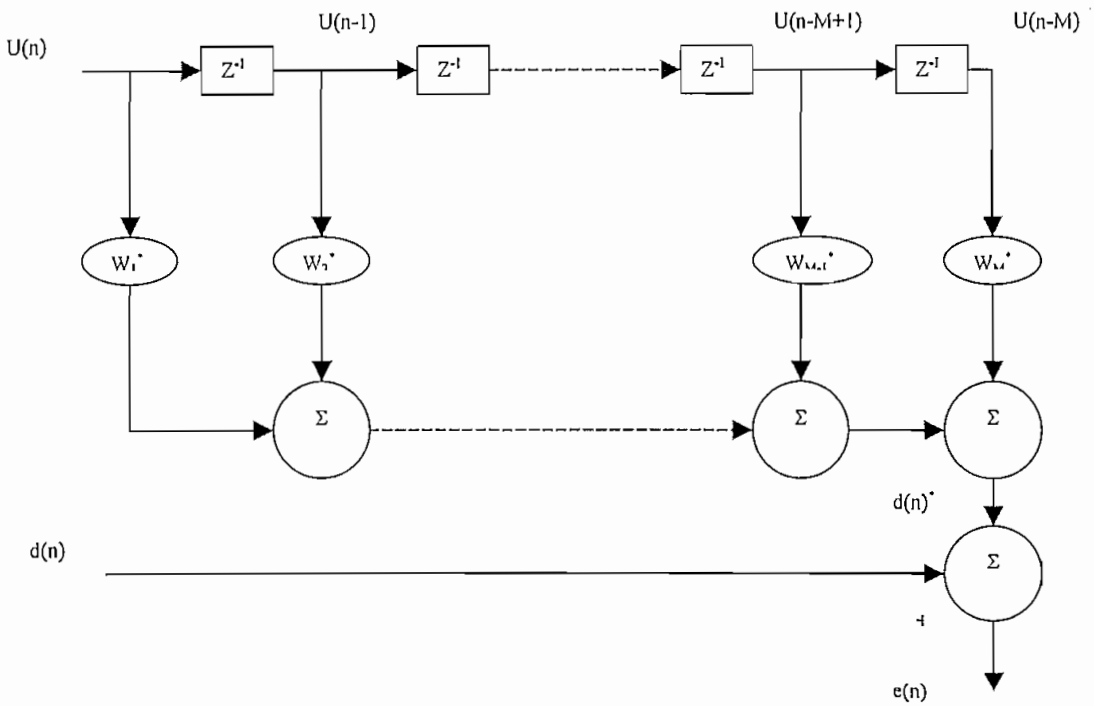



Figura 2.1 Esquema de un Filtro de Wiener.

Donde:

d(n): es la señal deseada.

u(n): es la señal de entrada al filtro, es decir, la señal que se utiliza para estimar d(n).

$\hat{d}(n)$: es la salida del filtro y

e(n) es el error en la estimación.

## 2.1.2 FILTROS ADAPTIVOS

El filtro de Weiner sólo funciona para procesos estacionarios, en los cuales se conoce de antemano la correlación entre la señal de interés y la que se usa para estimarla. Para lograr los objetivos se tiene que utilizar un filtro que se adapte a procesos no estacionarios también, ya que en el mundo real, los procesos estacionarios no existen. Por eso se usa un filtro adaptivo para alcanzar el objetivo.

Un filtro adaptivo es un filtro cuyos coeficientes son actualizados por un algoritmo adaptivo para optimizar la respuesta del filtro según un criterio deseado de desempeño.

En general los filtros adaptivos consisten en dos partes: un filtro cuya estructura es diseñada para lograr la función deseada sobre la señal de entrada, y un algoritmo adaptivo que ajusta los coeficientes del filtro de forma de lograr el mejor desempeño posible.

Los filtros adaptivos pueden ser usados en varias aplicaciones que requieren operaciones en tiempo real, por ejemplo, predicción adaptiva, ecualización de canales, cancelación de eco y cancelación de ruido. La implementación de un filtro adaptivo basada en procesadores digitales de señales (DSP) tiene muchas ventajas sobre otros enfoques. No solo por que los requerimientos de potencia, espacio y manufactura, son muy reducidos, sino que además, la programabilidad provee flexibilidad para la actualización del sistema y mejoras del *software*.

Las capacidades de flujo y manipulación de datos de un DSP son además los mayores factores a la hora de implementar un sistema de filtrado adaptivo.

## 2.2 FILTRO DIGITAL ADAPTIVO

Cuando se habla de "filtrado" se refiere a un proceso lineal diseñado para alterar el contenido espectral de una señal de entrada ( o una secuencia de datos) de un modo específico. El filtrado es realizado por FILTROS, cuya magnitud y/o fase satisfacen ciertas especificaciones en el dominio de la frecuencia.

El término "FILTRADO ADAPTIVO" implica que los parámetros que caracterizan al filtro, tales como ancho de banda, frecuencias de los ceros, etc cambian con el tiempo, esto es, los coeficientes, también llamados PESOS, de los filtros adaptivos cambian con el tiempo, en contraposición a los coeficientes de los filtros fijos que son invariantes con el tiempo.

El estudio se centra en los FILTROS ADAPTIVOS DIGITALES, que son aquellos en los que la entrada, la salida y los pesos del filtro están cuantificados y codificados en forma binaria. El tener los coeficientes del filtro no fijos sino variables es necesario cuando no se conocen a priori las características estadísticas de la señal a filtrar, o cuando se conocen y se sabe que son cambiantes con el tiempo, y, así, es en esos casos donde se precisa de un FILTRADO ADAPTIVO.

La ecuación de entrada-salida de un filtro adaptivo digital que se cumple en señales y filtros causales es :

$$y(n) = \sum_{i=0}^{N} a_i(n)x(n-i) - \sum_{j=1}^{M} b_j(n)y(n-j) \quad n \geq 0 \qquad (2.1)$$

Donde x(n) e y(n) son las muestras de entrada y salida, respectivamente, en el instante n, $a_i(n)$ y $b_j(n)$ son los pesos del filtro i-ésimo y j-ésimo en el instante

n, y N+M+1 es el número total de coeficientes del filtro. Si en lugar de usar $a_i(n)$ y $b_j(n)$ se utilizan $a_i$ y $b_j$, los coeficientes ya no serían variantes con el tiempo, y se encuentra ante un filtro fijo en lugar de ante un filtro adaptivo. Si $b_j(n) = 0$ para M, resulta un filtro adaptivo FIR, esto es, de respuesta impulsional finita:

$$y(n) = \sum_{i=0}^{N} a_i(n)x(n-i) \quad n \geq 0 \tag{2.2}$$

El filtro digital digital adaptivo podría perfectamente implementarse mediante un filtro IIR (respuesta impulsional infinita), pero los filtros FIR son mucho menos susceptibles que los IIR de ser INESTABLES. Hay que recordar que los filtros IIR tienen tanto polos como ceros, y, con solo la observación directa de la ecuación de entrada-salida del filtro en el dominio del tiempo (n), no se sabe dónde están los polos ni los ceros, lo que posibilita que los polos queden fuera de la circunferencia de radio unidad haciendo que el filtro sea inestable.

Además, aunque si se conociera teóricamente los coeficientes a utilizar para tener los polos y ceros donde se necesitan, y consiguiendo siempre la estabilidad del filtro, dado que se esta trabajando con filtros digitales, los coeficientes (pesos) del filtro siempre estarán cuantificados y codificados en forma binaria, con lo que es posible que por problemas de cuantificación los polos queden desplazados respecto del lugar teórico donde debieran estar, pudiendo salirse de la circunferencia de radio unidad, haciendo el filtro inestable. Ello no quiere decir que los filtros FIR sean siempre estables, de hecho, su estabilidad depende del algoritmo que se use para ajustar sus coeficientes. Sin embargo, se utilizan generalmente filtros FIR porque su estabilidad e inestabilidad es más controlable que en los IIR.

Existen muchos criterios que pueden adoptarse para llevar a cabo la adaptación de los pesos del filtro a las variaciones de la señal o señales de entrada. Se va a centrar aquí en el criterio de optimización de los coeficientes M.S.E. (*mean square error*), error cuadrático medio mínimo. Este criterio conduce

a una superficie N-dimensional (con N el número de coeficientes del filtro) que posee un único mínimo, que es al que se pretende llegar.

Para implementar el FIR digital adaptativo se utilizan, o bien la forma directa o la estructura LATTICE (podrían emplearse otras, pero éstas son las más habituales).

La forma directa es la más sencilla de implementar, conduciendo a algoritmos igualmente sencillos. La estructura LATTICE (enrejado, celosía) presenta mejores propiedades que la forma directa, pues ofrece mayor robustez frente a errores de redondeo y una mayor eficiencia computacional. Sin embargo, aumenta la complejidad de los algoritmos. Se adopta aquí la primera estructura debido a su sencillez.

La forma del filtro FIR adaptativo será entonces:

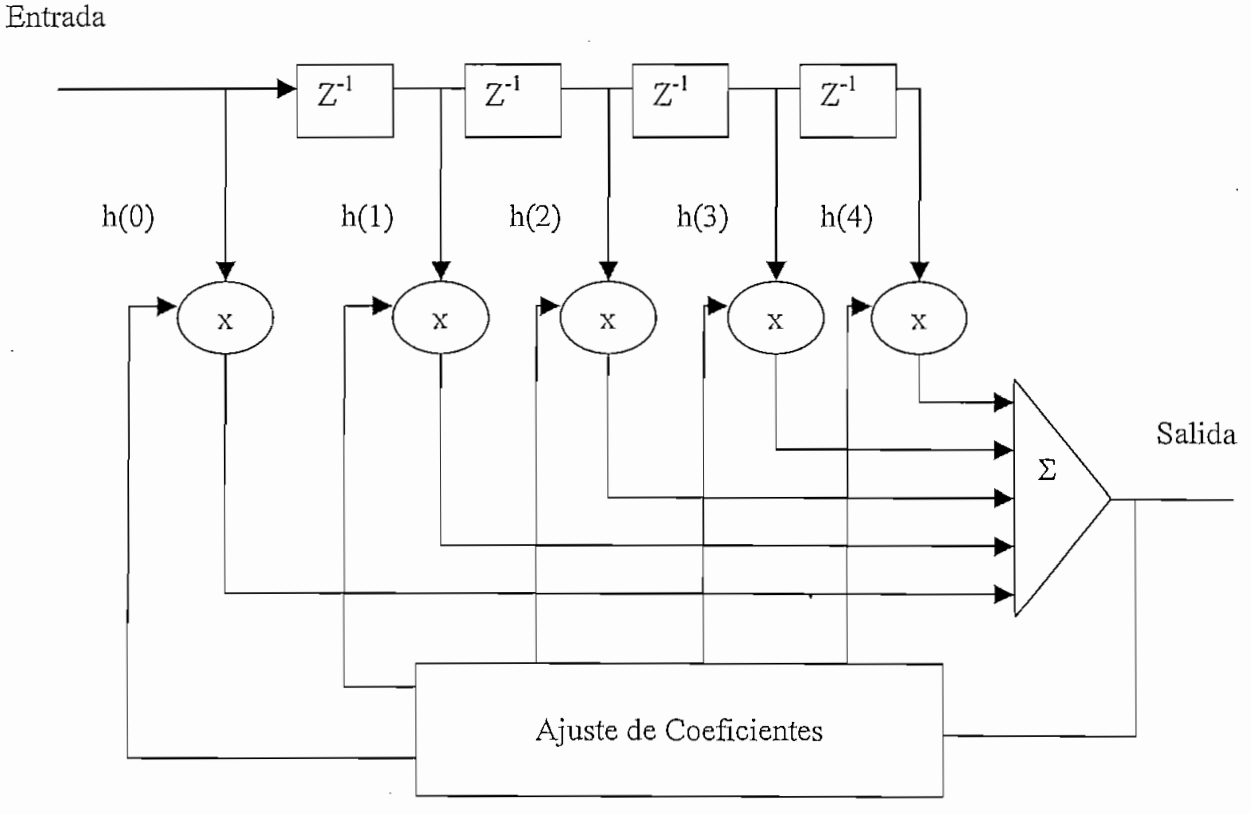



Figura 2.1 Estructura básica de un filtro FIR adaptivo.

Un modelo general de un sistema de filtrado adaptivo se muestra en la figura siguiente:

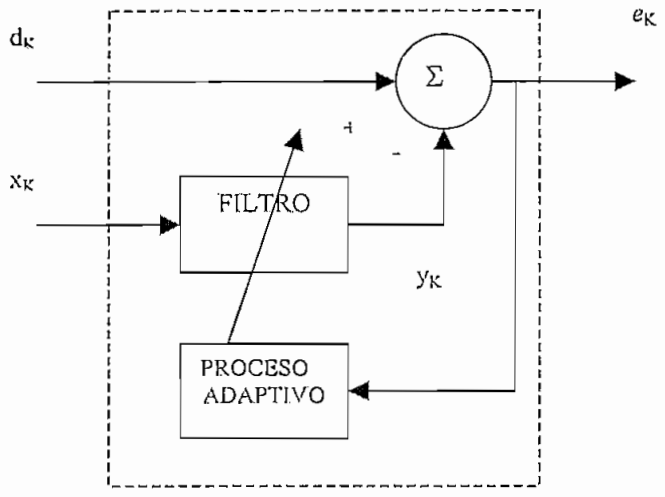



Figura 2.2 Modelo general de un sistema de filtrado adaptivo

El sistema tiene 6 componentes principales definidos como:

- $X_K$ señal de entrada
- $d_K$ señal deseada
- $Y_K$ señal de salida
- $\varepsilon_K$ señal de error
- FILTRO, proceso de filtrado
- PROCESO ADAPTIVO, alguna clase de algoritmo

El propósito general de un sistema de filtrado adaptivo es filtrar la señal de entrada $X_K$ para que se asemeje (de alguna forma) a la señal deseada de entrada.

La señal de entrada $X_K$ es filtrada y se obtiene $Y_K$, la señal de error $\varepsilon_K$ es la diferencia entre la señal deseada $d_K$ y la señal de salida $Y_K$ .

Esta diferencia se retroalimenta al proceso adaptivo que evalúa la "similitud" entre las dos señales según algún criterio de rendimiento y modificando la respuesta de frecuencia del filtro para aumentar esa similitud.

La eficiencia de un filtro adaptivo depende de una serie de factores, como el tipo de filtro (IIR o FIR), la estructura del mismo, y la forma en que se define la medida de prestaciones (error cuadrático medio, mínimo error cuadrático, etc.).

El estudio se centra en el filtro FIR por varias razones:

- El error cuadrático medio para un filtro transversal es una función cuadrática de los pesos del filtro. La superficie del error es un paraboloide con un solo mínimo, y por ello la búsqueda del error cuadrático medio mínimo es relativamente sencilla
- Asegurando que los coeficientes del filtro estén acotados se puede controlar fácilmente la estabilidad del filtro
- Existen algoritmos para la actualización de los coeficientes simples y eficientes
- Las prestaciones de estos algoritmos son perfectamente conocidas en términos de convergencia y estabilidad
- El filtro adaptivo FIR funciona lo suficientemente bien para satisfacer el criterio de diseño

Hay tres conceptos principales en la medida de las prestaciones de un algoritmo adaptivo:

- **Complejidad** (computacional): definida como el número de operaciones que han de realizarse en cada instante para implementar el algoritmo adaptivo; generalmente es un importante factor para determinar si la implementación en tiempo real resulta viable

- **Velocidad de convergencia**: Es la tasa a la que el filtro adaptivo converge a la solución de Wiener y es inversamente proporcional a la complejidad
- **Desajuste**: mide la diferencia entre la solución de Wiener y la obtenida con el algoritmo adaptivo. Generalmente, el desajuste es inversamente proporcional a la velocidad de convergencia y a la complejidad

Los algoritmos adaptivos para filtros digitales pueden ser aproximadamente divididos en tres grupos

- Algoritmos *least mean squares* (LMS)
- Algoritmos *least squares* (LS)
- Algoritmos *blind*

La pregunta crucial de todos los algoritmos adaptivos es como calcular el vector óptimo de los coeficientes del filtro en tiempo discreto, los cuales produzcan el menor error posible $\varepsilon_n$

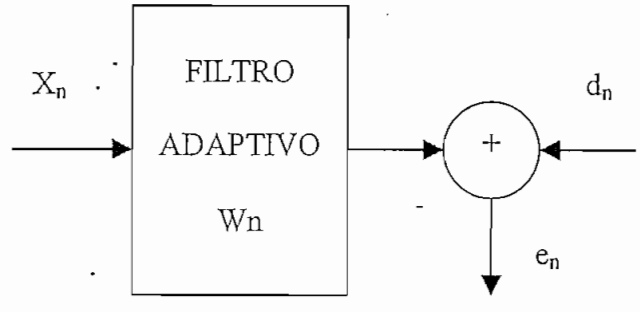



Figura 2.4 Señales que intervienen en el filtrado adaptivo

$$Y(n) = W(n)^T X(n) = \sum_{i=0}^{N-1} w_{i,n} x(n-i)$$

A continuación se menciona las principales ventajas y desventajas que tienen los tres grupos de algoritmos adaptivos mencionados anteriormente.

- Algoritmos LMS

  Ventajas:
  - Complejidad de aplicación baja
  - Muy robusto
  - Numéricamente estable

  Desventajas:
  - Adaptación lenta

  Como ya se trato en el capitulo 1 el algoritmo LMS trata de minimizar el error MSE (mean squared error) y según la ecuación (2.19), el tamaño del paso $\mu$ determina la velocidad de convergencia.

- Algoritmos LS

  Ventajas:
  - Adaptación muy rápida
  - Converge muy cerca del óptimo

  Desventajas
  - Complejidad de implementación alta
  - Problemas numéricos
  - Necesita una señal de referencia o una secuencia de entrenamiento

- Algoritmos Blind

  Ventaja:

- No necesita una señal de referencia o una secuencia de entrenamiento

Desventaja:

- Complejidad alta, convergencia lenta, o ambas

## 2.3 SIMULACIÓN DE UN FILTRO DIGITAL ADAPTIVO

Como ejemplo de la utilización de un Filtro Digital Adaptivo se toma la identificación de sistemas.

Se refiere aquí a la utilización de un sistema adaptativo para encontrar el filtro FIR que mejor reproduce la respuesta de otro sistema cuya respuesta de frecuencia no es conocida.

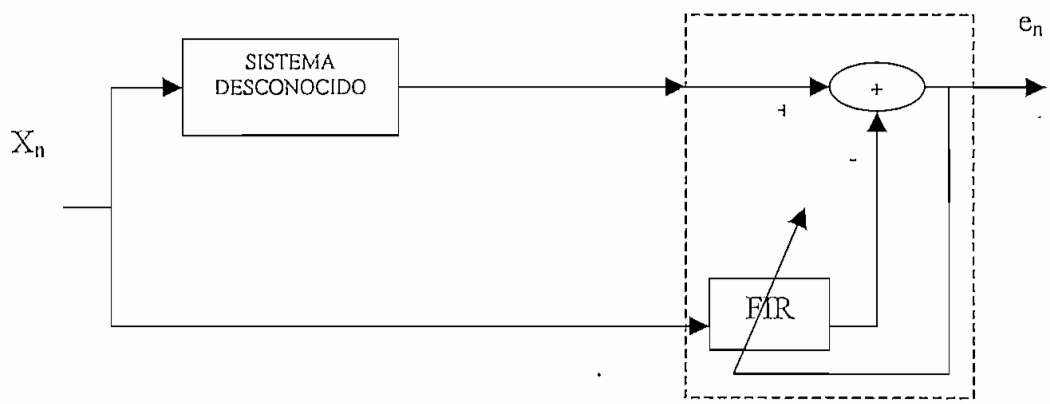



Figura 2.5 Sistema adaptivo

Cuando la salida del sistema, $e_n$, es nula, los coeficientes del filtro FIR, está dando la misma salida que el sistema desconocido para la misma entrada, con lo que está reproduciendo el comportamiento de éste.

Esto funciona perfectamente únicamente en el caso en que el sistema desconocido tenga la respuesta frecuencial de un FIR, pero, si por ejemplo, es un filtro todo-polos o tiene polos y ceros (en lugar de sólo ceros como un FIR), el sistema no será capaz de dar salida cero porque la respuesta frecuencial del FIR no será exactamente igual a la del sistema desconocido, pero se conseguirá la mejor aproximación del sistema a modelar que un filtro FIR puede dar. Mientras más coeficientes tenga el filtro, mejor será la aproximación.

A continuación se muestra el listado del programa utilizando MATLAB.

```
M = 10;                 % Orden del Filtro
L=256;                  % Longitud de los vectores
mu=.04;                 % Tamaño del paso para el filtrado adaptivo

H=randn(M,1);           % Números del azar normalmente distribuidos
W=zeros(M,1);           % Pesos del filtro
x=randn(L,1);           % Números del azar normalmente distribuidos
e=zeros(L,1);           % Error
etap=zeros(L,1);        % Error de coeficientes en cada etapa

for n=M:L-M-1,
  X=x(n:-1:n-M+1);
  d=H'*X;               % Cálculo de la salida del sistema desconocido
  y=W'*X;               % Cálculo de la salida del filtro
  e(n)=d-y;             % Error
  W=W+mu*e(n)*X;        % Actualización de los coeficientes del filtro
  etap(n)=norm(W-H); % Error entre los coeficientes estimados y los coeficientes
del sistema
end
```

```
hold off
plot(e(M:L))
xlabel('indice de tiempo (n)')
ylabel('e(n) - error residual')
title('Error Residual')
hold off

plot(abs(e(M:L)))
xlabel('Indice de tiempo (n)')
ylabel('|e(n)| - Error Residual')
title('Curva de aprendizaje del Error Residual')


pause
plot(20*log10(etap(M:L)));
xlabel('Indice de tiempo (n)')
ylabel('etap(n) - Error de pesos en cada interaccion (dB) ')
title('Curva de aprendizaje del error de pesos')

pause

plot(1*H)
hold

plot(W,':')
xlabel('Indice de tiempo (n)')
ylabel('Vector de coeficientes')
title('Convergencia del vector de coeficientes (...) vs.Verdadero (---)')
```

Como resultado se tiene los siguientes gráficos.

Para $\mu = 0.04$ :

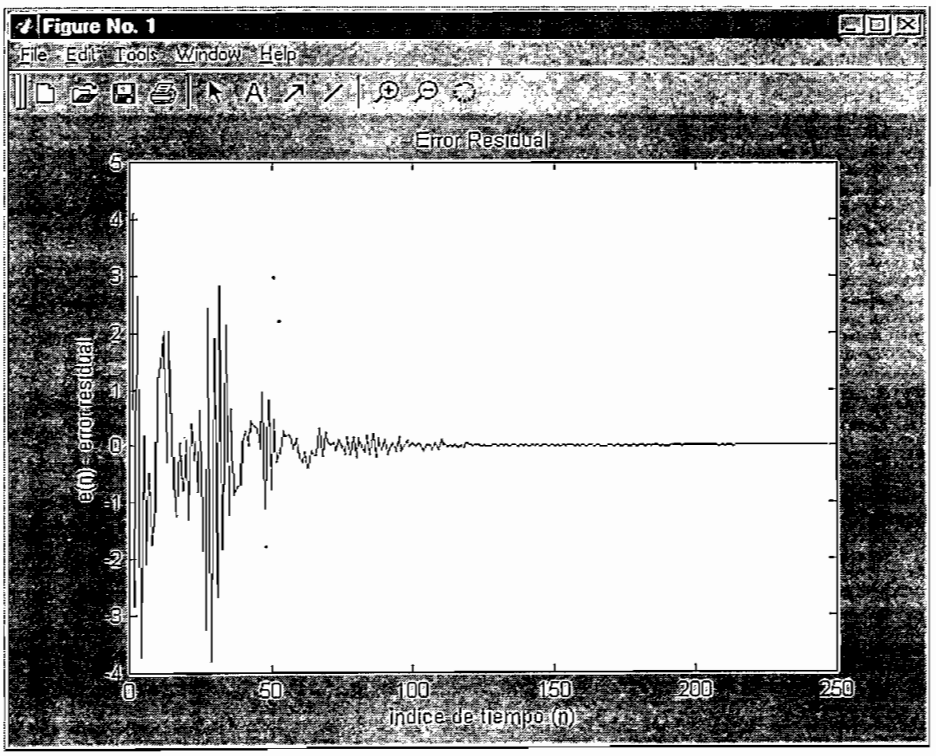



Figura 2.6 Error residual

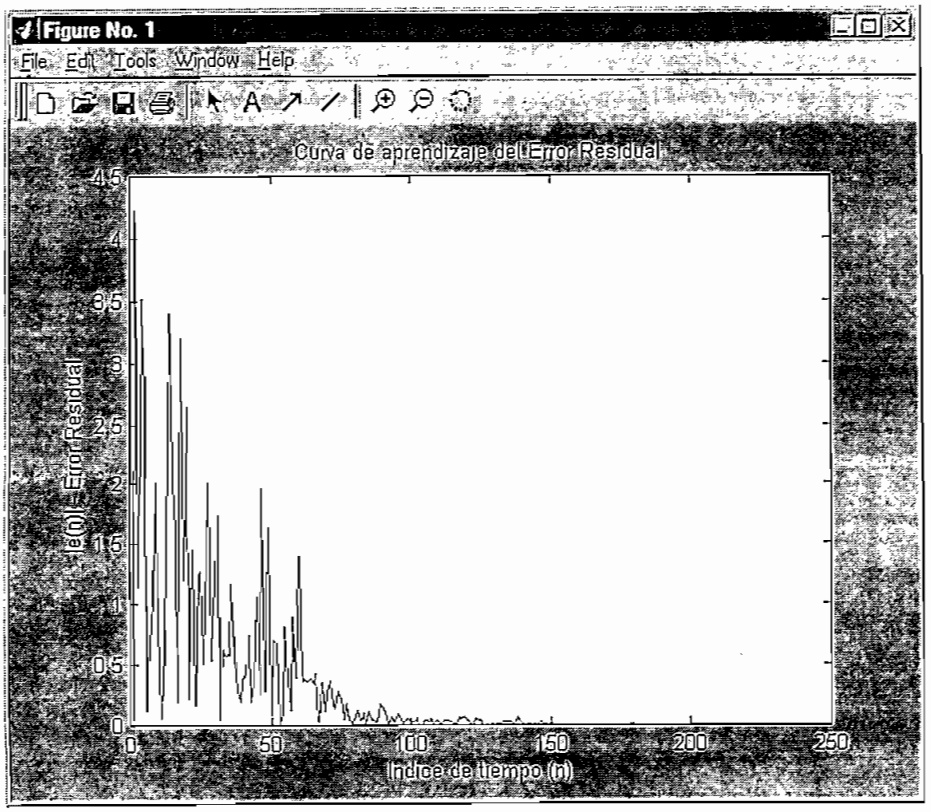


Figura 2.7 Curva de aprendizaje del error residual

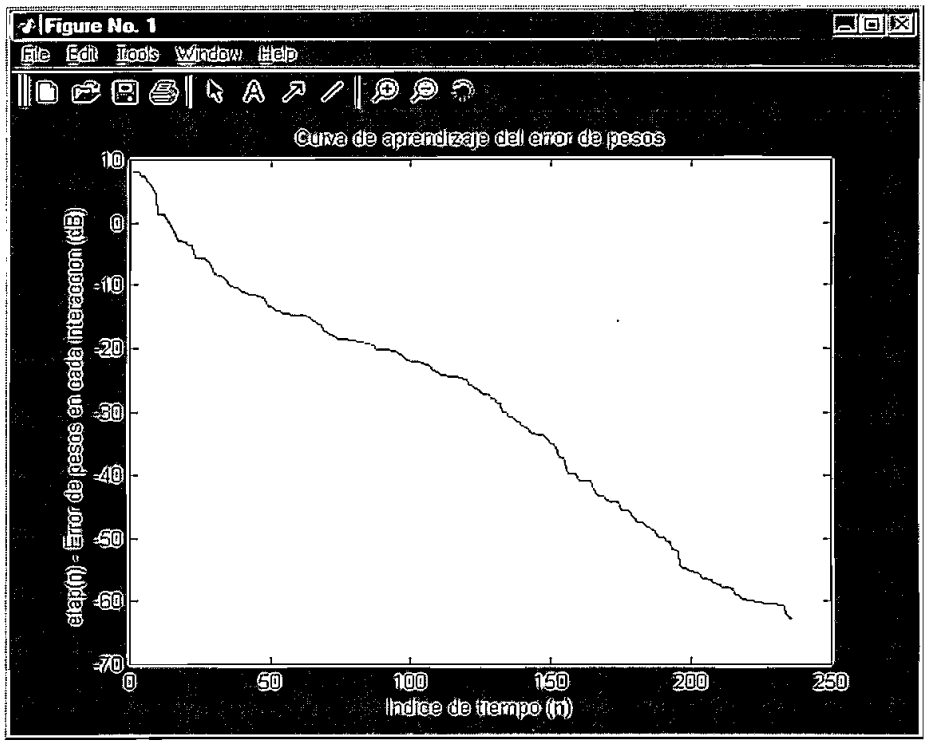



Figura 2.8 Curva de aprendizaje del error de pesos

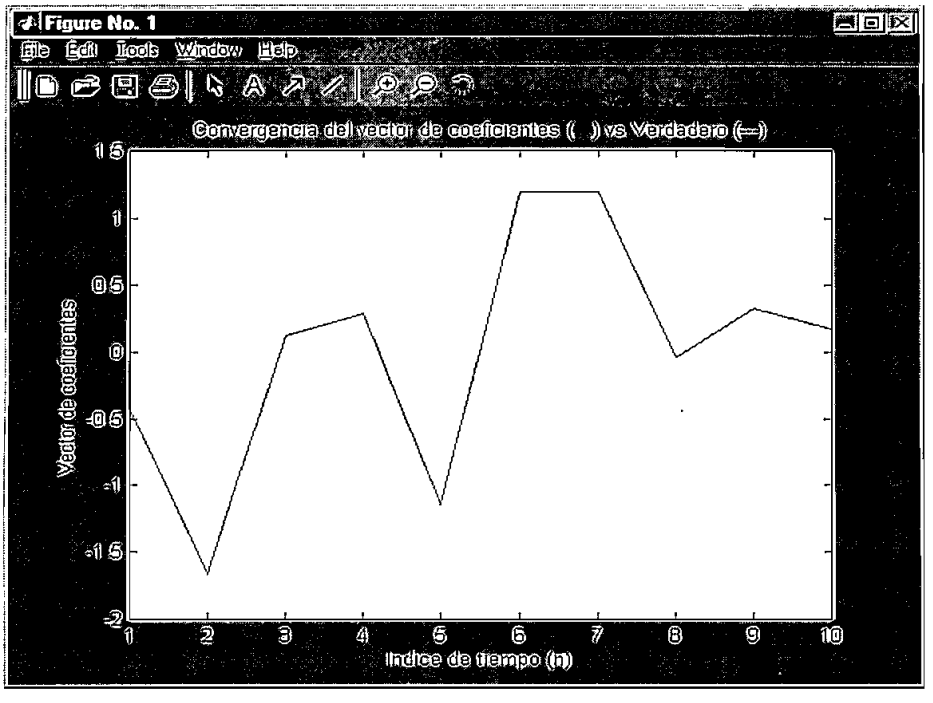



Figura 2.9 Convergencia del vector de coeficientes

Nota: En la figura 2.9 las dos curvas coinciden.

Para $\mu = 0.01$

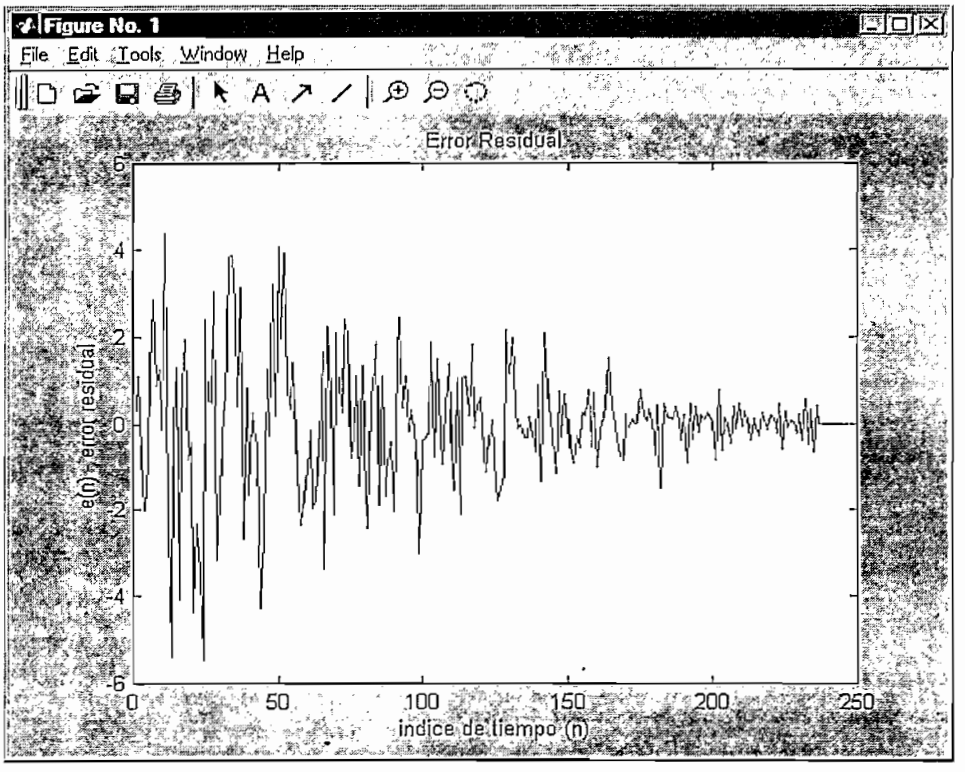



Figura 2.10 Error residual

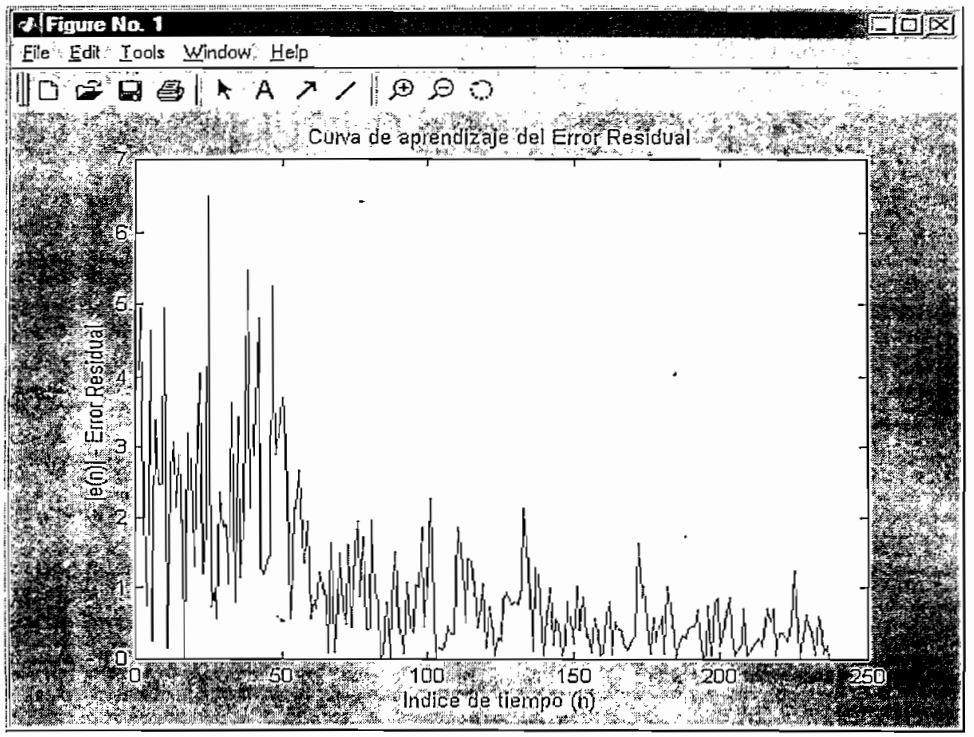


Figura 2.11 Curva de aprendizaje del error residual

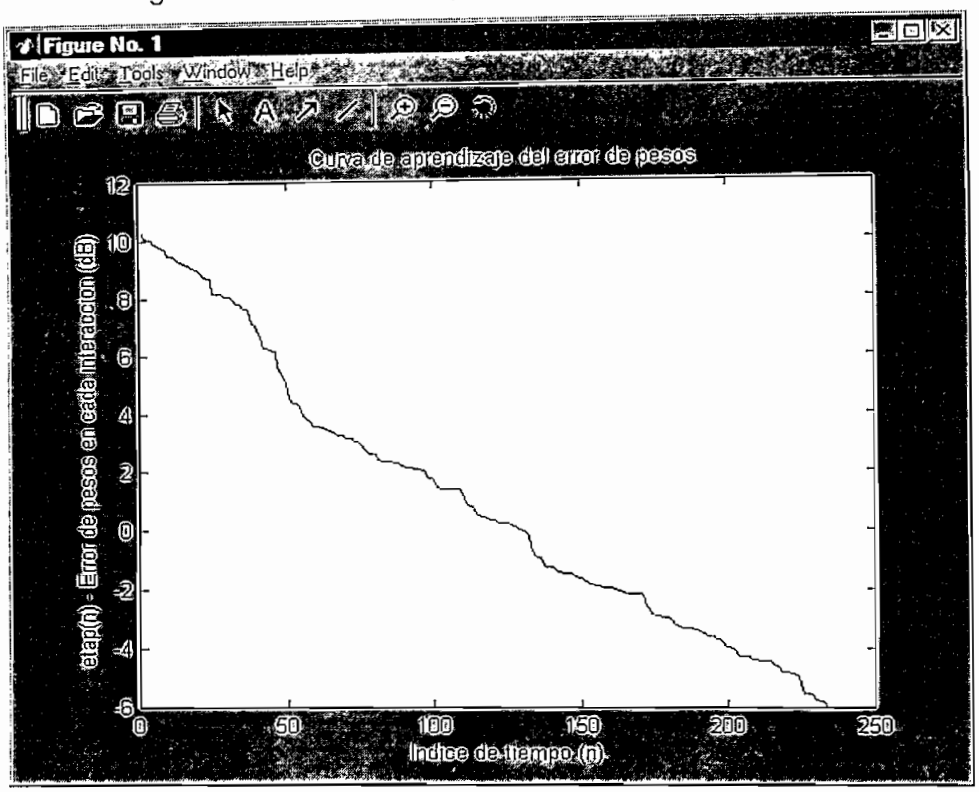



Figura 2.12 Curva de aprendizaje del error de pesos

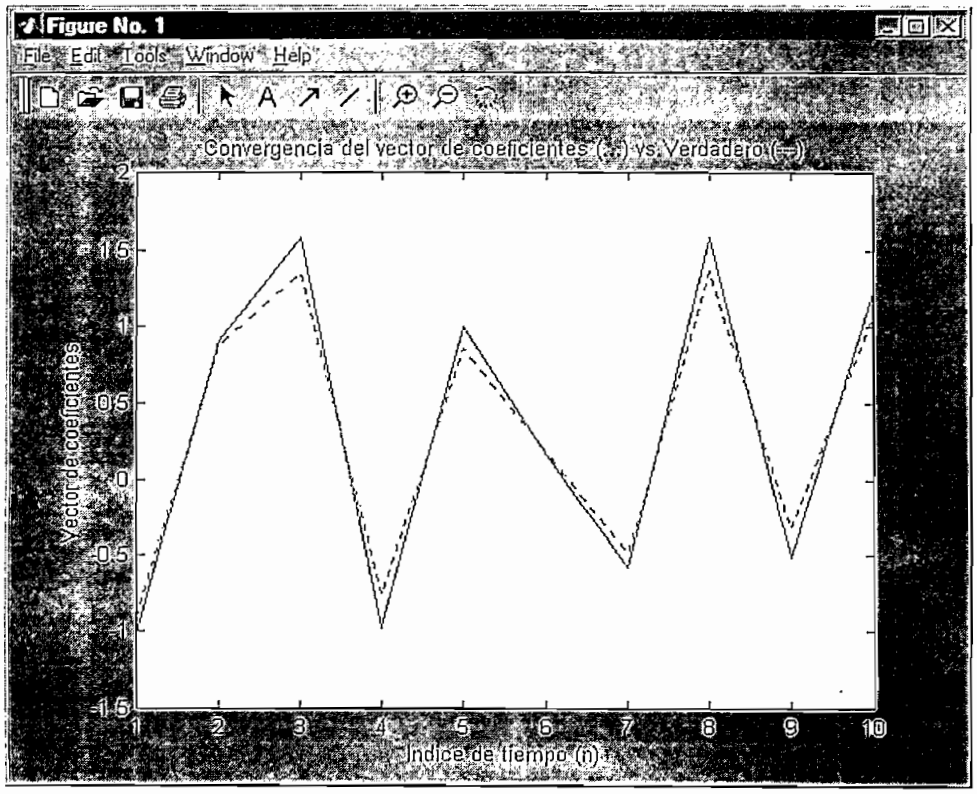



Figura 2.13 Convergencia del vector de coeficientes

Como se puede observar el tamaño del paso $\mu$ es de importancia para la velocidad de convergencia del algoritmo adaptivo LMS, un valor de $\mu$ menor implica una convergencia mas lenta hacia el estacionario, por lo que el error de los coeficientes del filtro FIR llegara con mas seguridad pero demandará mas tiempo al valor óptimo que en este caso son los coeficientes del sistema desconocido que se eligió al azar, si el algoritmo LMS converge rápidamente en la simulación se llega al vector óptimo de coeficientes del filtro FIR los cuales representan a los coeficientes del sistema desconocido.

# CAPITULO 3. SISTEMA DE CANCELACIÓN DE ECO EN CANALES TELEFÓNICOS

## 3.1 INTRODUCCIÓN

El eco juega un papel importante en determinados sistemas de telecomunicaciones. El eco en sistemas de telefonía es generalmente indeseable ya que éste se da en la conexión telefónica entre abonado y la central local que se realiza a 2 hilos, realizándose la conversión a 4 hilos en la híbrida de la central local, las cuales no operan perfectamente y su efecto genera muchos problemas en la fidelidad de una conversación entre abonados, por lo cual se ve la necesidad de minimizar este efecto con la utilización de sistemas de supresión de eco.

Cualquier punto a lo largo de un sistema de transmisión donde exista una discontinuidad en la impedancia puede ser causante de eco. Dado que los sistemas de transmisión telefónicos están compuestos por diferentes sistemas con distintas posibilidades de interconexión, en cada conexión puede existir una discontinuidad de impedancia significativa . A pesar de esto, las centrales que conforman la red troncal pueden ser diseñadas para evitar discontinuidades de impedancia significativa entre ellas. Sin embargo, la conexión entre el subscriptor y la central local presenta un inconveniente debido a que los lazos locales que llevan la señal a cada subscriptor varían de uno a otro por las diferencias en longitud y calibre de los cables empleados en los lazos, por las condiciones del medio ambiente, etc .. Este desacople de impedancias causa una reflexión, o eco, de la señal de la persona que habla que se regresa a través del canal por el cual está escuchando.

En su forma mas simple, un sistema de transmisión telefónico consiste en un par de cables cuyos terminales conectan dos teléfonos. Aquí, cualquier reflexión puede ser causante de eco. Los sistemas de *transmisión* que intervienen en una conexión telefónica pueden ser de dos o de cuatro líneas.

Para pasar de un sistema de dos líneas a uno de cuatro y viceversa se usa un acoplador pasivo bidireccional de cuatro puertos conocido como híbrido . Éste producirá un eco cercano. El híbrido del terminal opuesto también genera un eco que retornará al canal de recepción del sistema de cuatro líneas mezclado con la señal transmitida por la otra persona.

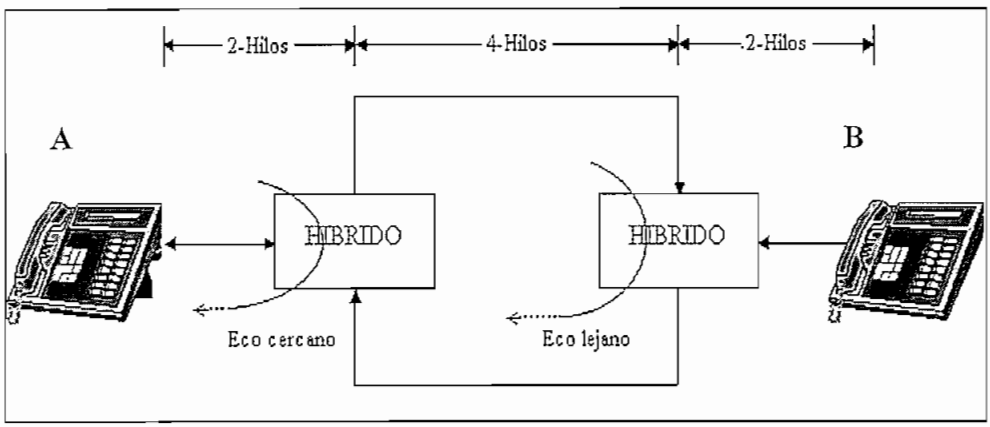



Figura 3.1 Esquema general de un sistema de transmisión telefónico

En una conversación telefónica, la tolerancia al eco dependerá de la magnitud del mismo, así como del retardo con que éste se perciba. La magnitud del eco dependerá de las pérdidas que éste experimente a su paso por el canal de ida y vuelta y mas la pérdida de retorno en el híbrido distante; mientras que las causas de retardos son debidas comúnmente a largas distancias físicas, procesamiento de las señales y dispersión en los canales telefónicos.

Los retardos mas grandes que se generan en un sistema telefónico se deben a largas distancias físicas. Por ejemplo en una transmisión por satélite.

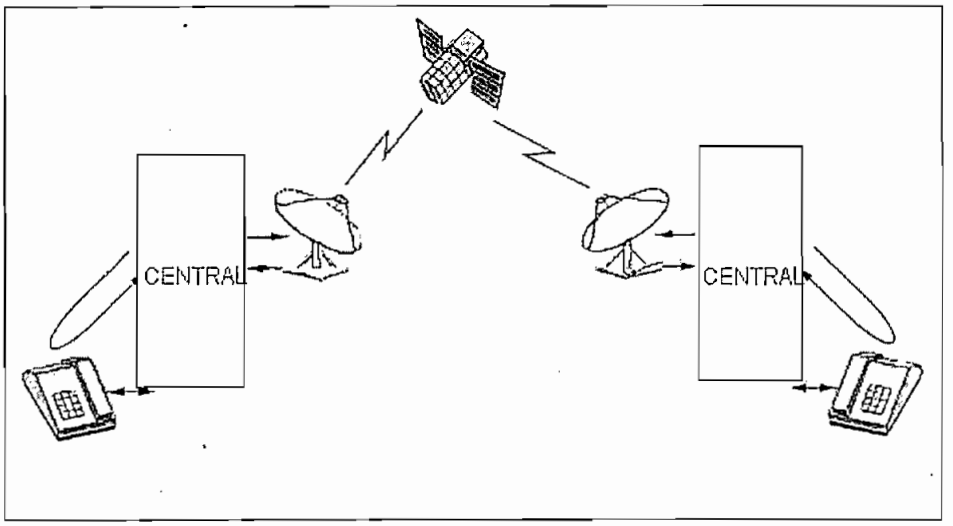



Figura 3.2 Transmisión por satélite

La forma moderna para manejar este fenómeno es el empleo de un cancelador de eco.

En aplicaciones de telefonía celular estos dispositivos se ubican en el centro de interrupción móvil. En circuitos de telefonía de larga distancia se localizan usualmente en el centro de conmutación internacional. Un esquema general de cancelación que se podría aplicar en telefonía es el siguiente:

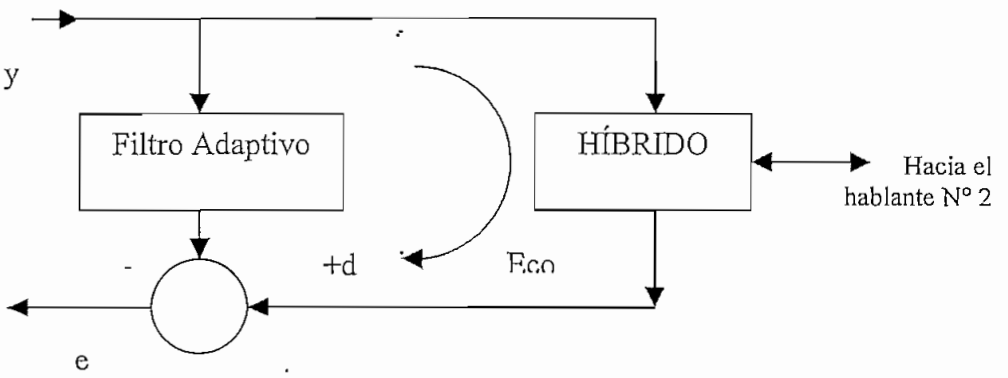



Figura 3.3 Esquema general del un sistema de cancelación de eco aplicado en telefonía

y =Señal transmitida por el hablante N° 1

d =Señal del hablante Nº 2+eco

e = Señal del hablante Nº 2 sin eco.

## 3.2 TIPOS DE ECO [4]

### 3.2.1 ECO ACÚSTICO

El eco acústico se genera tanto en micro teléfonos analógicos como digitales, el nivel del eco se relaciona al tipo y calidad del equipo usado. Esta forma de eco es producida por la existencia de acoplamiento entre el auricular y el micrófono en microteléfonos y dispositivos de manos libres.

En la Figura 3.4 se muestra una situación típica, cuando las personas están manejando en sus automóviles. Es esta situación el sonido del altavoz es oído por un oyente, sin embargo este sonido es recogido por el micrófono, directa e indirectamente, después de que éste se refleje desde el techo, ventanas y asientos del automóvil. El resultado de esta reflexión es la creación de eco multivía y múltiples armónicos del eco, que si no es eliminado, es transmitido hacia atrás al extremo distante y es oído por el locutor como eco. El uso predominante de teléfonos de manos libres en las oficinas ha exacerbado el problema del eco acústico.

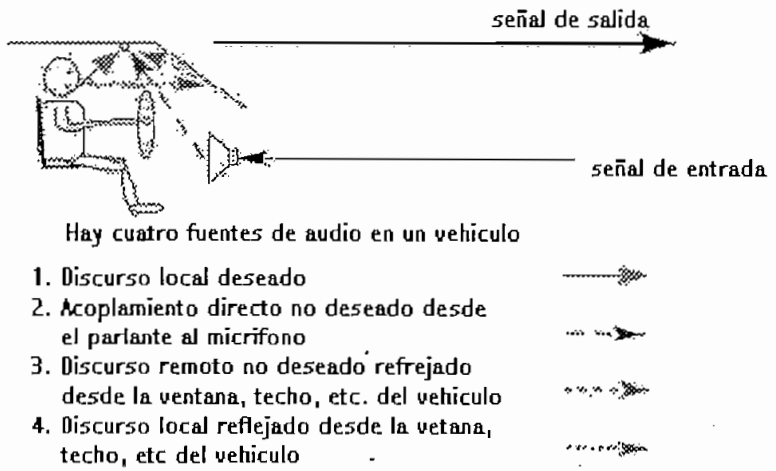



Figura 3.4 Eco acústico en un teléfono celular digital

[4] www.iec.org./online/tutorials/echo_cancel/index.html

La cancelación del eco acústico funciona modelando el discurso, es decir, estimando la señal y luego la pasa al altavoz ya quitado cualquier eco recogido por el micrófono. Este tipo de operación necesita una unidad mas compleja que se usa en telefonía para quitar muchos ecos acústicos (multipath) generados por cada silaba del discurso.

### 3.2.2 ECO HÍBRIDO

El eco híbrido, es la fuente primaria de eco en la Red Pública Conmutada (PSTN). Este eco es generado eléctricamente como señal de voz que se transmite por la red del lazo local (2 alambres) al lazo remoto (4 alambres) a través de la híbrida, reflejando energía eléctrica hacia el portavoz del circuito de 4 alambres. La cantidad real de señal reflejada depende del acoplamiento entre el circuito balanceado de la híbrida y el lazo local.

Normalmente el acoplamiento es pobre, produciendo un nivel considerable de señal reflejada que es llamado como ERL (perdida de retorno de eco). A más alto ERL, menos señal reflejada al hablador, y viceversa. Si el retraso del viaje redondo total ocurre dentro de justo unos milisegundos (es decir: dentro de 20 ms), el eco hace una contribución positiva a la calidad de la llamada. Sin embargo, cuando el retraso de la red total excede 30 ms, los beneficios desaparecen , y dan como resultado ecos molestos.

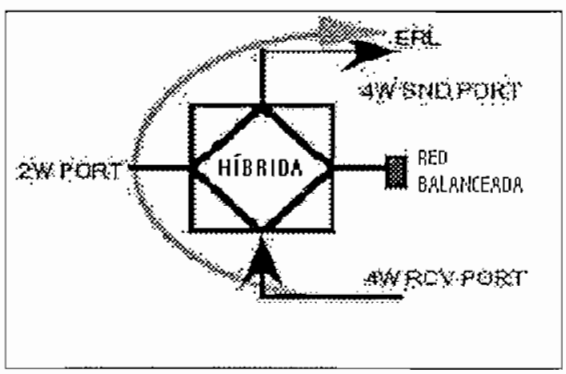



Figura 3.5 Acoplamiento entre el circuito balanceado de la *híbrida* y el lazo local

El eco acústico presenta particularidades respecto al eco eléctrico, ya que:

- el retardo que sufre la señal de eco acústico es mucho mayor que el que sufre la señal de eco eléctrico (cientos de ms frente a decenas de ms debido a las distintas velocidades de propagación de la señal por el cable o por el aire) con lo que es preciso que el filtro de cancelación acústico sea largo, esto es, tenga muchos coeficientes, para así poder adaptarse a ese retardo.
- Las características del eco acústico cambian muy rápidamente con el movimiento de objetos y/o individuos en el entorno del terminal, que pueden poner obstáculos o favorecer el camino de vuelta de la señal del auricular al micrófono.
- El eco acústico no llega atenuado por la híbrida, como el eléctrico, sino que se ve amplificado por el amplificador previo al auricular y por el posterior al micrófono. Así, si no se considera la atenuación que sufre la señal al viajar por el cable o por el aire, la señal de eco eléctrico que sale de la híbrida está atenuada respecto a la original debido a la propia híbrida, y se ve amplificada una sola vez, por el amplificador previo al auricular antes de salir por él. Sin embargo, la señal de eco acústico sale por el auricular y sin sufrir ninguna atenuación extra, vuelve a ser amplificada por el amplificador posterior al micrófono, con lo que adquiere niveles importantes y, además, podría realimentarse positivamente si después vuelve a rebotar en la híbrida volviendo al auricular (esto en ausencia de ambos canceladores, por supuesto),y pudiendo así crecer indefinidamente. Para evitar esto, se ha de conseguir que el filtro sea muy bueno en su adaptación, y que tenga buena velocidad de convergencia (que anule cuanto antes ese eco para evitar que se de esa realimentación positiva)

El hecho de que tenga que ser más largo el cancelador de eco acústico, para ajustarse a ese retardo mayor, hace que aumente la carga computacional y disminuya la velocidad de convergencia (a más largo es, más coeficientes hay que adaptar en cada interación, y menor es la velocidad de convergencia) hacia el vector de coeficientes óptimo del FIR adaptivo, lo cual es incompatible con las

características del canal, y, además, siempre quedará un eco residual que será molesto.

### 3.2.3 REDES DE LARGA DISTANCIA [5]

En redes de larga distancia, los canceladores de eco se utilizan para quitar el eco, hecho notable por el retraso inherente en medios de transmisión. Generalmente se utilizan dos canceladores de eco en el camino de transmisión de redes de larga distancia como se muestra en la Figura 3.6

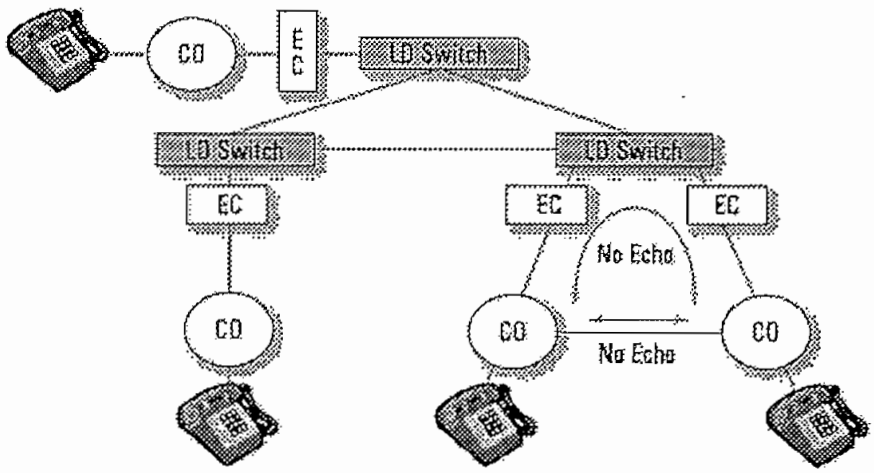



Figura 3.6 Red de larga distancia

Donde:

CO: Central telefónica local.

EC: Cancelador de eco

### 3.2.4 REDES INALÁMBRICAS [5]

En una típica llamada telefónica de la Red Pública Conmutada al un teléfono inalámbrico, el eco eléctrico se genera en la híbrida no balanceada de la

---

[5] http://tetes.cba.ufl.edu/ism6223f00/romanip/echo.html

Red Pública Conmutada en el extremo de la llamada telefónica. En aplicaciones de corta distancia este eco es imperceptible al oído humano, porque el retraso es tan corto que el oído humano no diferencia entre la señal fuente del eco y el eco en si. Las aplicaciones de larga distancia inducen retraso principalmente debido a la propagación y así requieren canceladores de eco para el funcionamiento correcto.

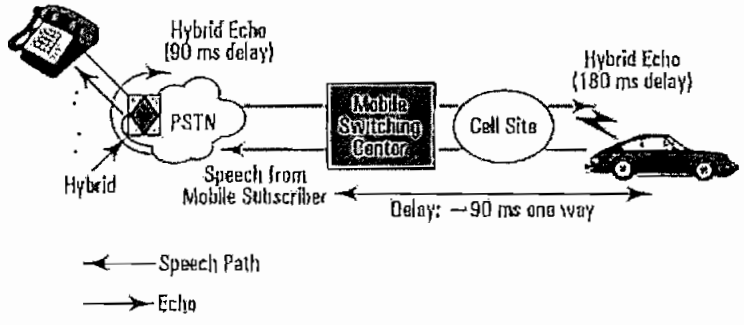



Figura 3.7 Eco generado en la híbrida

## 3.3 CANCELACIÓN DE ECOS EN CANALES TELEFÓNICOS

Se supone dos abonados conectados a su central local correspondiente. La conexión telefónica entre abonado y central local se realiza a 2 hilos, realizándose la conversión a 4 hilos en la híbrida de la central. Así, dado que la híbrida no está perfectamente equilibrada, existen varios caminos posibles para la señal que va de A hacia B:

- Camino normal de A hacia B
- ECO DEL HABLANTE: la señal que llega hasta la híbrida que está cercana a B, rebota en ella y así, parte de dicha señal vuelve hacia A de nuevo, de modo que A vuelve a oírse, y el resto va hacia B. Este eco eléctrico (eléctrico y no acústico porque va por cable y no por aire) se debe a una insuficiente atenuación en la híbrida, y provoca que el hablante oiga una versión retardada y atenuada de sí mismo.

- **ECO DEL OYENTE**: la señal que va de A hacia B, rebota en la híbrida cercana a B y así, parte de ella vuelve hacia A, y al llegar a la híbrida cercana a A, parte de ella va hacia A (que así vuelve a oírse) y el resto vuelve hacia B, con lo que vuelve a llegar a B, aunque atenuada y retardada.

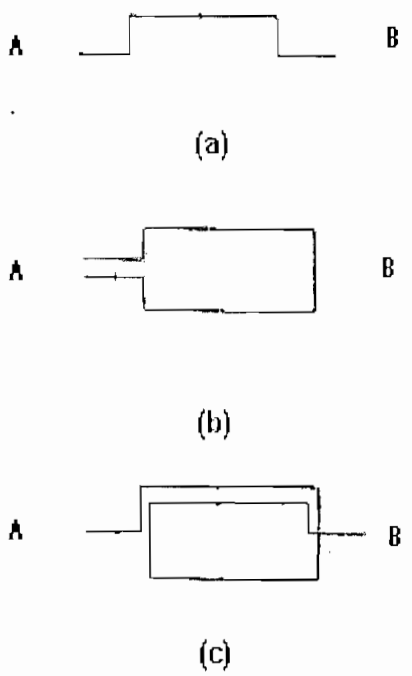



Figura 3.8 Caminos posibles de la señal que va de A hacia B

Como por la línea telefónica se puede mandar voz y datos, existen dos tipos de cancelación de eco en canales telefónicos: cancelación en transmisión de voz y cancelación en transmisión de datos. Se va a analizar aquí únicamente la primera de ellas.

### 3.3.1 CANCELACIÓN EN TRANSMISIÓN DE VOZ

El cancelador se coloca en el camino a 4 hilos más próximo al origen del eco, esto es, para cancelar el eco que se da debido al rebote en la híbrida de B, el cancelador se ha de colocar próximo a dicha híbrida, porque es donde más amplitud tiene el eco , ya que es ahí donde se ha generado, y si se colocase el cancelador en cualquier otro punto del camino a 4 hilos, el eco estaría más atenuado (porque el cable atenúa la señal) y más desfasado, lo que exigiría el uso de un filtro de más coeficientes (pues con M coeficientes sólo se puede corregir un desfase de M*fs muestras, con fs la frecuencia de muestreo de la señal).

Así, si y(n) es la señal que emite A hacia B, al llegar a la híbrida de B no es atenuada suficientemente, y así, además de ir hacia B, parte vuelve hacia A, sólo que lo que regresa hacia A es una versión atenuada y retardada de si mismo, señal que denotaremos r(n), siendo, así, $r(n)=\alpha y(n-n_o)$, donde $\alpha$ representa un coeficiente que da cuenta de la atenuación que sufre la señal en su "rebote" en la híbrida, y $n_o$ representa el retardo que experimenta (ese retardo será pequeño en general, pero la atenuación será considerable, ya que, en teoría, debiera ser infinita):

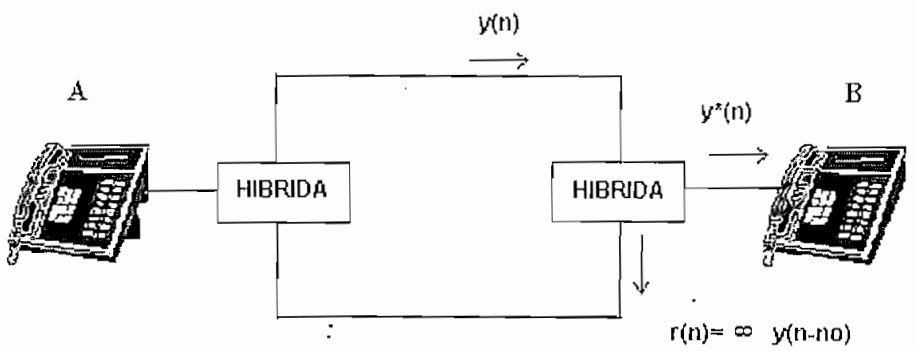



Figura 3.9 Esquema de conexión entre el oyente y el hablante

y*(n) es una versión de y(n) (hacia B no pasa y(n) tal cual, aunque sí pasa la mayor parte de y(n)). La señal r(n) ha de ser eliminada, para lo cual usaremos la propia señal y(n) y un algoritmo adaptivo (adaptivo porque no se conoce la señal que atraviesa la híbrida). Así, alimentando a un FIR adaptativo con la propia y(n) se obtendrá una estimación de r(n), $r_e(n)$, que restada a r(n) produciría una señal de eco nula hacia A, o, al menos, muy reducida.

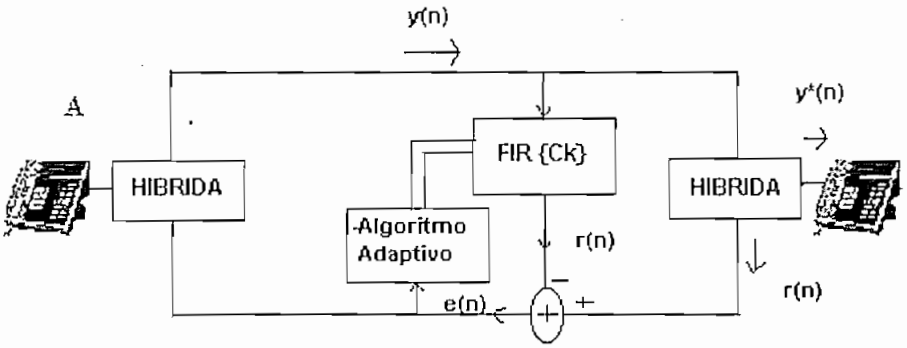



Figura 3.10 Localización del filtro adaptivo

Así, el error es e(n)=r(n)-$r_e$(n), y siguiendo el criterio del M.S.E. hay que minimizar:

$$E=\sum_{n=0}^{\infty}\left[r(n)-\sum_{k=0}^{M-1}h(k)y(n-k)\right]^2 \qquad (3.1)$$

Lo cual conduce a un sistema de M ecuaciones con M incógnitas que son los M coeficientes del filtro:

$$\sum_{n=0}^{\infty}h(k)r_{yy}(l-k)=r_{RY}(l) \quad \text{con l=0,1.....M-1} \qquad (3.2)$$

De hecho, en el fondo, aquí el FIR adaptivo está modelando la híbrida.

El problema que aquí se plantea es que , además de la señal y(n) que va de A hacia B, también puede existir señal de B hacia A, x(n), de modo que la híbrida cercana a B dejaría pasar hacia A la suma r(n)+x(n).

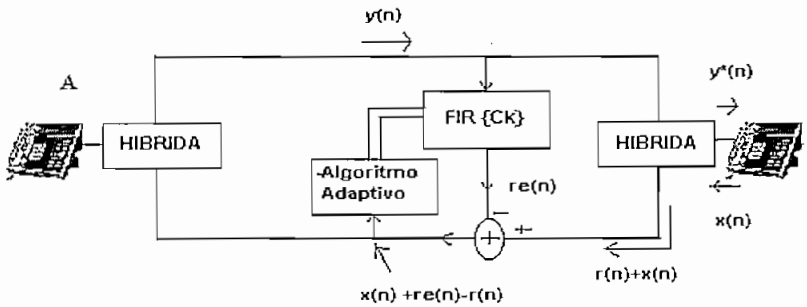



Figura 3.11 Localización del filtro adaptivo

Las ecuaciones que nos conducen a los M coeficientes del filtro en este caso son:

$$\sum_{k=0}^{M-1} h(k) r_{yy}(l-k) = r_{RY}(l) \qquad \text{con l=0,1...M-1} \tag{3.3}$$

Sólo que en la realidad lo que se tiene en (l) no es sólo r(n) sino también x(n), esto es , que se dispone de $r_{(R+X)Y}(l)$ y no de $r_{RY}(l)$. Si x(n) e y(n) son no correlacionadas, entonces $r_{(R+X)y}(l) \approx r_{RY}(l)$, y no hay ningún problema en la adaptación del FIR hacia r(n), pero si son correlacionadas, esto es, si tienen componentes espectrales comunes, no se cumple la aproximación anterior y eso hace que la adaptación de los coeficientes no sea la adecuada, por lo que , para mejorar el funcionamiento, en presencia de x(n), se detiene la adaptación (no se actualiza más, y el filtro permanece en el estado de adaptación que hubiese alcanzado antes de aparecer x(n) hasta que ésta señal desaparezca).

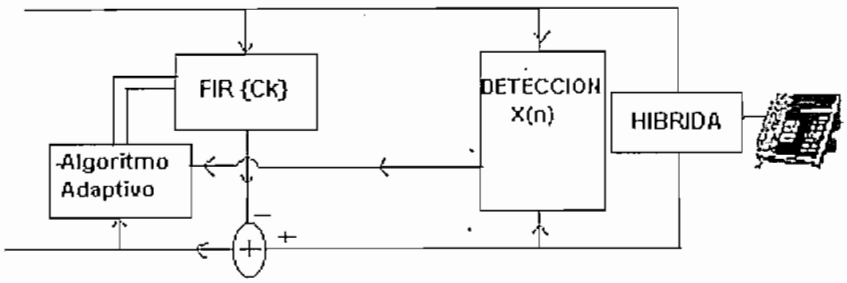



Figura 3.12 Filtro adaptivo con detección de X(n)

Para cancelar el eco en el otro sentido, esto es, el que se da en la híbrida cercana a A y vuelve hacia B, se coloca el cancelador en el extremo de A.

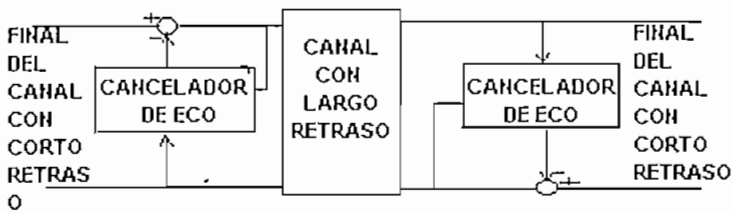


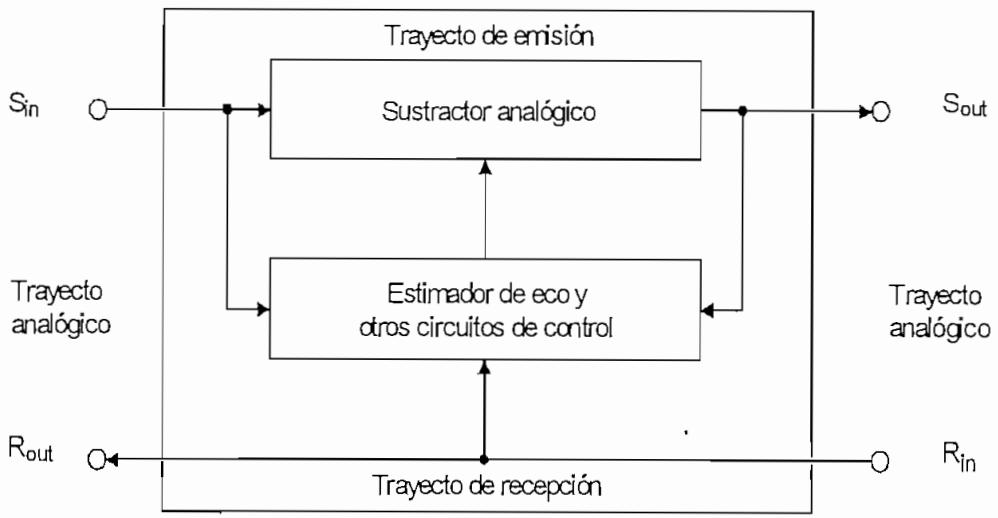

Figura 3.13 Cancelación en un canal con largo retraso

Podría utilizarse un único cancelador que realizara las dos tareas, pero eso conllevaría usar un filtro excesivamente largo, esto es, con demasiados coeficientes, y, a más coeficientes tenga el filtro, más difícil es adaptarlos. Hay que tener en cuenta que la longitud (el número de pesos) del filtro es proporcional a la distancia que recorre el eco, esto es, al retardo que sufre ( ya que con M coeficientes sólo puede adaptarse a un retardo de M*fs muestras, siendo fs la frecuencia de muestreo de la señal de entrada), y, así, a más lejos del punto de origen del eco se coloque el cancelador, se ha de adaptar a más retardo (más desfase), y así, más largo ha de ser el filtro y más difícil resulta adaptar todos sus coeficientes simultáneamente.

## 3.4 COMPENSADORES DE ECO [6]

### 3.4.1 CONSIDERACIONES GENERALES

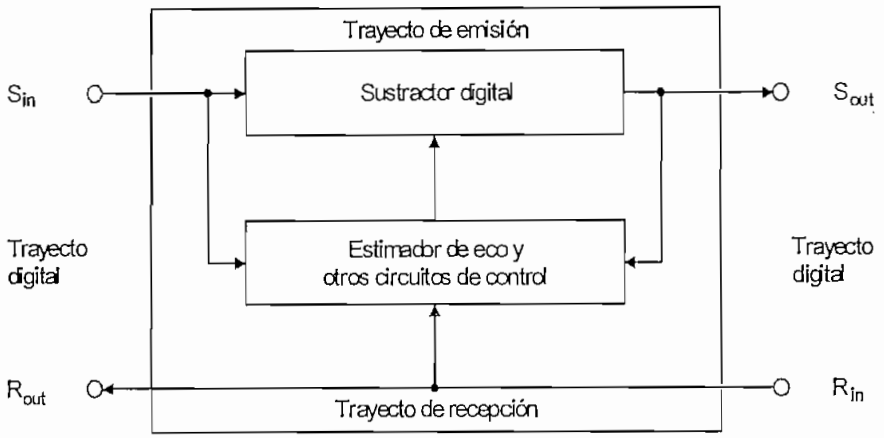
Los compensadores de eco son dispositivos activados por la voz, instalados en la parte a cuatro hilos de un circuito (que puede ser un trayecto de circuito individual o un trayecto que curse una señal multiplexada) y que tienen por función reducir el eco del circuito, para lo cual se sustrae de éste un valor estimado del eco. Pueden caracterizarse atendiendo al hecho de que el trayecto de transmisión o la sustracción del eco se efectúan por medios analógicos o digitales (véanse las Figuras 3.14 , 3.15 y 3.16).




$R_{in}$ Puerto de entrada recepción (*receive-in*)
$R_{out}$ Puerto de salida recepción (*receive-out*)
$S_{in}$ Puerto de entrada emisión (*send-in*)
$R_{out}$ Puerto de salida emisión (*send-out*)

Figura 3.14 Compensador de eco tipo A

[6] Recomendación UIT-T G.165,Características Generales de las Conexiones y Circuitos Telefónicos Internacionales, Compensadores de Eco




NOTA – Funcionalmente, la interfaz de un compensador de eco digital (DEC) (*digital echo canceller*) de tipo C es a 64 kbit/s. Sin embargo, se pueden combinar, por ejemplo, 24 ó 30 compensadores de eco digitales, que corresponden respectivamente a los niveles de la jerarquía digital primaria de 1544 kbit/s y 2048 kbit/s.

Figura 3.15 compensador de eco tipo C

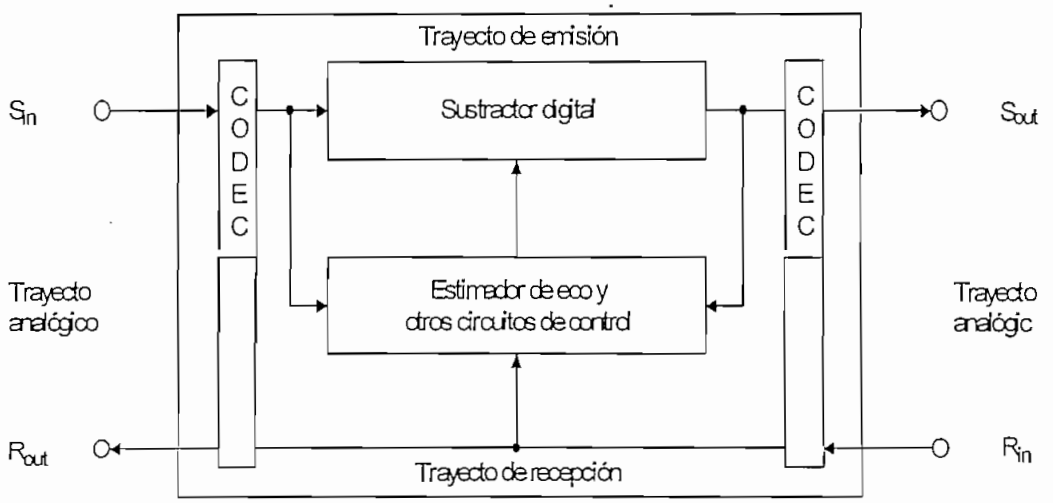



Figura 3.16 Compensador de Eco

## 3.4.2 DEFINICIONES RELATIVAS A LOS COMPENSADORES DE ECO

### 3.4.2.1 Compensador de eco

Dispositivo activado por la voz, instalado en la parte a cuatro hilos de un circuito y que tiene por función reducir el eco del extremo cercano presente en el trayecto emisión, para lo cual se sustrae un valor estimado de ese eco, del eco del extremo cercano (véase la Figura 3.17).

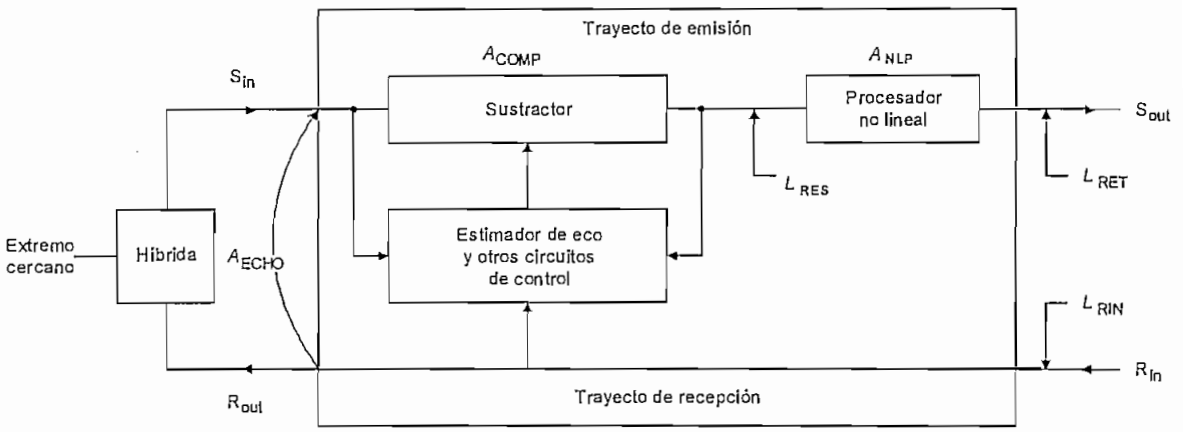



Figura 3.17 Compensador de eco

### 3.4.2.2 Atenuación del eco ($A_{ECHO}$)

Atenuación de una señal desde el puerto de salida recepción ($R_{out}$) hasta el puerto de entrada emisión ($S_{in}$) de un compensador de eco, debido a la pérdida de transmisión y en la híbrida, es decir, la atenuación en el trayecto del eco (eco cercano).

### 3.4.2.3 Retardo puro ($t_r$)

Retardo desde el puerto $R_{out}$ al puerto $S_{in}$ debido a los retardos intrínsecos de las facilidades de transmisión del trayecto de eco (eco cercano). En este caso se supone que el tiempo de tránsito a través de la híbrida es nulo.

#### 3.4.2.4 Retardo trayecto de eco (cercano) o retardo de extremo ($t_d$)

Suma del retardo puro ($t_r$) y el tiempo de dispersión. El tiempo de dispersión es el tiempo necesario para incluir los efectos de la limitación de banda, las reflexiones múltiples y el tránsito por la híbrida (ilustración en la Figura 3.18 más adelante). Cabe observar que esta definición supone un trayecto de eco único. Si hay varios trayectos de eco, el retardo global del trayecto de eco es el valor máximo de cada uno de los retardos de trayecto de eco. Como el tiempo de dispersión varía en función de las diversas redes nacionales, la capacidad de retardo de trayecto de eco del compensador viene dada por esta definición. Con una frecuencia de muestreo de 8 KHz y una versión FIR (respuesta de impulso limitado) de compensador de eco, el número de derivaciones del compensador de eco para un determinado valor de $t_d$ en milisegundos es igual a 8 veces $t_d$].

#### 3.4.2.5 Compensación ($A_{CANC}$)

Atenuación de la señal de eco al pasar por el trayecto-emisión de un compensador de eco. Esta definición excluye específicamente todo tratamiento no lineal de salida del compensador para proporcionar una atenuación mayor.

#### 3.4.2.6 Nivel de eco residual ($L_{RES}$)

Nivel de la señal de eco que subsiste en el puerto de salida-emisión de un compensador de eco en funcionamiento después de una compensación imperfecta del eco de circuito. Estará relacionado con la señal de entrada del lado recepción $N_{Rin}$ de acuerdo con la siguiente fórmula:

$$L_{RES} = N_{Rin} - A_{ECO} - A_{CANC}$$

No se incluyen eventuales tratamientos no lineales.

#### 3.4.2.7 Procesador no lineal (NLP)

Dispositivo con un umbral de supresión definido y en el que:

a) se suprimen las señales detectadas con un nivel inferior al umbral, y

b) se dejan pasar las señales detectadas con un nivel superior al umbral, aunque éstas pudieran ser distorsionadas

NOTAS

1 El funcionamiento concreto de un procesador no lineal depende del algoritmo de detección y de control utilizado.

2 Ejemplo de procesador no lineal es un recortador de señales analógicas, en el cual las señales de niveles inferiores a un umbral definido son forzadas a un cierto valor mínimo.

#### 3.4.2.8 Atenuación por procesamiento (o tratamiento) no lineal ($A_{NLP}$)

Atenuación adicional del nivel del eco residual mediante un procesador no lineal situado en el trayecto emisión de un compensador de eco.

NOTA – En un orden estricto, la atenuación causada por un proceso no lineal no puede expresarse en decibelios. No obstante, para facilitar la explicación y discusión del funcionamiento del compensador de eco, una utilización inteligente de $A_{NLP}$ resulta útil.

#### 3.4.2.9 Nivel del eco devuelto ($L_{RET}$)

Nivel de la señal en el puerto de salida-emisión de un compensador de eco en funcionamiento que volverá a la persona que habla. Se incluye la atenuación causada por un procesador no lineal, si está normalmente presente. $L_{RET}$ está relacionado con $N_{Rin}$ por la fórmula:

$$L_{RET} = L_{Rin} - (A_{ECHO}\ A_{CANC}\ A_{NLP})$$

En ausencia de un tratamiento no lineal, obsérvese que $L_{RES} = L_{RET}$.

#### 3.4.2.10 Atenuación combinada ($A_{COM}$)

La suma de la atenuación del eco, la atenuación por compensación y la atenuación por tratamiento no lineal (si existiera). Esta atenuación permite establecer la siguiente relación entre $L_{RIn}$ y $L_{RET}$.

$L_{RET} = L_{RIn} - A_{COM}$, donde $A_{COM} = A_{ECHO} + A_{CANC} + A_{NLP}$.

#### 3.4.2.11 Convergencia

Proceso de elaboración de un modelo del trayecto de eco que se utilizará en el estimador de eco para obtener la estimación del eco de circuito.

#### 3.4.2.12 Tiempo de convergencia

Para un determinado trayecto de eco, el intervalo que transcurre entre el instante en que una señal de prueba definida se aplica al puerto de entrada recepción de un compensador de eco con la respuesta impulsional estimada del trayecto de eco inicialmente puesta a cero, y el instante en que el nivel de eco devuelto en el puerto de salida emisión alcanza un nivel determinado.

#### 3.4.2.13 Tiempo de fuga

Intervalo entre el instante en que deja de aplicarse una señal de prueba al puerto de entrada recepción de un compensador de eco que ha alcanzado la plena convergencia y el instante en que el modelo de trayecto de eco en el compensador de eco cambia, de modo que, cuando se aplica una señal de prueba al puerto de entrada recepción con los circuitos de convergencia desactivados, el eco devuelto alcanza un nivel determinado.

Esta definición se refiere a compensadores de eco que emplean, por ejemplo, integradores con fugas en los circuitos de convergencia.

## 3.4.3 CARACTERÍSTICAS DE LOS COMPENSADORES DE ECO

### 3.4.3.1 Consideraciones generales

Se supone que los compensadores de eco son " semicompensadores de eco ", es decir, que la compensación sólo se produce en el trayecto emisión como consecuencia de señales presentes en el trayecto recepción.

### 3.4.3.2 Finalidad, funcionamiento y campo de aplicación

En un circuito telefónico cualquiera a dos hilos, o mixto a dos hilos y cuatro hilos, el eco es causado por desequilibrios de impedancia. Puede emplearse un compensador de eco para reducir el eco a niveles admisibles.

El eco presente en el puerto de entrada emisión de un compensador de eco es una reproducción deformada y retardada de la señal vocal procedente del extremo distante, es decir, el eco es la señal vocal entrante modificada por el trayecto de eco. El trayecto de eco se describe corrientemente por su respuesta impulsional (véase la Figura 3.18). Esta respuesta de un trayecto de eco típico muestra la señal afectada por un retardo $t_r$, debido a los retardos propios de los medios de transmisión que forman el trayecto de eco, y una dispersión debida a la limitación de la banda de frecuencias y a las reflexiones múltiples. Su suma representa el retardo del trayecto de eco, $t_d$. Los valores del retardo y la dispersión variarán según las propiedades de los trayectos de eco, es decir, pueden ser diferentes en las distintas redes nacionales. (Obsérvese que en el trayecto de eco puede haber más de una fuente de eco; existen numerosas configuraciones de red en las cuales se producen múltiples conversiones de dos hilos a cuatro hilos en el trayecto final de un compensador de eco).

Se supone que los trayectos de eco son esencialmente lineales y no varían de forma continua en función del tiempo. La calidad de funcionamiento del compensador de eco depende esencialmente de la linealidad del trayecto de eco entre Rout y Sin (véase la Figura 3.17).

Un compensador de eco deberá poder sintetizar una reproducción de la respuesta impulsiònal del trayecto de eco. Muchos compensadores de eco modelan el trayecto de eco mediante datos tomados por muestreo a la velocidad de Nyquist (8000 Hz). Para que este compensador de eco pueda funcionar debidamente deberá tener una memoria con capacidad suficiente para almacenar el número requerido de muestras. Se han efectuado con éxito demostraciones de compensadores de eco con capacidades de memoria de 8 ms a 64 ms. El máximo retardo de eco, $t_d$ admisible en la red en que se emplea el compensador de eco determinará la capacidad de memoria necesaria. La cuestión típica es la siguiente: si el número de elementos de almacenamiento es insuficiente, no será posible obtener una síntesis correcta de todos los trayectos de eco; si, por el contrario, el número de elementos de almacenamiento es demasiado elevado, se producirá un ruido adicional no deseado, como consecuencia de los elementos no utilizados que, debido al ruido introducido por el proceso de estimación, generalmente no tienen valor cero.

Debe señalarse que un compensador de eco introduce un trayecto de eco paralelo adicional. Si la diferencia entre las respuestas impulsionales del modelo de trayecto de eco y del trayecto de eco propiamente dicho alcanza cierta magnitud, el eco total devuelto puede ser mayor que el debido al trayecto de eco solamente.

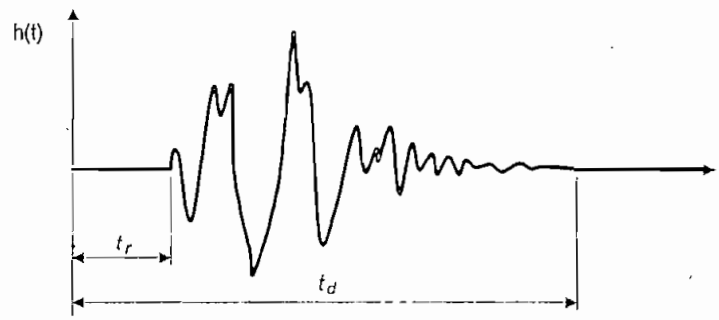



Figura 3.18 Ejemplo de respuesta impulsoidal de un trayecto de eco

Cuando el abonado local comienza a hablar al tiempo que recibe aún señales vocales («habla simultánea»), el compensador de eco puede interpretar la señal en emisión como un nuevo eco y tratar de adaptarse al mismo. Esto puede degradar considerablemente la calidad subjetiva de la conexión. No sólo se reduce la compensación del eco, sino que puede producirse la distorsión de las señales vocales en habla simultánea al tratar el compensador de eco de adaptarse dinámicamente.

Por tanto, los compensadores de eco habrán de satisfacer los siguientes requisitos fundamentales:

1) Tener una convergencia rápida;

2) Tener un bajo nivel de eco devuelto en monólogo;

3) Tener una divergencia pequeña en habla simultánea.

Cuando los compensadores de eco están situados en el lado del abonado del equipo de señalización internacional, los tonos de señalización no pasan a través de los compensadores, por lo que no es necesaria ninguna acción especial. Cuando los compensadores están en el lado internacional del equipo de señalización, son normalmente neutralizados por el conmutador durante los intervalos activos de intercambio de señalización a fin de evitar la distorsión de los tonos de señalización por el compensador de eco. Cuando aparecen simultáneamente tonos de señalización en los puertos de recepción y emisión del compensador (habla simultánea), la señal de recepción será procesada a través del modelo de trayecto de eco contenido en el compensador. La estimación de la señal producida por el compensador puede distorsionar suficientemente la señal del lado emisión, por lo que no será correctamente reconocida por la unidad de recepción de señalización.

#### 3.4.3.3 Activación y neutralización externas

Conviene que los compensadores de eco de tipo A y D posean una interfaz que pueda ser activada o neutralizada por una puesta a tierra exterior, derivada del circuito interurbano y controlada por la central o centro de conmutación internacional (ISC) correspondiente. Este dispositivo debe permitir o impedir, según el caso, el funcionamiento normal del compensador de eco. Ciertos compensadores de eco de tipo C pueden ser neutralizados directamente por una señal digital. Los compensadores de eco de tipo C deben mantener la integridad de la secuencia de bits a 64 kbit/s (es decir que, si la conversión de ley A a ley μ está integrada, será también neutralizada) en el estado externamente desactivado.

En el Anexo C se indica las pruebas y requisitos de funcionamiento con señales de entrada aplicadas a las trayectos de emisión y recepción.

## 3.5 ANÁLISIS DE LA PRECISIÓN DE LOS COEFICIENTES DEL FILTRO EN EL DESEMPEÑO DEL ALGORITMO LMS PARA CANCELADORES DE ECO G.165/G.168 [7]

En la implementación digital del algoritmo LMS, los coeficientes del filtro y el nivel de la señal son cuantificados , por lo que se introduce un error de cuantificación. El efecto del error de cuantificación en los coeficientes del filtro es de un interés particular, porque este determina el procesador que va a ser utilizado en la implementación para satisfacer los requerimientos de la aplicación. La discusión se enfoca en el efecto de la precisión finita de los coeficientes del filtro en el desempeño del algoritmo LMS en la cancelación de eco usando el procesador digital TMS320C6201.

---

[7] Zhaohong Zhang, Gunter Schmer, Analysis of Filter Coefficient Precision on LMS Algorithm Performance for G.165/g.168 Echo Cancellation, Aplication Report Spra561 – February 2000, Texas Instruments

### 3.5.1 ANÁLISIS TEÓRICO

En este análisis, es usado un filtro FIR transversal, que es usado para predecir, la forma del eco, el eco residual es calculado como:

$$e(n) = d(n) - \frac{1}{A}\sum_{k=0}^{M-1} H_K(n)x(n-k) \tag{3.4}$$

donde *e(n)* es el valor del residuo al tiempo *n*, *d(n)* es el valor del eco al tiempo *n*, $H_k(n)$ es el k - ésimo coeficiente del filtro al tiempo *n*, y *x(n-k)* es el valor de la señal con eco al tiempo *n-k*. *M* es la longitud del filtro, el cual es determinado por el tamaño del retraso del eco. *A* es un factor normalizado tal que la salida del filtro tenga la misma precisión de *d(n)*. El filtro es actualizado por el algoritmo LMS como:

$$H_K(n+1) = H_K(n) + \mu.e(n)x(n-k) \tag{3.5}$$

donde $\mu \geq 0$ e el tamaño del paso de la adaptación.

Primero hay dos efectos debido a la cuantificación del error en los coeficientes del filtro. El primer efecto es la degradación de la exactitud en el calculo de la predicción del eco debido a la cuantificación del error en los coeficientes del filtro. Se asume el modelo siguiente para los coeficientes del filtro $H_k$:

$$H_K = H_{OK} + e_K \tag{3.6}$$

donde $e_K$ es el error. La salida del filtro es calculada como

$$y = \frac{1}{A}\sum_{K=0}^{M-1} H_K x_K = \frac{1}{A}\sum_{K=0}^{M-1} H_{OK} x_K + \frac{1}{A}\sum_{K=0}^{M-1} e_K x_K \tag{3.7}$$

Para evaluar el termino del error

$$\Delta = \frac{1}{A}\sum_{K=0}^{M-1} e_K x_K \tag{3.8}$$

donde $e_K$ y $x_K$ son asumidas como variables independientes, la varianza del término del error es dada por:

$$\sigma^2{}_\Delta = \frac{1}{A} M \sigma^2{}_e \sigma^2{}_x \quad \text{y} \quad \sigma_\Delta = \frac{1}{A}\sqrt{M}\,\sigma_e \sigma_x \tag{3.9}$$

Para los coeficientes del filtro con una precisión de 16 bits y para asegurar que la salida del filtro tenga la misma precisión que *d(n)* se utiliza un valor de $A$ = *65536* . la especificación G.165/g.168, requiere que el eco residual, este debajo de 30 dB con referencia a la señal con eco.

La calidad de la cancelación es garantizada con la siguiente relación:

$$\sigma_\Delta << \frac{\sigma_x}{31.6} \tag{3.10}$$

Como un ejemplo, se asume que $e_K$ es uniformemente distribuido en {-0.5,0.5} y $x_K$ es tratado como una señal PCM uniformemente distribuida en {-4096,4096}, para un canal que da un eco con un retraso de 48 ms ($M$=384), se tiene:

$$\sigma_\Delta = \frac{1}{65536}\sqrt{384\frac{1}{12}\frac{4096^2}{3}} = 0.204 \tag{3.11}$$

Para un canal que da un eco con un retraso de 64 ms ($M$=512), se tiene:

$$\sigma_\Delta = \frac{1}{65536}\sqrt{512\frac{1}{12}\frac{4096^2}{3}} = 0.250 \tag{3.12}$$

se observa que $\sigma_\Delta$ es menor que 1, el error introducido con una cuantizacion de 16 bits para los coeficientes del filtro es despreciable en el calculo de la salida del filtro.

Un segundo efecto en la cuantificación de los coeficientes del filtro es la terminación temprana del algoritmo, el algoritmo cesa de ajustar los coeficientes, cuando el término de corrección $\mu\ e(n)\ x(n-k)$ en la ecuación 2 es menor en magnitud que la mitad del intervalo de la cuantificación de los coeficientes del filtro. El rendimiento se degrada porque la adaptación se termina por un efecto de cuantificación.

El algoritmo LMS continua la adaptación si:

$$\left|\mu e(n)x(n-k)\right| \geq 2^{-B-1} \tag{3.13}$$

donde B es el número de bits utilizados para representar los coeficientes del filtro. Esta condición puede ser aproximada reemplazando la magnitud por su valor RMS, eso es:

$$\mu \cdot \sigma_e \cdot \sigma_x \geq 2^{-B-1} \tag{3.14}$$

El valor máximo permisible para el paso de la adaptación es:

$$\mu_{\max} = \frac{1}{M\sigma_x^{\,2}} \tag{3.15}$$

es común tomar como valor del paso la mitad del valor máximo permisible, entonces:

$$\frac{1}{2M\dfrac{\sigma_x}{\sigma e}} \geq 2^{-B-1} \tag{3.16}$$

tenemos,

$$B \geq 3.322(\log_{10} M + \log_{10}(\frac{\sigma_x}{\sigma_e})) \qquad (3.17)$$

La relación anterior indica que el mínimo número de bits requerido para representar los coeficientes del filtro es proporcional al logaritmo de la longitud del filtro y la relación $\sigma_x / \sigma_e$.

Para estar dentro de las especificaciones ITU G.165 y G.168. el cancelador de eco debe lograr por lo menos una cancelación de 30 dB, $(\log_{10}(\sigma x/\sigma_e)) \geq 1.5$, cuando converge el filtro. Para un canal con un retraso del eco de 48 ms, M = 384. El mínimo valor de B es 13.6.

Para un canal con un retraso del eco de 64 ms, M = 512. el mínimo valor de B es 14.

Por lo anteriormente dicho, los coeficientes del filtro con una precisión de 16 bits, es adecuada para estas aplicaciones.

Varios casos de la simulación del algoritmo LMS, se pone a consideración. En la figura 3.19 y en la figura 3.20 se ilustra los resultados de la simulación para 32 ms y 48 ms de retardo del eco, con varios niveles de señal con eco. El nivel de la señal cambia desde 0 dB (0 dB = 4096) hasta –24 dB en pasos de 6 dB , puede verse que el filtro para 32 ms de retardo del eco converge después de alrededor de 200 ms, mientras el filtro para 48 ms de retardo del eco, converge después de 300 ms. Esto se debe a que el paso de la adaptación es pequeño para el filtro.

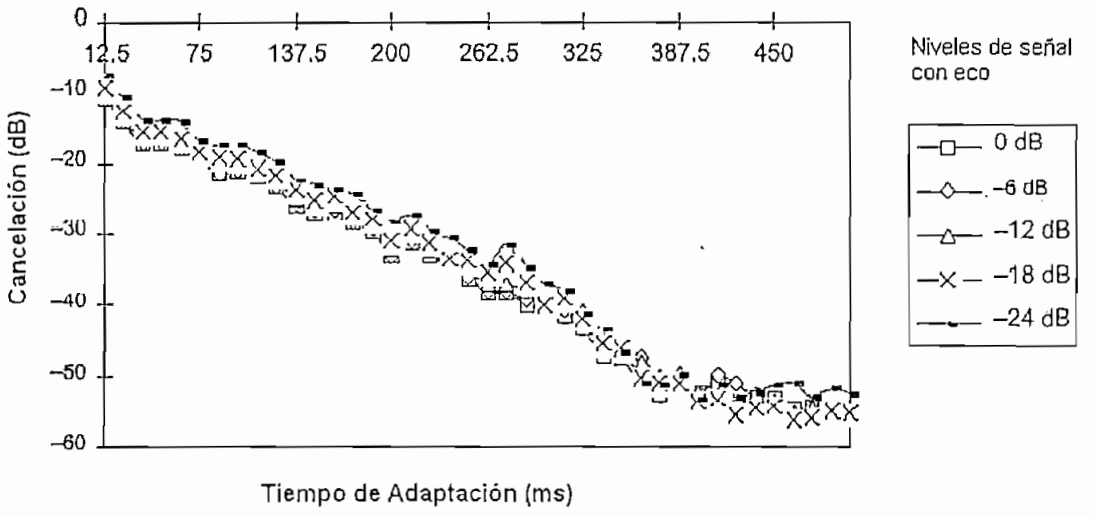



Figura 3.19 Cancelación de eco con un retardo de 32 ms (16 bits)

Este caso es comparado, con una implementación de 32 bits . En la figura 3.21, figura 3.22 y figura 3.23 para retraso de eco de 32 ms, 48 ms, y 64 ms respectivamente. El nivel de la señal de entrada al filtro es escogida en 0 dB. El paso en la adaptación es el mismo para los dos casos. En la figura 3.24 y figura 3.25 , el nivel de la señal de entrada es puesta en –30 dB para ilustrar la características del filtro con niveles pequeños de señal. Se observa que no discrepa la velocidad de convergencia. Esto verifica la teoría de que el número de bits usados para representar los coeficientes del filtro no afectan la velocidad de convergencia , pero si en el nivel de convergencia.

Esto muestra que la velocidad de convergencia depende de tres factores:

- el tamaño del paso
- la longitud del filtro
- la naturaleza de la señal de entrada.

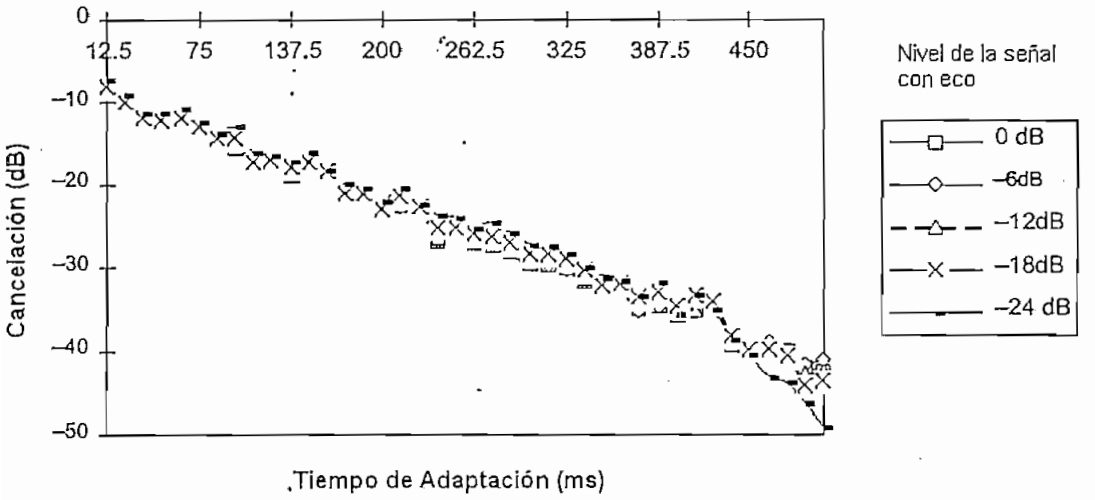



Figura 3.20 Cancelación de eco con un retardo de 48 ms (16 bits)

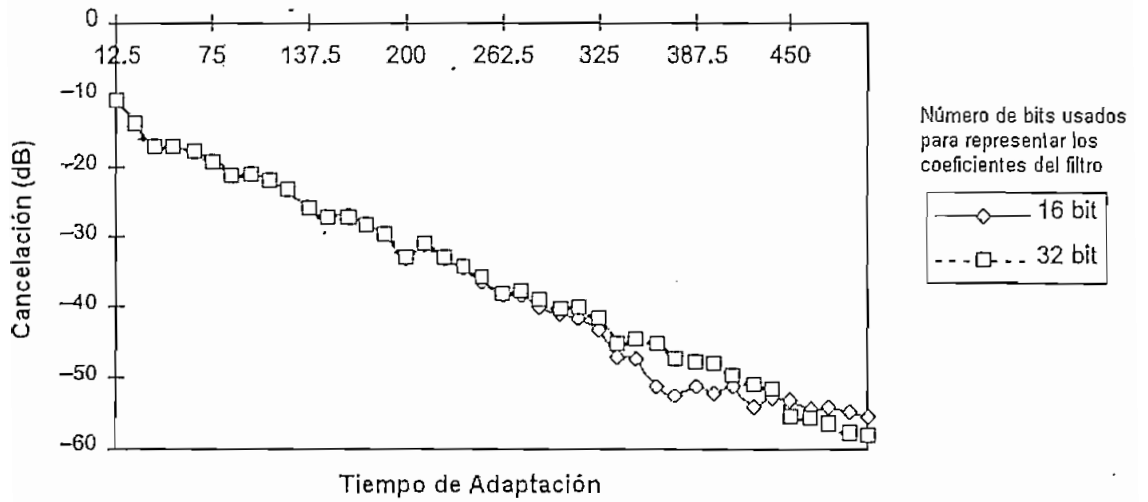



Figura 3.21 Cancelación de eco con un retardo de 32 ms (entrada 0dB)

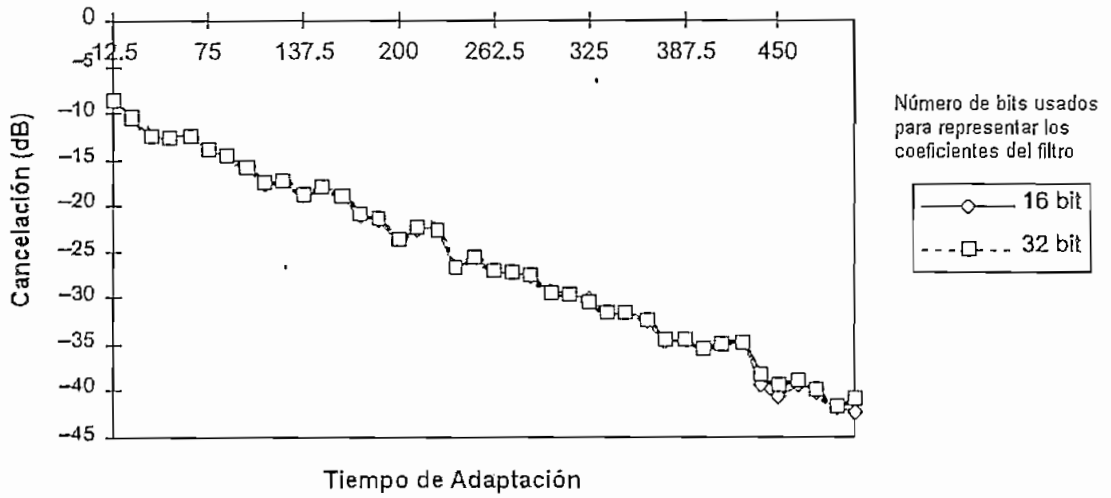



Figura 3.22 Cancelación de eco con un retardo de 48 ms (entrada 0dB)

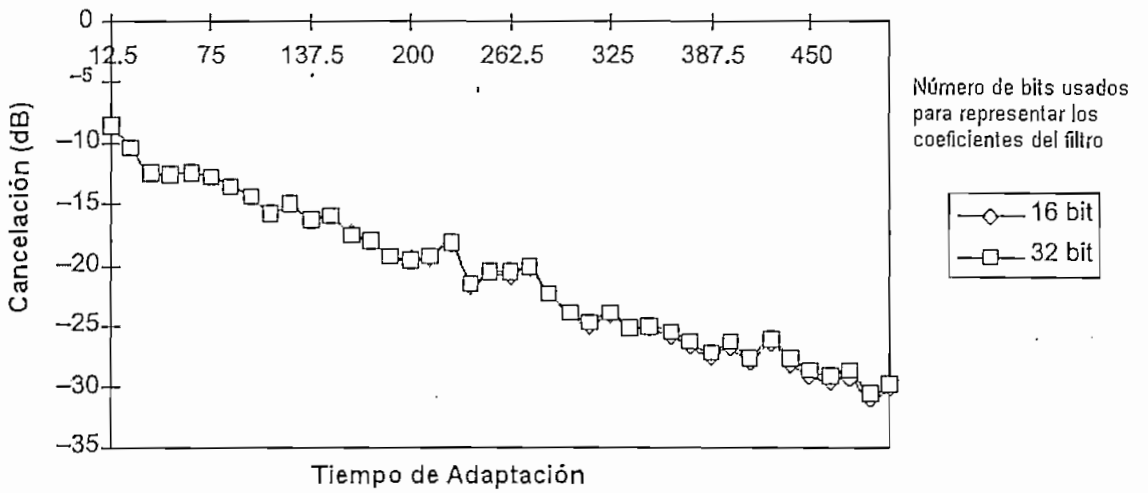



Figura 3.23 Cancelación de eco con un retardo de 64 ms (entrada 0dB)

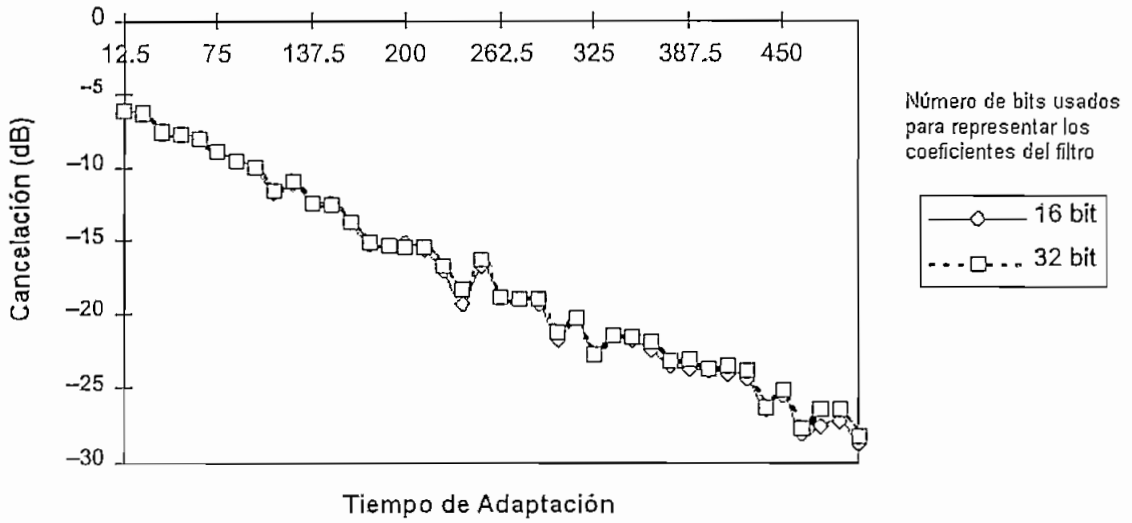



Figura 3.24 Cancelación de eco con un retardo de 48 ms (entrada -30dB)

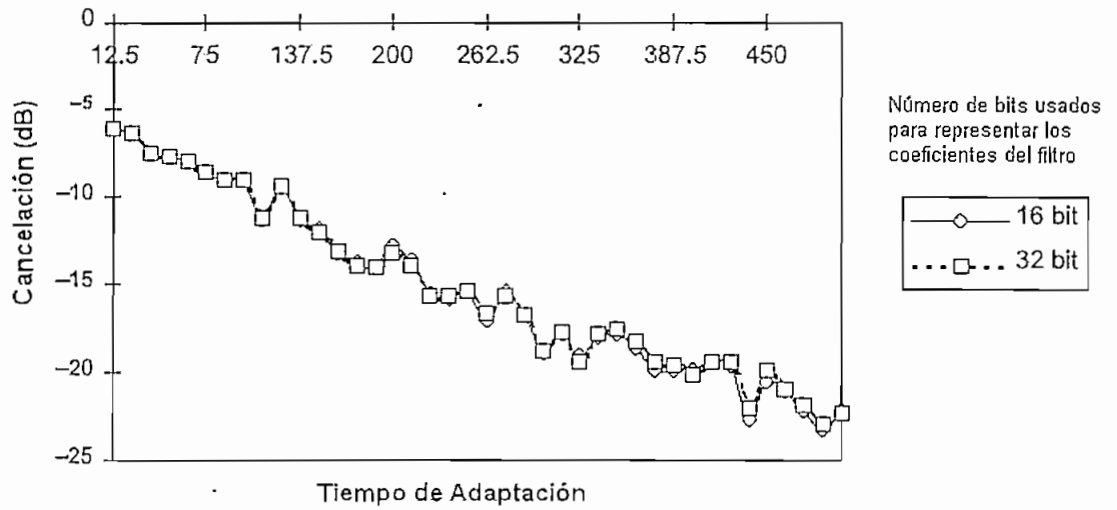



Figura 3.25 Cancelación de eco con un retardo de 64 ms (entrada -30dB)

### 3.5.2 ESTIMACIÓN DE LA POTENCIA DE UNA SEÑAL [8]

Las operaciones básicas para estimar la potencia de la una señal son multiplicación y suma, se utiliza un filtro de respuesta impulsiva infinita (IIR). IIR porque este filtro es recursivo, cada salida del filtro depende de la salida anterior, es decir se utiliza un integrador con un factor de olvido 1-α.

La siguiente ecuación , es una ecuación genérica para la estimación de la potencia:

$$P(n) = L^2(n) \tag{3.18}$$

donde: L(n) es la magnitud estimada de la señal de entrada (señal con eco) en instante n.

Estas ecuaciones, muestran dos formas para estimar la potencia, dependiendo de que se tenga como datos:

$$L(n) = (1-\alpha)\cdot L(n-1) + \alpha \cdot |X(n)| \tag{3.19}$$

$$P(n) = (1-\alpha)\cdot P(n-1) + \alpha \cdot X^2(n) \tag{3.20}$$

donde

α=1/32 (ventana muy pequeña para estimar la potencia, 4 ms)

α=1/128 (ventana pequeña para estimar la potencia, 16ms)

α=1/16384 (ventana larga para estimar la potencia, 2048 ms)

X(n)=señal de entrada

1-α = factor de olvido

---

[8] Jelena Nikolic, Implementing a Line Echo Canceller Using The Block Update and NLMS Algorithms on the TMS320C54x DSP, Associate Technical Staff, DSP Applications, SC Group Technical Marketing, Texas Instruments, April 1997

## 3.6 SIMULACIÓN DEL ALGORITMO LMS APLICADO A LA CANCELACIÓN DEL ECO EN CANALES TELEFONICOS

Para la realización de la simulación es necesario conocer la repuesta impulsiva del canal telefónico, ya que el filtro FIR modela la híbrida en las centrales telefónicas, y con el modelo construido, se puede encontrar un error el cual sirve para actualizar los coeficientes del filtro. Entonces la construcción de la respuesta impulsiva de un canal con eco es de la siguiente manera:

```
ch1=[1.3,1.1,.7,.2,-.2,-.25,-.2,-.5,-.07,.08,.1,.4,.09,.04,0,-.04,-.02]
ch2=[1,.8,.6,.45,.2,.1,.07,-.08,-.1,-.4,-.09,.04,0,-.04,-.02];
ch=[zeros(1,3*(N-32)/4),fliplr(ch1),ch2,zeros(1,(N-32)/4)];
```

El último vector obtenido da un vector de 128 puntos, con un eco retardado por 75 muestras.

```
impul = ch'/(2*norm(ch));
```

Esta última modificación da una inserción al canal de –6 dB de pérdida. El resultado final es el siguiente:

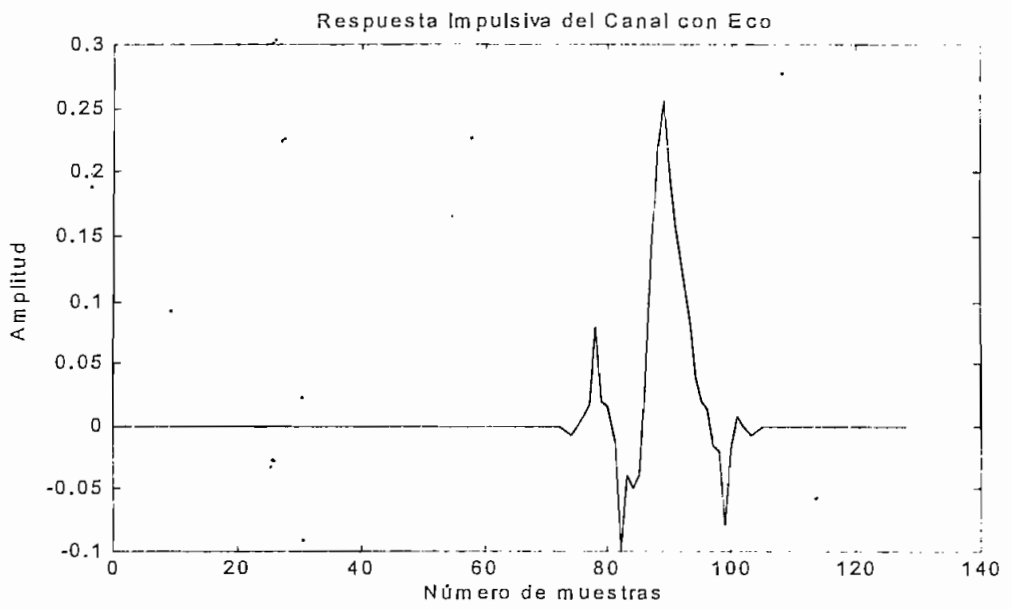



Figura 3.26 Respuesta impulsiva de un canal telefónico

Luego se necesita la señal telefónica digitalizada, la cual es la siguiente:

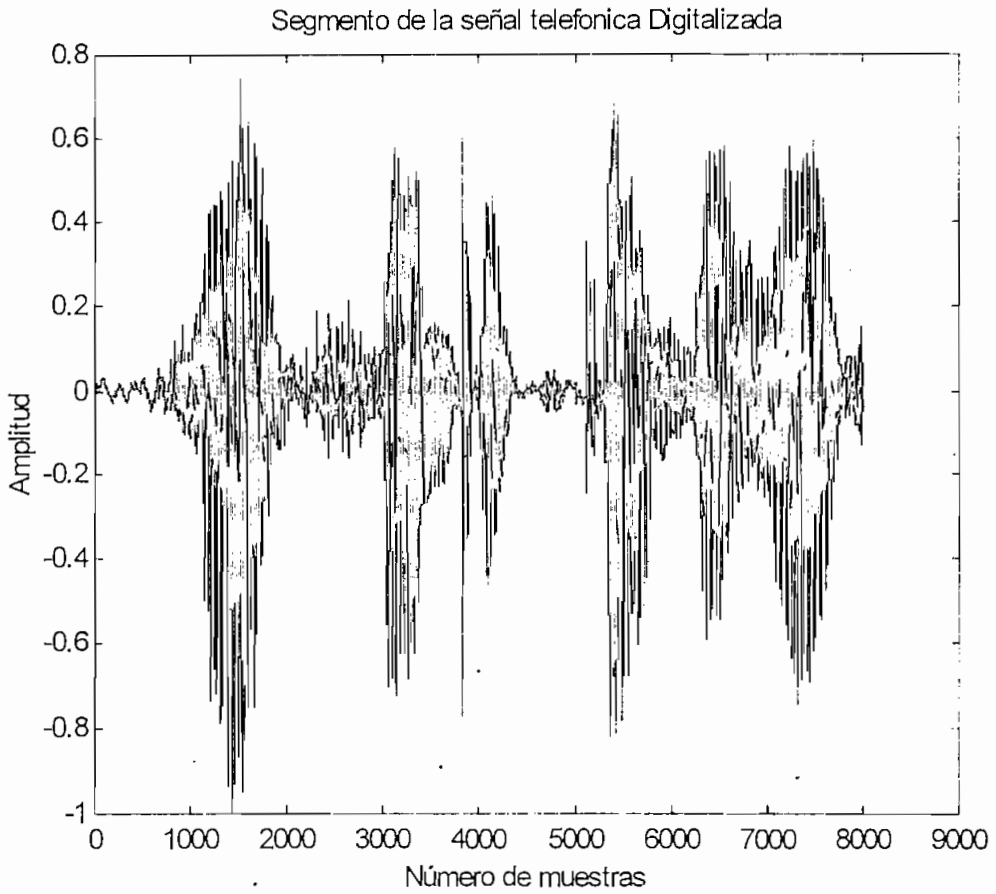
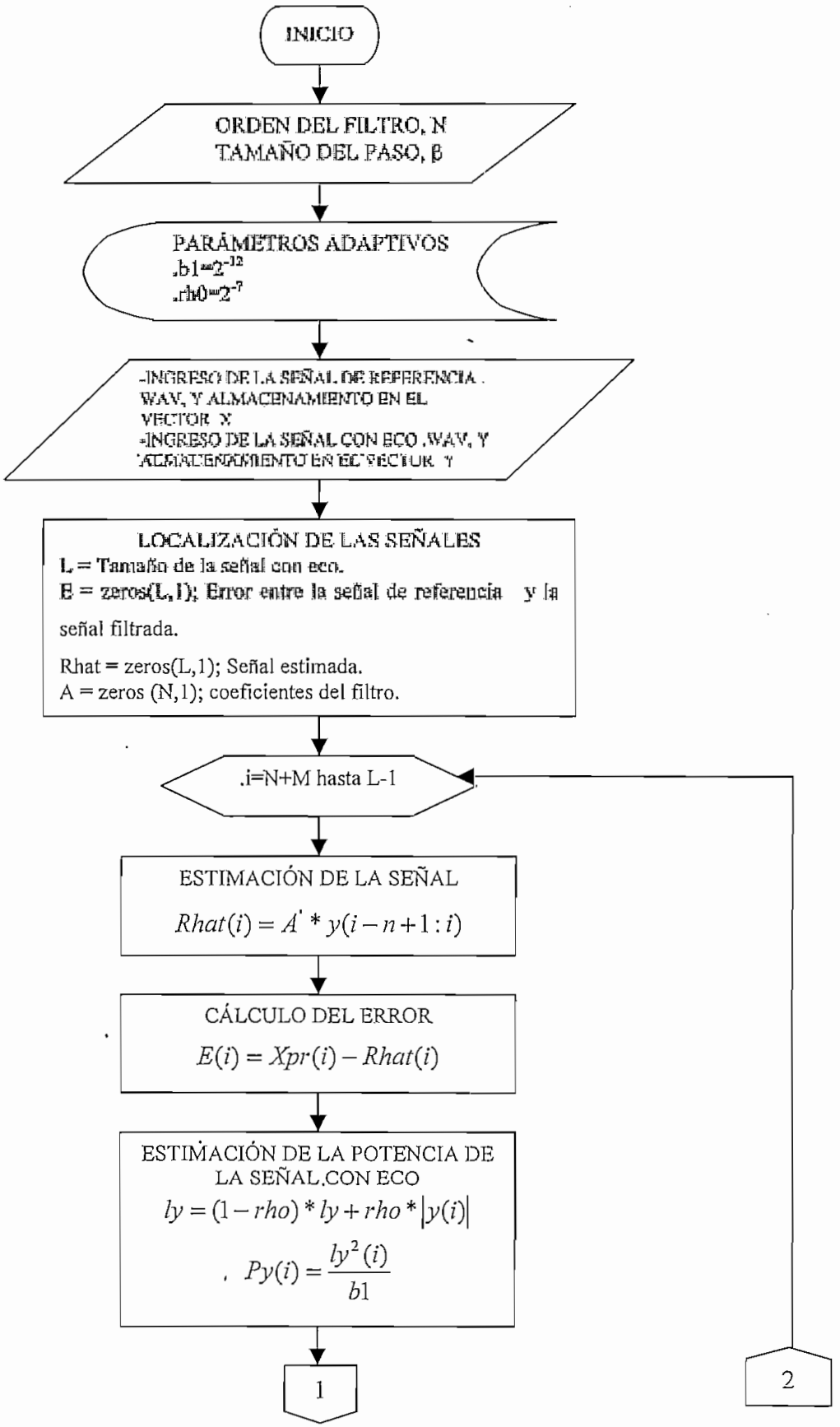



Figura 3.27 segmento de una señal telefónica digitalizada

Con las dos señales anteriores, se procede a utilizar el filtrado adaptivo, cuyo programa fuente se encuentra en le Anexo E. En la figura 3.28 se muestra un diagrama de flujo del filtrado LMS.

Para tener la señal original, como ya se tiene la respuesta impulsiva del sistema, se realiza una convolución con la señal telefónica digitalizada, como se sabe esta es la señal con eco.

Luego de dimensionar adecuadamente los vectores con los cuales se trabaja, poniendo valores adecuados a los parámetros adaptivos y variables como la longitud del cancelador, se obtiene los siguientes resultados.


INICIO
ORDEN DEL FILTRO, N
TAMAÑO DEL PASO, β
PARÁMETROS ADAPTIVOS
.b1=2^-12
.rh0=2^-7
-INGRESO DE LA SEÑAL DE REFERENCIA .WAV, Y ALMACENAMIENTO EN EL VECTOR X
-INGRESO DE LA SEÑAL CON ECO .WAV, Y ALMACENAMIENTO EN EL VECTOR Y
LOCALIZACIÓN DE LAS SEÑALES
L = Tamaño de la señal con eco.
E = zeros(L,1); Error entre la señal de referencia y la señal filtrada.
Rhat = zeros(L,1); Señal estimada.
A = zeros (N,1); coeficientes del filtro.
.i=N+M hasta L-1
ESTIMACIÓN DE LA SEÑAL
Rhat(i) = A' * y(i-n+1:i)
CÁLCULO DEL ERROR
E(i) = Xpr(i) - Rhat(i)
ESTIMACIÓN DE LA POTENCIA DE LA SEÑAL.CON ECO
ly = (1-rho) * ly + rho * |y(i)|
, Py(i) = ly^2(i) / b1
1
2

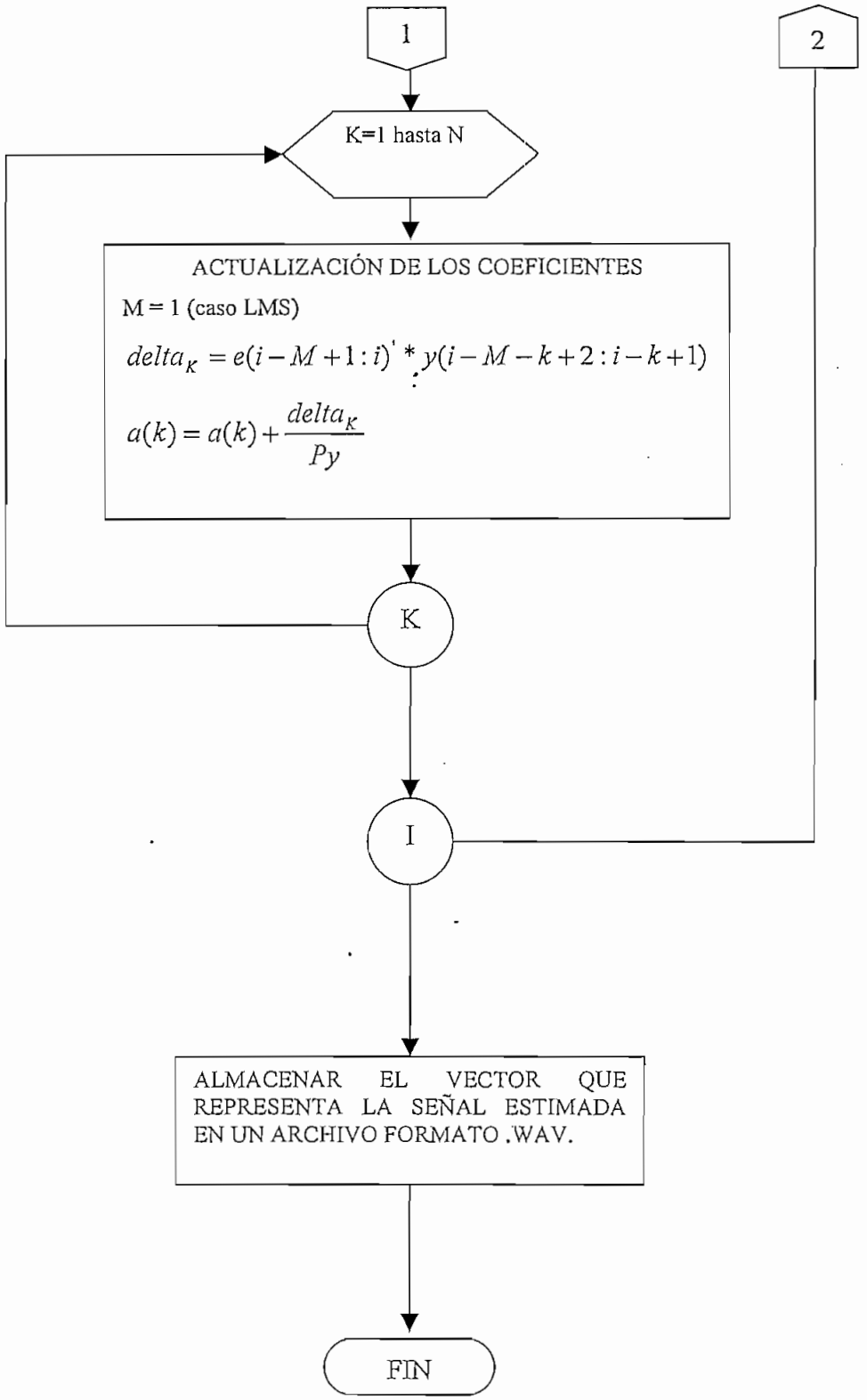



Figura 3.28 Diagrama de flujo del filtrado LMS

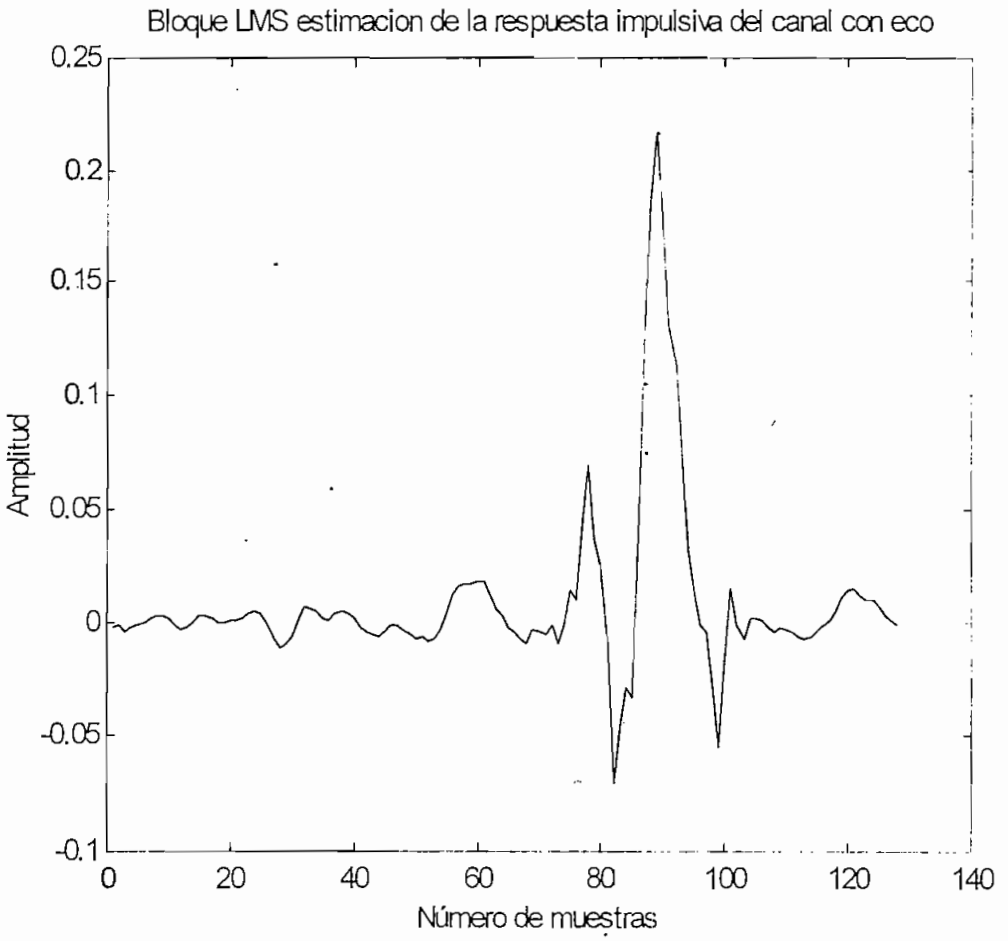



Figura 3.29 Respuesta impulsiva estimada de un canal telefónico que produce eco

Como se observa la estimación del bloque LMS es similar a la respuesta impulsiva construida anteriormente, no es idéntica porque el modelo construido es ideal.

En la figura 3.30 se presenta la señal estimada, es la señal que da el filtro adaptivo, la cual tiene un eco residual, que está dentro de los niveles permitidos para los canceladores de eco según la normas G.165 y G.168.

El nivel de potencia que tiene el eco residual, se lo presenta en la figura 3.31.

En la figura 3.32, se presenta una curva de aprendizaje, la cual muestra el error que se tiene al final de proceso versus el número de interacciones realizadas.

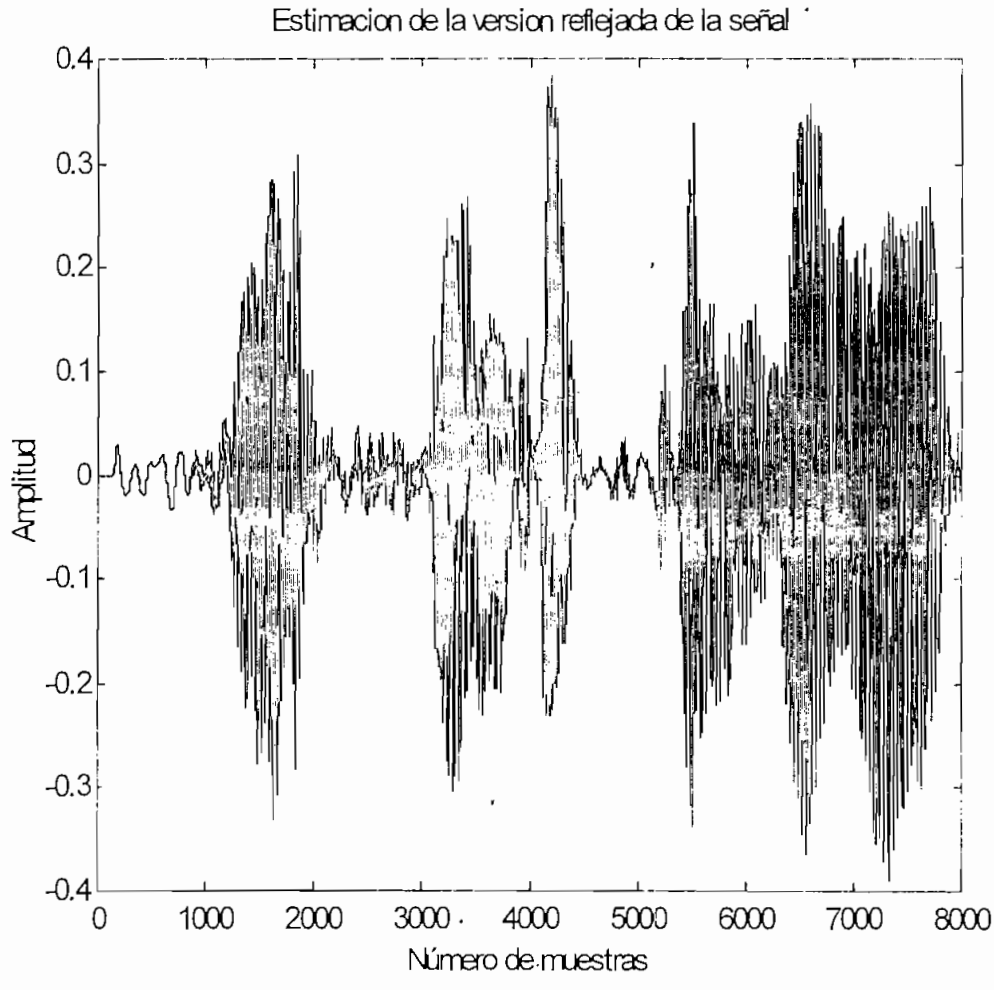



Figura 3.30 Estimación de la señal por el filtro adaptivo.

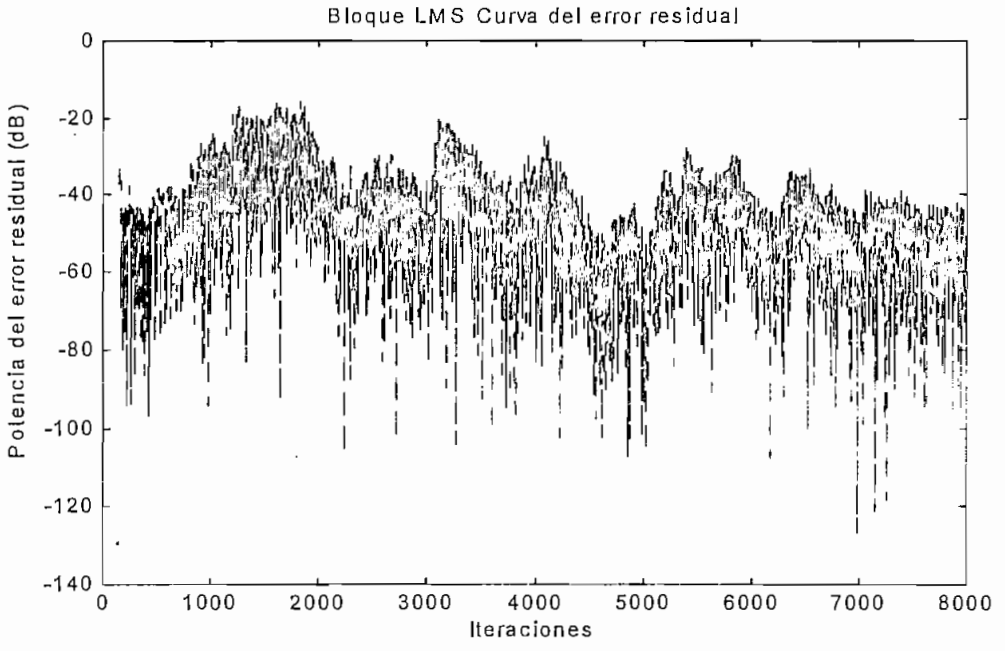



Figura 3.31 Potencia del eco residual.

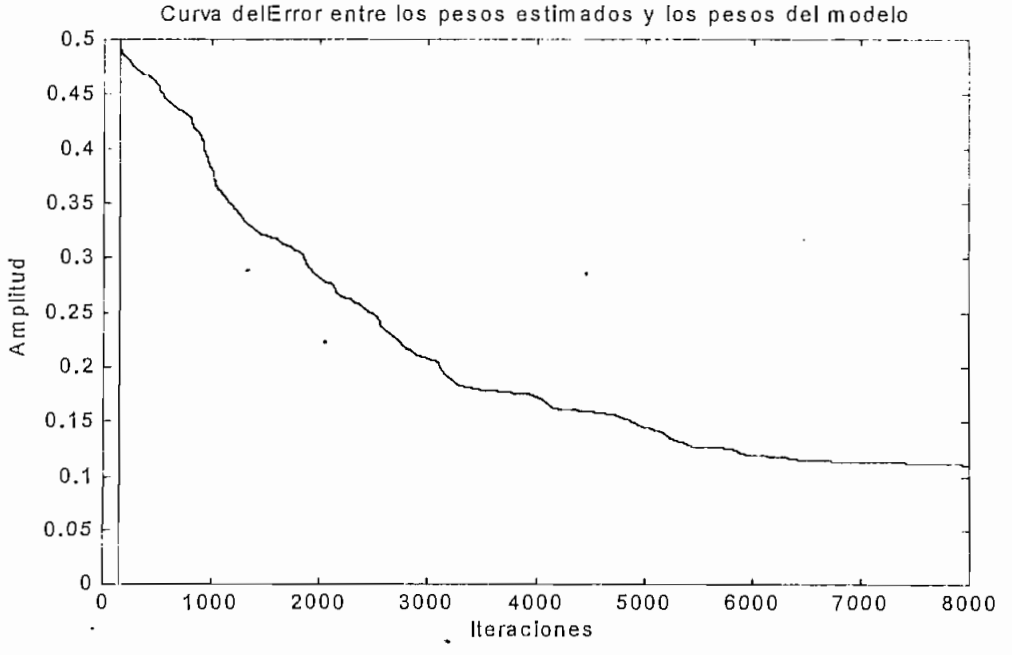



Figura 3.32 Curva de aprendizaje.

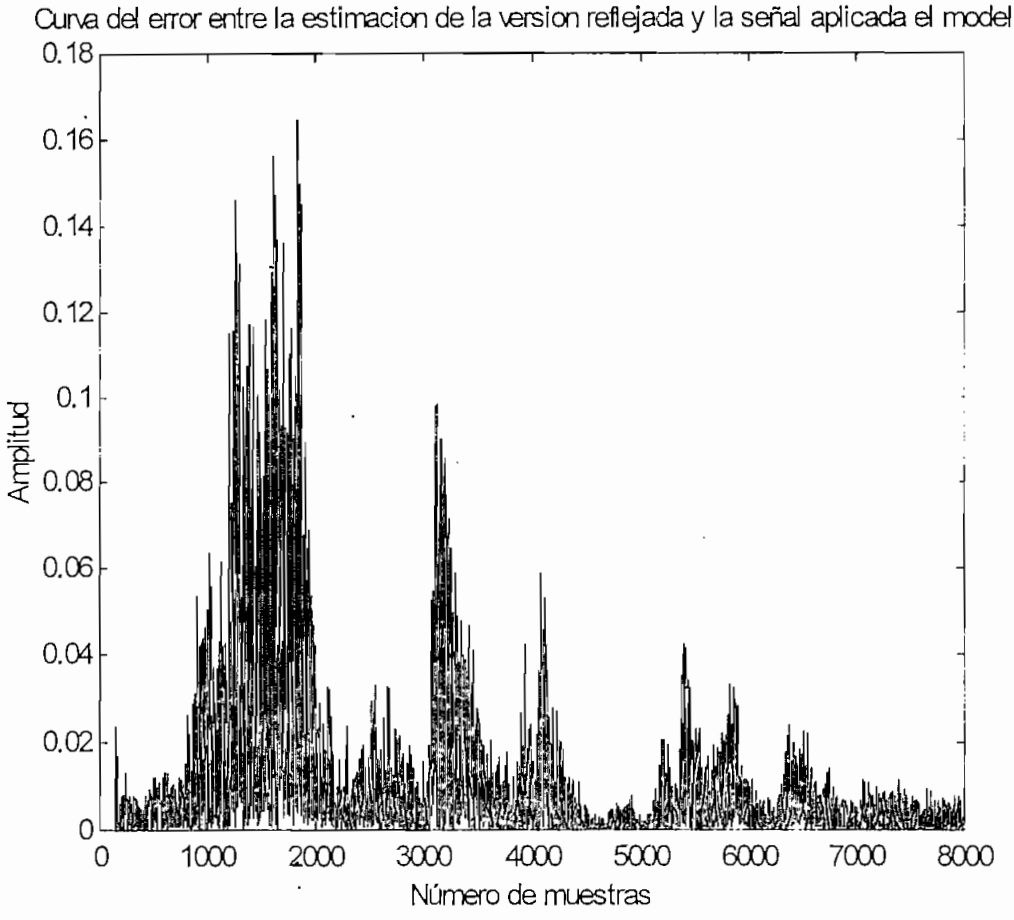



Figura 3.33 Error entre la señal aplicada al modelo y la señal estimada por el filtro.

En la figura 3.33 se muestra el error entre la señal aplicada al modelo y la señal estimada por el filtro.

### 3.6.1 RESULTADOS

Visualizando los resultados anteriores en la simulación realizada , se tiene:

En la Figura 3.34 se muestra la señal telefónica digitalizada y la señal estimada.

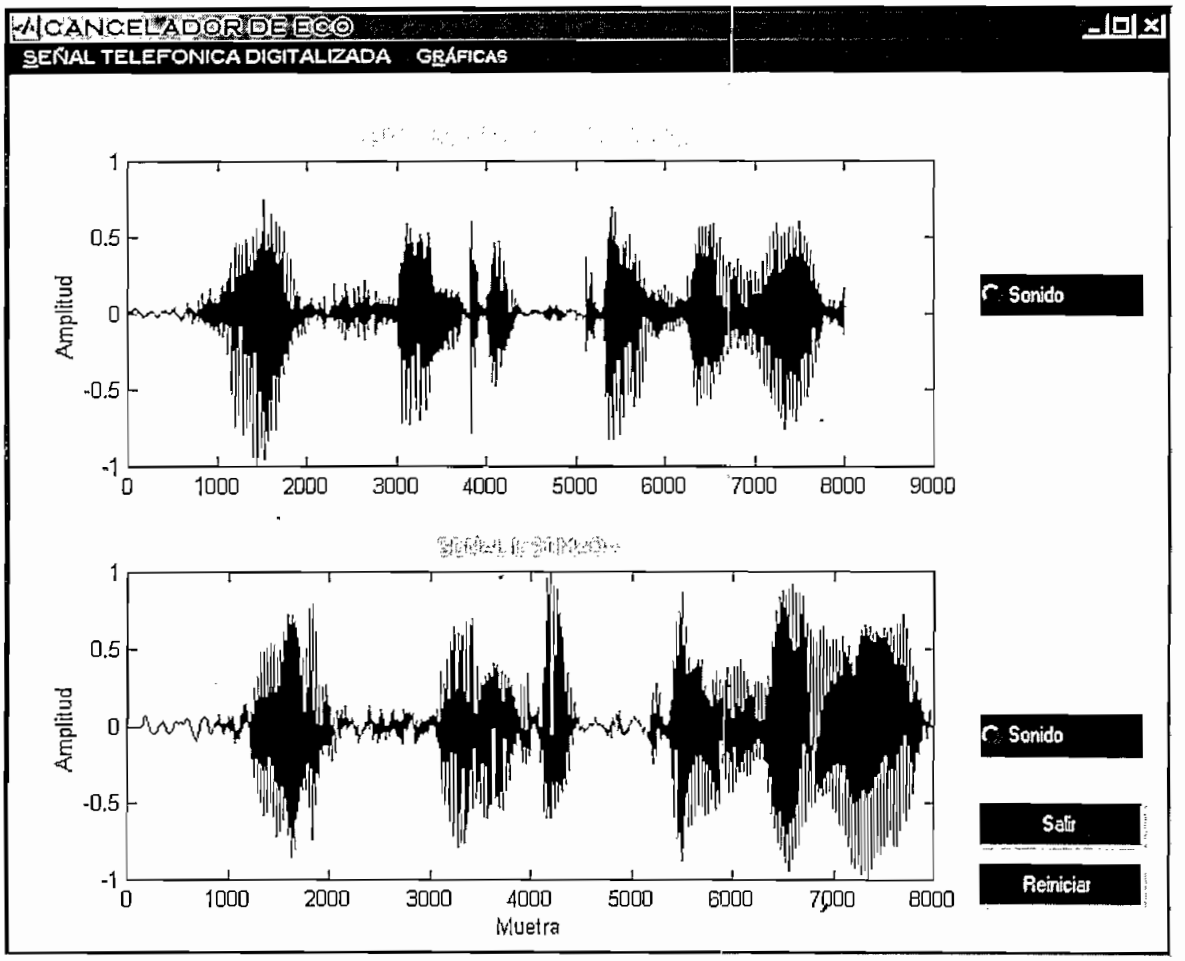



Figura 3.34 Señal Telefónica y Señal Estimada

A continuación en las figuras 3.35, 3.36, 3.37, 3.38 se presentan los resultados antes mencionados y además se presentan gráficas de densidad espectral de potencia , de un espectrograma[3] , y de la transformada de Fourier de las señales patrón y estimadas.

Además, se tiene una opción en la interfaz gráfica de poder escuchar la señal con eco y la señal estimada.

[3] Espectrograma: Representación de las variaciones de la frecuencia - eje vertical- y la intensidad - nivel de grises - en el habla a lo largo del tiempo - eje horizontal -

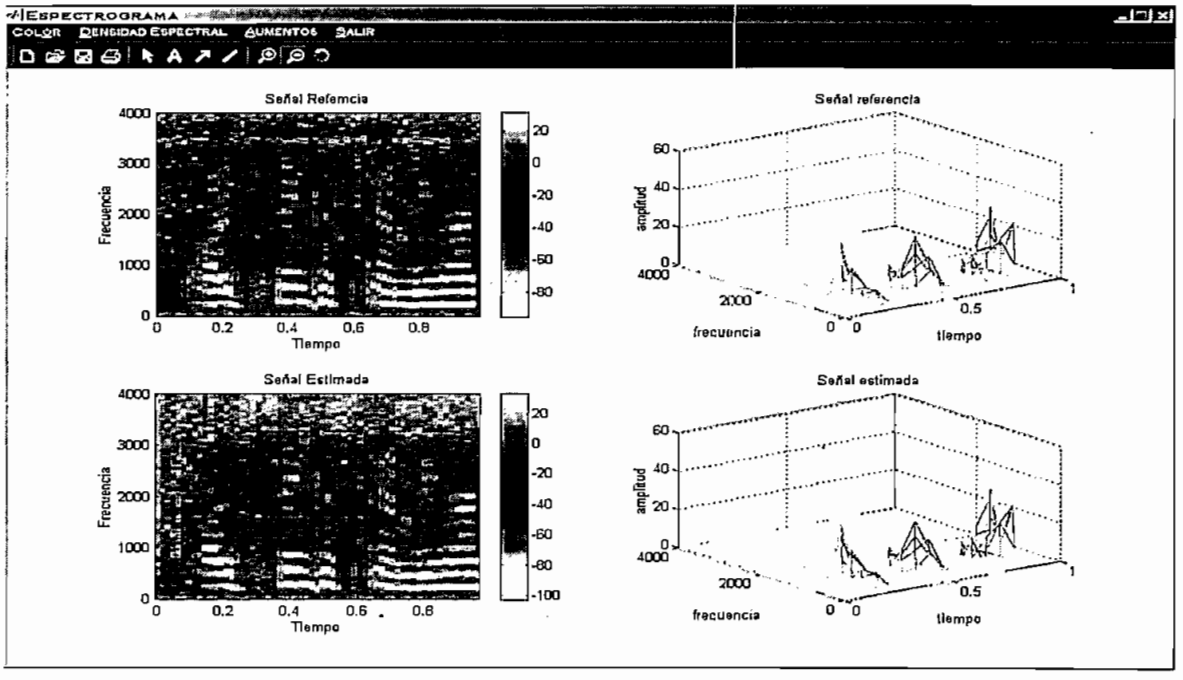



Figura 3.35 Espectrograma

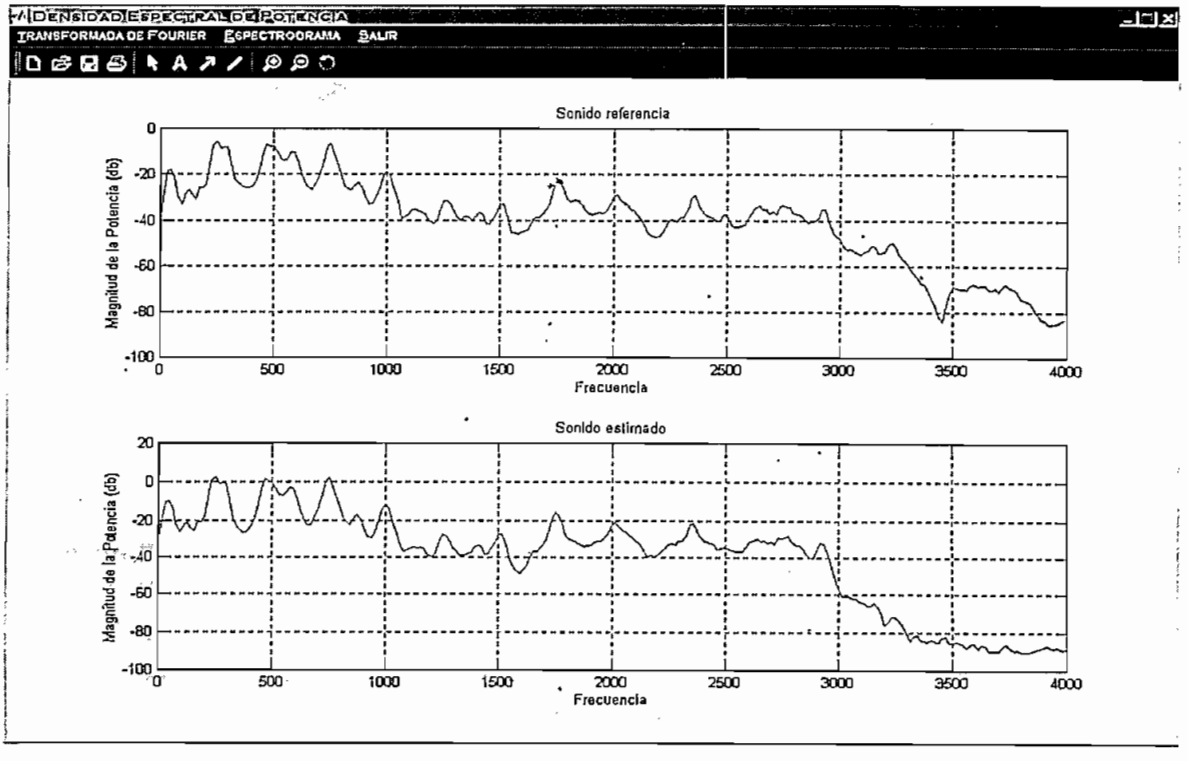



Figura 3.36 Densidad Espectral de Potencia

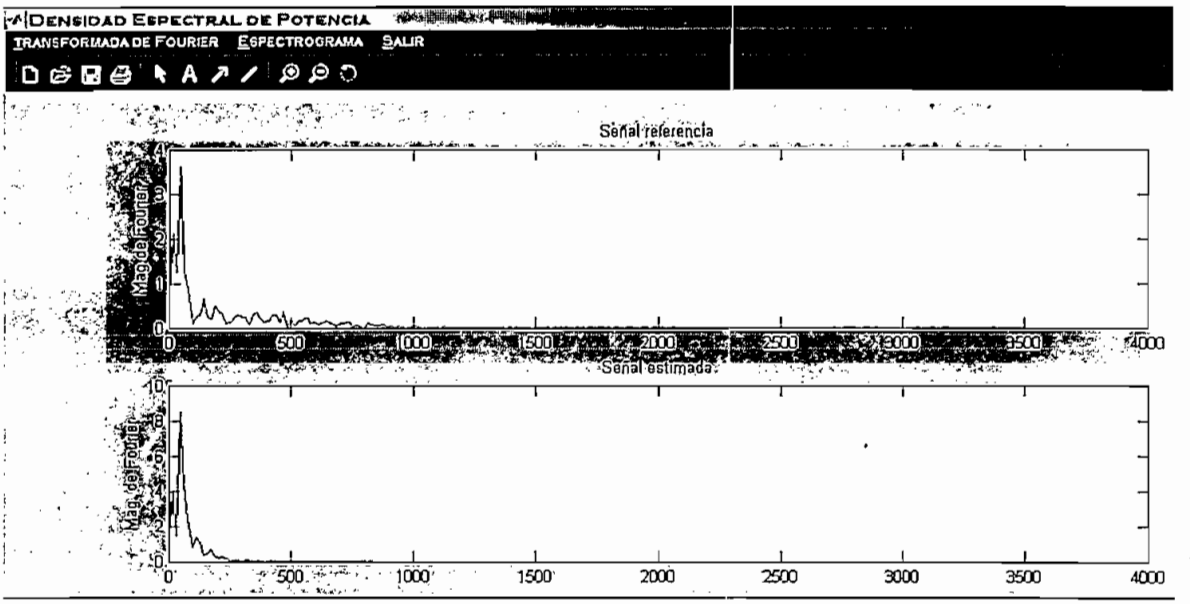



Figura 3.37 Transformada de Fourier de las señales.

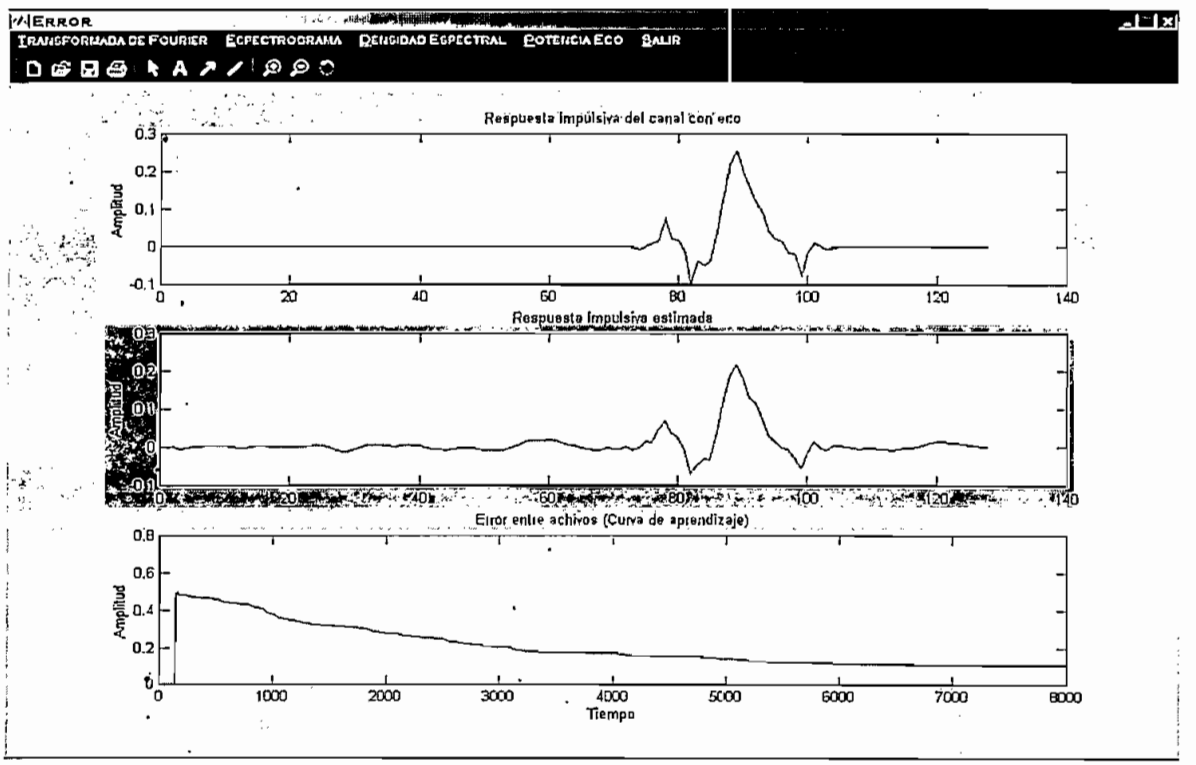



Figura 3.38 Error de los coeficientes del filtro construido, y los coeficientes estimados.

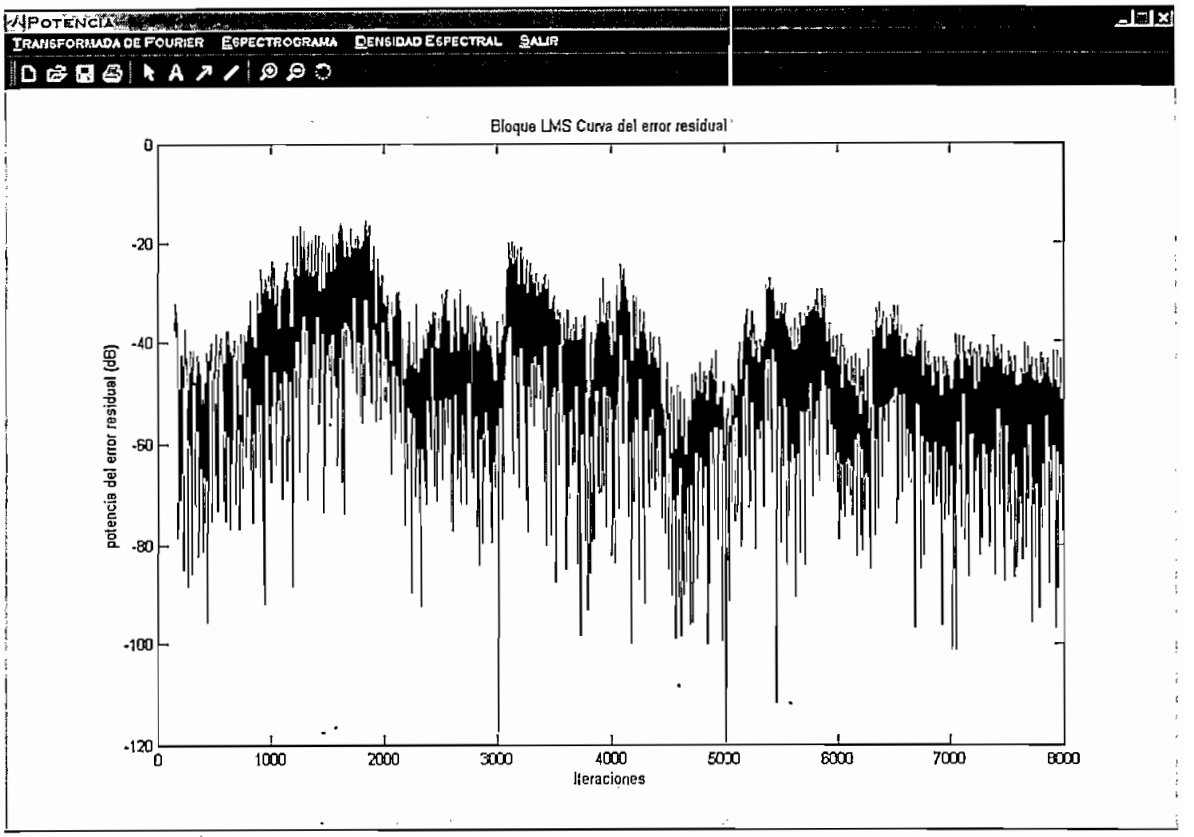



Figura 3.39 Potencia del error residual.

A continuación se presenta los resultados que se obtiene al utilizar señales patrón, a las cuales se les da el efecto de eco con programas comerciales. El archivo resultante se lo ingresa al modelo programado, el error para actualizar los coeficientes del filtro se lo obtiene entre la señal original y la señal estimada por el filtro.

Se presentan los resultados con una señal de voz digitalizada que se la utiliza como señal de referencia a la cual se le da el efecto de eco con una atenuación del 60% y con un retraso de 0.075 segundos, entonces se tiene las siguientes gráficas:

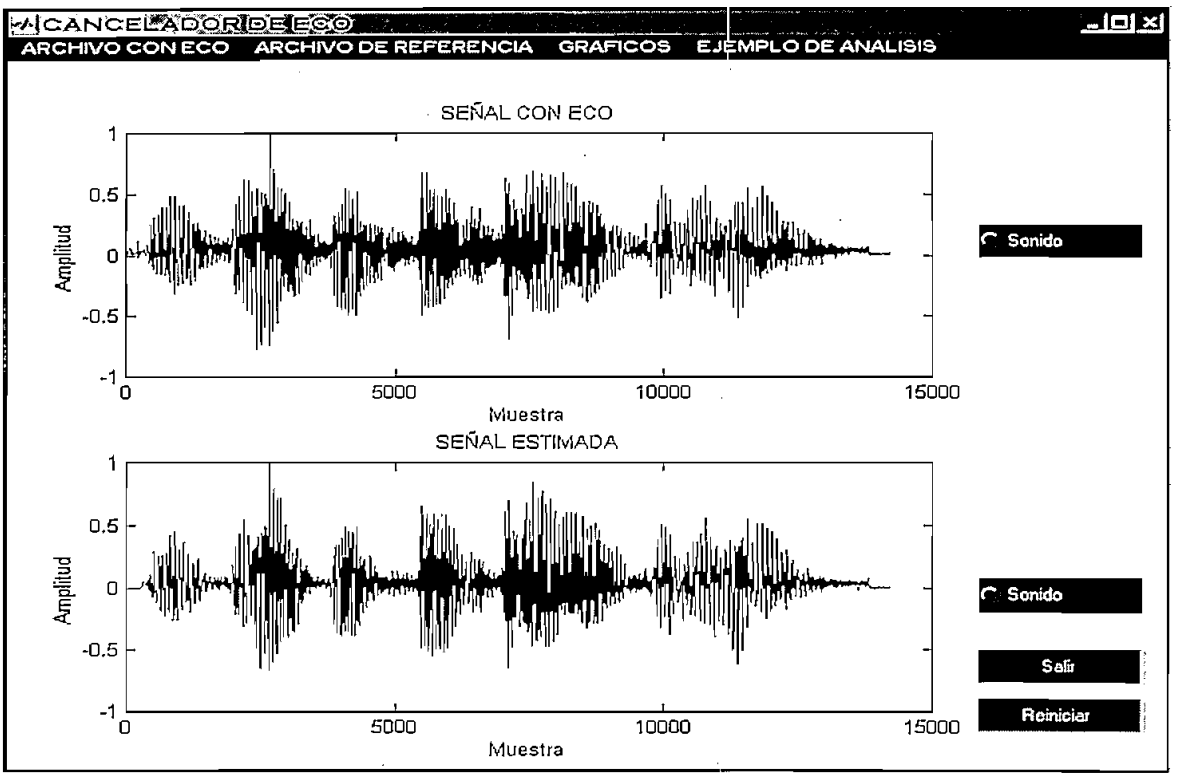



Figura 3.40 Presentación de la señal con eco y de la señal estimada.

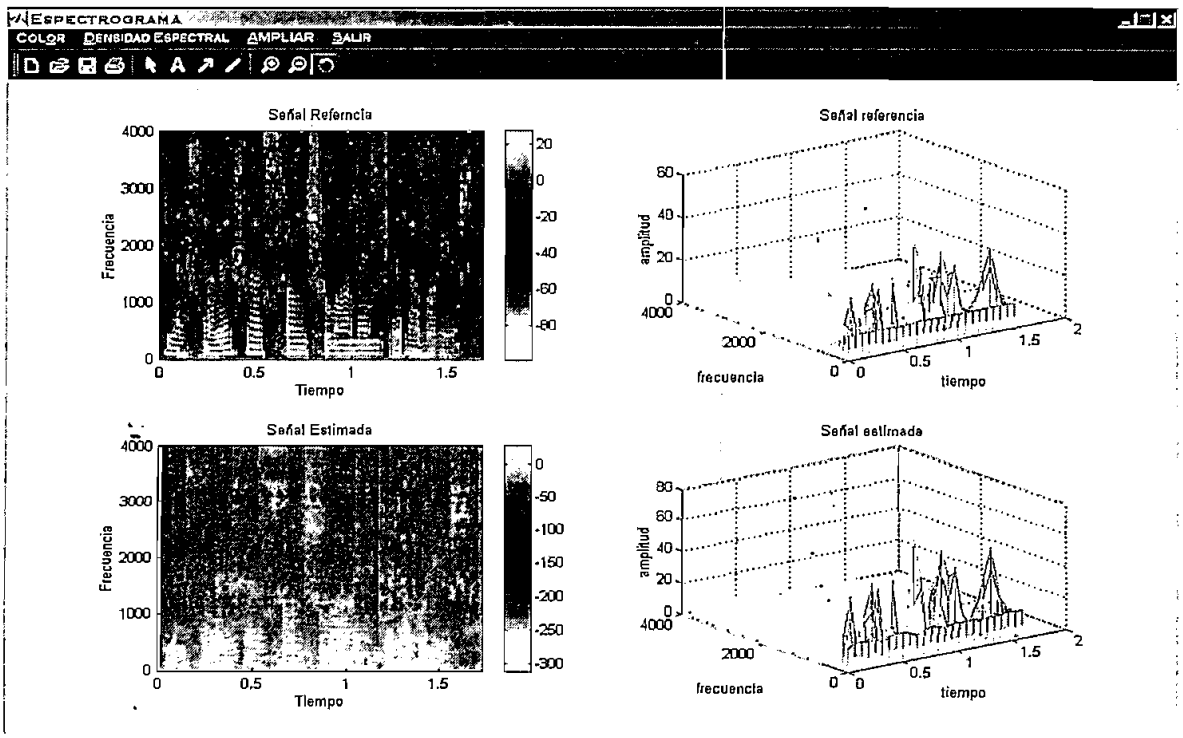



Figura 3.41 Espectrograma.

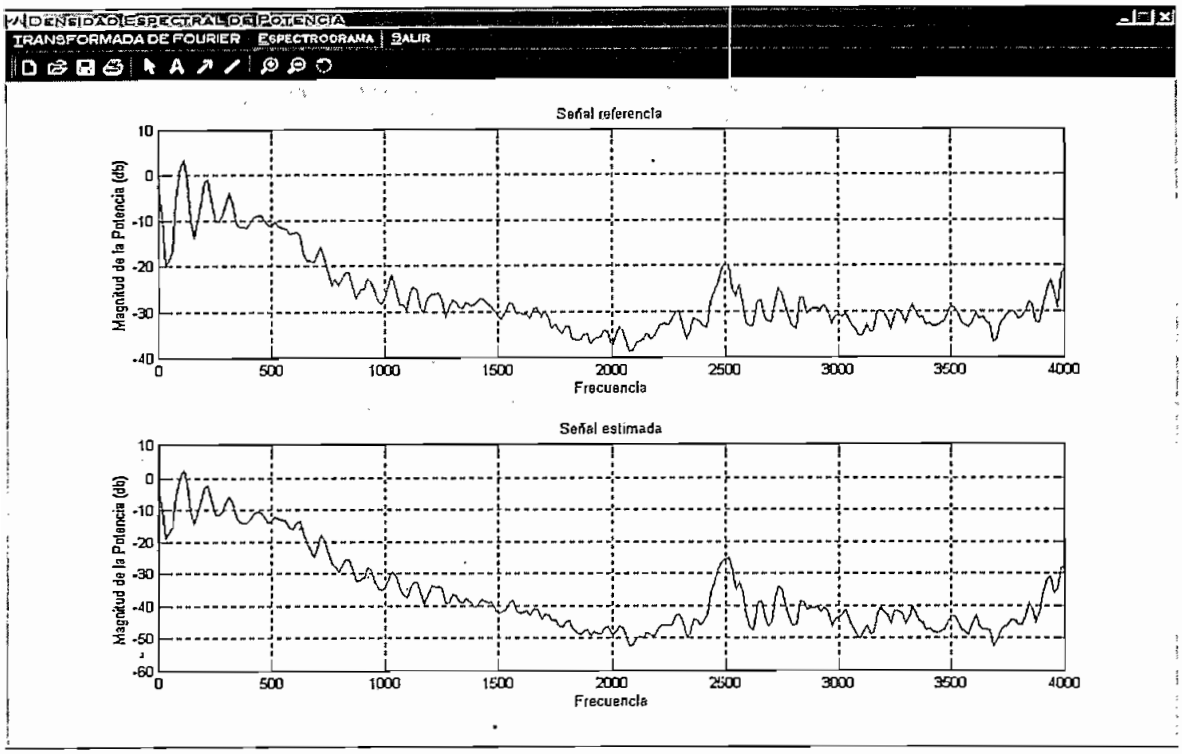



Figura 3.42 Densidad espectral de potencia.

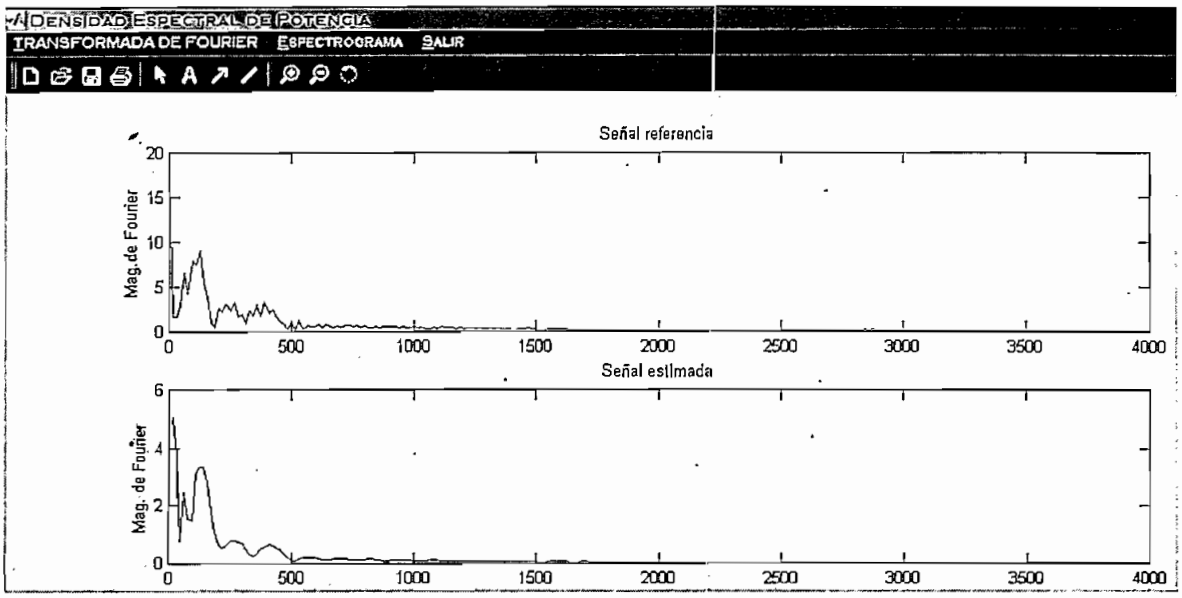



Figura 3.43 Transformada de Fourier.

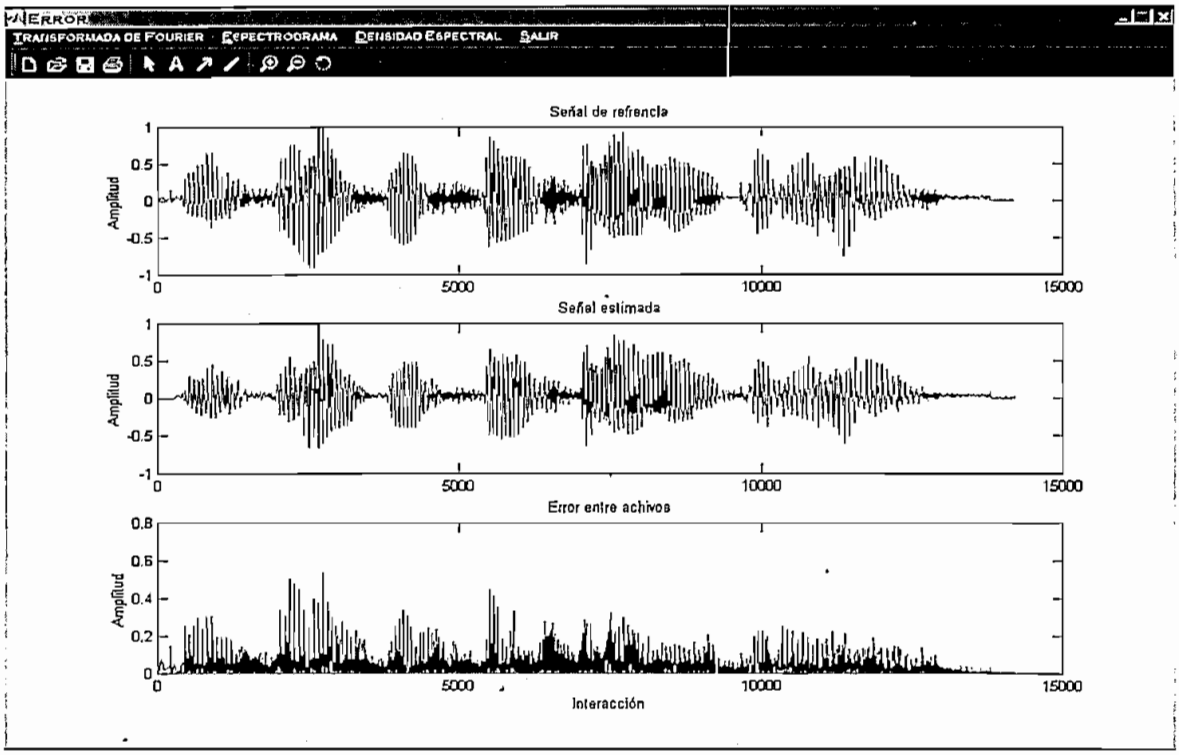



Figura 3.44 Comparación entre la señal de referencia y la señal estimada.

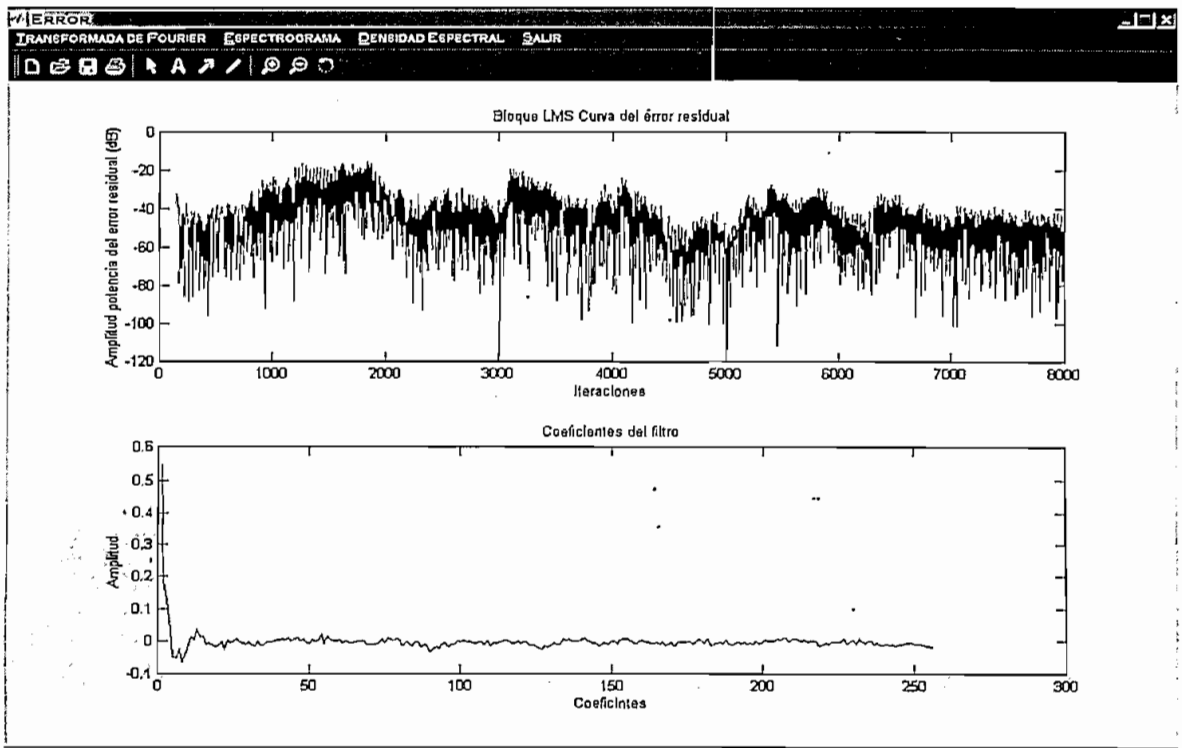



Figura 3.45 Potencia del error residual, y coeficientes finales del filtro.

A continuación se presenta otro caso con la misma señal de referencia anterior ahora se le da un retraso de 0.3 segundos y con una atenuación del 60%, entonces se tiene:

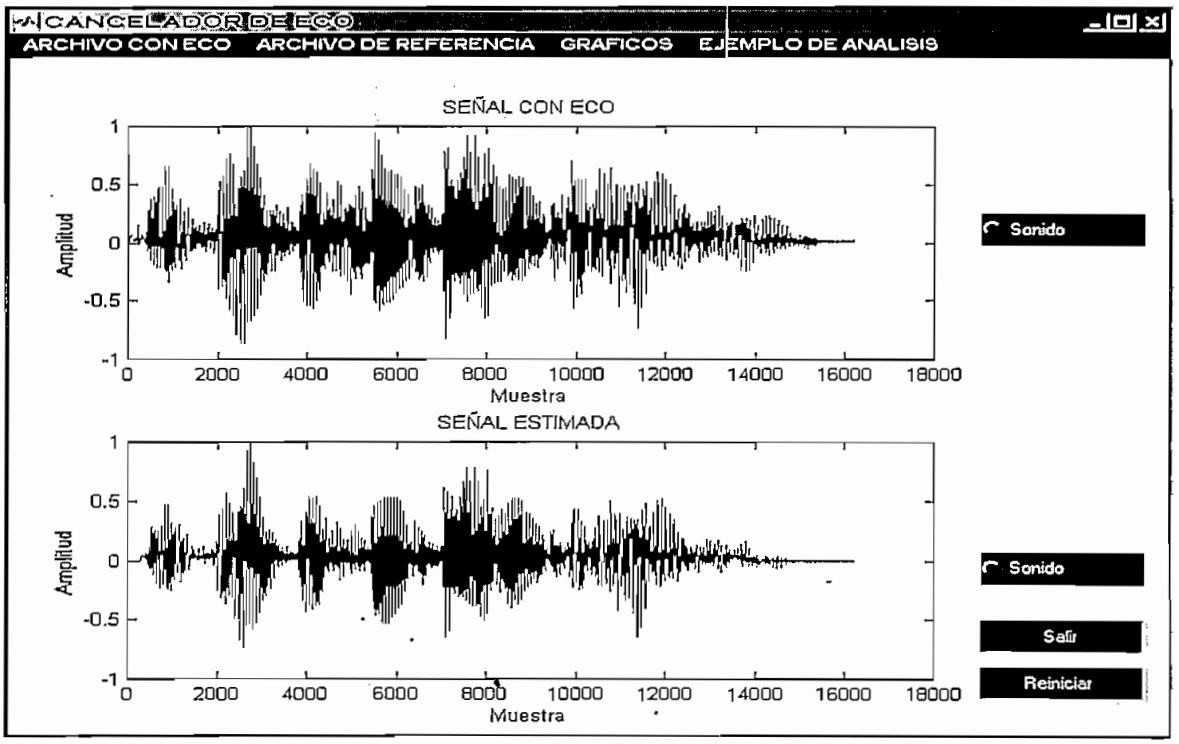



Figura 3.46 Señal con eco y señal estimada.

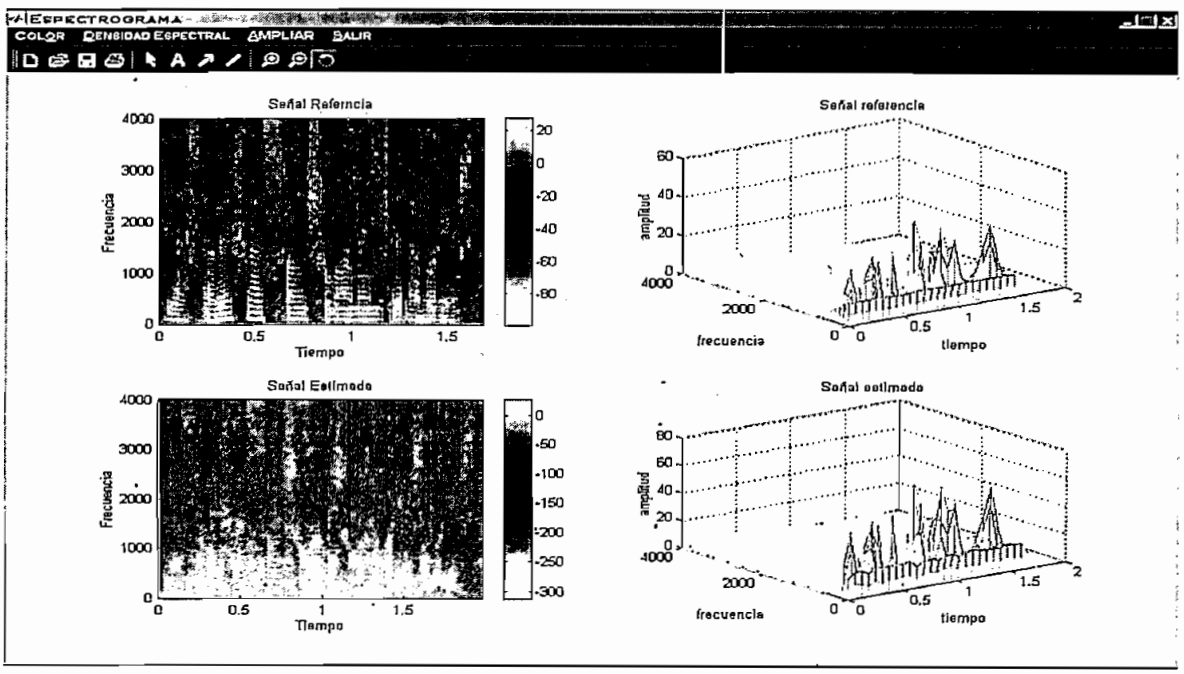



Figura 3.47 Espectrograma.

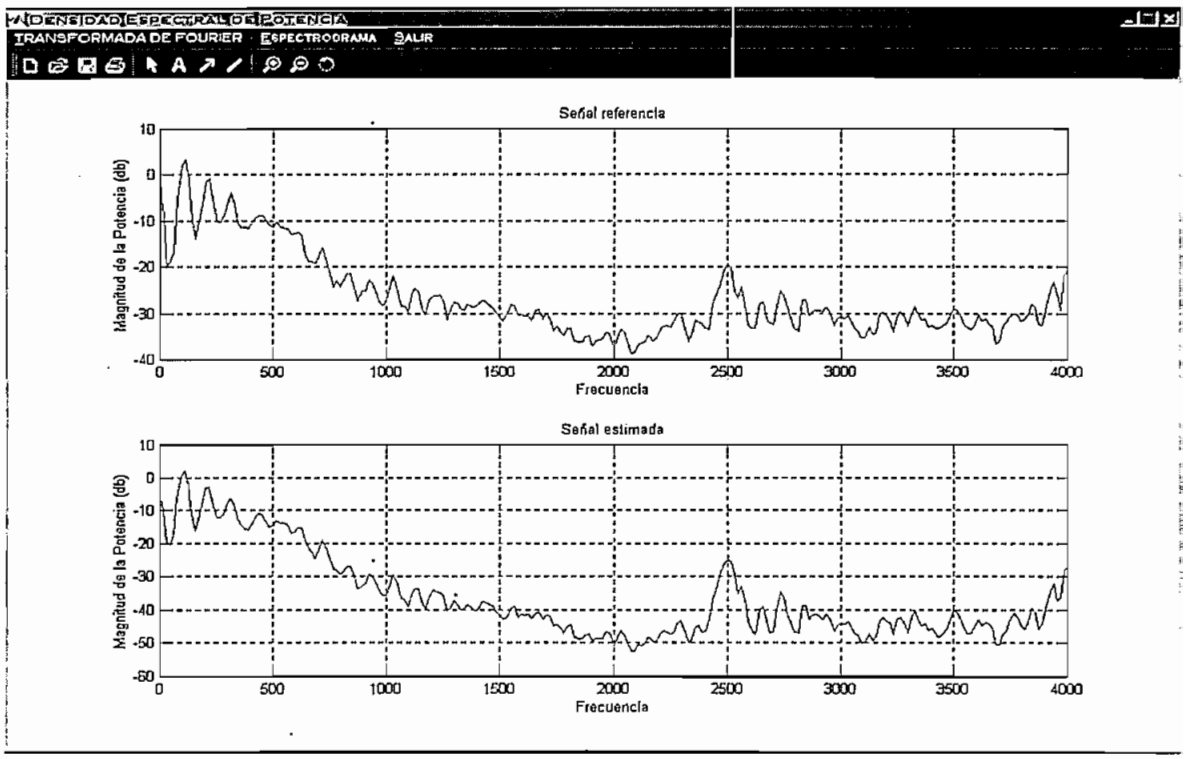



Figura 3.48 Densidad espectral de potencia.

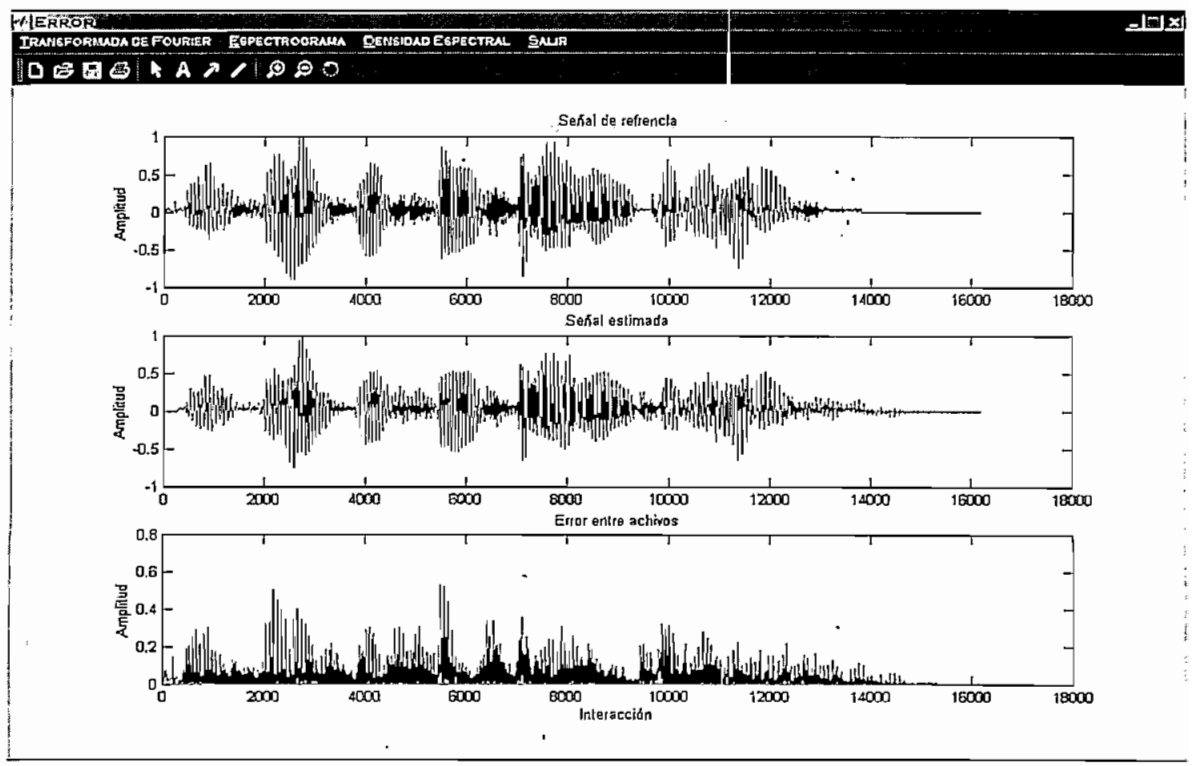



Figura 3.49 Comparación entre la señal estimada y la señal referencia.

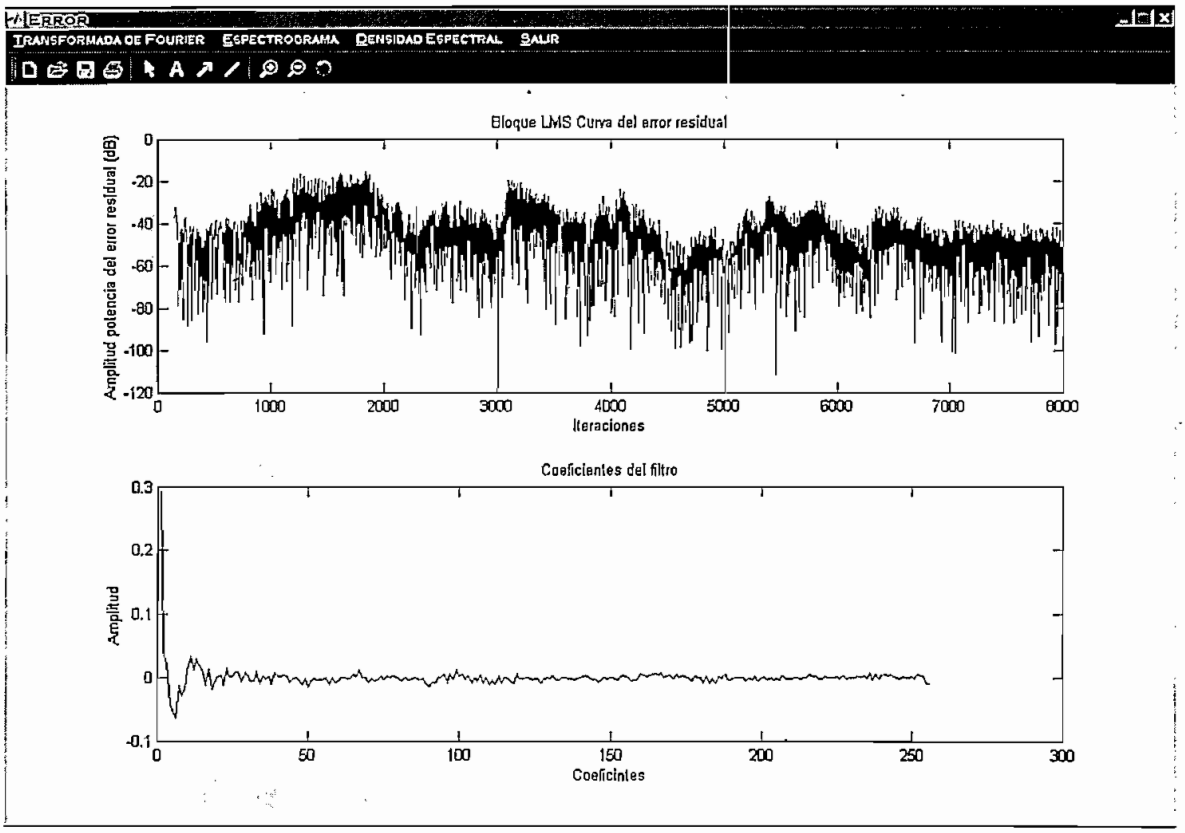



Figura 3.50 Potencia del eco residual y coeficientes finales del filtro.

La forma de utilizar el programa de simulación se presenta en el Anexo C.

El listado del programa .m, con las respectivas subrutinas se presenta en el Anexo E.

# CAPITULO 4. CONCLUSIONES Y RECOMENDACIONES

## 4.1 CONCLUSIONES

- Se ha podido cumplir el objetivo del proyecto de titilación, que es analizar el algoritmo adaptivo LMS, el cual al formar parte de un sistema de cancelación de eco que permite minimizar el nivel de eco en sistemas de telefonía.

- Para la elección de tamaño de paso en el filtrado adaptivo ,μ, se debe utilizar los criterios citados en este proyecto de titilación como son: Potencia de la señal de entrada al sistema de cancelación de eco, tamaño de la ventana utilizada para estimar dicha potencia, y el número de bits utilizados para representar los coeficientes del filtro adaptivo, con lo cual se asegura la convergencia del sistema con una velocidad adecuada . Con un valor menor de μ y si la convergencia está garantizada , ésta es mas lenta y los coeficientes necesitan más interacciones para aproximarse a los valores deseados, pero este avance hacia los valores óptimos se realiza de manera mas suave, sin tantas fluctuaciones, hasta llegar a unos valores más cercanos a los óptimos

- El número de coeficientes utilizados en el filtro es de gran importancia en lo que tiene que ver tanto en la estimación de la señal como en la velocidad del filtro adaptivo, entre más coeficientes se utilice mejor será la estimación pero el sistema se vuelve mas lento, también hay que tomar en cuenta que según el retraso del eco también se utilizará un determinado número de coeficientes

- Se ha demostrado la eficiencia del algoritmo LMS aplicado en la cancelación de eco, a pesar de no disponer la respuesta impulsiva del sistema que distorsiona las señales originales por lo que el error utilizado para actualizar los coeficientes del filtro se lo obtiene de la señal original y de la señal estimada por el filtro adaptivo

- La estabilidad del filtro queda asegurada por: la utilización del filtros FIR , el algoritmo LMS utilizado para actualizar los coeficientes y principalmente del factor $\mu$

- Del análisis matemático del filtro adaptivo, se concluye que el FIR adaptivo está modelando un sistema desconocido que en este caso es la híbrida

- La búsqueda del error cuadrático medio mínimo es sencilla ya que éste es una función cuadrática con un solo mínimo

- El número de coeficientes del filtro adaptivo esta determinado por el retardo que tenga el eco

- Con la utilización del filtro adaptivo LMS se cumple con una de las recomendaciones de la UIT-T G.165 en lo que tiene que ver con el nivel del eco devuelto en la señal estimada, que como se puede observar en el presente trabajo llega a valores menores a los –30 dB

- MATLAB es una herramienta con gran potencial para realizar trabajos de simulación en ingeniería para saber con anterioridad el comportamiento de los sistemas matemáticos antes de ser implementados en tiempo real

## 4.2 RECOMENDACIONES

- Realizar un estudio comparativo entre los diferentes algoritmos adaptivos como son el LMS. NLMS (LMS Normalizado), RLS (Algoritmos de mínimos cuadrados recursivos) y todas sus variantes en lo que respecta a su desempeño en sistemas de cancelación de eco telefónico para saber las ventajas y desventajas de utilizar dichos algoritmos

- Implementar en tiempo real el algoritmo LMS aplicado para la cancelación de eco en canales telefónicos, y poder apreciar las propiedades de dicho algoritmo, principalmente su sencillez computacional

- Ampliar el estudio de sistemas de cancelación de eco para transmisión de datos con la utilización del protocolo IP ya que este tipo de tecnología está utilizándose en nuestro país en lo que tiene que ver en telefonía pública

- Sería recomendable contar con un laboratorio de Procesamiento digital de señales para así facilitar las investigaciones que realizan los estudiantes y así sus estudios sean más completos

# REFERENCIAS BIBLIOGRÁFICAS


1. MathWorks INC, MATLAB User's Guide, Massachusetts, 1995
2. E. Part-Enander, A. Sjöberg, B. Melin, and P. Isaksson, The MATLAB Handbook Addison - Wesley, New York , 1996
3. Escuela Superior de Ingenieros Industriales, Universidad de Navarra, Javier García de Jalón, José Ignacio Rodríguez, Alfonso Brazales, Aprenda MATLAB, San Sebastián Agosto 1999
4. Samirs Soliman, Mandyam Srinath, Señales y Sistemas Continuos y Discretos, Segunda Edición
5. Rodger E. Ziemer, William H.Tranter, D.Ronald Fannin, Signals and Systems: Continuos and Discrete, Third Edition
6. John G. Proakis, Dimitris G. Manolakis, Tratamiento Digital de Señales: Principios, Algoritmos y Aplicaciones,
7. C.W.K. Gritton, D.W.Lin, Echo Cancellation Algorithms, IEEE ASSP Magazine, April 1984
8. Zhaohong Zhang, Gunter Schmer, Analysis of Filter Coefficient Precision on LMS Algorithm Performance for G.165/g.168 Echo Cancellation, Aplication Report Spra561 – February 2000, Texas Instruments
9. David G. Messerschmitt, Echo Cancellation in Speech and Data Transmission, IEEE Jornual selected Areas in Comunications, Vol Sac-2, No.2, March !984
10. Vincent Allen, Mark E Henderson, Erik S Wheeler, and John Williams., Acoustic Echo Cancellation in a Reverberatory Chamber Using an Adaptive Least Means Square Algorithm, The Echo Cancellation Group, Department of Electrical and Computer Engineering, Department of Electrical and Computer Engineering, 1995
11. David Qi, Acustic Echo Cancellation, Algorithms and Implementation on the TMS320C8x, Digital Signal Processing Solutions, Texas Instruments, SPRA 063, May 1996
12. Jelena Nikolic, Implementing a Line Echo Canceller Using The Block Update and NLMS Algorithms on the TMS320C54x DSP, Associate

Technical Staff, DSP Applications, SC Group Technical Marketing, Texas Instruments, April 1997

13. Recomendación UIT-T G.165, Características Generales de las Conexiones y Circuitos Telefónicos Internacionales, Compensadores de Eco

## PAGINAS WEB:

http://prof.usb.ve/tperez/investigacion/cancelacion/Cancelacion.htm

http://www.itu.int/home/

http://www.iie.edu.uy/ense/asign/sisdsp/proyectos/1999/nlms/dsp1.htm

http://www.ece.rice.edu/~dodeedo/elec422/LMSAlgorithm.htm

http://www.ee.vt.edu/~ssgandhi/AFLib/AFLib.html

http://www.iie.edu.uy/ense/asign/sisdsp/proyectos/1998/lms/index.html

http://www.ece.rice.edu/~dodeedo/elec422/LMSAlgorithm.htm

http://www.siglab.ece.umr.edu/ee341/dsp/filters/fir1.html

http://www.cdrewes.de/adaptive.html

http://bips.bi.ehu.es/prj/ruido/

http://www.gtc.cps.unizar.es/~eduardo/docencia/tds/librohtml/adapt1.htm

http://www.cc.isel.ipl.pt/Default.htm

http://teets.cba.ufl.edu/ism6223f00/romanip/echo.html

http://prof.usb.ve/tperez/investigacion/cancelacion/telefono/TelefoCancela.htm

http://www.iec.org/online/tutorials/echo_cancel/index.html

http://www.herberts.org/wayne/proj431/projdes.htm

http://esiiss.ceit.es/asignaturas/tratamiento%20digital/TEMA9/index.htm

http://www.cps.unizar.es/~eduardo/docencia/tds/LibroHtml/lms1.htm

http://www.mathtools.net/

# ANEXO A

# INTRODUCCIÓN AL MATLAB

## A.1 INTRODUCCIÓN

MATLAB es un programa interactivo para computación numérica y visualización de datos. Es ampliamente usado por Ingenieros en el análisis y diseño, posee además una extraordinaria versatilidad y capacidad para resolver problemas en matemática aplicada, física, química, ingeniería, finanzas y muchas otras aplicaciones. Está basado en un sofisticado software de matrices para el análisis de sistemas de ecuaciones.

MATLAB integra análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en que se escribirían tradicionalmente, sin necesidad de hacer uso de la programación tradicional.

En el módulo principal del programa vienen las funciones básicas del procesamiento de matrices además de funciones básicas para graficar datos. A este módulo principal se le agregan módulos complementarios llamados Toolbox, cada uno de los cuales provee funciones matemáticas específicas requeridas para trabajar en un área determinada. Los toolbox instalados se activan automáticamente al correr Matlab. Al instalar un toolbox determinado se instala con este una interfaz gráfica que permite trabajar sistemas más complejos en forma de diagrama de bloques.

Estos Toolboxes cubren en la actualidad prácticamente casi todas las áreas principales en el mundo de la ingeniería y la simulación, destacando entre ellos el 'toolbox' de proceso de imágenes, señal, control robusto, estadística, análisis financiero, matemáticas simbólicas, redes neurales, lógica difusa, identificación de sistemas, simulación de sistemas dinámicos, etc

El Lenguaje de Computación Técnica MATLAB es un ambiente de computación técnica integrada que combina computación numérica, gráficos, visualización avanzada y un lenguaje de programación de alto nivel.

Los poderosos y amplios métodos de cómputo numérico y graficación permiten la prueba y exploración de ideas alternativas con facilidad, mientras que el ambiente de desarrollo integrado facilita producir resultados prácticos fácilmente.

Combinando MATLAB con Signal Processing Toolbox, Wavelet Toolbox y un conjunto de herramientas complementarias - tales como Image Processing, Neural Network, Fuzzy Logic, Statistics y otras – se puede crear un ambiente de análisis personalizado de señales y desarrollo de algoritmos DSP. Para simulación y desarrollo de prototipos usted puede agregar Simulink y el DSP Blockset para modelar y simular sus sistemas DSP, y luego usar Real-Time Workshop para generar código C para su hardware designado.

## A.2 REQUIRIMIENTOS DE HARDWARE

**Espacio requerido en RAM**: 16 Mb RAM mínimo (24 Mb o más recomendable)

**Espacio requerido en disco**: Depende de la partición del disco y de la cantidad de archivos de ayuda instalados: de 30 Mb a 250 Mb.

## A.3 USO DE MATRICES

MATLAB emplea matrices porque con ellas se puede describir infinidad de objetos de una forma altamente flexible y matemáticamente eficiente. Una matriz de píxeles puede ser una imagen o una película. Una matriz de fluctuaciones de una señal puede ser un sonido o una voz humana. Y tal vez más significativamente, una matriz puede describir una relación lineal entre los componentes de un modelo matemático. En este último sentido, una matriz puede describir el comportamiento de un sistema extremadamente complejo. Por

ejemplo una matriz puede representar el vuelo de una avión a 40.000 pies de altura, o un filtro digital de procesamiento de señales.

## A.4 ORIGEN

MATLAB fue originalmente desarrollado en lenguaje FORTRAN para ser usado en computadoras *mainframe*. Fue el resultado de los proyectos *Linpack y Eispack* desarrollados en el *Argonne National Laboratory*, para proveer acceso fácil a su software matricial.

Su nombre proviene de *MATrix LABoratory*. Al pasar de los años fue complementado y reimplementado en lenguaje C. Actualmente la licencia de MatLab es propiedad de *Math Works Inc.*

La arquitectura abierta facilita usar MATLAB y los productos que lo acompañan para explorar datos y crear herramientas personalizadas que proveen visiones profundas, rápidas y ventajas competitivas.

## A.5 PLATAFORMAS

MatLab está disponible para una amplio número de plataformas: estaciones de trabajo SUN, Apollo, VAXstation y HP, VAX, MicroVAX, Gould, Apple Macintosh y PC AT compatibles 80386 o superiores. Opera bajo sistemas operativos UNIX, Macintosh y Windows.

Para Windows requiere adaptador y acelerador gráfico y con display de 8 bits, tarjeta de sonido, Word 7.0 o superior, Navigator 3 o Explorer 4 (para el escritorio de ayuda).

## A.6 ENTORNO DE TRABAJO

**MATLAB** provee una serie de instrucciones o herramientas que facilitan la labor al usuario. Provee herramientas que le facilitan la administración de las variables de su área de trabajo y además para la importación y exportación de

datos. Además provee las herramientas para el desarrollo, manejo, corrección y demarcación de los archivos M, los cuales con la base de la aplicación **MATLAB**.

**MATLAB** viene con una extensa librería. Esta está compuesta de algoritmos computacionales que van desde los comandos sencillos de sum, sine, cosine, hasta funciones de aritmética compleja, y funciones sofisticadas como la inversa de una matriz, los valores característicos, las funciones de Bessel y las transformadas rápidas de Fourier.

Características:

- Cálculos intensivos desde un punto de vista numérico.
- Gráficos y visualización avanzada.
- Lenguaje de alto nivel basado en vectores, arreglos y matrices.
- Colección muy útil de funciones de aplicación.

## A.7 SALIDAS O PRESENTACIONES

MATLAB provee acceso inmediato a las características gráficas especializadas requeridas en ingeniería y ciencias. La potente graficación orientada a objetos gráficos le permite graficar los resultados de su análisis, incorporar gráficos en sus modelos de sistemas, presentar rápidamente objetos complejos 3-D objetos, y crear resultados de presentación, entre los cuales se destacan:

- Representaciones 2-D y 3-D, incluyendo datos triangulados y reticulados
- Control de fuentes, letras Griegas, símbolos, subíndices y superíndices
- Selección expandida de símbolos marcadores de curvas
- Gráficos de torta, de barras 3-D y gráficos de barras horizontales
- Gráficos 3-D y sólido modelado
- Representación de imágenes y archivos I/O .
- Gráficos comentados
- Leer/Escribir archivos de datos Hierarchical Data Format (HDF)
- Presentación de OpenGL software y hardware
- Animación

- Soporte de colores verdaderos (24-bit RGB)
- Fuentes múltiples de luz para superficies coloreadas
- Vista basada en cámara y control de perspectiva
- Iluminación Plana, Gouraud y Phong
- Soporte eficiente de imagen de datos de 8-bit
- Control de eje y cámara
- Propiedades de superficie y patch
- Modelos de iluminación
- Control gráfico de objetos
- Impresión y representación de copias
- Formatos gráficos exportables
- Soporte de publicación de escritorio

### A.7.1 DESCRIPTIVOS GRÁFICOS PARA EXPLORAR Y PRESENTAR SUS DATOS

Los gráficos de propósitos generales y de aplicación específica le permiten visualizar al instante señales, superficies paramétricas, imágenes y más. Todos los atributos de los gráficos de MATLAB son personalizables, desde los rótulos de ejes al ángulo de la fuente de luz en las superficies 3-D . Los gráficos están integrados con las capacidades de análisis, de modo que usted puede mostrar gráficamente cualquier conjunto de datos, ecuación o resultado funcional sin editar.

### A.7.2 I/O DIRECTO DE DATOS

Usted puede ingresar y sacar datos de MATLAB rápidamente. Las funciones están disponibles para leer y escribir archivos de datos formateados en MATLAB, llamados archivos MAT. Funciones adicionales ejecutan programas ASCII e I/O binario de bajo nivel desde los archivos de programas M, C, y Fortran, permitiéndole trabajar con todos los formatos de datos. MATLAB también incluye soporte incorporado para formatos populares de archivos estándar.

### A.7.3 COMPUTACIÓN SIMBÓLICA INTEGRADA

Integrando el motor simbólico Maple V® con MATLAB, los Symbolic Math Toolboxes le permiten mezclar libremente computación simbólica y numérica, una sintaxis simple e intuitiva.

### A.7.4 ANÁLISIS DE DATOS CONFIABLE, RÁPIDO Y EXACTO

Los métodos usados comunmente para análisis de datos multidimensional generalizados 1-D, 2-D están incorporados en MATLAB. Interfaces gráficas fáciles de usar, específicas para aplicaciones, la línea de comando interactiva y herramientas de programación estructuradas le permiten elegir el mejor camino para sus tareas de análisis.

### A.7.5 ANÁLISIS DE DATOS PARA DSP

MATLAB ofrece muchas herramientas para realizar la funcionalidad indispensable en procesamiento de señales, tales como Transformadas Rápidas Fourier y Transformadas Rápidas Inversas de Fourier. La visualización de datos de procesamiento de señales está soportada por funciones tales como gráficos stem y periodogramas. El lenguaje de MATLAB, inherentemente orientado a matrices hace que la expresión de coeficientes de filtros y demoras de buffers sean muy simples de expresar y comprender.

### A.7.6 ANÁLISIS DE DATOS EN APLICACIONES DE IMÁGENES

MATLAB y la Image Processing Toolbox ofrece un amplio conjunto de herramientas que le permite fácilmente manipular, procesar y analizar datos de imágenes, mostrar interactivamente pantallas de imágenes 2-D o 3-D, visualizar datos temporales cuando es necesario, y comentar sus resultados para publicaciones técnicas. La orientación basada en matrices del lenguaje de MATLAB le permite expresar en forma compacta operaciones matemáticas de forma similar a cómo las expresaría sobre papel. Como resultado, es fácil e intuitivo efectuar procesamiento de imágenes y operaciones de análisis tales como FFTs, filtrado 2-D, morfología binaria, manipulación geométrica, conversión de espacios de colores, compresión, análisis de componentes conectados y más. Sea que usted esté usando los algoritmos del sistema o esté inventando los suyos

propios, MATLAB le provee un ambiente en el que usted puede experimentar. A diferencia de C y C++, MATLAB le permite desarrollar algoritmos desde cero o trabajar con interfaces complicadas a bibliotecas externas.

## A.8 EL MATLAB Y LA ESTADÍSTICA

Statistics Toolbox: Combina poderosos algoritmos estadísticos con interfaces gráficas interactivas

Las Statistics Toolbox le da un rango ancho de herramientas para realizar cálculos estadísticos. Los despliegues gráficos interactivos le permitieron aplicar métodos estadísticos fáciles y de forma consistente, mientras el lenguaje de MATLAB le permite fácilmente crear los acostumbrados métodos estadísticos y de análisis.

## A.9 LIBRERÍAS

Librería de Aplicaciones de MATLAB

### A.9.1 SIGNAL PROCESSING TOOLBOX

MATLAB tiene una gran colección de funciones para el procesamiento de señal en el Signal Processing Toolbox. Este incluye funciones para:

- Análisis de filtros digitales incluyendo respuesta en frecuencia, retardo de grupo, retardo de fase.
- Implementación de filtros, tanto directo como usando técnicas en el dominio de la frecuencia basadas en la FFT.
- Diseño de filtros IIR, incluyendo Butterworth, Chebyschev tipo I, Chebyshebv tipo II y elíptico.
- Diseño de filtros FIR mediante el algoritmo óptimo de Parks-McClellan.

### A.9.2 THE MATLAB C MATH LIBRARY

La MATLAB C Math Library proporciona al usuario la capacidad computacional de MATLAB en una librería en formato objeto enlazable. El objetivo principal de la C Math Library es soportar el desarrollo de aplicaciones 'stand alone' utilizando MATLAB y su compilador. Puede ser utilizada

independientemente de MATLAB por programadores avezados en lenguaje C que necesiten prestaciones computacionales robustas y de alto rendimiento.

La MATLAB C Math Library proporciona una amplia gama de funciones clásicas del programa MATLAB , proporcionadas como librerías objeto, incluyendo básicamente las siguientes categorías de funciones presentes en MATLAB y archivos M compilados:

- Algebra lineal.
- Funciones matemáticas elementales y especializadas.
- Operadores lógicos y aritméticos.
- Matrices elementales y manipulación de vectores.
- Matrices especiales.
- Estadística básica y análisis de datos.
- Polinomios e interpolación.
- Gestión de cadenas de caracteres.
- Entradas y Salidas.
- Gestión de memoria y errores.

Para construir una aplicación del tipo 'stand alone' que incorpore código originalmente desarrollado como archivos M de MATLAB , deberán de seguirse los pasos siguientes:

- Utilizar el compilador de MATLAB para convertir archivos M en C mediante la utilización de la instrucción mcc -e (la cual es externa a MATLAB).
- Compilar el código C fuente en código objeto utilizando un compilador ANSI C.
- Enlazar el código resultante con la MATLAB C Math Library y con cualquier tipo de archivos y programas específicos que hayan sido previamente definidos por el usuario.

Mediante la compilación de los archivos M podemos obtener un rendimiento significativo. La velocidad de mejora de este rendimiento, depende fuertemente de cada aplicación. En algunos casos el rendimiento puede mejorar hasta en 200 veces la ejecución si la comparamos con el modo de trabajo interpretado del

programa. Las operaciones matriciales y vectoriales ejecutadas desde MATLAB ya están fuertemente optimizadas en su diseño. Sin embargo, mediante la utilización del compilador se obtendrán significativas mejoras. Ciertas instrucciones, como load y eval, no están soportadas por el compilador de MATLAB . Este no puede generar código de los diagramas de bloques de SIMULINK.

### A.9.3 SYMBOLIC MATH TOOLBOX

Los principales tipos de operaciones soportados son los siguientes:

- Álgebra simbólica: Derivación, integración y simplificación de expresiones matemáticas
- Álgebra lineal exacta: Inversas, determinantes, autovalores y formas canónicas de matrices simbólicas
- Aritmética de precisión variable: Evaluación de expresiones matemáticas con diversos grados de precisión
- Resolución de ecuaciones: Resolución numérica y simbólica de ecuaciones algebráicas y diferenciales
- Funciones matemáticas especiales: Evaluación de la mayoría de las funciones utilizadas en matemáticas aplicadas

### A.9.4 OPTIMIZATION TOOLBOX

El toolbox de optimización consta de un conjunto de funciones que resuelven problemas de extremos, con o sin condiciones, de funciones reales las cuales son generalmente multivariables y no lineales. Así mismo, posee funciones para la resolución de algunos tipos de problemas matriciales en extremos. Resulta conveniente para una comprensión y mejor manejo de la toolbox poseer conocimientos básicos previos de análisis de funciones reales, matrices y teoría de extremos.

Algunas de las áreas básicas que cubre este toolbox para MATLAB son las siguientes:

- Cálculo de un extremo local (máximo o mínimo) de una función real f(x), en general multivariable y no lineal, sin imponer ninguna restricción o condición a la solución. Como caso particular, se incluye una rutina especial para problemas de mínimos cuadrados no lineales.
- Cálculo de un extremo local (máximo o mínimo) de una función real f(x), en general multivariable y no lineal, condicionado a que la solución satisfaga ciertas condiciones de desigualdad (g(x)<=0) y/o igualdad (g(x)=0).
- Problemas de aproximación a un conjunto de objetivos.
- Cálculo de soluciones de un sistema de ecuaciones continuas y, en general, no lineales.
- Solución de problemas minimax.
- Programación lineal.
- Programación cuadrática.
- Problemas de mínimos cuadrados no negativos.
-

### A.9.5 IMAGE PROCESSING TOOLBOX

Este Toolbox proporciona a MATLAB de un conjunto de funciones que amplia las capacidades del producto para realizar desarrollo de aplicaciones y de nuevos algoritmos en el campo del proceso y análisis de imágenes. El entorno matemático y de creación de MATLAB es ideal para el procesado de imágenes, ya que estas imágenes son, al fin y al cabo, matrices. Este toolbox incorpora funciones para:

- Diseño de filtros.
- Mejora y retocado de imágenes.
- Análisis y estadística de imágenes.
- Operaciones morfológicas, geométricas y de color.
- Transformaciones 2D.

Los programas actuales de procesado y análisis de imágenes se clasifican actualmente en dos categorías: librerías de bajo nivel para programadores

profesionales y paquetes de aplicación con capacidades limitadas de personalización.

El Image Processing Toolbox entra dentro de la categoría de familias de funciones que, desde el entorno de trabajo de MATLAB , permitirá al profesional efectuar una exploración exhaustiva y desde un punto de vista matemático de las imágenes y gráficos que se deseen tratar o analizar.

Algunas de las funciones más importantes incluidas dentro de este toolbox son las siguientes:

- Análisis de imágenes y estadística.
- Diseño de filtros y recuperación de imágenes.
- Mejora de imágenes.
- Operaciones morfológicas.
- Definición de mapas de colores y modificación gráfica.
- Operaciones geométricas.
- Transformación de imágenes.
- Proceso de bloques
-

### A.9.6 NEURAL NETWORK TOOLBOX

Este toolbox proporciona funciones para el diseño, inicialización, simulación y entrenamiento de los modelos neuronales de uso más extendido en la actualidad.

Mediante la inclusión de un amplio abanico de funciones y procedimientos escritos para MATLAB, el usuario puede mediante el Neural Network Toolbox efectuar el diseño de arquitecturas complejas, combinando los modelos que ya están proporcionados por defecto en el toolbox. Asimismo, el usuario puede definir sus propias funciones de transferencia e inicialización, reglas de aprendizaje, funciones de entrenamiento y estimación de error para usarlas posteriormente con las funciones básicas. Este toolbox incluye un manual de introducción al campo de las redes neuronales junto con una colección de demostraciones y aplicaciones muy didácticas, útiles para el estudio y la profundización en las cuestiones fundamentales de los paradigmas de redes

neuronales básicos. Asimismo, se proporcionan las referencias bibliográficas más significativas referidas a los distintos modelos que aparecen en la aplicación.

Dentro de las aplicaciones básicas de este toolbox, cabe destacar aquellas que están orientadas al campo de la industria aeroespacial y automoción (simulación, sistemas de control, autopilotaje), banca, defensa (reconocimiento de patrones, procesamiento de señales, identificación de imágenes, extracción de características, compresión de datos), electrónica (control de procesos, análisis de errores, modelado no lineal, síntesis de voz, visión por ordenador), economía (análisis financiero, análisis predictivo), industria (control de procesos, identificación en tiempo real, sistemas de inspección), medicina, robótica (control de trayectorias, sistemas de visión), reconocimiento y síntesis del habla, telecomunicaciones (control de datos e imágenes, servicios de información automatizada, traducción del lenguaje hablado en tiempo real, diagnosis, sistemas de enrutamiento), etc. El toolbox contiene muchos ejemplos de algunas de estas aplicaciones.

### A.9.7 NON LINEAR CONTROL DESIGN TOOLBOX

Este toolbox está pensado para ser utilizado exhaustivamente por ingenieros que diseñan controladores para industrias avanzadas, destacando el sector del automóvil, ingeniería aeroespacial, control de procesos y empresas petroquímicas. El toolbox NCD permite por primera vez a los ingenieros diseñar directamente sus controladores en un ambiente no lineal, obviando la aproximación lineal y otros procedimientos auxiliares que antes se necesitaban de modo imperativo. Los resultados ahora son de elevada calidad, controladores más robustos y un ciclo de diseño mucho más rápido.

### A.9.8 NAG FOUNDATION TOOLBOX

Este toolbox proporciona un acceso interactivo, desde dentro de MATLAB, a un amplio conjunto de funciones matemáticas y estadísticas contenidas en las clásicas NAG Fortran Libraries de la empresa The Numerical Algorithms Group Incorpora más de 200 archivos M, los cuales cubren un amplio espectro de áreas de interés, entre las que cabe destacar optimización, ecuaciones diferenciales ordinarias y en derivadas parciales, cuadratura, estadística, etc.

Algunas de las áreas de cobertura de la NAG Foundation Toolbox son las siguientes:

- Ceros de polinomios
- Raíces de una o más ecuaciones de tipo trascendental.
- Suma de series.
- Cuadraturas.
- Ecuaciones diferenciales ordinarias.
- Ecuaciones diferenciales en derivadas parciales.
- Estadística no paramétrica.
- Análisis de series temporales.
- Rutinas de clasificación.
- Aproximación de funciones especiales.
- Aproximación de curvas y superficies.
- Maximización y minimización de funciones.
- Factorización de matrices.
- Valores y vectores propios.
- Resolución de ecuaciones lineales simultáneas.
- Estadística básica.
- Análisis de correlación y regresiones.
- Métodos multivariantes.
- Generación de números aleatorios.

## A.10 SIMULINK

Simulink es una herramienta para el modelaje, análisis y simulación de una amplia variedad de sistemas físicos y matemáticos, inclusive aquellos con elementos no lineales y aquellos que hacen uso de tiempos continuos y discretos. Como una extensión de MatLab, Simulink adiciona muchas características específicas a los sistemas dinámicos, mientras conserva toda la funcionalidad de propósito general de MatLab. Así Simulink no es completamente un programa separado de MatLab, sino un anexo a él. El ambiente de MatLab está siempre disponible mientras se ejecuta una simulación en Simulink. Tiene dos fases de uso: la definición del modelo y el análisis del modelo.

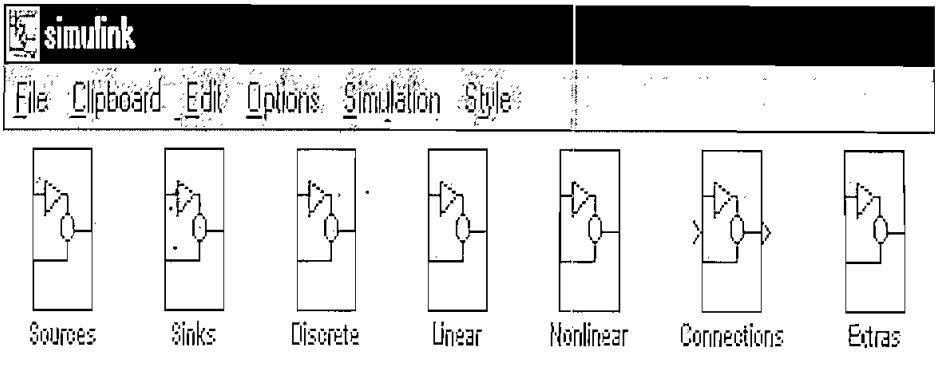
La definición del modelo significa construir el modelo a partir de elementos básicos construidos previamente, tal como, integradores, bloques de ganancia o servomotores. El análisis del modelo significa realizar la simulación, linealización y determinar el punto de equilibrio de un modelo previamente definido.

Para simplificar la definición del modelo Simulink usa diferentes clases de ventanas llamadas ventanas de diagramas de bloques. En estas ventanas se puede crear y editar un modelo gráficamente usando el mouse. Simulink usa un ambiente gráfico lo que hace sencilla la creación de los modelos de sistemas.

Después de definir un modelo éste puede ser analizado seleccionando una opción desde los menús de Simulink o entrando comandos desde la línea de comandos de MatLab.

Simulink puede simular cualquier sistema que pueda ser definido por ecuaciones diferenciales continuas y ecuaciones diferenciales discretas. Esto significa que se puede modelar sistemas continuos en el tiempo, discretos en el tiempo o sistemas híbridos.

Simulink usa diagramas de bloques para representar sistemas dinámicos. Mediante una interface gráfica con el usuario se pueden arrastrar los componentes desde una librería de bloques existentes y luego interconectarlos mediante conectores y alambre. La ventana principal de Simulink se activa escribiendo simulink en la línea de comandos de MatLab, y se muestra a continuación:




SIMULINK Block Library (Version 1.2c)

Figura A.1 Ventana principal del simulink

Haciendo doble click en cualquiera de las librerías presentes en esta ventana se abrirá otra ventana conteniendo una cantidad de bloques relativos a dicha librería.

Para realizar un sistema debe abrirse una nueva ventana de diagrama de bloques seleccionando la opción file del menú principal del Simulink y luego la opción new. En esta nueva ventana se colocarán todos los bloques interconectados que formarán el sistema deseado.

Como ejemplo se ha tomado un generador de ondas seno de la librería de fuentes "sources" y un osciloscopio de la librería "sinks", ambos se unieron mediante un conector usando el mouse. Este sistema se almacena como un archivo-m.

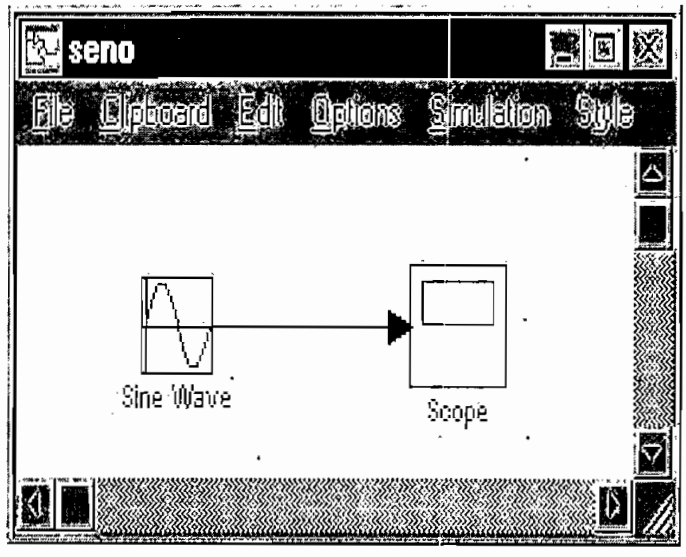



Figura A.2 Ejemplo de utilización

Haciendo doble click sobre cada elemento del sistema se pueden ver y modificar sus características. Por ejemplo, al generador seno se le puede modificar su amplitud, frecuencia y fase. Al osciloscopio se le definen las escalas horizontal y vertical.

Para ejecutar el programa se usa la opción simulation en el menú de la ventana del archivo-m creado. En este submenú está la opción start que permite

ejecutar el programa. También está la opción parameters que activa el panel de control de Simulink en donde se definen los métodos y parámetros usados para la simulación, tal como se muestra a continuación:

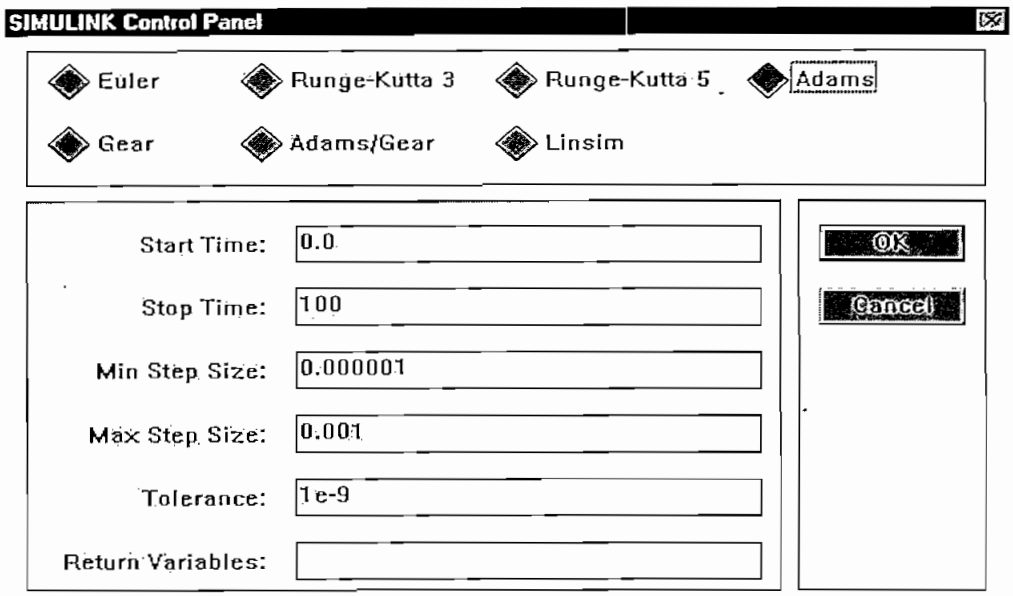
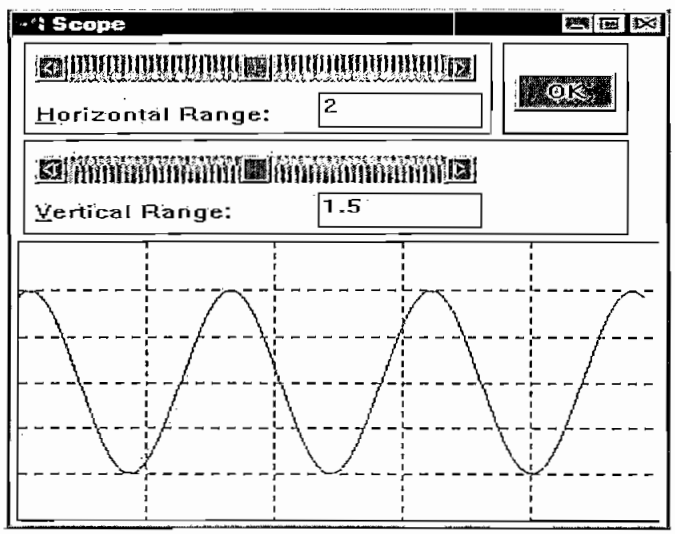



Figura A.3 Ejecución del programa

Al ejecutar el programa senoi.m creado mediante simulink, se puede observar la respuesta al hacer doble click en el osciloscopio.




A.4 Salida

Existen numerosos bloques y funciones incorporados en las librerías de simulink que pueden ser empleados para simular cualquier sistema.

### A.10.1 ACELERADOR DE SIMULINK

Para incrementar la velocidad de Simulink se debe instalar el acelerador "Accelerator". Este permite automáticamente generar una versión mejorada de los modelos los cuales correrán diez veces más rápido que el original. El acelerador puede ser usado sobre modelos continuos, discretos en el tiempo e híbridos.

El acelerador trabaja generando y compilando un código-C para un modelo dado. Una vez se completa la compilación, la simulación es ejecutada en la ventada de modelos de Simulink exactamente igual que antes sólo que más rápidamente. El propósito del acelerador es aumentar la velocidad de simulación. Si el programa MatLab tiene instalado el "Accelerator" podrá iniciarse la acción aceleradora seleccionando la opción simulation en el menú principal del Simulink y dentro de ésta seleccionando la opción Accelerate. Esta acción es totalmente transparente en el sentido de que el incremento de la velocidad se presenta sin ningún otro requerimiento por parte del usuario.

### A.10.2 GENERADOR DE CÓDIGO C EN SIMULINK

Una vez que se ha creado un modelo dinámico en Simulink, se puede invocar el generador de código-C que permite convertir el diagrama de bloques implementado en un código C. Este puede ser útil para varios propósitos: puede ser usado para control en tiempo real, simulación en tiempo real o simulación acelerada en tiempo no real. Sus aplicaciones pueden ser control de movimiento, control de procesos, sistemas automotores, equipos médicos, robótica, etc.

El código-C es tal que puede ser ejecutado en tiempo real. No requiere ser escrito manualmente por un programador pues es creado a nivel de diagramas de bloques en Simulink. El código generado puede correr sobre un amplio rango de hardware ubicado en estaciones de trabajo, PC o microprocesadores. Este código es la forma en la que puede usare el Simulink para adquisición de datos.

## A.11 VENTANAS

Como se muestra en la figura A.5, la interfase de usuario de MATLAB no es muy distinta a la de otras aplicaciones a las cuales estamos acostumbrados, pero la verdadera diferencia consiste en la utilidad que presta como aplicación para la investigación y el desarrollo de modelos matemáticos y estadísticos los cuales son tratados de forma interactiva, y con superposición de ventanas en un entorno de fácil comprensión e interpretación de los datos arrojados como resultados de los distintos rangos de cálculo que se pueden proporcionar a cada modelo de tal forma que se puede hacer estudios de comportamiento y tratar de determinar como se comportará una determinada variable a través de una experimentación virtual en tiempo real.

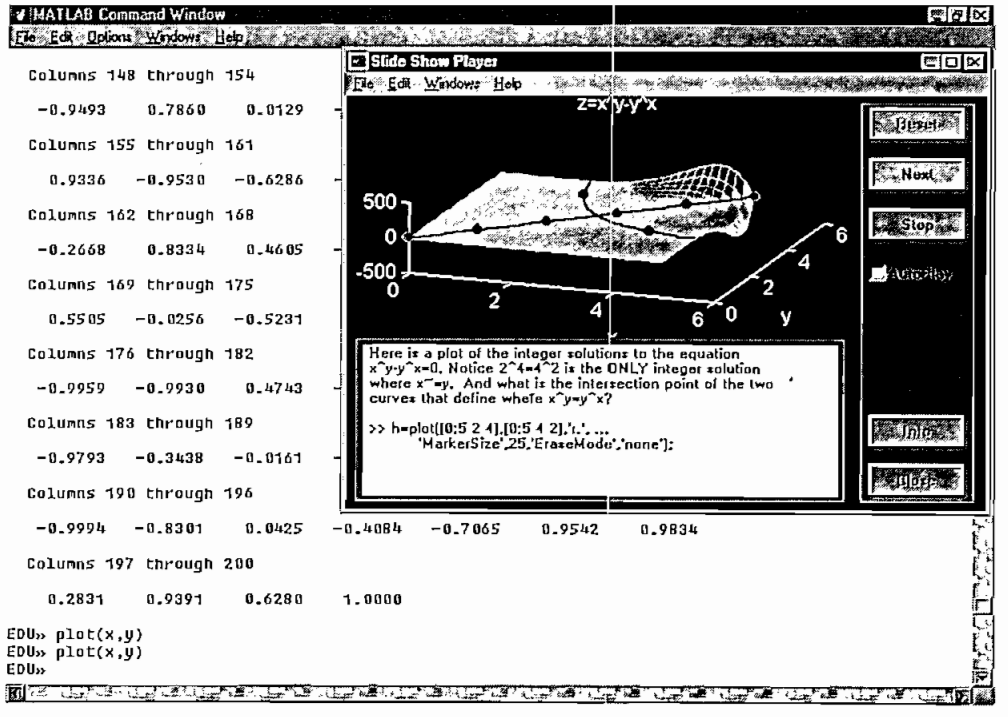



Figura A.5 Interfaz de usuario

Las ventanas de despliegue gráfico son muy similares, en éstas el énfasis de la presentación se pone en la gráfica generada y no en el entorno de trabajo, es por esta razón que puede parecer que el diseño de esta aplicación es escueto, pero se debe recordar que como todo este tipo de aplicaciones su desarrollo está orientado al logro de un objetivo específico como es el resolver modelos matemáticos.

## A.12 INICIANDO MATLAB

Después de ejecutar el programa MatLab desde el sistema operativo empleado, por ejemplo haciendo doble click sobre el icono de MatLab en ambientes Windows, aparece el indicador de comandos el cual está listo para recibir instrucciones en lenguaje MatLab. Este indicador es de la siguiente forma:

>>

Al iniciar el uso de MatLab están disponibles dos comandos de ayuda y demostración. Para ejecutarlos se escribe el comando en la línea de comandos después del símbolo >> y se presiona la tecla Enter. Por ejemplo:

>>help

permite obtener una ayuda sobre los diferentes comandos de MatLab.

>>demo

hace una demostración de las diferentes aplicaciones de MatLab.

Para cerrar o finalizar el uso de MatLab se usa el comando quit.

>>quit

### A.12.1 USO A NIVEL DE COMANDOS

La primera forma de interactuar con MatLab es a través de la línea de comandos. Puede ejecutarse un comando escribiendolo después del símbolo >> y presionando la tecla Enter.

Ya que MatLab se basa en el álgebra de matrices, como ejemplo crearemos una matriz. Estas pueden estar formadas por un sólo elementos (escalar), por una fila o una columna (vector) o por una serie de filas y columnas (matriz propiamente dicha).

>>A=1

define A como un escalar de valor 1. Al definir A automáticamente MatLab despliega en pantalla su valor.

A=

1

Para no desplegar el valor de la variable creada, debe agregarse punto y coma (;) al final del comando.

Luego de crear una variable, puede desplegarse su valor en pantalla escribiendo la variable después del prompt (>>).

```
>>A
```

Se pueden redefinir variables, por ejemplo:

```
>>A=[1   2   3 ]
```

define A como un vector de tres elementos, A(1)=1, A(2)=2 y A(3)=3. Estos elementos deben separase con espacios en blanco o comas (,). Para definir una matriz se deben separar las filas con punto y coma (;) o con retorno (Enter).

```
>>A=[1   2   3 ;  4   5   6]
```

o

```
>>A=[1      2      3

4       5      6]
```

ambos comandos producen el mismo efecto:

```
A =

1       2      3

3       5      6
```

El álgebra de matrices es posible mediante los operadores:

+ suma

- resta

* multiplicación

^ potencia

' transpuesta

\ división izquierda

/ división derecha

>>A=[1 2 3;4 5 6]; B=[6 5 4; 3 2 1]; define las matrices A y B. Para sumarlas se escribe la operación:
>>A+B
El resultado de la operación es por defecto almacenado en la variable ans, e inmediatamente desplegado en pantalla:

ans =

7 7 7

7 7 7

Para almacenar la suma de A y B en la variable C:
>>C=A+B
C =

7 7 7

7 7 7

A continuación se presentan los comandos de MatLab donde por ejemplo se encuentran numerosas operaciones de manipulación de matrices.

- min(A), max(A) - dan el mínimo y máximo respectivamente por columnas de A
- sum(A), prod(A) - producen la suma y producto respectivamente por columnas de A
- norm(A,p) - norma p de la matriz A donde p=1,2, ó inf
- eig(A) - vector cuyos componentes son los valores propios de A
- det(A) - el determinante de A
- inv(A) - la matriz inversa de A

### A.12.2 USO A NIVEL DE PROGRAMACIÓN

Programar en MatLab es usar una serie de comandos que permitan realizar una tarea o función específica. Estos pueden ser escritos uno por uno a través de la línea de comandos:

```
>>A=[1 2 3;4 5 6;7 8 9]
A =
1    2    3
4    5    6
7    8    9

>>A'
ans =
1    4    7
2    5    8
3    6    9
```

El primer comando A=[1 2 3;4 5 6;7 8 9] define la matriz A y el siguiente comando A' calcula y despliega en pantalla la transpuesta de A.

Es posible hacer una colección de comandos y agruparlos en un archivo de tipo texto y de extensión m (.m) llamado archivo-m. Estos archivos pueden ser scripts o funciones. El script es un archivo-m que contiene una serie de comandos que se ejecutarán al ejecutar dicho archivo en MatLab. La función, es un archivo-m que permite la entrada y salida de argumentos además de la ejecución de comandos.

Para crear un archivo-m se usa cualquier editor de textos, asegurándose de almacenar dicho archivo con la extensión (.m). MatLab incluye un editor de archivos-m que puede accesarse mediante la opción file del menú principal. Por ejemplo, el siguiente archivo ejemplo.m usa el comando for para crear el vector x.

```
% Ejemplo de un archivo-m
% Creación del vector x usando el comando for
```

```
n=5;
for i=1:n
x(i)=i^2;
end
x
% Fin del archivo-m
```

Este ejemplo es un archivo-m tipo script. El símbolo % permite hacer comentarios. Para ejecutarlo, en la línea de comandos se debe escribir el nombre del archivo:

```
>>ejemplo
x =
1    4    9    16    25
```

Para crear funciones, debe crearse el respectivo archivo-m, con el nombre con el que se desea llamar a la función. Por ejemplo, para crear una función llamada promedio debe guardarse en un archivo llamado promedio.m. Este archivo debe incluir la declaración de la función mediante function. Esta función calcula el promedio de los elementos de un vector y grafica dicho vector mediante el comando plot.

```
% Calcula el promedio de los elementos de un vector y dibuja dicho vector
% Sintaxis: promedio(x) donde x es el vector a promediar
function p = promedio(x)
n=length(x);
p=0;
for i=1:n
p=p+x(i);
end
p=p/n;
plot(x);
```

Para ejecutar la función, se hace el llamado en la línea de comandos incluyendo

el parámetro. La función promedio usa por parámetro un vector. Este vector debe ser definido previamente.

```
>>A=[1    2    4    3    7    5    6    1    2    0    8    5];
>>promedio(A)
ans =
3.6667
```

MatLab despliega las imágenes en una ventana de figuras. Al observar el contenido de dicha ventana luego de ejecutar la función promedio, se tiene:

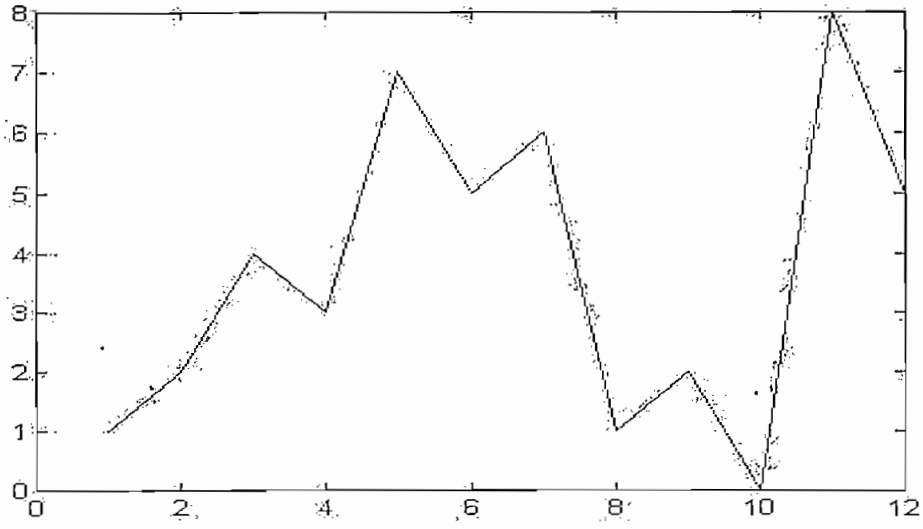



Figura A.6 Despliegue de resultados gráficos

Esta imagen es el resultado del comando plot(x) al ejecutar la función promedio.

MatLab poseé un conjunto de archivos-m incorporados (built-in). Puede agregársele archivos-m definidos por el usuario almacenando los mismos en el directorio principal de MatLab. Los comentarios incluidos en estos scripts y funciones se visualizan al usar el comando help seguido del nombre del archivo.

```
>>help promedio
```

Calcula el promedio de los elementos de un vector y dibuja dicho vector Sintaxis: promedio(x) donde x es el vector a promediar

Para ver el contenido de un archivo-m se usa el comando type seguido del nombre del archivo.

## A.13 FUNCIONES ESPECIALES

Lista parcial de funciones

### A.13.1 FUNCIONES MATEMÁTICAS

### A.13.2 FUNCIONALES ESPECIALES Y ELEMENTALES

- Funciones gamma, beta y elípticas.
- Transformación de sistemas de coordenadas.
- Matriz identidad y otras matrices elementales.
- Matrices de Hilbert, Toeplitz, Vandermonde, Hadamard, etc.
- Partes reales, imaginarias y complejas conjugadas.

### A.13.3 ÁLGEBRA LINEAL NUMÉRICA

- Valores propios y descomposición de matrices.
- Funciones generales de evaluación de matrices.
- Determinantes, normas, rangos, etc.
- Matrices inversas y factorización de matrices.
- Matriz exponencial, logarítmica y raíces cuadradas.

### A.13.4 POLINOMIOS E INTERPOLACIÓN

- Interpolación 1-D y 2-D.
- Construcción polinomial.
- Interpolación por splines cúbicos.
- Diferenciación de polinomios.
- Evaluación de polinomios.

- Multiplicación y división de polinomios.
- Residuos de polinomios y residuos.

### A.13.5 MÉTODOS NUMÉRICOS NO LINEALES

- Búsqueda de ceros en funciones de una única variable.
- Minimización de funciones de una o más variables.
- Resolución numérica de integrales.
- Solución numérica de ecuaciones diferenciales ordinarias.

### A.13.6 ESTADÍSTICA Y ANÁLISIS DE FOURIER

- Convolución 1-D y 2-D.
- Filtros digitales 1-D y 2-D.
- Transformadas de Fourier 1-D y 2-D y su inversa.
- Coeficientes de correlación y matrices de covarianza.
- Deconvolución.
- Magnitudes y ángulos de fase.
- Funciones max, min, sum, mean y otras funciones de estadística básica.

### A.13.7 OPERACIONES ALGEBRAICAS Y LÓGICAS

- Suma, resta, multiplicación, división y potencias de matrices.
- Matrix traspuesta.
- Operadores lógicos AND, OR, NOT y XOR.

### A.13.8 UTILIDADES

- Gestión y mantenimiento de errores.
- Conversión de tipos de datos Fortran.
- Funciones de fecha y hora.
- Clasificación de matrices.
- Conversión de números a cadenas y viceversa.

# ANEXO B

# B3 CREACIÓN DE INTERFACES GRÁFICAS

MATLAB permite desarrollar de manera simple un conjunto de pantallas con botones, menús, ventanas, etc., que permiten utilizar de manera muy simple programas realizados en el entorno ***Windows***. Este conjunto de herramientas se denomina ***interface de usuario***. Las posibilidades que ofrece MATLAB 5.3 han mejorado mucho respecto a versiones anteriores, aunque no son muy amplias en comparación con otras aplicaciones de ***Windows*** como ***Visual Basic***.

Para poder hacer programas que utilicen las capacidades gráficas avanzadas de MATLAB hay que conocer algunos conceptos que se explican en los apartados siguientes. Aunque MATLAB dispone ahora de la herramienta GUIDE, que permite generar interfaces de usuario de una forma muy cómoda y sencilla, es conveniente conocer los fundamentos de lo que se está haciendo, e incluso ser capaz de hacerlo programando si ayudas.

## B Estructura de los gráficos de MATLAB

Los gráficos de MATLAB tienen una estructura jerárquica formada por ***objetos*** de distintos ***tipos***. Esta jerarquía tiene forma de *árbol*, con el aspecto mostrado en la Figura B.1.

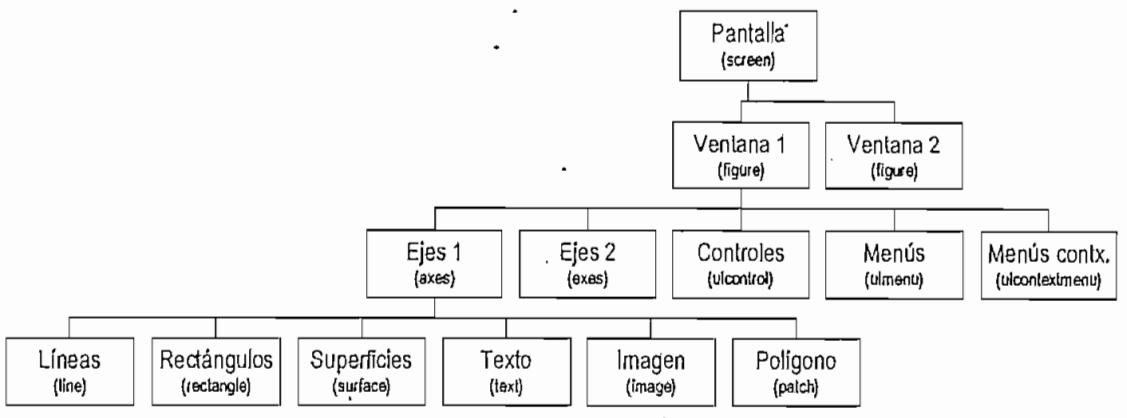



Figura B.1 Jerarquía gráfica de MATLAB.

## B1.1 OBJETOS GRÁFICOS DE MATLAB

Según se muestra en la Figura B.1, el objeto más general es la ***pantalla (screen***). Este objeto es la raíz de todos los demás y sólo puede haber un objeto pantalla. Una pantalla puede contener una o más ***ventanas (figures***). A su vez cada una de las ventanas puede tener uno o más ***ejes*** de coordenadas *(axes*) en los que se puede representar otros objetos de más bajo nivel. Una ventana puede tener también ***controles (uicontrol*** tales como *botones*, *barras de desplazamiento*, *botones de selección o de opción*, etc.) y ***menús (uimenu***). Finalmente, los ejes pueden contener los seis tipos de elementos gráficos que permite MATLAB: ***líneas (line***), ***rectángulos (rectangle***), ***polígonos (patches***), ***superficies (surface***), ***imágenes bitmap (image***) y ***texto (text***).

La jerarquía de objetos mostrada en la Figura 24 indica que en MATLAB hay ***objetos padres*** e ***hijos***. Por ejemplo, todos los objetos ***ventana*** son hijos de ***pantalla***, y cada ***ventana*** es padre de los objetos ***ejes***, ***controles*** o ***menús*** que están por debajo. A su vez los elementos gráficos (líneas, polígonos, etc.) son hijos de un objeto ***ejes***, y no tienen otros objetos que sean sus hijos.

Cuando se borra un objeto de MATLAB automáticamente se borran todos los objetos que son sus descendientes. Por ejemplo, al borrar unos ejes, se borran todas las líneas y polígonos que son hijos suyos.

## B1.2 IDENTIFICADORES *(HANDLES)*

Cada uno de los objetos de MATLAB tiene un ***identificador único (handle***). En este escrito a los identificadores se les llamará ***handle*** o ***id***, indistintamente. Algunos gráficos tienen muchos objetos, en cuyo caso tienen múltiples *handles*. El ***objeto raíz*** (pantalla) es siempre único y su identificador es el ***cero***. El identificador de las ventanas es un entero, que aparece en la barra de nombre de dicha ventana. Los identificadores de otros elementos gráficos son números *float*, que pueden ser obtenidos como valor de retorno y almacenados en variables de MATLAB.

MATLAB puede tener varias ventanas abiertas, pero siempre hay una y sólo una que es la ***ventana activa***. A su vez una ventana puede tener varios *ejes (axes*), pero sólo unos son los ***ejes activos***. MATLAB dibuja en los ejes activos de la ventana activa. Los identificadores de la ventana activa, de los ejes activos y del

objeto activo se pueden obtener respectivamente con los comandos ***gcf*** *(get current figure)*, ***gca*** *(get current axes)* y ***gco*** *(get current object)*:

gcf devuelve un entero, que es el ***handle*** de la ventana activa

gca devuelve el ***handle*** de los ejes activos

gco devuelve el ***handle*** del objeto activo

Los objetos se pueden borrar con el comando ***delete***:

delete(handle) borra el objeto correspondiente y todos sus hijos

MATLAB dispone de funciones gráficas de alto y bajo nivel. Son funciones de alto nivel las funciones ***plot***, ***plot3***, ***mesh***, ***surf***, ***fill***, ***fill3***, etc., utilizadas previamente. Cada una de estas funciones llama a una o más funciones de bajo nivel. Las funciones de bajo nivel crean cada uno de los 9 tipos de objetos disponibles y de ordinario tienen el nombre inglés del objeto correspondiente: ***figure***, ***axes***, ***uicontrol***, ***uimenu***, ***line***, ***rectangle***, ***patch***, ***surface***, ***image*** y ***text***.

Como ejemplo, ejecútense los siguientes comandos, observando la evolución de lo dibujado en la ventana y como MATLAB devuelve el ***id*** de cada objeto como valor de retorno:

```
» fig = figure
» li1 = line([0,5],[0,5])
» li2 = line([0,5],[5,0])
» po1 = patch([1,4,3],[1,1,4],'g')
» delete(po1)
» delete(li1)
```

Estos valores de retorno pueden ser almacenados en variables para un uso posterior. En ocasiones el ***id*** es un vector de valores, como por ejemplo al crear una superficie que consta de líneas y polígonos. Los comandos anteriores han abierto una ventana y dibujado sobre ella dos líneas cruzadas de color amarillo y un triángulo de color verde.

## B2 PROPIEDADES DE LOS OBJETOS

Todos los objetos de MATLAB tienen distintas ***propiedades***. Algunas de éstas son el *tipo*, el *estilo*, el *padre*, los *hijos*, si es *visible* o no, y otras propiedades

particulares del objeto concreto de que se trate. Algunas de las propiedades comunes a todos los objetos son: ***children***, ***clipping***, ***parent***, ***type***, ***UserData***, ***Visible*** y ***Tag***. Otras propiedades son propias de un tipo determinado de objeto.
Las propiedades tienen ***valores por defecto***, que se utilizan siempre que el usuario no indique otra cosa. Es posible cambiar las propiedades por defecto, y también devolverles su valor original (llamado ***factory***, por ser el valor por defecto con que salen de fábrica). El usuario puede consultar ***(query)*** los valores de las propiedades de cualquier objeto. Algunas propiedades pueden ser modificadas y otras no (son ***read only***). Hay propiedades que pueden tener cualquier valor y otras que sólo pueden tener un conjunto limitado de valores (por ejemplo, ***on*** y ***off***).

## *B2.1* FUNCIONES *SET()* Y *GET()*

MATLAB dispone de las funciones ***set*** y ***get*** para consultar y cambiar el valor de las propiedades de un objeto. Las funciones ***set(id)*** lista en pantalla todas las propiedades del objeto al que corresponde el *handle* (sólo los *nombres*, sin los valores de las propiedades). La función ***get(id)*** produce un listado de las propiedades y de sus *valores*. Como ejemplo, siendo ***li1*** el ***id*** de la primera línea dibujada anteriormente, la respuesta a ***set(li1)*** es:

**» set(li1)**

```
    Color
    EraseMode: [ {normal} | background | xor | none ]
    LineStyle: [ {-} | -- | : | -. | none ]
    LineWidth
    Marker: [ + | o | * | . | x | square | diamond | v | ^ |
    > | < | pentagram |
    hexagram | {none} ]
    MarkerSize
    MarkerEdgeColor: [ none | {auto} ] -or- a ColorSpec.
    MarkerFaceColor: [ {none} | auto ] -or- a ColorSpec.
    XData
    YData
    ZData
```

```
ButtonDownFcn
Children
Clipping: [ {on} | off ]
CreateFcn
DeleteFcn
BusyAction: [ {queue} | cancel ]
HandleVisibility: [ {on} | callback | off ]
HitTest: [ {on} | off ]
Interruptible: [ {on} | off ]
Parent
Selected: [ on | off ]
SelectionHighlight: [ {on} | off ]
Tag
UIContextMenu
UserData
Visible: [ {on} | off ]
```

que muestra las propiedades del objeto *línea*. Las propiedades que tienen un conjunto finito de valores presentan estos valores entre *corchetes* [ ] separados por barras verticales. La opción por defecto se muestra entre *llaves* { }. En general, el significado de cada propiedad es bastante evidente a partir de su nombre.

El comando ***get(id)*** devuelve las propiedades del objeto junto con sus valores:

**» get(li1)**

```
Color = [0 0 1]
EraseMode = normal
LineStyle = -
LineWidth = [0.5]
Marker = none
MarkerSize = [6]
MarkerEdgeColor = auto
MarkerFaceColor = none
XData = [0 5]
```

```
YData = [0 5]
ZData = []
ButtonDownFcn =
Children = []
Clipping = on
CreateFcn =
DeleteFcn =
BusyAction = queue
HandleVisibility = on
HitTest = on
Interruptible = on
Parent = [12.0001]
Selected = off
SelectionHighlight = on
Tag =
Type = line
UIContextMenu = []
UserData = []
Visible = on
```

Para conocer el valor de una propiedad particular de un objeto se utiliza la función ***get(id,'propiedad')***.

**» get(li1,'color')**

```
ans =
1 1 0
```

Las propiedades de un objeto pueden ser cambiadas o modificadas (salvo que sean *read-only*) con el comando ***set(id,'propiedad','valor')***. Por ejemplo, para cambiar el color de la segunda línea en el ejemplo anterior:

**» set(li2,'color','r')**

Es interesante hacer pruebas con los distintos tipos de objetos gráficos que se pueden crear y manipular con MATLAB. Por ejemplo, ejecútense los siguientes comandos observando cómo van cambiando el grosor de las líneas y los colores:

**» close(gcf)**

```
» fig=figure
» li1=line([0,5],[0,5])
» li2=line([0,5],[5,0],'color','w')
» po1=patch([1,4,3],[1,1,4],'g')
» pause(3)
» set(li1,'LineWidth',2), pause(1);
» set(li2,'LineWidth',2,'color','r'), pause(1);
» set(po1,'LineWidth',2,'EdgeColor','w','FaceColor','b')
```

El comando ***set*** permite cambiar varias propiedades a la vez, poniendo sus nombres entre apóstrofos seguidos de sus valores. Los ejemplos anteriores demuestran que es esencial disponer de los ***id*** si se desea modificar un gráfico o utilizar propiedades distintas de las de defecto.

Es posible también establecer las propiedades en el momento de la creación del objeto, como en el ejemplo siguiente que crea una figura con fondo blanco:

```
» fig = figure('color','w')
```

Se puede utilizar la propiedad ***type*** para saber qué tipo de objeto *(línea, polígono, texto*, ...) corresponde a un determinado *id*. Esto es especialmente útil cuando el ***id*** es un vector de valores correspondientes a objetos de distinto tipo.

```
» get(li2,'type')
     line
```

## B2.2 PROPIEDADES POR DEFECTO

Anteponiendo la palabra ***Default*** al nombre de un objeto y de una propiedad se puede acceder al valor por defecto de una propiedad, bien para consultar su valor, bien para modificarlo. Por ejemplo, ***DefaultLineColor*** representa el color por defecto de una *línea*, y ***DefaultFigureColor*** representa el *color de fondo* por defecto de las ventanas. Cambiando un valor por defecto a un determinado nivel de la jerarquía de objetos se cambia ese valor para todos los objetos que están por debajo y que se creen a partir de ese momento. Por ejemplo, el siguiente comando cambia el color de fondo de todas las ventanas (hijas de *screen*) que sean creadas a partir de ese momento:

```
» set(0,'DefaultFigureColor','w')
```

Cuando se crea un objeto se busca el valor por defecto de sus propiedades a su nivel, y si no se encuentra se sube en la jerarquía hasta que se encuentra un valor por defecto, y ese valor es el que se utiliza. Para devolver una propiedad a su valor original se utiliza el valor ***'factory'***, como por ejemplo:

**» set(id, 'FaceColor', 'factory')**

De forma análoga, el valor ***'remove'*** para una propiedad elimina un valor introducido previamente. Por ejemplo, para que el fondo de las ventanas deje de ser blanco se debe ejecutar el comando:

**» set(0,'DefaultFigureColor','remove')**

### B2.3 FUNCIONES DE UTILIDAD

MATLAB dispone de algunas funciones que permiten modificar las propiedades de algunos objetos de una forma más directa y sencilla que con las funciones ***get*** y ***set***. Algunas de éstas funciones, ya mencionadas al hablar de los gráficos 2-D y 3-D, son ***axis***, ***cla***, ***colormap*** y ***grid***.

Para obtener más información sobre estas funciones puede utilizarse el ***Help*** de MATLAB.

## B3 CREACIÓN DE CONTROLES GRÁFICOS: COMANDO *UICONTROL*

MATLAB permite desarrollar programas con el aspecto típico de las aplicaciones de ***Windows***. En este apartado se estudiará cómo crear algunos de los controles más utilizados. Para todos los controles, se utilizará la función ***uicontrol***, que permite desarrollar dichos controles. La forma general del comando ***uicontrol*** es la siguiente:

```
id_control = uicontrol(id_parent,...
'Propiedad1',valor1,...
'Propiedad2',valor2,...
... (otras propiedades)
'callback','sentencias')
```

Las propiedades son las opciones del comando, que se explican en el apartado siguiente. Éstas tendrán comillas simples (') a su izquierda y derecha, e irán seguidas de los parámetros necesarios.

En caso de que el conjunto de propiedades de un control exceda una línea de código, es posible continuar en la línea siguiente, poniendo tres puntos seguidos (...).

El comando ***uicontrol*** permite definir los controles gráficos de MATLAB descritos en los siguientes apartados. Estos controles reciben las acciones de los usuarios, que se denominan ***eventos*** (por ejemplo, clicar en un botón, cambiar el valor de una barra de desplazamiento, ...). A continuación se explican algunas de las propiedades de ***uicontrol***.

### B3.1 COLOR DEL OBJETO *(BACKGROUNDCOLOR)*

Controla el color del objeto. Por defecto éste suele ser gris claro, aunque puede tomar distintos valores: ***green, red, white,*** etc. Éstos irán definidos entre comillas (por ejemplo '***green***') o con un vector de tres elementos que indican las componentes RGB *(Red, Green, Blue)*.

### B3.2 ACCIÓN A EFECTUAR POR EL COMANDO *(CALLBACK)*

Este comando especifica la ***acción*** a efectuar por MATLAB al actuar sobre el control. Se trata de una expresión u orden, almacenada en una cadena de caracteres o en una función, que se ejecutará al activar el control. Esta instrucción es equivalente a ejecutar la función o a realizar ***eval****('expresión')*.

Algunos controles tienen distintos tipos de callback según el evento que reciban del usuario.

### B3.3 CONTROL ACTIVADO/DESACTIVADO *(ENABLE)*

Este comando permite desactivar un control, de tal forma que una acción sobre el mismo (por ejemplo, apretar sobre el mismo con el ratón) no produce ningún efecto. Los valores que puede tomar esta variable son ***on*** u ***off***. Cuando un control está desactivado suele mostrar un aspecto especial (una tonalidad más gris, por ejemplo) que indica esta situación.

### B3.4 ALINEAMIENTO HORIZONTAL DEL TÍTULO *(HORIZONTALALIGNMENT)*

Esta opción determina la posición del título del control en el mismo (propiedad ***String***). Los valores que puede tomar son: ***left***, ***center*** o ***right***. En los botones y en otros controles la opción por defecto es ***center***.

### B3.5 VALOR MÁXIMO *(MAX)*

Esta opción determina el máximo valor que puede tomar la propiedad ***Value*** cuando se utilizan cajas de textos, botones de opción o barras de desplazamiento. En caso de botones tipo ***on/off***, que solamente admiten las dos posiciones de abierto y cerrado, esta variable corresponde al valor del mismo cuando está activado.

### B3.6 VALOR MÍNIMO *(MIN)*

Análogo a la opción anterior para el valor mínimo.

### B3.7 IDENTIFICADOR DEL OBJETO PADRE *(PARENT)*

Esta opción especifica el ***id*** del objeto *padre*. Debe ir siempre en primer lugar.

### B3.8 POSICIÓN DEL OBJETO *(POSITION)*

En esta opción se determina la posición y el tamaño del control dentro del objeto *padre*. Para ello se define un vector de cuatro elementos, cuyos valores siguen el siguiente orden: *[izquierda*, *abajo*, *anchura*, *altura]*. Aquí *izquierda* y *abajo* son la distancia a la esquina inferior izquierda de la figura y *anchura* y *altura* las dimensiones del control. Por defecto se mide en ***pixels***, aunque con la propiedad ***Units*** las unidades se pueden cambiar.

### B3.9 NOMBRE DEL OBJETO *(STRING)*

Esta opción define el nombre que aparecerá en el control. Cuando el control sea una opción de menú desplegable ***(popup menu)***, los nombres se sitúan en orden dentro del *String*, separados por un carácter barra vertical (|).

### B3.10 TIPO DE CONTROL (*STYLE*)

Esta opción puede tomar los siguientes valores: ***pushbutton, togglebuttons***, ***radiobutton, checkbox, slider, edit, popupmenu, list, frames*** y ***text,*** según el tipo de control que se desee (Como se verá en los ejemplos siguientes, pueden usarse nombres abreviados: así, ***pushbutton*** puede abreviarse en ***push***).

### B3.11 UNIDADES *(UNITS)*

Unidades de medida en las que se interpretará el comando ***Position***. Los valores que puede tomar ***Units*** son: ***pixels*** (puntos de pantalla), ***normalized*** (coordenadas de 0 a 1), ***inches*** (pulgadas), ***cent*** (centímetros), ***points*** (equivalente a 1/72 parte de una pulgada). La opción por defecto es ***pixels***.

### B3.12 VALOR *(VALUE)*

Permite utilizar el valor que tiene del control en un momento dado. Por ejemplo, un botón de chequeo ***(checkbutton)*** puede tener dos tipos de valores, definidos por las variables ***Max*** y ***Min***.

Según sea uno u otro el programa ejecutará una acción. La variable ***Value*** permite controlar dicho valor. Esta propiedad es especialmente importante en las barras de desplazamiento ***(sliders)***, pues son controles especialmente diseñados para que el usuario introduzca valores. También en los controles ***list*** y ***popupmenu***, en donde el valor representa el número de orden que en la lista de opciones ocupa el elemento elegido por el usuario.

### B3.13 VISIBLE *(VISIBLE)*

Puede tomar dos valores: ***on/off***. Indica si el control es visible o no en la ventana. Este comando es similar al ***Enable***, si bien con ***Visible*** en ***off***, además de quedar inhabilitado, el control desaparece de la pantalla.

## B4 TIPOS DE *UICONTROL*

Existen ocho tipos de controles diferentes. La utilización de cada uno vendrá dada en función de sus características y aplicación.

## B4.1 BOTONES *(PUSHBUTTONS)*

La Figura B.2 muestra el aspecto típico de un botón. Al clicar sobre él con el ratón se producirá un evento que lanza una acción que deberá ser ejecutada por MATLAB.

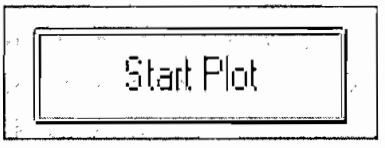



Figura B.2. Botón ***(push button)***.

**Ejemplo**: La siguiente instrucción dibujará en la figura activa de MATLAB un botón como el de la Figura B.2, que tiene como nombre ***Start Plot***. Al pulsarlo se ejecutarán las instrucciones contenidas en el fichero ***mifunc***.m:

```
pbstart = uicontrol(gcf,...
'Style','push',...
'Position',[10 10 100 25],...
'String','Start Plot',...
'CallBack','mifunc');
```

donde es necesario crear un fichero llamado ***mifunc.m,*** que será ejecutado al pulsar sobre el botón. En lugar de este fichero también puede ponerse una cadena de caracteres conteniendo distintos comandos, que será evaluada como si se tratase de un fichero *.m. Como ejemplo se puede crear un fichero ***mifunc.m*** que contenga sólo la sentencia:

```
disp('Ha pulsado en Start Plot')
```

## B4.2 BOTONES DE SELECCIÓN *(CHECK BOXES)*

Los botones de selección permiten al usuario seleccionar entre dos opciones (Figura B.3). Los botones de selección actúan como interruptores, indicando un estado ***on*** (si el botón está activado) u ***off*** (si el botón está desactivado). El valor

de ambos estados viene definido por las opciones ***Max*** y ***Mi***n, respectivamente. Los botones de selección deben ser independientes unos de otros.

**Ejemplo**: El siguiente conjunto de instrucciones crea una caja con dos opciones, ambas permiten el control de los ejes, de manera independiente, dentro de la función ***plo***t:

```
% Definir un texto fijo como título para los botones de selección
txt_axes = uicontrol(gcf,...
'Style','text',...
'Units','normalized','Position',[0.4 0.55 0.25
0.1],...
'String','Set Axes Properties');
% Definir la checkbox para la propiedad Box de los
ejes
cb_box = uicontrol(gcf,...
'Style','checkbox',...
'Units','normalized','Position',[0.4 0.475 0.25
0.1],...
'String','Box=on',...
'CallBack',['set(gca,''Box'',''off''),',...
'if get(cb_box,''value'')==1,',...
'set(gca,''Box'',''on''),',...
'end']);
% Definir la checkbox para la propiedad TickDir de los ejes
cb_tdir = uicontrol(gcf,...
'Style','checkbox',...
'Units','normalized','Position',[0.4 0.4 0.25 0.1],...
'String','TickDir=out',...
'CallBack',['set(gca,''TickDir'',''in''),',...
'if get(cb_tdir,''value'')==1,',...
'set(gca,''TickDir'',''out''),',...
'end']);
```

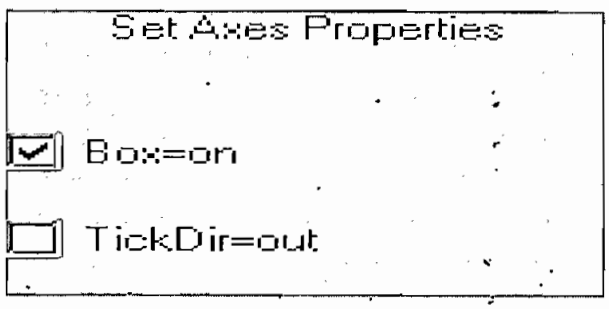



Figura B.3 Botones de selección ***(Check Boxes)***.

## B4.3 BOTONES DE OPCIÓN *(RADIO BUTTONS)*

Al igual que los botones de selección, los botones de opción permiten al usuario seleccionar entre varias posibilidades. La diferencia fundamental reside en que en los botones de opción, las opciones son excluyentes, es decir, no puede haber más de uno activado, mientras que el control anterior permite tener una o más cajas activadas.

**Ejemplo**: El siguiente ejemplo corresponde a los controles de la Figura B.4. Estos botones permiten cambiar de dirección los indicadores de los ejes: ***In*** para orientarlos hacia dentro de la figura, o ***Out*** para que se sitúen en el exterior de la gráfica.


Figura B.4. Botones de opción ***(Radio Buttons)***.

```
% Definir el texto de título para este grupo de
controles
txt_tdir = uicontrol(gcf,...
'Style','text', 'String','Seleccionar posición
marcas',...
'Position',[200 200 150 20]);
% Definir la propiedad TickDir In con radiobutton
(defecto)
td_in = uicontrol(gcf,...
'Style','radio', 'String','In',...
'Position',[200 17  150 25],...
'Value',1,...
'CallBack',[...
```

```
'set(td_in,''Value'',1),'...
'set(td_out,''Value'',0),'...
'set(gca,''TickDir'',''in''),']);
% Definir la propiedad TickDir Out con radiobutton
td_out = uicontrol(gcf,...
'Style','radio', 'String','Out',...
'Position',[200 150 150 25],...
'CallBack',[...
'set(td_out,''Value'',1),'...
'set(td_in,''Value'',0),'...
'set(gca,''TickDir'',''out''),']);
```

## B4.4 BARRAS DE DESPLAZAMIENTO *(SCROLLING BARS O SLIDERS)*

Las barras de desplazamiento (ver Figura B.5) permiten al usuario introducir un valor entre un rango de valores. El usuario puede cambiar el valor clicando sobre la barra, clicando en las flechas laterales o bien arrastrando el elemento central con el ratón.

**Ejemplo**: El siguiente ejemplo muestra como se utilizan las barras de desplazamiento para mover un sistema de referencia espacial:

```
% Obtener un id de la ventana activa y borrar su contenido
fig = gcf; clf
% Los callbacks definen la propiedad View de los ejes
% a partir de os valores de las barras de desplaz. (Value property)
% y escriben sus valores en los controles text
%
% Definir la barra para el ángulo de azimut
sli_azm = uicontrol(fig,'Style','slider','Position',[50 50 120
20],...
'Min',-90,'Max',90,'Value',30,...
'CallBack',[...
'set(azm_cur,''String'','...
'num2str(get(sli_azm,''Val'')),',...
'set(gca,''View'',',...
'[get(sli_azm,''Val''),get(sli_elv,''Val'')]),']);
% Definir la barra para el ángulo de elevación
```

```
sli_elv = uicontrol(fig,...
'Style','slider',...
'Position',[240 50 120 20],...
'Min',-90,'Max',90,'Value',30,...
'CallBack',[...
'set(elv_cur,''String'',',...
'num2str(get(sli_elv,''Val'')),',...
'set(gca,''View'',',...
'[get(sli_azm,''Val''),get(sli_elv,''Val'')])']);
% Definir los controles de texto para los valores mínimos
azm_min = uicontrol(fig,...
'Style','text',...
'Pos',[20 50 30 20],...
'String',num2str(get(sli_azm,'Min')));
elv_min = uicontrol(fig,...
'Style','text',...
'Pos',[210 50 30 20],...
'String',num2str(get(sli_elv,'Min')));
% Definir los controles de texto para los valores máximos
azm_max = uicontrol(fig,...
'Style','text',...
'Pos',[170 50 30 20],...
'String',num2str(get(sli_azm,'Max')),...
'Horiz','right');
elv_max = uicontrol(fig,...
'Style','text',...
'Pos',[360 50 30 20],...
'String',num2str(get(sli_elv,'Max')),...
'Horiz','right');
% Definir las etiquetas de las barras
azm_label = uicontrol(fig,...
'Style','text',...
'Pos',[50 80 65 20],...
'String','Azimuth');
elv_label = uicontrol(fig,...
'Style','text',...
'Pos',[240 80 65 20],...
'String','Elevación');
% Definir los controles de texto para los valores actuales
```

```
% Los valores son inicializados aquí y son modificados por los
callbacks
% de las barras cuando el usuario cambia sus valores
azm_cur = uicontrol(fig,...
'Style','text',...
'Pos',[120 80 50 20],...
'String',num2str(get(sli_azm,'Value')));
elv_cur = uicontrol(fig,...
'Style','text',...
'Pos',[310 80 50 20],...
'String',num2str(get(sli_elv,'Value')));
```

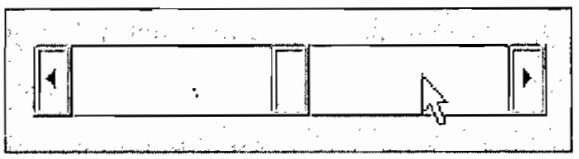


Figura B.5. Barra de desplazamiento ***(slider)***.

## B4.5 CAJAS DE SELECCIÓN DESPLEGABLES *(POP-UP MENUS)*

Las cajas de selección desplegables permiten elegir una opción entre varias mostradas en una lista. Eligiendo una de las opciones, MATLAB realizará la opción elegida. Según la Figura B.6, el menú se despliega pulsando sobre la flecha de la derecha. La opción sobre la que pase el ratón (en este caso ***re***d) aparecerá de otro color.

**Ejemplo**: El siguiente código abre una ventana con un determinado tamaño y posición y permite cambiar su color de fondo con un menú pop-up (lista desplegable, que aparece desde el primer momento en la pantalla):

```
close all;
pantalla = get(0,'ScreenSize'); xw=pantalla(3);
yw=pantalla(4);
fig=figure('Position',[xw/4 yw/4 xw/2 yw/2])
% Definir el menú pop-up
popcol = uicontrol(fig,...
'Style','popup',...
'String','red|blue|green|yellow',...
```

```
'Position',[xw/16 yw/16 xw/12 yw/32],...
'CallBack',['cb_col = [''R'',''B'',''G'',''Y''];',...
'set(gcf,''Color'',cb_col(get(popcol,''Value'')))']);
```

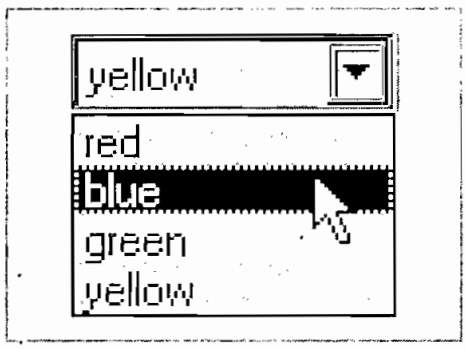



Figura B.6. Menú desplegable *(**Pop-up menu**)*.

## B4.6 CAJAS DE TEXTO *(STATIC TEXTBOXES)*

Son controles especiales, ya que no permite realizar ninguna operación con el ratón. Permiten escribir un texto en la pantalla. Una aplicación de este ***uicontrol*** aparece en la variable ***txt_tdir*** del ejemplo de los ***radiobuttons***. En este caso particular el texto indica la función de los botones de opción.

## B4.7 CAJAS DE TEXTO EDITABLES *(EDITABLE TEXTBOXES)*

Las cajas de texto se utilizan para introducir y modificar cadenas de caracteres (ver Figura B.7). Puede tener una o más líneas, y la llamada a la opción de ejecución ***Callback*** será efectiva una vez que se pulsen las teclas ***Control-Return*** o se clique con el ratón fuera del control.

**Ejemplo**: El siguiente ejemplo escribe un texto en la ventana de MATLAB que el usuario puede modificar.

```
fig=gcf; clf;
edmulti = uicontrol(fig,...
'Style','edit',...
'BackgroundColor','white',...
'FontSize',14,'FontName','Arial',...
'String','Cambie este texto',...
```

```
'Position',[40 200 150 50],...
'Max',2,...
'CallBack','get(edmulti,''String'')'...
);
```

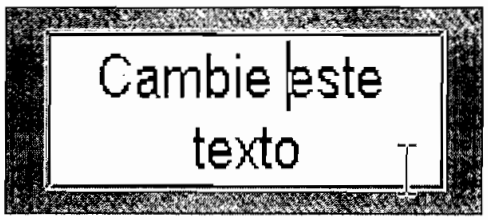



Figura B.7. Texto editable (***text***).

## B4.8 MARCOS *(FRAMES)*

Un marco tampoco es un control propiamente dicho. Su función es la de englobar una serie de opciones (botones, cajas de texto, etc....) con el fin de mantener una estructura ordenada de controles, separando unos de otros en función de las características del programa y del gusto del programador. En la Figura B.8 pueden observarse dos marcos, en los que se sitúan dos botones.

**Ejemplo**: El ejemplo dibuja un marco y dos botones, según se muestra en la Figura B.8. Al pulsar el botón se imprime un mensaje en la ventana de MATLAB.

```
fig=gcf; clf;
ft_dir = uicontrol(gcf,...
'Style','frame',....
'Position',[30 30 120 85]);
pbstart = uicontrol(gcf,...
'Style','push',...
'Position',[40 40 100 25],...
'String','Start Plot',...
'CallBack',['disp(''Start Plot'')']);
pbstart = uicontrol(gcf,...
'Style','push',...
```

```
'Position',[40 80 100 25],...
'String','Stop Plot',...
'CallBack','disp(''Stop Plot'')');
```

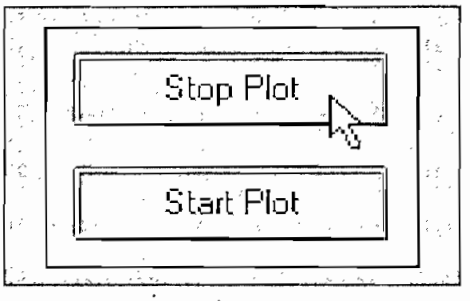



Figura B.8. Marcos *(**Frames**)*.

## B5 CREACIÓN DE MENÚS

En MATLAB se pueden construir aplicaciones basadas en ventanas, con menús definidos por el usuario. En este apartado se describe cómo crear diferentes menús y submenús. Para crear los menús se utiliza la función ***uimenu***. El aspecto general del comando ***uimenu*** es el siguiente:

```
id_menu = uimenu(id_parent,...
'Propiedad1', 'valor1',...
'Propiedad2', 'valor2',...
'Propiedad3', 'valor3',...
'Otras Propiedades', 'Otros valores');
```

Normalmente una de las propiedades suele ser la propiedad ***CallBack***, aunque no siempre es necesario, ya que algunos menús no ejecutan ningún comando de MATLAB, sino que se encargan de desplegar nuevos submenús. Por lo tanto no es imprescindible definirla en todos los menús.

Para generar submenús basta con crear un menú nuevo, donde el ***id_parent*** sea el identificador del menú padre.

En la Figura B.9 se muestra un detalle de los menús que componen una aplicación en particular (en este caso ***Microsoft Word 97***).

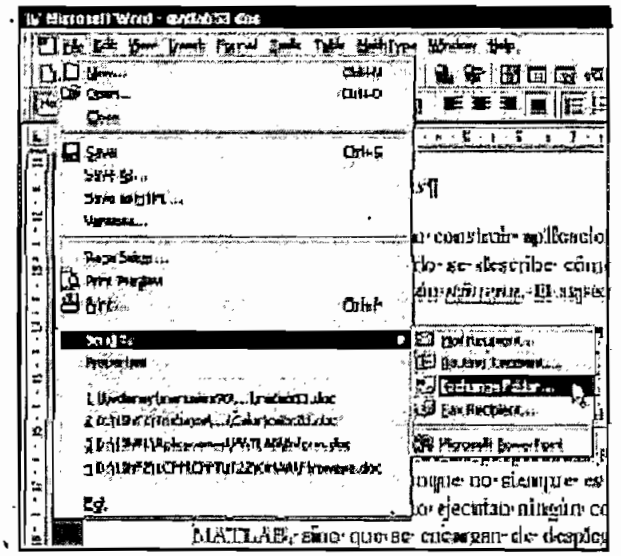


Figura B.9 Detalle de los menús de ***Word 9***7

## B6 DESCRIPCIÓN DE LAS PROPIEDADES DE LOS MENÚS

### B6.1 ACELERADOR *(ACCELERATOR)*

Esta propiedad es opcional y permite definir un carácter con el que activar el menú desde el teclado, sin necesidad de utilizar el ratón. Para activar el menú hay que teclear simultáneamente ALT + el carácter elegido.

```
uimenu(id_parent,...
'Accelerator','carácter',...
);
```

### B6.2 ACCIÓN A EFECTUAR POR EL MENÚ *(CALLBACK)*

Esta propiedad es muy importante, ya que determina la acción a realizar por MATLAB al actuar sobre el menú correspondiente.

```
uimenu(id_parent,...
'CallBack','string',...
```

```
);
```

### B6.3 CREACIÓN DE SUBMENUS *(CHILDREN)*

Permite crear submenús a partir de menús creados previamente.

```
uimenu(id_parent,...
'Children',vector de handles,...
);
```

### B6.4 MENÚ ACTIVADO/DESACTIVADO *(ENABLE)*

Esta propiedad permite modificar la posibilidad de acceso al menú correspondiente. A veces resulta interesante que un determinado menú o submenú esté inactivo, porque su acción no tenga sentido en algún momento de la ejecución del programa, (por ejemplo, ***Word*** no permite guardar un fichero, si no hay ninguno abierto). Los menús desactivados se muestran en color gris.

```
uimenu(id_parent,...
'Enable','on'/'off',...
);
```

### B6.5 NOMBRE DEL MENÚ *(LABEL)*

Esta propiedad establece el texto correspondiente del menú. Esta opción se puede combinar con la del acelerador, con el fin de señalar al usuario del programa, qué tecla debe pulsar para que se ejecute el comando. Así, anteponiendo el carácter (&) a la letra que se desee, ésta aparece subrayada. Esto no significa que este método sustituya al acelerador, solamente es una indicación para el usuario, que le indica la tecla con la que se activa el menú.

```
uimenu(id_parent,...
'Label','string',...
);
```

### B6.6 CONTROL DEL OBJETO PADRE *(PARENT)*

Esta opción especifica el ***id*** del objeto padre. Debe ir siempre en primer lugar. El padre de un ***uimenu*** es siempre una **figura** u otro ***uimenu***.

```
uimenu(id_parent,...
'Parent',handle,...
);
```

### B6.7 POSICIÓN DEL MENÚ *(POSITION)*

Esta propiedad se define mediante un escalar, que determina la posición relativa del menú. Un orden creciente en los menús, empezando a contar por uno, equivale a situarlos de izquierda a derecha en la barra de menús. En los submenús ese orden creciente significa que se colocan de arriba hacia abajo.

```
uimenu(id_parent,...
'Position',scalar,...
);
```

### B6.8 SEPARADOR *(SEPARATOR)*

Esta propiedad permite colocar una línea de separación encima del menú al que se aplica. Cuando se pone a ***on*** se traza la línea y cuando está en ***off*** no se coloca. Por defecto esta propiedad está en ***off***.

```
uimenu(id_parent,...
'Separator', 'on'/'off',...
);
```

### B6.9 VISIBLE *(VISIBLE)*

Puede tomar dos valores: ***on/off***. Indica si el menú es visible en la ventana o no.

```
uimenu(id_parent,...
'Visible', 'on'/'off',...
);
```

## B7 EJEMPLO DE UTILIZACIÓN DEL COMANDO *UIMENU*

A continuación se muestra un ejemplo de utilización de los menús de usuario. El ejemplo de la Figura B.10 consta de tres menús (***File***, ***Time*** y ***Curve***) en la barra de menús. El menú ***File*** incluye tres submenús, ***Load***, ***Save*** y ***Exit***. El submenú ***Load*** permite cargar el fichero de variables ***data.mat***. El submenú ***Save*** salva las variables actuales en el fichero ***data.mat***, y el submenú ***Exit***, detiene la ejecución del programa. El menú ***Time*** permite seleccionar entre tres límites para la variable t, que es un vector. El menú ***Curve*** despliega a su vez dos submenús (***Sinus*** y ***Cosinus***), que trazan las gráficas de seno y coseno respectivamente, en función del tiempo seleccionado en el menú ***Time***. Después de elegir el tiempo hay que volver a elegir la curva que se desea visualizar. En la Figura B.10 se muestra un aspecto general del programa.

Para realizar esta aplicación, se crea en primer lugar la ventana del programa. A continuación se muestra cómo se debe crear una ventana sin ningún elemento estándar en la barra de menús. Los menús se irán incluyendo posteriormente.

```
% Crear una figura sin barra de menús
id_Fig = figure('Units','normalized',...
'Numbertitle','off',...
'Name','Ejemplo de uimenus',...
'menubar','none');
% Creación de los diferentes menús
% Menú File
id_File = uimenu(id_Fig,'Label','&File',...
'Accelerator','f');
% Menú Time
id_Time = uimenu(id_Fig,'Label','&Time',...
'Accelerator','t');
% Menú Curve
id_Curve = uimenu(id_Fig,'Label','&Curve',...
'Accelerator','c');
% Creación de los diferentes submenús
id_Load = uimenu(id_File,'Label','&Load',...
'Accelerator','L',...
```

```
'CallBack','load data.mat');
id_Save = uimenu(id_File,'Label','&Save',...
'Accelerator','s',...
'CallBack','save data.mat');
id_Exit = uimenu(id_File,'Label','E&xit',...
'Accelerator','x',...
'CallBack','close');
% Time
id_10 = uimenu(id_Time,'Label','10',...
'CallBack','t=0:.1:10;');
id_20 = uimenu(id_Time,'Label','20',...
'CallBack','t=0:.1:20;');
id_30 = uimenu(id_Time,'Label','30',...
'CallBack','t=0:.1:30;');
% Curve
id_Sinus = uimenu(id_Curve,'Label','Sinus',...
'CallBack','plot(t,sin(t));grid;title(''Sinus'')');
id_Cosinus = uimenu(id_Curve,'Label','Cosinus',...
'CallBack','plot(t,cos(t));grid;title(''Cosinus'')',..
'Separator','on');
```

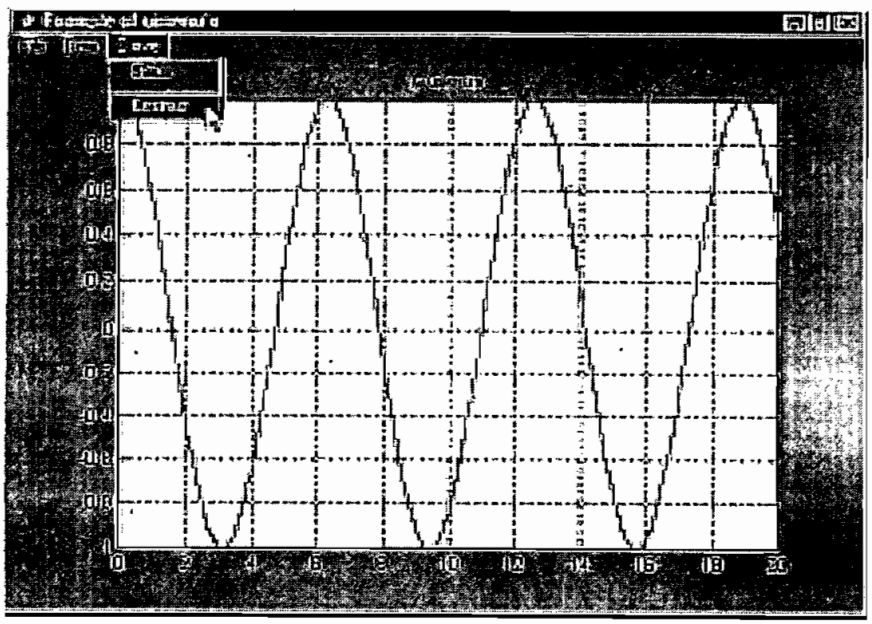


Figura B.10 Ejemplo de menús programados por el usuario

## B8 MENÚS CONTEXTUALES *(UICONTEXTMEN*U)

Los ***menús contextuales*** son menús que se abren cuando el usuario realiza una determinada acción (normalmente clicar con el botón derecho) sobre un objeto de la figura. Los menús contextuales se crean también con la función ***uimen***u, poniendo en ***on*** la propiedad ***UIContextMenu*** del objeto al que se quiere unir el menú contextual. para más información consultar el ***Help*** de MATLAB.

## B9 CONSTRUCCIÓN INTERACTIVA DE INTERFACES DE USUARIO (GUIDE)

MATLAB, a partir de la versión 5.0, ha incorporado un módulo llamado GUIDE ***(Graphical User Interface Development Environmen***t) que permite crear de modo interactivo la interfase de usuario, al modo de ***Visual Basi***c, aunque todavía con unas posibilidades mucho más limitadas. En cualquier caso, si no es uno de los avances más importantes de MATLAB, si ha sido uno de los más agradecidos por los usuarios, que ya no tienen que escribir sin ayuda los ***callbacks*** .En la Figura B.11 se puede ver la interfase de usuario de un programa de análisis de estructuras reticulares planas, desarrollada con GUIDE en unos pocos minutos (la interfase de usuario, se entiende; el programa completo precisa lógicamente de más tiempo).

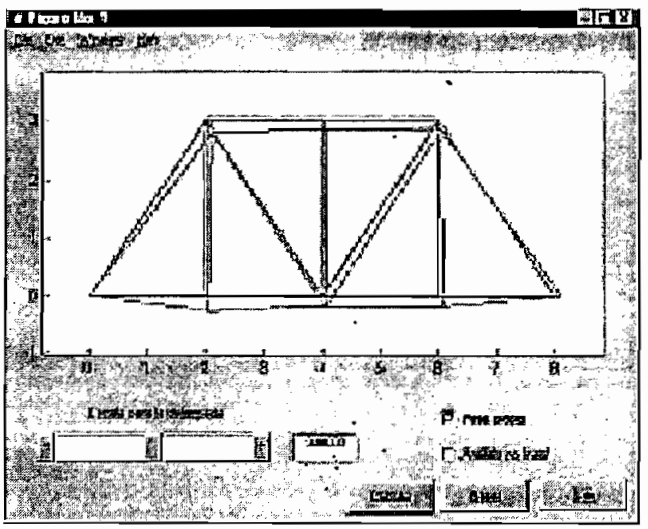


Figura B.11. Interfase de usuario creada con GUIDE

## B9.1 Guide Control Panel

Como cualquier otro programa, GUIDE se ejecuta a partir de la línea de comandos de MATLAB, tecleando:

```
» guide
```

y pulsando ***Intro***. A continuación se abre la ventana ***Guide Control Panel (GCP)***, mostrada en la Figura B.12. Se abre también una figura en blanco, sobre la que el diseñador deberá ir situando los distintos controles con el ratón, hasta terminar con el aspecto requerido, similar al de la aplicación mostrada en la Figura B.11. La ventana ***GCP*** ocupa el papel central de la generación de los controles y menús de la interface de usuario. Dicha ventana contiene tres partes principales, dispuestas una debajo de otra. La ***parte superior*** contiene cuatro grandes botones o iconos, correspondientes a los otros cuatro grandes módulos de GUIDE. De izquierda a derecha aparecen los iconos correspondientes a:

- el *Editor de Propiedades* (***Property Editor***),
- el *Editor de Llamadas* (***Callback Editor***),
- el *Editor de Alineamientos* (***Alignement Editor***) y
- el *Editor de Menús* (***Menu Editor***).

En lo sucesivo se hará referencia a estos módulos con la nomenclatura inglesa. Estos editores se pueden hacer visibles desde la línea de comandos de MATLAB (***propedit***, ***cbedit***, ***align*** y ***menuedit***), clicando en dichos iconos o seleccionándolos en el menú ***Tools*** de la ventana en la que se está haciendo el diseño (deberá estar en estado ***Controlled***).

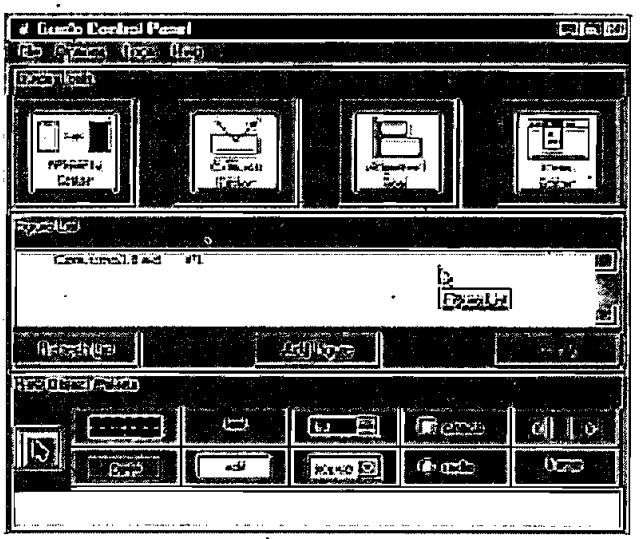


Figura B.12. Ventana ***Guide Control Panel***

La ***parte central*** de la ***GCP*** contiene la lista de ventanas o figuras de MATLAB controladas por GUIDE. En este caso sólo hay una, pero podría haber varias. Cada una de las figuras pueden estar en dos estados: *controlada* ***(Controlled)*** y *activa* ***(Active)***. Estos dos estados se corresponden con losmodos "de diseño" y "de ejecución" de otras aplicaciones. Para que una figura pase de ***Controlled*** a ***Active*** hay que realizar dos acciones: 1.- Clicar sobre la línea correspondiente en la lista de figuras, con lo cual el mensaje en dicha lista pasa de un estado a otro, y, 2.-Clicar sobre el botón ***Apply***, con lo cual el estado seleccionado pasa a ser el estado real de la figura. Junto al botón ***Apply*** aparece otro botón llamado ***Add Figure*** que permite crear una nueva figura cuando se desee.

La ***parte inferior*** de la ***GCP*** contiene iconos correspondientes a los elementos de interface de usuario (o controles) soportados por MATLAB 5.3, que son los siguientes (de izquierda a derecha y de arriba a abajo): ***axes***, ***text***, ***listbox***, ***checkbox***, ***slider***, ***pushbutton***, ***edit***, ***popupmenu***, ***radiobutton*** y ***frame***. Para crear uno de estos controles basta clicar sobre el icono correspondiente y luego ir a la figura en que se desee introducirlo (que deberá estar en estado ***Controlled***), clicar y arrastrar con el botón izquierdo del ratón pulsado para dar al control la posición y tamaño deseado. El nuevo control puede desplazarse y cambiarse de tamaño con ayuda del ratón, al igual que la propia ventana en la que ha sido situado Se puede observar que falta el icono correspondiente a ***togglebutto***n. Para crear un botón de este tipo se crea un ***pushbutton*** y se cambia la propiedad ***Style*** a ***togglebutton***.

## B9.2 El Editor de Propiedades *(Property Editor)*

En un momento dado, sólo puede estar abierto un ***Property Editor (PE)***. Este editor tiene el aspecto mostrado en la Figura B.13. Nótese que para que tenga este aspecto, los dos ***checkboxes (Show Objecty Browser y Show Property List)*** tienen que estar activados. En la parte superior del ***PE*** aparece una lista jerarquizada de los controles presentes en la figura ***(Object Browser)***. En este caso todos los controles son "hijos" de la figura nº 1, lo cual se muestra en el hecho de que todos aparezcan algo desplazados hacia la derecha. Para cada elemento se muestra el tipo (propiedad ***Style***) y un nombre ***(EditText1*** para el elemento seleccionado). Este nombre se define por medio de la propiedad ***Tag***.

Los nombres utilizados en este caso son los nombres por defecto propuestos por GUIDE.

En la parte central del ***PE*** aparece una lista o relación de propiedades del objeto seleccionado en la parte superior. Las propiedades no se pueden editar directamente sobre esta lista: hay que seleccionar una propiedad y darle valor en la caja de texto que aparece inmediatamente encima de la lista, a la derecha (a la izquierda aparece el nombre de la propiedad). Para las propiedades que sólo pueden tomar ciertos valores, aparece una lista desplegable que los muestra y ayuda a elegirlos. En el caso de la Figura B.13, como se trata de colores, aparece un botón con tres puntos, que da paso a un cuadro de diálogo en el que se puede elegir el color deseado en una paleta de colores. Para seleccionar una propiedad basta teclear sus primeras letras en la caja de texto que contiene el nombre y pulsar ***Intro***. El ***PE*** se encarga de buscar la propiedad deseada.

Es posible seleccionar varios objetos y establecer sus propiedades conjuntamente (las que tenga sentido hacerlo, como por ejemplo el color, el tamaño, etc.).

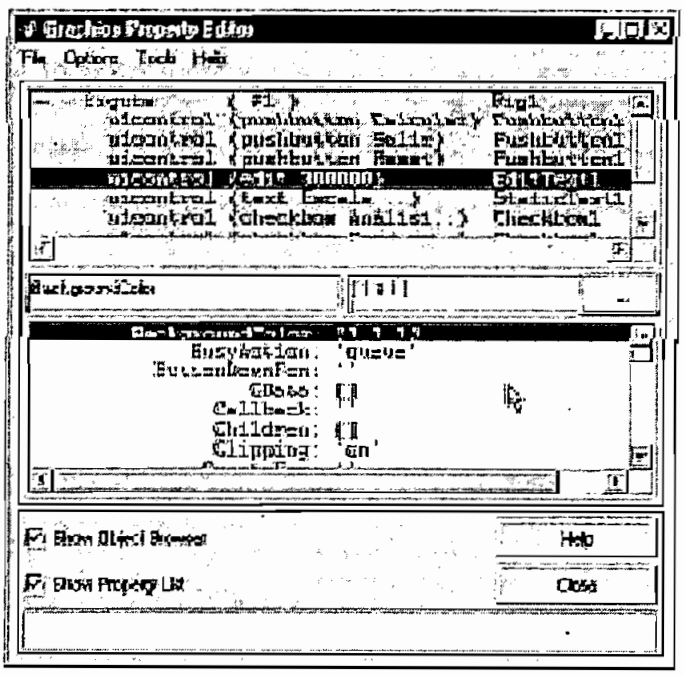



Figura B.13 El Editor de Propiedades ***(Property Editor)***.

Es muy instructivo ver con calma las propiedades de un determinado objeto. Algunas pueden ser muy poco significativas en la mayor parte de los casos y otras se utilizan con mucha frecuencia:

Entre las que casi siempre son las más importantes se pueden citar las siguientes:

**String** El texto mostrado en casi todos los controles (botones, botones de opción o selección, cajas de texto, listas de selección, menús pop-up)

**Label** Propiedad de ***uimenu*** que especifica el texto que aparece en el menú. Se puede utilizar el carácter **&** para especificar la tecla aceleradora, que aparecerá subrayada en el menú de la aplicación.

**Tag** Un nombre interno para el objeto. No lo ve el usuario, pero se utiliza mucho en programación para localizar un determinado objeto. GUIDE asigna un ***Tag*** por defecto a cada objeto que se crea. Este nombre puede respetarse o ser sustituido por otro elegido por el usuario.

**Style** La clase de objeto de que se trata *(pushbutton | togglebutton | radiobutton | checkbox | edit | text | slider | frame | listbox | popupmenu)*.

**Position** Vector de cuatro elementos que indican la posición y tamaño del control *[left, bottom, width, height]*. Recuérdese que el origen está en la esquina inferior izquierda. Las unidades se expresan mediante la propiedad ***Units***.

**Extent** Vector de cuatro elementos que indica el tamaño del ***String*** de un objeto o elemento (etiqueta o texto mostrado).

**Units** Unidades en que se miden las dimensiones: ***normalized*** miden desde (0,0) a (1.0,1.0). También ***pixel***s, ***inch***es, ***centimeter***s, y ***points*** (1/72 de una pulgada), que son unidades absolutas.

**BackgroundColor** Vector de tres números entre 0 y 1.0 que indican las componentes RGB del color de fondo de un objeto.

**ForegroundColor** Vector de tres números entre 0 y 1.0 que indican las componentes RGB del color del texto de un objeto.

**Parent** El ***handle*** de la figura o control padre.

**Children** Los ***handle*** de los controles hijos.

**Enable** Si el control está activo o no, es decir si el usuario puede o no actuar sobre dicho control.

**Visible** Si la figura o el control es visible o no.

**FontName** El tipo de letra que se desea utilizar: 'Times New Roman', 'Arial', 'Courier New', etc.

**FontSize** El tamaño de la letra en puntos *(points)*.

**FontWeight** Uno de estos valores: 'normal', 'ligth', 'demi' y bold'

**UserData** Cualquier dato que el usuario quiera asociar con el control. No es utilizado por MATLAB. Los valores de ***UserData*** se pueden escribir con ***set()*** y leer con ***get()***.

**Value** Valor asociado con algunos controles: posición del cursor en la barra de desplazamiento, valor de la propiedad ***max*** cuando están en ***on*** y ***min*** cuando están en ***off*** en los ***checkbuttons*** y ***radiobuttons***; número ordinal (empezando por 1) del elemento seleccionado en las ***listbox*** y ***popupmenu***. Las cajas de texto, los botones y los frames no tienen esta propiedad.

Para obtener ayuda sobre las propiedades de un objeto se puede proceder del siguiente modo.

A partir del ***Help*** de MATLAB se abre el ***Matlab Help Desk*** en ***Netscape Navigator*** o ***Internet Explorer***; a continuación se clica en el enlace ***Handle Graphics Objects*** y se abre una página con la jerarquía de objetos gráficos (ver Figura 24). Los elementos de esta jerarquía contienen ***enlaces*** a páginas de ayuda sobre cada uno de sus objetos ***(figure***, ***axes***, ***uicontrol***, etc.***)***. Cada una de esas páginas contiene información sobre las propiedades de cada elemento.

Algunos controles son un poco especiales. A continuación se hacen algunas consideraciones sobre ellos:

- Los controles ***listbox*** y ***porpupmenu*** contienen un conjunto de opciones. El texto de esas opciones se pueden almacenar en un ***cell array*** del espacio de trabajo base y luego la variable se introduce en la propiedad ***String*** del control (crea una copia llamada ***towork***).
- Para las barras de desplazamiento ***(slider***s) los valores máximos y mínimos vienen dados por las propiedades ***max*** y ***min***. Los incrementos pequeño y grande vienen definidos por la propiedad ***SliderStep***, que es un vector de dos elementos.

### B9.3 El Editor de Llamadas *(Callback Editor)*

El ***Callback Editor (CE)*** es uno de los componentes más importantes de GUIDE, porque permite definir la forma en la que los distintos controles responden a las

acciones del usuario *(eventos)*. La ventana del ***CE*** se puede ver en la Figura B.14.

La respuesta que se desea obtener cuando el usuario realiza una determinada acción sobre un control se obtiene programando los ***callbacks***, que definen el código que se debe ejecutar en ese caso.

Este código se asocia con el control por medio del ***C***E, en cuya parte superior puede hacerse aparecer una lista de objetos ***(Object Browse***r). Seleccionando un objeto, en la caja de texto editable que aparece en el centro puede escribirse el código que se desea ejecutar al producirse el evento correspondiente. El evento se puede seleccionar en la lista desplegable que hay debajo del ***Object Browse***r. Este código puede escribirse de dos maneras:

- Por medio de una cadena de caracteres que contiene los comandos de MATLAB (no es necesario poner las comillas simples externas porque el ***CE*** se ocupa de ello.

- Por medio de una llamada a una función que realice todas las operaciones necesarias. Este segundo método es claramente preferible al anterior, por dos motivos: 1) las cadenas de caracteres se interpretan cada vez que se ejecutan y esto es más lento que ejecutar una función, que se compila la primera vez que es llamada y se mantiene en memoria. 2) las funciones mantienen su propio espacio de trabajo, diferente del espacio de trabajo base y por tanto sin posibilidad de chocar o interferir con nombres de variables que estuvieran definidos en dicho espacio de trabajo base.

El ***CE*** permite definir ***callbacks*** correspondientes a eventos más especializados. Estos callbacks aparecen en el menú o lista desplegable que se ve en la Figura B14, encima de la caja de texto a la izquierda.

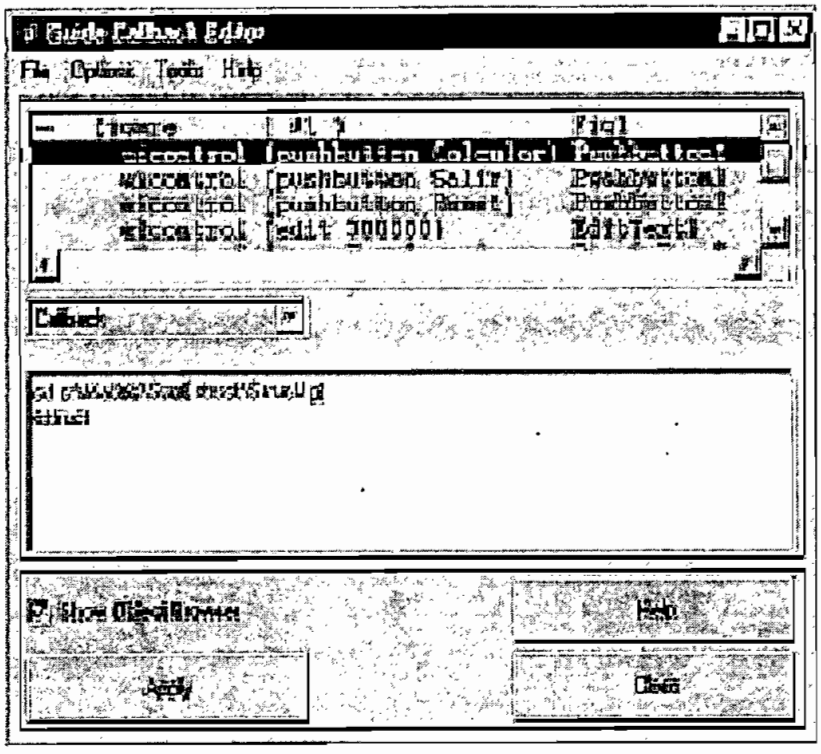


Figura B.14. El ***Callback Editor***

## B9.4 El Editor de Alineamientos *(Alignment Editor)*

El ***Alignment Editor (AE)*** es una herramienta auxiliar que permite que los controles situados en una ventana por medio de GUIDE, aparezcan uniformemente alineados o distribuidos. Su funcionamiento es bastante sencillo e intuitivo, por lo que no se darán muchos detalles aquí. La ventana del ***AE*** se muestra en la Figura B.15. En la parte superior se muestra de nuevo la lista de objetos ***(Object Browse***r), en la que deberán estar seleccionados los objetos que se desea alinear o distribuir (para seleccionar varios objetos basta clicar sucesivamente sobre ellos manteniendo pulsada la tecla de ***Mayúsculas***). Cuando se quiere alinear o distribuir varios objetos hay que tener en cuenta que el ***AE*** considera una caja "imaginaria" que comprende los objetos a ser alineados o distribuidos, y que realiza su operación de alineamiento o distribución dentro de esa caja.

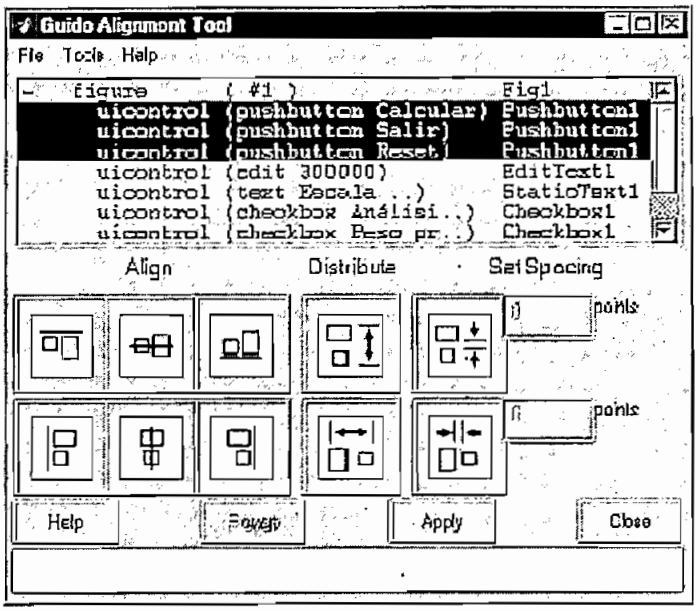



Figura B.15 El ***Alignement Editor***.

## B9.5 El Editor de Menús *(Menu Editor)*

El ***Menu Editor (ME)*** es el último componente de GUIDE. Su ventana aparece en la Figura B.16. Hay que señalar que en toda ventana de MATLAB ***(figure)*** aparecen por defecto cuatro menús ***(File***, ***Edit***, ***Windows*** y ***Help***). Los menús creados con el ***Menu Editor*** son menús adicionales. MATLAB permite hasta 4 niveles de menús. En la parte superior del ***Menu Editor*** aparece una representación jerárquica de los menús. En este caso se ha introducido un menú ***Color***, con dos sub-menús, ***Deformada*** y ***Estructura***. Cada uno de ellos tiene dos colores opcionales.

A la izquierda de la lista de menús aparecen cuatro botones que permiten mover los menús por la jerarquía, aumentando o reduciendo su nivel en dicha jerarquía *(promote* y *demote*), cambiando su orden o pasándolos de un menú a otro.

Debajo de la jerarquía de menús del ***ME*** se muestran existen tres cajas de texto que permiten fijar el ***Label*** (lo que aparece como título del menú), el ***Tag*** (el nombre interno del menú) y el ***callback*** (la función o los comandos que se ejecutarán al elegir ese menú).

Para introducir un nuevo menú en la jerarquía basta pulsar el botón ***New Menu*** y establecer los datos que se deseen, situándolo con los botones en la posición deseada del menú deseado.

Pulsando ***Apply*** los datos de las cajas de texto se introducen en la jerarquía visible en la parte superior de la ventana.

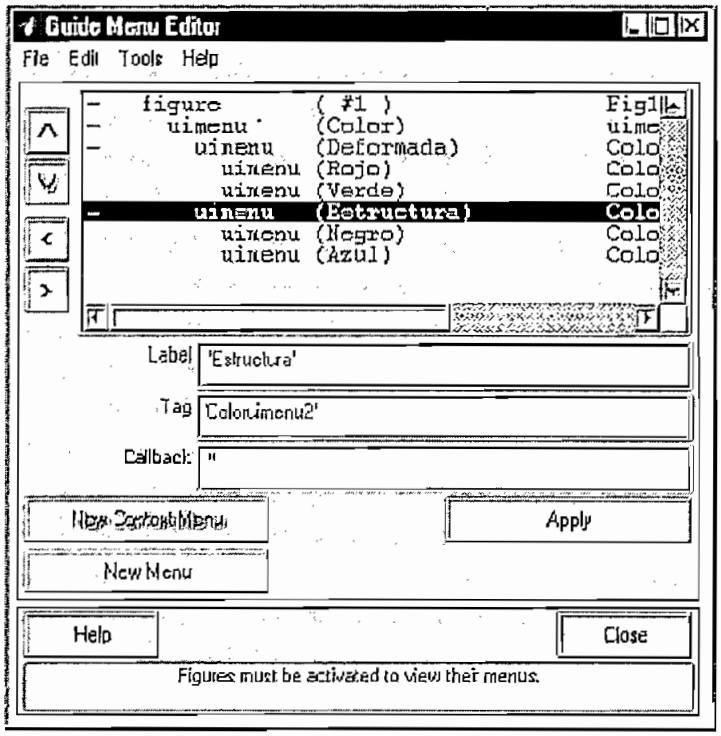



Figura B.16. El ***Menu Editor***.

## B9.6 PROGRAMACIÓN DE *CALLBACKS*

### B9.6.1 ALGUNAS FUNCIONES ÚTILES

Las siguientes funciones de MATLAB son muy útiles para la programación de interfaces de usuario gráficas interactivas:

- gcf *(get current figure)* devuelve el ***handle*** a la figura activa
- gca *(get current axes)* devuelve el ***handle*** de los ejes activos
- gcbo *(get callback object)* devuelve el ***handle*** del objeto sobre el que se ha producido el evento al que está tratando de responder el ***callback***
- gcbf *(get callback figure)* devuelve el ***handle*** de la figura sobre la que se ha producido el evento
- findobj(gcbf, 'prop', 'propvalue') devuelve el ***handle*** de un objeto de la figura activa cuya propiedad ***prop*** tiene el valor ***propvalue***. Los valores admitidos para la propiedad son los mismos que admite la función ***set***

- get(handle, 'prop') devuelve el valor de la propiedad ***prop*** en el objeto cuyo ***handle*** se pasa como primer argumento set(handle, 'prop', 'propvalue') establece el valor ***propvalue*** en la propiedad ***prop*** del objeto cuyo ***handle*** se pasa como primer argumento

Recuérdese que el ***handle*** es un número que identifica a cualquier objeto o elemento de la jerarquía gráfica de MATLAB. El ***handle*** se obtiene como valor de retorno en el momento de la creación del objeto y puede ser guardado en una variable para hacer referencia posteriormente a ese objeto. También puede ser hallado más tarde con alguna de las funciones vistas en este apartado, que devuelven el ***handle*** del objeto que cumple una determinada condición.

Hay que tener en cuenta que el ***handle*** no es un valor conocido de antemano (como puede ser por ejemplo la propiedad ***Tag*** de un objeto), sino que cambia cada vez que se ejecuta el programa.

Por eso no se puede introducir en el código como constante numérica o alfanumérica, sino que hay que obtenerlo de algún modo en cada ejecución.

Los ***handles*** son el método utilizado por MATLAB para hacer referencia a los objetos de la interface gráfica de usuario y esto condiciona mucho el tipo de programación. En MATLAB no se puede utilizar la terminología típica ***objeto.propiedad*** de otros entornos de programación.

### B9.6.2 ALGUNAS TÉCNICAS DE PROGRAMACIÓN

Ya se ha comentado que es preferible gestionar los ***callbacks*** por medio de funciones que por medio de cadenas de caracteres que contengan comandos de MATLAB. Ésta es una de las primeras ideas que hay que tener en cuenta. Lo mismo se puede decir respecto a los ficheros de comandos ****.m*** que no son funciones: siempre será preferible utilizar funciones.

Recuérdese que las funciones de MATLAB tienen su propio espacio de trabajo y que la información necesaria para realizar su tarea les llega por uno de los siguientes caminos:

- por medio de los ***argumentos*** con los que se llama a la función,
- a través de ***variables globales*** (técnica que conviene evitar en la medida de lo posible),

- es también posible que la propia función se las arregle para encontrar de alguna manera la información que necesita, bien buscándola en algún recurso (un fichero, una variable accesible, ...) al que se tenga acceso, bien llamando a otras funciones que se la proporcionen.

En los ***callback*** de MATLAB hay que recurrir con mucha frecuencia a la tercera posibilidad: las funciones tienen que buscar la información que necesitan, de ordinario por medio de llamadas a las funciones vistas en el apartado anterior (***gcbo***, ***gcbf*** y ***findobj***).

Se hace referencia a las componentes u objetos de MATLAB por medio del ***handle***. Por ejemplo, para dibujar sobre unos ejes, para cambiar el texto que aparece sobre un botón, hace falta el ***handle*** del objeto sobre el que se quiere actuar. De forma análoga, para saber sobre qué objeto se ha producido el evento que da origen a un ***callback***, hay que localizar el ***handle*** de dicho objeto. Los ***handle*** se pueden determinar de las siguientes formas:

- Para obtener el ***handle*** de la figura en la que se ha producido un evento se utiliza la función ***gcbf***
- Para obtener el ***handle*** del objeto en el que se ha producido un evento se utiliza la función ***gcbo***
- Para obtener el ***handle*** de un objeto en el que se desea cambiar alguna propiedad se utiliza la función ***findobj(gcbf, 'prop', 'propvalue')***.
- MATLAB permite distintos tipos de evento sobre su jerarquía de objetos. La acción callback es la que está más directamente relacionada con la función del control (Clicar en un ***pushbutton***, cambiar el estado de un ***checkbutton*** o un ***radiobutton***, mover el cursor de una ***slider***, etc.). Hay otros eventos que disparan otras funciones ***callback***, tales como ***CreateFcn*** y ***DeleteFcn*** (que se ejecutan cuando se crea o destruye un objeto), ***WindowsButtonDownFcn***, ***WindowsButtonMotionFcn*** y ***WindowsButtonUpFcn*** (que se ejecutan cuando se pulsa, mueve o suelta un botón del ratón sobre una ventana) o ***ButtonDownFcn*** y ***KeyPressFcn*** (cuando se pulsa un botón del ratón o una tecla del teclado). Para más información mirar las propiedades de ***Figure***, ***Axes*** y ***Uicontrol*** con el ***Help Desk*** (HTML).

Es conveniente agrupar la gestión de los ***callbacks*** de varios componentes, objetos o controles en una misma función. Esta función puede calcular o recibir como argumento el objeto en el que se ha producido el evento, y luego con una sentencia ***switch*** (o con varios ***if... ifelse*** encadenados) ejecutar el código apropiado para el objeto sobre el que ha actuado el usuario y para la acción que éste ha realizado. Puede ser muy útil construir estas funciones utilizando ***sub-funciones***, es decir funciones definidas en el mismo fichero ****.m*** y que no pueden ser llamadas desde fuera de dicho fichero.

Para pasar valores de unas funciones a otras (o para compartir valores entre varias funciones de la interface de usuario) no es posible utilizar los argumentos. Se podría utilizar variables globales, pero ésta es una técnica considerada como peligrosa. Una solución es guardar dichos valores en la propiedad ***UserData*** que tienen las figuras y los controles de MATLAB. Se pueden almacenar valores con la función ***set()*** y recuperarlos con la función ***get()***.

# ANEXO C

## MANUAL DE USUARIO DE LA INTERFAZ GRÁFICA

La interfaz gráfica diseñada es utilizada para visualizar la cancelación de eco señales de voz, presentando los resultados en el dominio del tiempo y en el dominio de la frecuencia. Para que se pueda visualizar en una manera rápida los resultados se debe disponer de una computadora personal con el hardware y software adecuado entre los cuales se puede mencionar:

- Procesador 660 Mhz (mínimo)
- 64 Mb en RAM
- Multimedia (tarjeta de sonido, video, altavoces)
- Software (Matlab versión 5.3 con tolbox de telecomunicaciones)

Una vez instalado Matlab versión 5.3, se copia todos los archivos que se tiene en el cd del programa de la interfaz gráfica en la siguiente carpeta:

C:\MATLABR11\WORK

Luego se digita la palabra **inicio**, con la cual se accede a la pantalla de presentación la cual se indica el la figura C.1

Aquí se observa las siguientes regiones:

1. datos del autor de la interfaz
2. botones de selección
    a. continuar
    b. salir

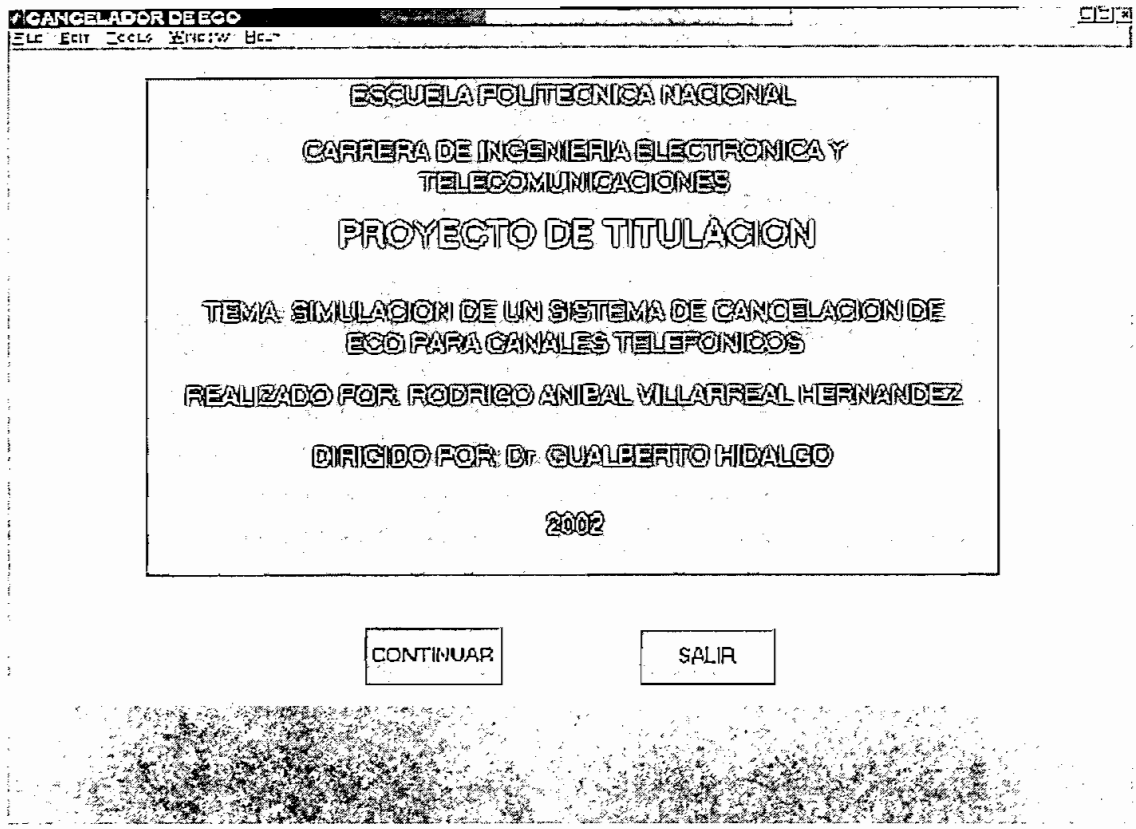



Figura C.1 Pantalla de inicio

Seguidamente dando un clic en continuar se tiene la pantalla principal figura C.2 donde se tiene las siguiente regiones:

1. Menú personalizado
    a. Archivo con eco
    b. Archivo de referencia
    c. Gráficos
    d. Ejemplo de análisis
2. Área de resultados
3. Radio botones para sonido
    a. señal con eco
    b. señal procesada
4. Botones de pulsación
    a. Salir
    b. Reiniciar

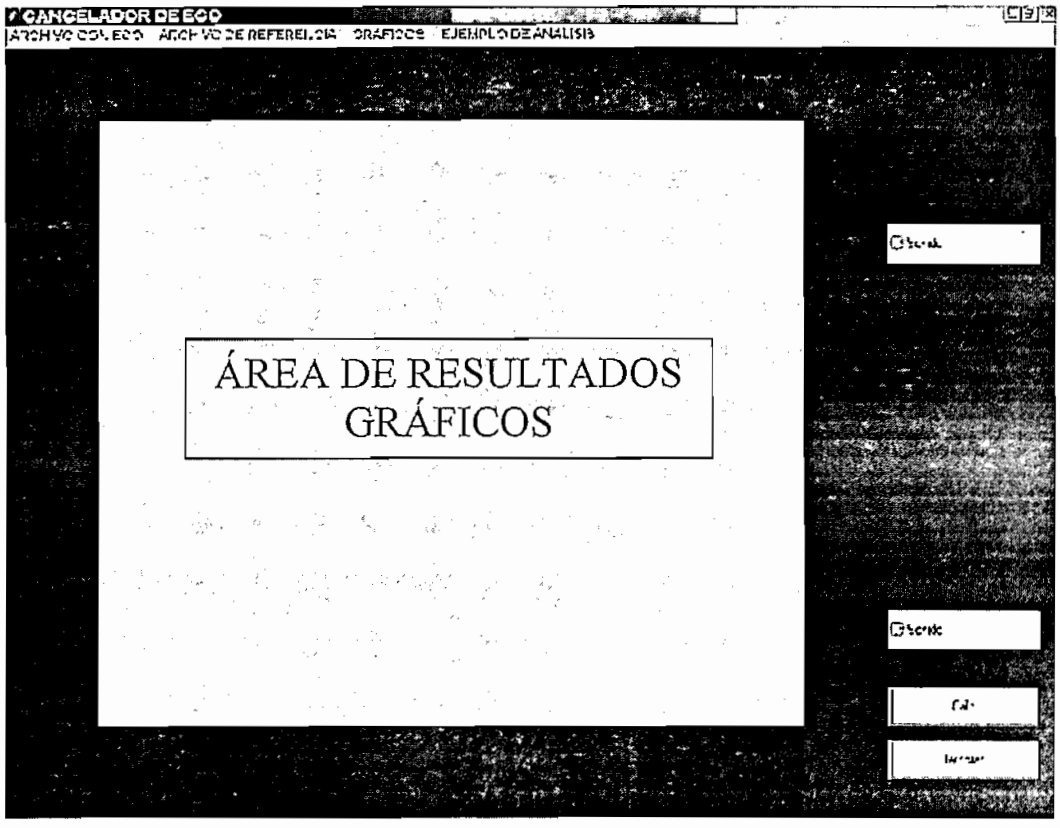



Figura C.2 Pantalla principal

Para iniciar en si el módulo del programa primeramente se debe seguir el siguiente orden:

1. **Archivo con eco,** se selecciona el archivo a procesar y se visualiza dicha señal
2. **Archivo de referencia,** según el archivo seleccionado anteriormente, se selecciona la señal de referencia, esta señal no se visualiza y empieza el procesamiento de la señal con eco. Una vez terminado el procesamiento se visualiza la señal procesada.

El resultado de los pasos anteriores se presenta en la figura C.3.

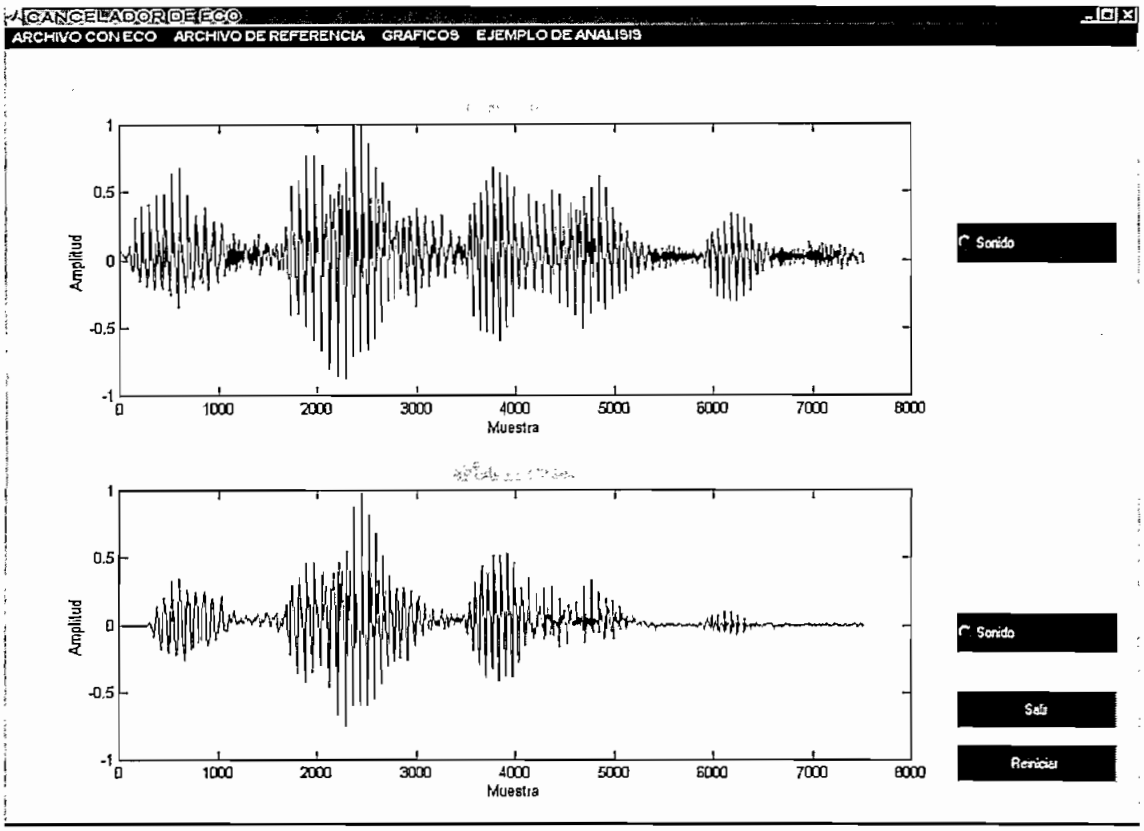



Figura C.3 Presentación de la señal con eco y la señal procesada

Una vez que se tiene la señal procesada se procede con las opciones de análisis que se implementa en esta simulación, las cuales están en la opción **Gráficos** en el menú personalizado, en el cual se tiene las siguientes opciones (figura C.4):

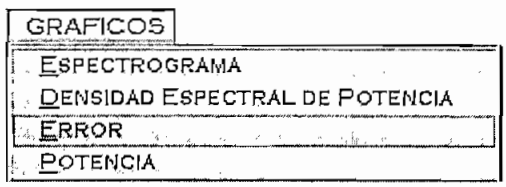



Figura C.4 Opciones del menú gráfico

**Espectrograma**: se presenta la gráfica en dos dimensiones entre la frecuencia y el tiempo, en el cual las componentes de mayor frecuencia se diferencian por colores más intensos. También se presenta una grafica en tres dimensiones de

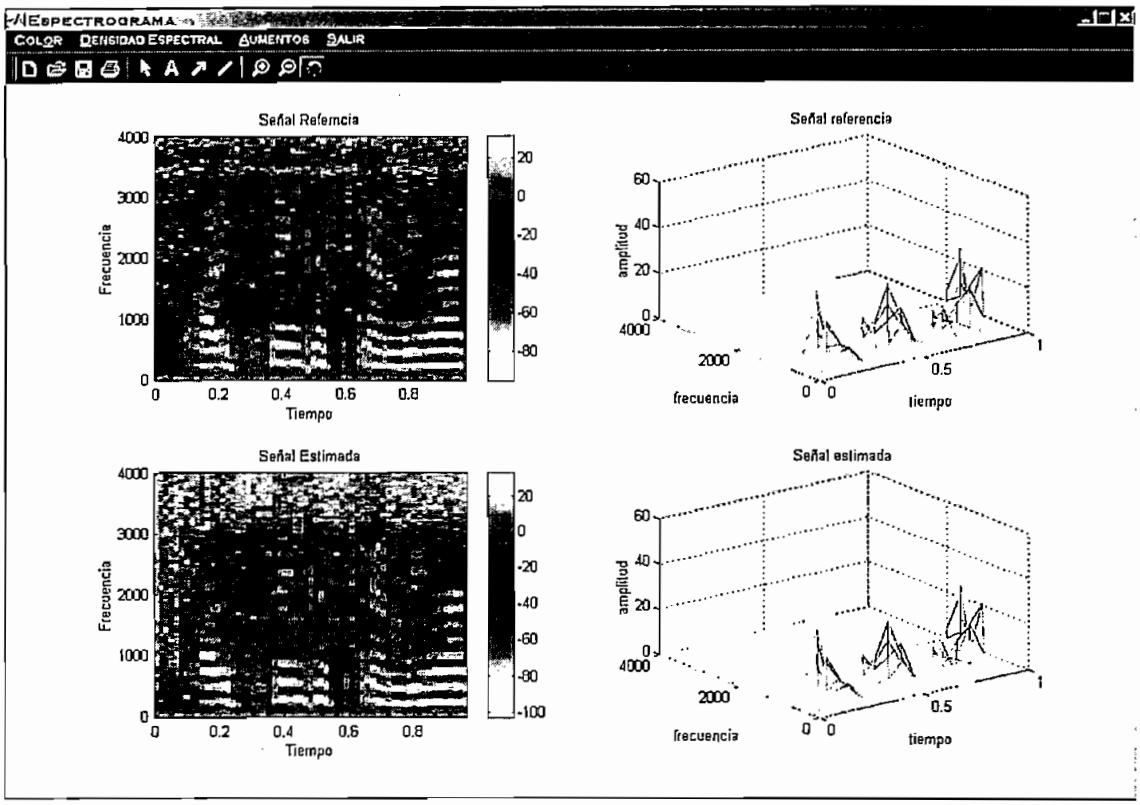



Figura C.5 Espectrograma

**Densidad espectral de potencia**: esta característica se obtiene al aplicar la trasformada rápida de Fourier. En la figura C.6 se presenta esta característica tanto para la señal de referencia, y la señal estimada.

También en cada pantalla se presenta la opción de obtener la trasformada de Fourier en la cual se tiene una representación tanto para la señal de referencia como para la señal estimada, figura C.7

**Error**: se presenta la diferencia entre la señal de referencia y la señal estimada, figura C.8

**Potencia**: se presentan dos gráficas:

1. potencia del error residual
2. Coeficientes finales del filtro adaptivo

Estas gráficas se presentan en la figura C.9

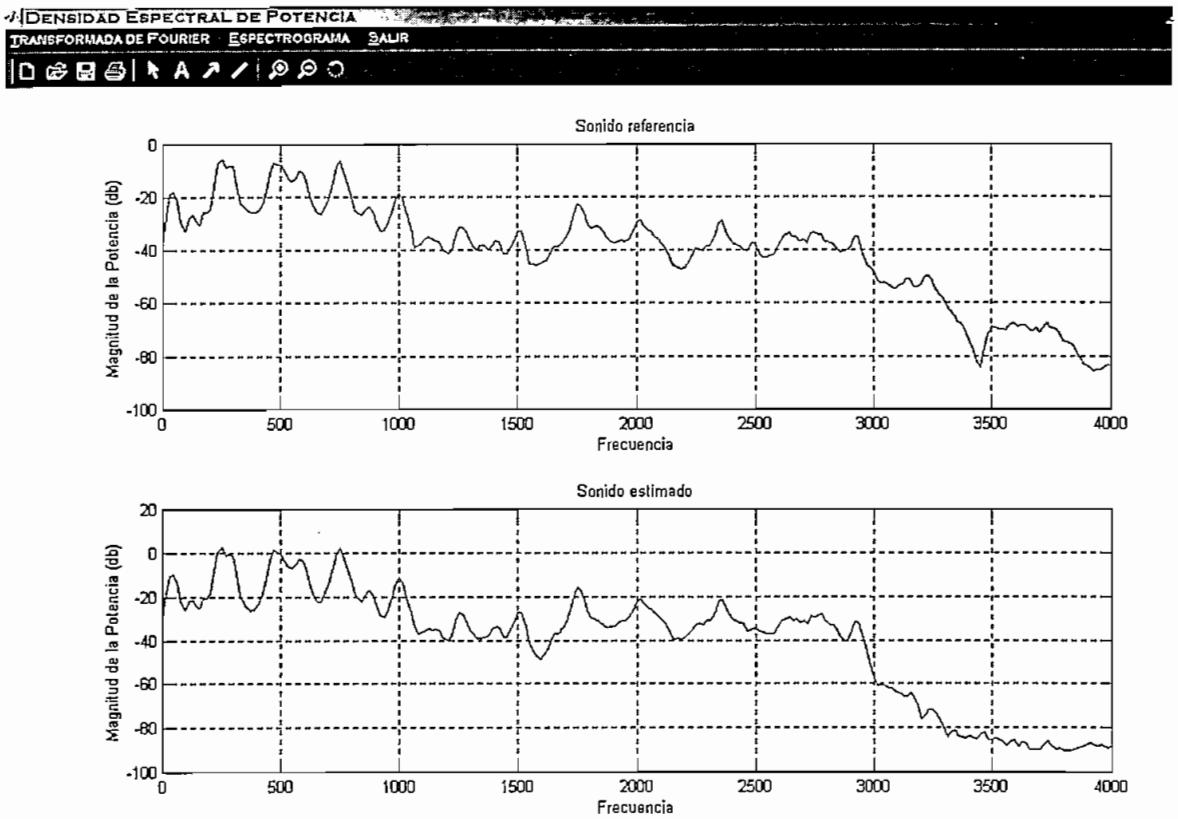



Figura C.6 Densidad espectral de potencia

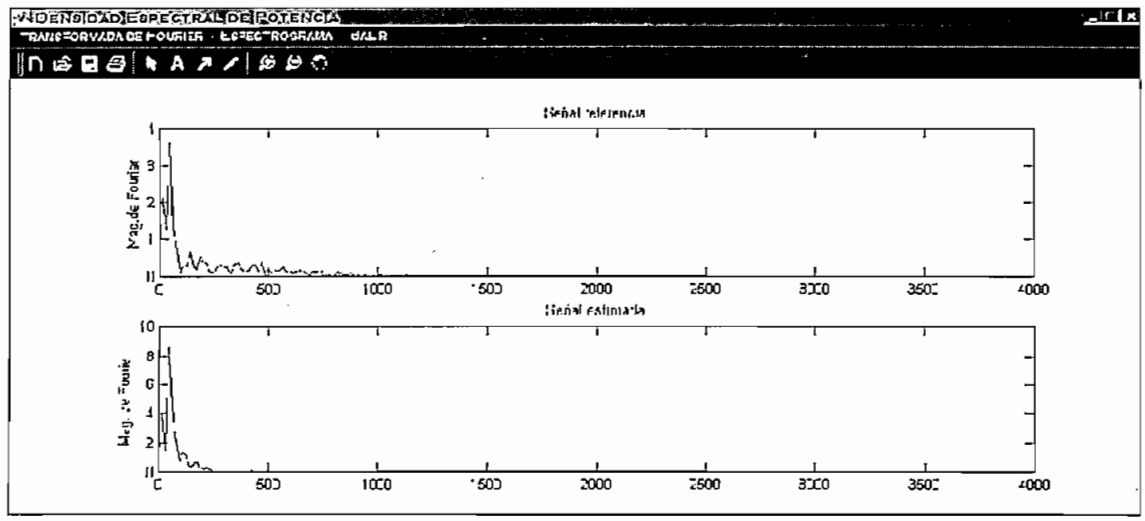



Figura C.7 Transformada de Fourier

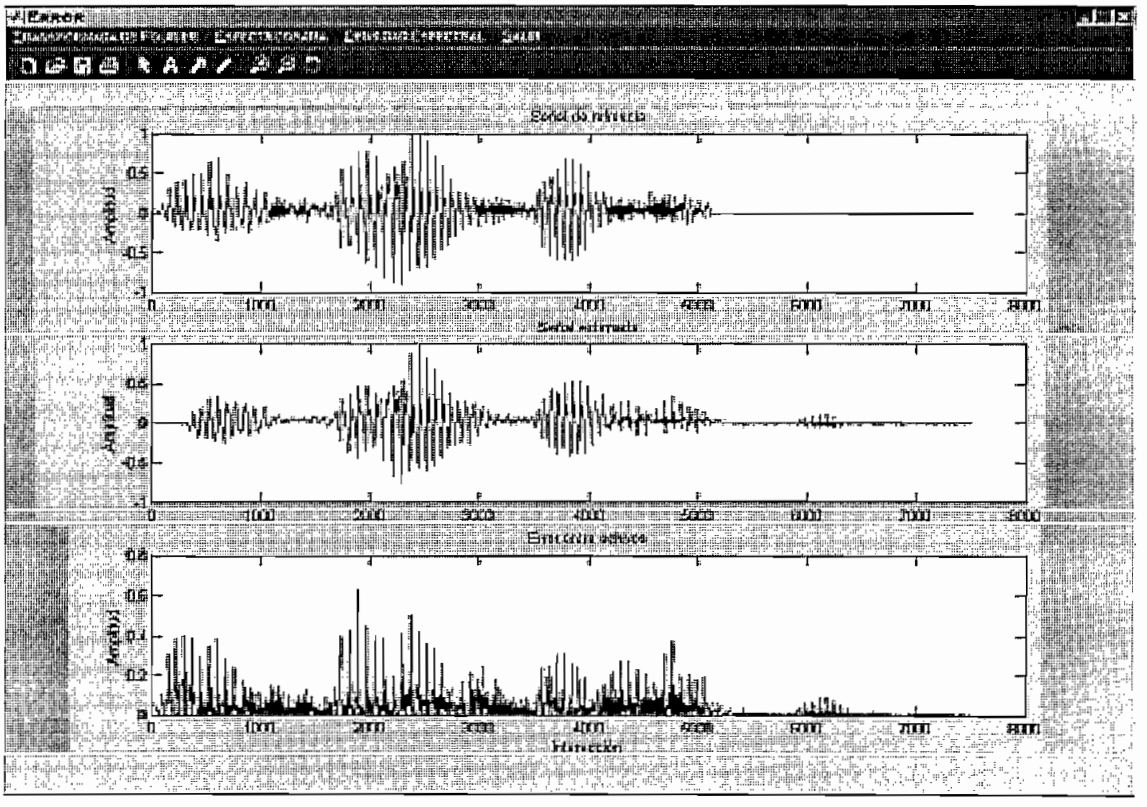


Figura C.8 Diferencia entre señales

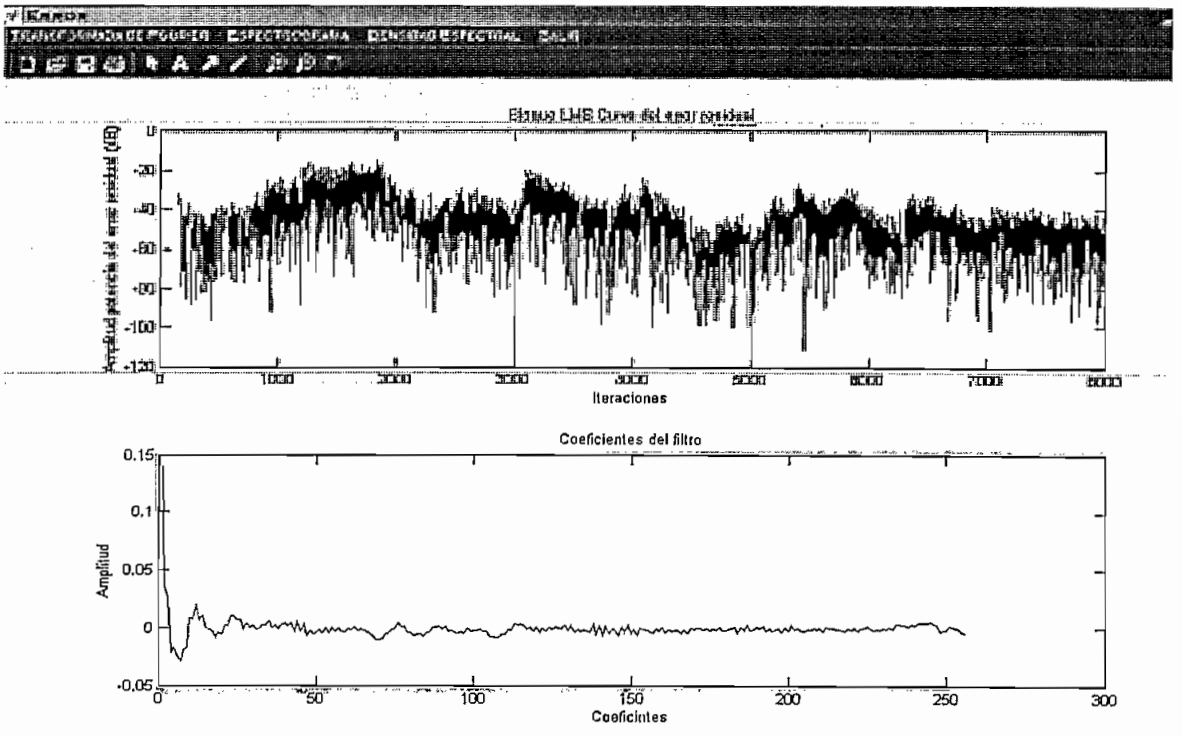



Figura C.9 Potencia del eco residual, Coeficientes de filtro adaptivo

Siguiendo con el menú personalizado se tiene la opción de **Ejemplo de análisis** (figura C.10) con la cual se realiza un análisis del los coeficientes del filtro adaptivo


Figura C.10 Opción ejemplo de análisis

Una vez seleccionada la opción nos aparece la siguiente pantalla (figura C.11) en la cual tenemos dos opciones.

1. Señal telefónica digitalizada , la cual nos permite cargar al programa una señal telefónica real que en este caso es F4.mat, seguidamente se ejecuta el programa y una vez finalizado el proceso se presenta la señal telefónica digitalizada y la señal estimada (figura C.12)

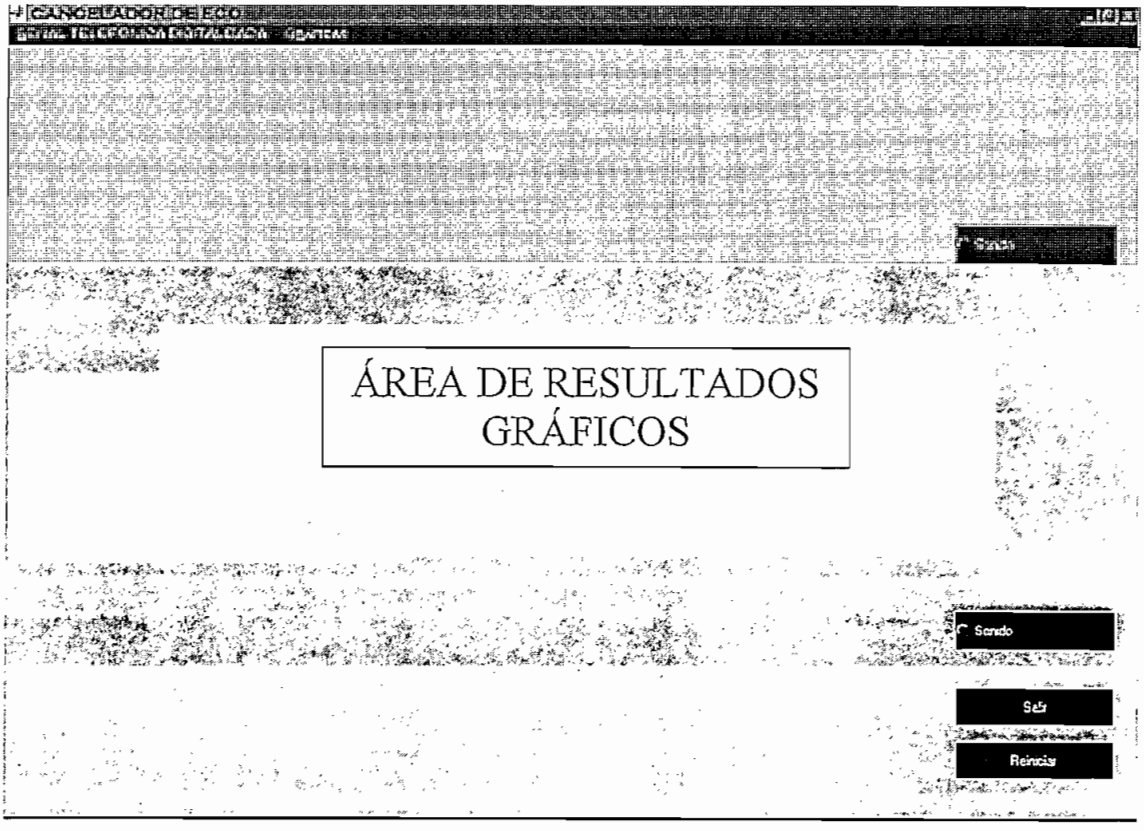



Figura C.11 Pantalla de inicio del ejemplo de análisis

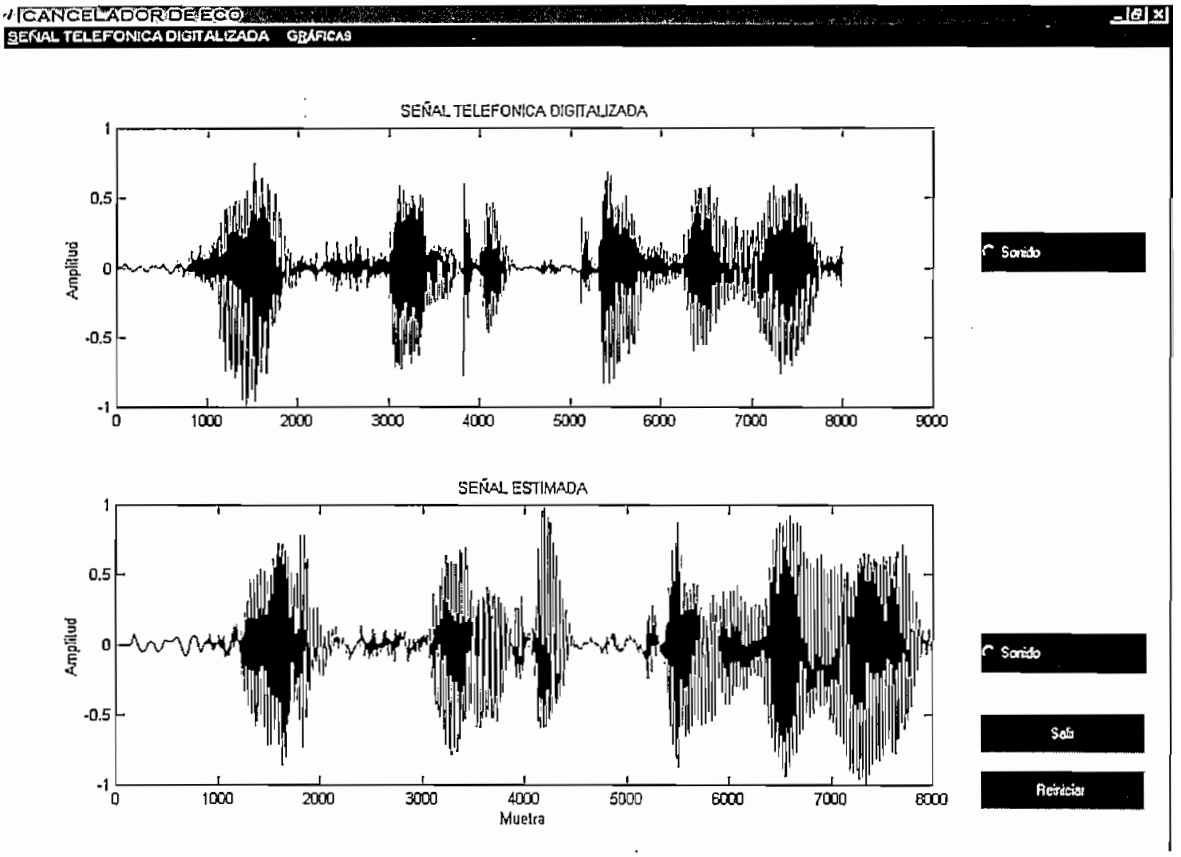



Figura C.12 Señal telefónica digitalizada y señal telefónica estimada

2. Gráficas: en la cual se tiene las opciones que se presenta en la figura C.13

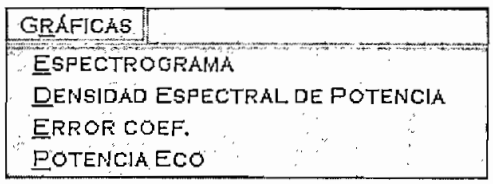



Figura C.13 Opción Gráficas

En la cual se tiene las mismas opciones que en el caso anterior con modificaciones en las dos últimas las cuales se presentan a continuación.

**Error coef**: en el cual se presenta el error entre la respuesta impulsiva del canal telefónico, la respuesta impulsiva estimada y la curva de aprendizaje (figura C.14).

**Potencia eco**: representa la potencia del eco residual (figura C.15)

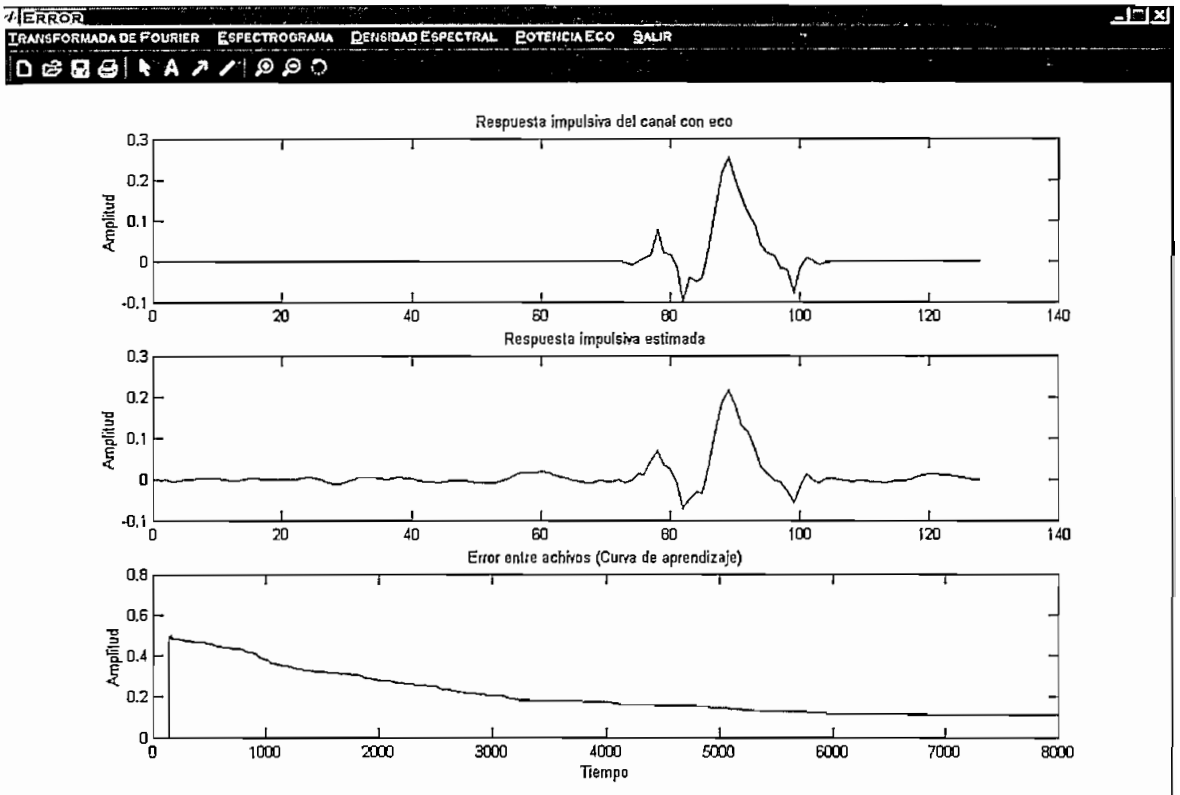



Figura C.14 Respuesta impulsiva del canal telefónico, respuesta impulsiva estimada, y curva de aprendizaje

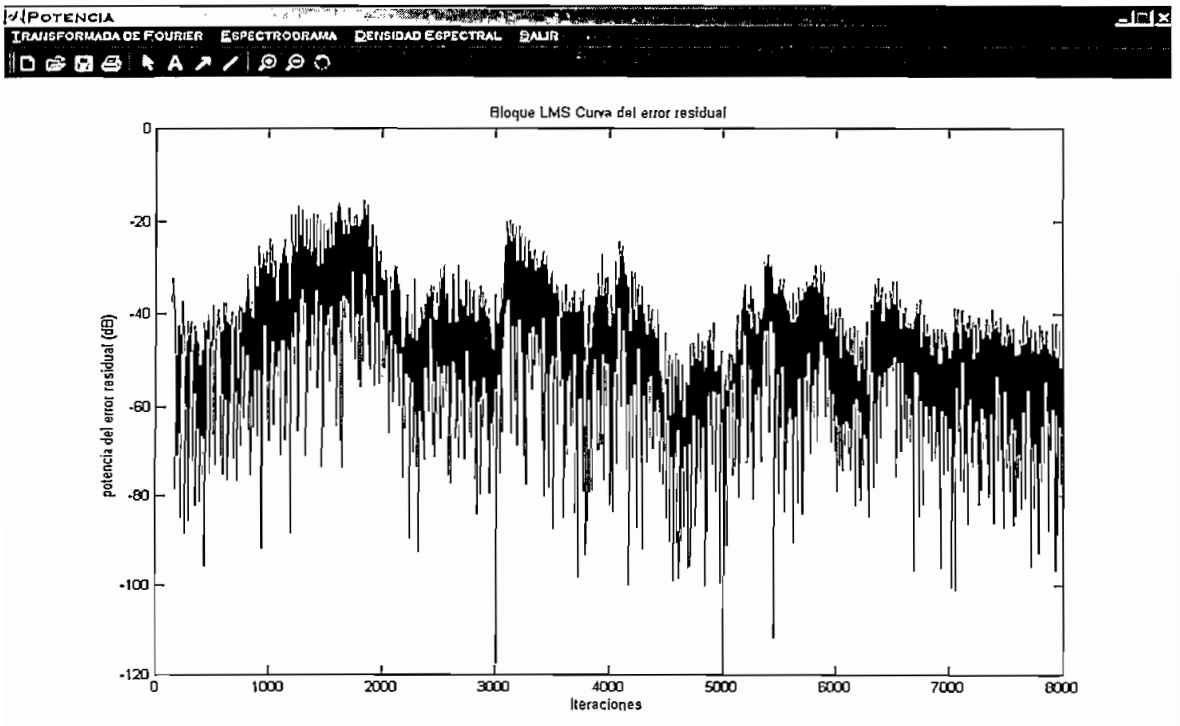



Figura C.15 Potencia del eco residual

En cualquiera de las subventanas para salir se debe dar un clic en **SALIR** y para reiniciar en la pantalla principal se debe dar un clic en **REINICIAR.**

Si se desea utilizar otros archivos Wav, éstos deben estar con una frecuencia de muestreo de 8 khz ya que la programación del algoritmo LMS está hecha para una frecuencia de muestreo antes mencionada.

La velocidad del filtrado adaptivo principalmente dependerá del tamaño de los archivos utilizados, velocidad del procesador utilizado, y de la memoria RAM con la que se cuente.

# ANEXO D [1]

## D Pruebas y requisitos de calidad de funcionamiento con señales de entrada aplicadas a los trayectos emisión y recepción

### D.1 Calidad de transmisión

#### D.1.1 Distorsión por retardo – Tipo A

La distorsión por retardo con respecto al retardo mínimo no excederá de los valores que se indican en la tabla 1.1.

| Banda de frecuencias (Hz) | Distorsión por retardo (µs) |
|---|---|
| 500 a 600 | 300 |
| 600 a 1000 | 150 |
| 1000 a 2600 | 50 |
| 2600 a 3000 | 250 |

Tabla 1.1

#### D.1.2 Distorsión de atenuación – Tipo A

La distorsión de atenuación será tal que, si Q dB es la atenuación a 800 Hz (o 1000 Hz) la atenuación a cualquier frecuencia de la banda de 300 a 3400 Hz estará comprendida entre (Q. 0,5) dB y (Q – 0,2) dB, y a 200 Hz, estará comprendida entre (Q 1,0) dB y (Q – 0,2) dB.

#### D.1.3 Retardo de grupo – Tipo C

El retardo de grupo en el trayecto emisión debe mantenerse lo más reducido posible y no debe exceder de 1ms. En el trayecto recepción no debe producirse ningún retardo apreciable.

---

[1] Recomendación UIT-T G.165, Características Generales de las Conexiones y Circuitos Telefónicos Internacionales, Compensadores de Eco

### D.1.4 Retardo de grupo – Tipo D

Los retardos de grupo en los trayectos emisión y recepción cumplirán los requisitos de los compensadores de eco tipo C, con la adición del retardo permitido para los codecs.

## D.2 Calidad de funcionamiento de los compensadores de eco

Las características de funcionamiento que se indican a continuación son las de los compensadores de eco que incluyen procesadores no lineales

En las pruebas se ha supuesto que el procesador no lineal puede neutralizarse, que el dispositivo para la memoria de la respuesta impulsiva del trayecto de eco (registro H) puede liberarse (ponerse a cero) y que se puede desactivar la adaptación.

Los requisitos se han descrito sobre la base de pruebas efectuadas aplicando señales a $R_{in}$ y $S_{in}$ de un compensador de eco y midiendo la señal en $S_{out}$.. El montaje de prueba se muestra en la Figura D.1. Se supone que los puertos son puntos de igual nivel relativo. Como señal de prueba en la entrada recepción se empleó ruido blanco limitado en banda. La atenuación del eco es independiente de la frecuencia.

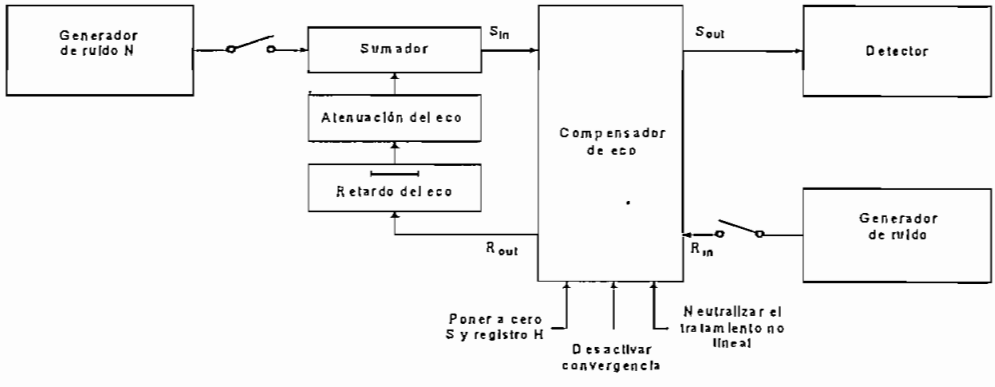



Figura D.1 Prueba de funcionamiento de los compensadores de eco

El compensador de eco tiene por finalidad principal controlar el eco de una señal vocal. Esto se consigue sintetizando una reproducción de la respuesta impulsional del trayecto de eco, que se utiliza para generar una estimación del eco que se sustrae del eco real del circuito. La síntesis debe realizarse utilizando una señal vocal de entrada. Dada la dificultad de definir una señal vocal de prueba, las descritas a continuación son pruebas tipo y se basan en la utilización de una señal de prueba de ruido limitado en banda, esencialmente por razones de conveniencia y posibilidad de repetición de las medidas. Estas pruebas sólo se realizarían en un compensador de eco después que se haya comprobado que sintetiza debidamente una reproducción de la respuesta impulsional del trayecto de eco a partir de una señal vocal de entrada y su eco correspondiente.

### D.2.1 Prueba de los niveles del eco residual y del eco devuelto en régimen permanente

Esta prueba tiene por objeto verificar que la compensación en régimen permanente ($A_{CANC}$) es suficiente para producir un nivel de eco residual suficientemente bajo para permitir la aplicación de un tratamiento no lineal sin depender excesivamente de éste.

Después de liberar inicialmente el registro H, se aplica una señal en recepción durante un plazo suficiente para que el compensador converja y produzca un nivel de eco residual en régimen permanente.

### D.2.2 Prueba de convergencia

Esta prueba tiene por objeto verificar que el compensador de eco converge rápidamente para todas las combinaciones de niveles de la señal de entrada y trayectos de eco, y que el nivel del eco devuelto es suficientemente bajo. Inicialmente, se libera el registro H y se desactiva la adaptación. El detector de habla simultánea, de haberlo, se pone en el modo habla simultánea, para lo cual se aplican señales a $S_{in}$ y $R_{in}$. Se quita entonces la señal de $S_{in}$ y simultáneamente se activa la adaptación. El grado de adaptación, medido por el nivel del eco devuelto, depende de las características de convergencia del

compensador de eco y del tiempo de bloqueo para la detección del habla simultánea.

### D.2.3 Calidad de funcionamiento en condiciones de habla simultánea

Las dos partes de esta prueba tienen por finalidad verificar la idoneidad del método elegido para asegurar un funcionamiento del compensador en distintas condiciones de habla sumultánea. En estas pruebas se ha partido del supuesto de que, una vez detectada el habla simultánea, se toman medidas para impedir la adaptación o reducir su velocidad a fin de evitar una reducción excesiva de la compensación.

#### D.2.3.1 Primera parte

Esta prueba tiene por objeto verificar que la sensibilidad de la detección de habla simultánea no sea tan alta que el eco y una señal vocal de bajo nivel procedente del extremo cercano provoquen un funcionamiento indebido del detector de habla simultánea hasta el punto de que no se produzca la adaptación.

#### D.2.3.2 Segunda parte

Esta prueba tiene por objeto asegurarse de que el detector de habla simultánea es lo suficientemente sensible y funciona con la suficiente rapidez para que no se produzca una gran divergencia durante el habla simultánea.

### D.2.4 Prueba del tiempo de fuga

Esta prueba tiene por objeto asegurarse de que el tiempo de fuga no es demasiado corto, es decir, que el paso del contenido del registro H al valor cero no es demasiado rápido. El procedimiento de prueba consiste en hacer alcanzar la plena convergencia al compensador de eco para un trayecto de eco

determinado y suprimir seguidamente todas las señales aplicadas al compensador de eco

### D.2.5 Prueba de convergencia con pérdida de retorno infinita

Esta prueba está destinada a verificar que el compensador de eco cuenta con medios para impedir la producción indeseada de eco. Esto puede producirse cuando el registro H contiene un modelo de trayecto de eco, correspondiente a una conexión anterior o a la conexión en curso, y se interrumpe el trayecto del eco (desaparece el eco de circuito) mientras está presente una señal en $R_{in}$.

### D.2.6 No divergencia con señales de banda estrecha (optativo)

Esta prueba tiene por objeto verificar que el compensador de eco permanece estable en presencia de señales de banda estrecha. El nivel de eco residual se mide antes y después de aplicar una onda sinusoidal o una onda compuesta de dos frecuencias.

### D.2.7 No convergencia con señales mono o bifrecuencia y en un protocolo de toma de un contacto(optativa)

Los compensadores de eco que no son neutralizados de manera externa, situados en el lado de línea de los sistemas de señalización N.º 5, 6 y 7 en centrales internacionales o asociados con centrales nacionales, deben funcionar de manera adecuada en presencia de tonos de señalización. Esta prueba sirve para verificar que los compensadores de eco no descarten una señal mono o bifrecuencia transmitida en un protocolo de toma de contacto en el sentido de emisión antes o después de recibir una señal idéntica (salvo el nivel y la fase) en el sentido de recepción. El objetivo es conseguir que ciertos tonos de señalización puedan transmitirse correctamente sin que neutralicen externamente los compensadores de eco.

## D.3 Características de un neutralizador por tono para compensadores de eco

### D.3.1 Consideraciones generales

Los compensadores de eco deben estar equipados con un detector de tonos. Este detector de tonos responde a una señal de neutralización y consiste en un tono de 2100 Hz en el que se introducen inversiones de fases periódicas. El neutralizador de tono sólo debe responder a esta señal especificada pero no a otras señales en banda, por ejemplo, vocales, ni a un tono de 2100 Hz sin inversión de fase. El neutralizador de tono debe detectar una señal de neutralización que puede estar presente en el trayecto de emisión o de recepción, y responder a la misma.

El término «neutralizado» (o «desactivado», o «inhabilitado») se refiere a una condición en la cual el compensador de eco está configurado de forma tal que ya no modifica las señales que pasan a través del mismo en uno u otro sentido. En tal condición, no se sustrae del trayecto emisión una estimación de eco, el procesador no lineal se hace transparente.

## D.4 Procesadores no lineales para uso en compensadores de eco

### D.4.1 Campo de aplicación

Tales procesadores no lineales pueden realizarse de diferentes maneras (a título de ejemplo se citan los recortadores del centro de las señales) y tener características de funcionamiento fijas o adaptivas, pero no se formula recomendación alguna sobre una realización particular.. Una información más detallada y concreta hay que buscarla por referencia a realizaciones específicas. Esto se hace en el Anexo C para el caso particular de un «procesador no lineal de referencia». Con este término se ha querido indicar una realización para orientación e ilustración solamente. No excluye otras realizaciones, ni tampoco implica que el procesador no lineal de referencia sea, necesariamente, la

realización más apropiada por cualesquiera razones técnicas, operacionales o económicas.

### D.4.2 Principios generales y directrices

#### D.4.2.1 Función

##### D.4.2.1.1 Consideraciones generales

El procesador no lineal está situado en el trayecto emisión entre la salida del substractor y el puerto de salida emisión del compensador de eco. Desde el punto de vista conceptual, es un dispositivo que bloquea las señales de bajo nivel y deja pasar las de alto nivel.

Tiene por función reducir el nivel de eco residual que queda después de una compensación imperfecta del eco del circuito, de manera que pueda conseguirse el nivel de eco devuelto del bajo valor requerido.

##### D.4.2.1.2 Calidad de funcionamiento

Una compensación imperfecta puede deberse a que los compensadores de eco pueden no ser capaces de modelar adecuadamente trayectos de eco que generan niveles apreciables de distorsión no lineal. Se recomienda, por tanto, que todos los compensadores de eco capaces solamente de modelar los componentes lineales de los trayectos de eco, pero que están destinados al uso en la red general, tengan incorporados procesadores no lineales.

##### D.4.2.1.3 Limitaciones

Esta utilización de procesadores no lineales representa una solución de compromiso en cuanto a la transparencia de circuito que sería posible obtener con un compensador de eco que pudiera lograr el $L_{RET}$ necesario utilizando solamente técnicas de modelado y compensación. Se recomienda que no se deba depender excesivamente del procesador no lineal y que el $L_{RES}$ deba ser lo suficientemente bajo para evitar que se produzcan ecos objetables en condiciones de habla simultánea.

### D.4.2.2 Umbral de supresión

#### D.4.2.2.1 Consideraciones generales

El nivel umbral de supresión ($T_{SUP}$, *suppression threshold level*) de un procesador no lineal se expresa en dBm0 y es igual al nivel más alto de una señal sinusoidal en el momento preciso en que es suprimida. Pueden utilizarse umbrales de supresión fijos o adaptivos.

Cuando se emplea un umbral de supresión fijo, el nivel apropiado que ha de utilizarse dependerá de la compensación obtenida y de las características de los niveles de conversación y las condiciones de línea propias de la red determinada en la que ha de utilizarse el compensador de eco. Se recomienda, por tanto, que el nivel real pueda seleccionarse en el punto de utilización a fin de que el usuario pueda ajustarlo al entorno real de la red.

Para un umbral de supresión adaptivo una buena solución de compromiso puede obtenerse utilizando un $T_{SUP}$ elevado para evitar que sea rebasado por un eco residual de una persona que habla alto, y utilizar un $T_{SUP}$ bajo para reducir la distorsión de la conversación al producirse una intervención, haciendo que el $T_{SUP}$ se adapte a las condiciones de circuito y a los niveles de conversaciones reales. Esto puede conseguirse de diversas maneras y no se recomienda una realización particular.

### D.4.2.3 Control de la activación del procesador no lineal

#### D.4.2.3.1 Consideraciones generales

Es necesario controlar la activación del procesador no lineal de modo que no esté activo cuando sea probable que haya señales vocales del extremo cercano. Cuando el procesador no lineal está «activo», deberá funcionar tal como está concebido para reducir el $L_{RES}$. Cuando está «inactivo», no debe realizar ningún tratamiento no lineal de ninguna señal que atraviese el compensador de eco.

#### D.4.2.3.2 Orientaciones sobre control

Se recomienda que el control de la activación de los procesadores no lineales se base en los dos principios siguientes. En primer lugar, puesto que tienen por objeto reducir aun más el $L_{RES}$, deben estar activos cuando $L_{RES}$ tenga un nivel apreciable. Segundo, puesto que no deben deformar las señales vocales del extremo cercano, deben estar inactivos en presencia de señales vocales del extremo cercano. Cuando estos dos principios estén en contradicción, la función de control deberá favorecer el segundo de ellos.

# ANEXO E

# LISTADO DEL PROGRAMA DE LA INTERFAZ GRAFICA

```
%****************************************
%                                       %
%         PANTALLA DE PRESENTACION      %
%                                       %
%****************************************



function fig = inicio()
% This is the machine-generated representation of a Handle Graphics object
% and its children.  Note that handle values may change when these objects
% are re-created. This may cause problems with any callbacks written to
% depend on the value of the handle at the time the object was saved.
% This problem is solved by saving the output as a FIG-file.
%
% To reopen this object, just type the name of the M-file at the MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.
%
% NOTE: certain newer features in MATLAB may not have been saved in this
% M-file due to limitations of this format, which has been superseded by
% FIG-files.  Figures which have been annotated using the plot editor tools
% are incompatible with the M-file/MAT-file format, and should be saved as
% FIG-files.

load inicio

h0 = figure('Color',[0.8 0.8 0.8], ...
    'Colormap',mat0, ...
    'FileName','C:\MATLABR11\work\inicio.m', ...
    'Name','CANCELADOR DE ECO', ...
    'NumberTitle','off', ...
    'PaperPosition',[18 180 576 432], ...
    'PaperUnits','points', ...
    'Position',[1 29 1024 702], ...
    'Tag','Fig1', ...
    'ToolBar','none');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 1], ...
    'FontSize',20, ...
    'ListboxTop',0, ...
    'Position',[94.5 167.25 582 338.25], ...
    'String','ESCUELA POLITECNICA NACIONAL', ...
    'Style','text', ...
    'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 1], ...
    'FontSize',20, ...
    'ListboxTop',0, ...
    'Position',[115.5 424.5 543.75 42.75], ...
    'String','CARRERA DE INGENIERIA ELECTRONICA Y TELECOMUNICACIONES', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 1], ...
    'FontSize',24, ...
    'ListboxTop',0, ...
    'Position',[117 375 542.25 37.5], ...
    'String','PROYECTO DE TITULACION', ...
    'Style','text', ...
    'Tag','StaticText3');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 1], ...
    'FontSize',20, ...
    'ListboxTop',0, ...
```

```
    'Position',[114 317.25 546.75 39.75], ...
    'String','TEMA: SIMULACION DE UN SISTEMA DE CANCELACION DE ECO PARA CANALES
TELEFONICOS', ...
    'Style','text', ...
    'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 1], ...
    'FontSize',20, ...
    'ListboxTop',0, ...
    'Position',[112.5 262.5 548.25 37.5], ...
    'String','REALIZADO POR: RODRIGO ANIBAL VILLARREAL HERNANDEZ', ...
    'Style','text', ...
    'Tag','StaticText5');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 1], ...
    'FontSize',20, ...
    'ListboxTop',0, ...
    'Position',[110.25 221.25 549.75 38.25], ...
    'String','DIRIGIDO POR: Dr. GUALBERTO HIDALGO', ...
    'Style','text', ...
    'Tag','StaticText6');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 1], ...
    'FontSize',20, ...
    'ListboxTop',0, ...
    'Position',[114 192 546 21.75], ...
    'String','2002', ...
    'Style','text', ...
    'Tag','StaticText7');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.847058823529412 0.752941176470588 0.627450980392157], ...
    'Callback','pantalla11', ...
    'FontSize',14, ...
    'ListboxTop',0, ...
    'Position',[243.75 93 93.75 39.75], ...
    'String','CONTINUAR', ...
    'Tag','Pushbutton1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.847058823529412 0.752941176470588 0.627450980392157], ...
    'Callback','pantalla1(''salir'')', ...
    'FontSize',14, ...
    'ListboxTop',0, ...
    'Position',[430.5 93 93 38.25], ...
    'String','SALIR', ...
    'Tag','Pushbutton2');
if nargout > 0, fig = h0; end
```

```
function pantalla11(proceso)
% profile on -detail operator;
global  pantalla1 paramet;
global  nombre
global y1 fm ;
global y ;
global time1 ampl Tiempo pa  tseg fm n_muestras n_segundos x Nm;
global time2 amp2 SIZ origen  p6 p5
global dnuevo1 dnuevo2 dinicial dinicial2 archivo;
global Nms Nmsil Silen s fms densi2 espec2 pant
global pox poy;
global  espera;
global xp yp tie frec ele;
global fi ff ff1 ff2 ff3 ff4 ff5 ff6 ff7 ff8
global fd0 fd1 fd2 fd3 fm1 fm2 fm3 fm4 eng n pfn



p=1;
if nargin<1
   proceso='inicio';
end
switch(proceso)

case 'inicio'

%************************************************************
%***** valores de posicionamiento de botones y pantallas *****
%************************************************************

   posx=0.02;
   posy =0.92;
   posancho=0.14;
   posalto=0.05;

%************************************************************
% valores de la pantalla1, color, márgenes, título
%************************************************************
%color .1 .7 .7
%'Color',[0.333333333333333 0.450980392156863 0.415686274509804], ...
%'Color',[0.250980392156863 0.501960784313725 0.501960784313725], ...
   pantalla1 = figure('Units','normalized', ...
   'color','b',...
   'colormap',[.1 .7 .7],...
   'DoubleBuffer','on', ...
   'MenuBar','none', ...
   'Name','CANCELADOR DE ECO', ...
   'NumberTitle','off', ...
   'PaperPosition',[18 180 576 432], ...
   'PaperUnits','normalized', ...
   'Position',[0.11 0.14 0.7 0.7], ...
   'Tag','Fig2', ...
   'ToolBar','none');

%************************************************************
%*********** definición de menus ***************************
%************************************************************

    fi=uimenu('Label','ARCHIVO CON ECO','CallBack','pantalla11(''abrir1'')');


    ff=uimenu('Label','ARCHIVO DE REFERENCIA','CallBack','pantalla11(''abrir2'')');


    fd=uimenu('Label','GRAFICOS');
    fd0=uimenu(fd,'Label','&Espectrograma','enable','on','Callback','pantalla11(''espec
2'')');
    fd2=uimenu(fd,'Label','&Densidad Espectral de
Potencia','enable','on','Callback','pantalla11(''densi2'')');

fd3=uimenu(fd,'Label','&Error','enable','on','Callback','pantalla11(''diferencia'')');
```

```
    fd4=uimenu(fd,'Label','&Potencia','enable','on','Callback','pantallall(''potencia''
)');

    fn=uimenu('Label','EJEMPLO DE ANALISIS');
    fn0=uimenu(fn,'Label','Coeficioentes del filtro
adaptivo','enable','on','Callback','pantallall(''ejemplol'')');
    %fn1=uimenu(fn,'Label','Potencia-
C.Aprendizaje','enable','on','Callback','pantallall(''potenciacoef'')');

  %   uimenu('Label','AYUDA','Callback','web
file:///c|/matlabr11/work/tesis/html/cancelador.html');



%*********************************************************
% ********** botón de salida ****************************
%*********************************************************

   p1 = uicontrol(pantallal, ...
   'Units','normalized', ...
   'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
   'Callback','pantallall(''salir'')', ...
   'ListboxTop',0, ...
   'Position',[posx+0.82 posy-0.8 posancho posalto], ...
   'String','Salir', ...
   'Tag','Pushbutton5');

%*********************************************************
% ********** botón de reseteo ****************************
%*********************************************************

   p2 = uicontrol(pantallal, ...
   'Units','normalized', ...
   'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
   'Callback','pantallall(''resetear'')', ...
   'ListboxTop',0, ...
   'Position',[posx+0.82 posy-0.87 posancho posalto], ...
   'String','Reiniciar ', ...
   'TooltipString','Reseteo del programa',...
   'Tag','Pushbutton6');

%***********************************************

% 'backgroundcolor',[0.1 0.7 0.7]
   origen = uicontrol(pantallal, ...
   'Units','normalized', ...
   'backgroundcolor',[.1 .7 .7],...
   'Position',[posx+0.25 posy+0.02 posancho posalto-0.01], ...
   'String','ARCHIVO CON ECO', ...
   'Horizontal','center',...
   'Style','text', ...
   'Visible','off',...
   'Tag','StaticText2');

   efecto = uicontrol(pantallal, ...
   'Units','normalized', ...
   'backgroundcolor',[.1 .7 .7],...
   'Position',[posx+0.25 posy- 0.47 posancho posalto-0.01], ...
   'String','Archivo ARCHIVO SIN ECO', ...
   'Horizontal','center',...
   'Style','text', ...
   'Visible','off',...
   'Tag','StaticText3');

sonidos1=uicontrol(pantallal,...
   'Units','normalized',...
   'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
   'Position',[posx+0.82 posy-0.2 posancho posalto], ...
   'Style','radiobutton', ...
   'enable','on',...
   'Value',0,...
```

```
    'String','Sonido', ...
    'Callback','pantalla11(''sonidoeco'')', ...
    'Tag','Radiobutton2');

sonidos2=uicontrol(pantalla1,...
    'Units','normalized',...
    'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
    'Position',[posx+0.82 posy-0.7 posancho posalto], ...
    'Style','radiobutton', ...
    'enable','on',...
    'Value',0,...
    'String','Sonido', ...
    'Callback','pantalla11(''sonidoest'')', ...
    'Tag','Radiobutton4');

orden=uicontrol( ...
      'Style','text', ...
        'Units','normalized', ...
      'Position',[posx+0.82 posy-0.4 posancho posalto], ...
        'Horiz','left', ...
        'String','Orden:', ...
        'Interruptible','off', ...
      'BackgroundColor',[0.5 0.5 0.5], ...
        'ForegroundColor','white');
        orderHndl = uicontrol( ...
      'Style','edit', ...
      'Units','normalized', ...
      'Position',[posx+0.87 posy-0.4 posancho posalto], ...
      'Horiz','left', ...
      'Background','white', ...
      'Foreground','black', ...
      'String','128','Userdata',128, ...
      'callback','pantalla11(''orden'')');

   paso=uicontrol( ...
      'Style','text', ...
        'Units','normalized', ...
      'Position',[posx+0.82 posy-0.5 posancho posalto], ...
        'Horiz','left', ...
        'String','Paso:', ...
        'Interruptible','off', ...
      'BackgroundColor',[0.5 0.5 0.5], ...
        'ForegroundColor','white');
        orderHndl = uicontrol( ...
      'Style','edit', ...
      'Units','normalized', ...
      'Position',[posx+0.87 posy-0.5 posancho posalto], ...
      'Horiz','left', ...
      'Background','white', ...
      'Foreground','black', ...
      'String','1','Userdata',1, ...
      'callback','pantalla11(''paso'')');

case 'abrir1'
   [nombre,ruta]=uigetfile('*.wav','CANCELADOR DE ECO'); % permite al usuario abrir un archivo
   SIZ=wavread(nombre,'size'); % Devuelve a SIZ el tamaño de wavread
   Y=wavread(nombre); % almacena en Y el archivo q se abrió
   Y=Y/max(abs(Y)); % Normalización

   wavwrite(Y,'c:\MATLABR11\WORK\eco')

   temp = subplot(2,1,1),plot(Y,'y');
   title('SEÑAL CON ECO');
   ejeposi=[.1 .55 .7 .35]
   set(temp, 'position', ejeposi);
   ylabel('Amplitud');
   xlabel('Muestra');
   set(gca,'Color','k'); set(gca,'Xcolor','w'); set(gca,'YColor','w');

case 'abrir2'
```

```
    [nombre,ruta]=uigetfile('*.wav','CANCELADOR DE ECO');
    SIZ=wavread(nombre,'size');
    X=wavread(nombre)
    wavwrite(X,'c:\MATLABR11\WORK\referencia')
    pa=SIZ(1,1);
    pantalla11('cancelador')
case 'orden'
    v = get(gco,'UserData');  % gco
    s = get(gco,'String');
    vv = eval(s,num2str(v))
    print vv
    save vv
case 'paso'
    v = get(gco,'UserData');
    s = get(gco,'String');
    bb = eval(s,num2str(v))
    print bb
    save bb
    pantalla11 ('cancelador')

case 'cancelador'
    load vv
    load bb
    N=vv;
    U=bb;
% Parmetros adaptivos

    b1 = 2^(-12);
    rho = 2^(-7);

    %X=referencia;
    %Y=eco;
    %ref=input('Introduzca nombre de la señal de referencia .WAV... ','s');

    [Y,fs,bits]=wavread('referencia');

%eco=input('Introduzca nombre de la señal con eco .WAV... ','s');
    [X,fs,bits]=wavread('eco');

    y=X;
    r=Y;
    xpr=r;
    print X
    Y=Y/max(abs(Y));
    X=X/max(abs(X));
    Ty=size(Y,1)

    Tx=size(X,1)

    corr=zeros(Tx-Ty,1);
    Y=[Y;corr];
    r=Y;
    xpr=r;
    L=Tx;


    % Localizaciones de las señales
    e = zeros(L,1);          % Error entre filtered far end speech y xpr;
    etap = zeros(L,1);       % Error entre filter adaptive estimate;
    rhat = zeros(L,1);       % estimacion de la version reflejada far end speech

h = waitbar(0,'estimando señal....');

    %*******************************
    %************* Bloque del filtro LMS
    M = 16;         % rank of correlation estimate - amount of smoothing for gradient
estimate
    a = zeros(N,1);       % pesos del Filtro
    P = 16;         % round robin factor
```

```
    ly = abs(y(N+M-1));

    for i=N+M:L-1

      rhat(i) = a' * flipud(y(i-N+1:i));  % flipud a ver mañana
      e(i) = xpr(i)- rhat(i);
      ly = (1-rho) * ly + rho*abs(y(i));
      py = (M * ly * ly) / (b1 * P);
     for k=rem(i,P)+1:P:N
      deltak = e(i-M+1:i)' * y(i-M-k+2:i-k+1);
      a(k) = a(k) + ((deltak*U) / py);
     end
     %etap(i) = norm(a - tail_filt);

  end
 waitbar(i/(L-1),h);

 close (h);
 save coef a;
 save e;
 rhat=rhat/max(abs(rhat));
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5



    temp= subplot (2,1,2), plot (rhat,'y');
    title ('SEÑAL ESTIMADA');
    ejeposi=[.1 .08 .7 .35];
    set(temp, 'position', ejeposi);
    ylabel('Amplitud');
    xlabel('Muestra');
    set(gca,'Color','k'); set(gca,'Xcolor','w'); set(gca,'YColor','w');

    wavwrite(rhat,fs,16,'c:\MATLABR11\WORK\estimacion')
    wavwrite(e,fs,16,'c:\MATLABR11\WORK\error')
%*****************************************************************
%***************boton para graficar la potencia del eco recidual***
%*****************************************************************
%***********************************************
%*** ejecución de los sonidos de los archivos****
%***********************************************

case 'sonidoeco'
    Y=wavread('eco.wav');
    sound (Y);

case 'sonidoest'
    Y=wavread('estimacion');
    sound (Y);

case 'resetear'
    close
    clear all
    pantalla11

case 'salir'
    close all
    clear all
    clc



case 'espera'
     h = waitbar(0,'estimando señal....');
       for i=1:1000,
        waitbar(i/1000,h)
       end
       close(h)


case 'espec2'
```

106 — Ref.

```
   pantalla2

case 'densi2'

   pantalla3

case 'diferencia'

   pantalla4

case 'ejemplo1'

   pantallaejem1

case 'potencia'

   pantalla5

case 'potenciacoef'

   pantalla51

case 'ejemplo2'
   pantallaejem2

case 'ejemplo3'
   pantallaejem3
end

%*************************************************
%profile report;
if nargout > 0, fig = h0; end
```

```
function pantalla2(opc)
global x fm y espec color pant;
global xlong;
global SD_PLOTCMD Mi Mf nombre;
global SD_DISP ;
global plotSpec ;
global xl x x3 xs;
global yp ypl ypsl yps;
global z zs zp tiempo tiempo2 ;
global dato frec dato2 frec2;

if nargin<1
   opc='inicio';
end

switch opc

case 'inicio'

   colordef black
   espec = figure('Units','normalized', ...
   'MenuBar','none', ...
   'Name','Espectrograma', ...
   'NumberTitle','off', ...
   'PaperPosition',[18 180 576 432], ...
   'PaperUnits','normalized', ...
   'ToolBar','figure',...
   'Position',[0 0.05 1 0.84], ...
   'Resize','off',...
   'color','b',...
   'Tag','Fig7');


%*********************************
%******* creación de menus ******

   %** creación de menus de color ***

   color = uimenu('Parent',espec, ...
   'interruptible', 'off', 'busyaction', 'cancel',  ...
   'Label','Col&or', 'Tag','mnuamplia');

   color1 = uimenu('Parent',color, ...
   'Callback','pantalla2(''jet'');', ...
   'interruptible', 'off', 'busyaction', 'cancel',  ...
   'Label','&Jet', 'Tag','jet');

   color2 = uimenu('Parent',color, ...
   'Callback','pantalla2(''cool'');', ...
   'interruptible', 'off', 'busyaction', 'cancel',  ...
   'Label','Coo&l', 'Tag','cool');

   color3 = uimenu('Parent',color, ...
   'Callback','pantalla2(''hot'');', ...
   'interruptible', 'off', 'busyaction', 'cancel',  ...
   'Label','ho&t', 'Tag','mnuampliaout', 'Separator', 'on');

   color4 =uimenu('Parent',color, ...
   'Callback','pantalla2(''prism'');', ...
   'interruptible', 'off', 'busyaction', 'cancel',  ...
   'Label','&Prism', 'Tag','cool');

   color5=uimenu('Parent',color, ...
   'Callback','pantalla2(''gray'');', ...
   'interruptible', 'off', 'busyaction', 'cancel',  ...
   'Label','&Gray', 'Tag','cool','Separator', 'on');

   color6=uimenu('Parent',color, ...
   'Callback','pantalla2(''pink'');', ...
   'interruptible', 'off', 'busyaction', 'cancel',  ...
   'Label','P&ink', 'Tag','cool');
```

```
   color7=uimenu('Parent',color, ...
   'Callback','pantalla2(''copper'');', ...
   'interruptible', 'off', 'busyaction', 'cancel',  ...
   'Label','Copp&er', 'Tag','cool','Separator', 'on');

   color8=uimenu('Parent',color, ...
   'Callback','pantalla2(''flag'');', ...
   'interruptible', 'off', 'busyaction', 'cancel',  ...
   'Label','&Flag', 'Tag','cool');

   densidad=uimenu('Parent',espec,...
   'Callback','pantalla3', ...
   'interruptible', 'off', 'busyaction', 'cancel',  ...
   'Label','&Densidad Espectral', 'Tag','densidad');

   aumento=uimenu('Parent',espec,...
      'interruptible', 'off', 'busyaction', 'cancel',  ...
      'Label','&AMPLIAR', 'Tag','aumento');

   aumento1=uimenu('Parent',aumento,...
   'Callback','pantalla2(''otro1'')',...
   'interruptible', 'off', 'busyaction', 'cancel',  ...
   'Label','Referencia', 'Tag','aumento1');

   aumento2=uimenu('Parent',aumento,...
   'Callback','pantalla2(''otro2'')',...
   'interruptible', 'off', 'busyaction', 'cancel',  ...
   'Label','Estimación', 'Tag','aumento2');

   mnuSalir = uimenu('Parent',espec, ...
   'interruptible', 'off', 'busyaction', 'cancel',  ...
   'Label','&Salir', 'Tag','mnuSpectrogram','Callback','close(gcf)');



%************************************************
% gráfico del espectrograma
%************************************************
   [x,fs,bits]=wavread('referencia.wav');
   yo=x/max(abs(x));t=(0:length(yo)-1)/fs;
   subplot(2,2,1);
   set(gcf,'UserData',[t(:) yo(:)]);
   specgram(yo,[],fs);
   set(gca,'XColor','w');
   set(gca,'YColor','w');
   xlabel('Tiempo');
   ylabel('Frecuencia');
   title(['Señal Referncia']);
   set(colorbar,'ycolor','w');
   set(colorbar,'xcolor','w');
   drawnow;


%**********GRAFICO DEL ESPECTRO DE POTENCIA DEL de la estimacion**********


 [y,fs,bits]=wavread('estimacion.wav');

   y2=y/max(abs(y));
   t=(0:length(y2)-1)/fs;
   subplot(2,2,3);
   set(gcf,'UserData',[t(:) y2(:)]);
   specgram(y2,[],fs);
   xlabel('Tiempo');
   set(gca,'XColor','w');
   set(gca,'YColor','w');
   ylabel('Frecuencia');
   title(['Señal Estimada']);
   colormap(jet);
```

```
    colorbar
    set(colorbar,'xcolor','w');
    set(colorbar,'ycolor','w');
    drawnow;


%*** calculo de parámetros para espectrograma***


    [x,fm,bits]=wavread('referencia.wav',[Mi Mf]);
    [x1,fm2,t]=specgram(x,1024,fm); % y,1024,fs
    dato=x1;
    frec=fm2;
    tiempo=t;
    [y,fm,bits]=wavread('estimacion.wav',[Mi Mf]);
    [x3,fm3,t3]=specgram(y,1024,fm);
    dato2=x3;
    frec2=fm3;
    tiempo2=t3;

%*** selección de rutinas de menus************


    subplot(2, 2, 2);
    [x2 yp] = meshgrid(tiempo,frec);
    x = abs(x2);
    yp1 = abs(yp);
    z = abs(dato);
    plotSpec = mesh(x,yp1,z);
    xlabel('tiempo');ylabel('frecuencia');zlabel('amplitud');
    set(gca,'xcolor','w');
    set(gca,'ycolor','w');
    rotate3d on;
    title('Señal referencia');

    subplot(2, 2, 4);
    [x2 yp] = meshgrid(tiempo2,frec2);
    x = abs(x2);
    yp1 = abs(yp);
    z = abs(dato2);
    plotSpec = mesh(x,yp1,z);
    xlabel('tiempo');ylabel('frecuencia');zlabel('amplitud');
    set(gca,'xcolor','w');
    set(gca,'ycolor','w');
    rotate3d on;
    title('Señal estimada');

case 'cool'
    colormap(cool)

case 'jet'
    colormap(jet)

case 'hot'
   colormap(hot)

case 'prism'
    colormap(prism)
case 'pink'
    colormap(pink)
case 'gray'
    colormap(gray)
case 'copper'
    colormap(copper)
case 'flag'
colormap(flag)

case 'otro1'
    figure(12)
    [x2 yp] = meshgrid(tiempo,frec);
    x = abs(x2);
```

```
    yp1 = abs(yp);
    z = abs(dato);
    plotSpec = mesh(x,yp1,z);
    xlabel('tiempo');ylabel('frecuencia');zlabel('amplitud');
    set(gca,'xcolor','w');
    set(gca,'ycolor','w');
    rotate3d on;
    title('Señal referencia');

case 'otro2'
    figure(13)
    [x2 yp] = meshgrid(tiempo2,frec2);
    x = abs(x2);
    yp1 = abs(yp);
    z = abs(dato2);
    plotSpec = mesh(x,yp1,z);
    xlabel('tiempo');ylabel('frecuencia');zlabel('amplitud');
    set(gca,'xcolor','w');
    set(gca,'ycolor','w');
    rotate3d on;
    title('Señal estimada');

end

if nargout > 0, fig = espec; end
```

```
function pantalla3(opc)
global x y fm densi;
global xlong;
global SD_PLOTCMD;
global SD_DISP ;
global plotSpec ;
global x1 x x3 xs;
global yp yp1 yps1 yps;
global z zs zp tiempo tiempo2  ;
global dato frec dato2 frec2;

if nargin<1
   opc='inicio';
end

switch opc
   case 'inicio'
   colordef black
   densi = figure('Units','normalized', ...
   'MenuBar','none', ...
   'Name','Densidad Espectral de Potencia', ...
   'NumberTitle','off', ...
   'PaperPosition',[18 180 576 432], ...
   'PaperUnits','normalized', ...
   'Resize','off',...
   'color','b',...
   'ToolBar','figure',...
   'Position',[0 0.05 1 0.84], ...
   'Tag','Fig8');


%******************************
%** creación de menus *********

      psdgraf=uimenu('parent',densi,...
      'interruptible', 'off', 'busyaction', 'cancel',  ...
      'Label','&TRANSFORMADA DE FOURIER',
'Tag','mnuSpectrogram','Callback','pantalla3(''fftgraf'')');


   espectro= uimenu('Parent',densi, ...
      'Callback','pantalla2', ...
      'interruptible', 'off', 'busyaction', 'cancel',  ...
      'Label','&Espectrograma', 'Tag','espectro','Separator', 'on');

   mnuSalir = uimenu('Parent',densi, ...
      'interruptible', 'off', 'busyaction', 'cancel',  ...
      'Label','&Salir', 'Tag','mnuSpectrogram','Callback','close(gcf)');


[x,fm,bits]=wavread('referencia.wav');
   x=x/max(abs(x))

   subplot(2,1,1);
   psd(x,512,fm,[],256);
   title(['Señal referencia']);
   xlabel('Frecuencia');ylabel('Magnitud de la Potencia (db)')
   drawnow;
   [y,fm,bits]=wavread('estimacion.wav');
   y=y/max(abs(y))
   subplot(2,1,2);
   psd(y,512,fm,[],256);
   title(['Señal estimada']);
   xlabel('Frecuencia');ylabel('Magnitud de la Potencia (db)')
   drawnow;

case 'fftgraf'
   x=wavread('referencia');
   x=x/max(abs(x));
   Ft=fft(x,512);
   w=(0:255)/256*(fm/2);
```

```
    subplot(2,1,1);
    plot(w,abs(Ft(1:256)));
    ylabel('Mag.de Fourier')
    title(['Señal referencia']);
    y=wavread('estimacion');
    y=y/max(abs(y))
    Ft1=fft(y,512);
    w=(0:255)/256*(fm/2);
    subplot(2,1,2);
    plot(w,abs(Ft1(1:256)));
    ylabel('Mag. de Fourier')
    title(['Señal estimada']);
    Ty=size(y,1)

    Tx=size(x,1)

    corr=zeros(Ty-Tx,1);
    x=[x;corr];

    Ft2=fft(zp,512);
    %w=(0:255)/256*(fm/2);
    %subplot(3,1,3);
    %plot(w,abs(Ft2(1:256)));
    %title(['Diferencia']);
    %xlabel('Frecuencia (Hz)');
    %ylabel('Mag. de Fourier')
end
```

```
function pantalla4(opc)
global x y fm densi;
global xlong;
global SD_PLOTCMD;
global SD_DISP ;
global plotSpec ;
global x1 x x3 xs;
global yp yp1 yps1 yps;
global z zs zp tiempo tiempo2  ;
global dato frec dato2 frec2;


if nargin<1
   opc='inicio';
end

switch opc
   case 'inicio'
  % colordef black
   densi = figure('Units','normalized', ...
   'MenuBar','none', ...
   'Name','Error', ...
   'NumberTitle','off', ...
   'PaperPosition',[18 180 576 432], ...
   'PaperUnits','normalized', ...
   'Resize','off',...
   'color','b',...
   'ToolBar','figure',...
   'Position',[0 0.05 1 0.84], ...
   'Tag','Fig9');


%*******************************
%** creación de menus *********

      psdgraf=uimenu('parent',densi,...
      'interruptible', 'off', 'busyaction', 'cancel',  ...
      'Label','&Transformada de Fourier',
'Tag','mnudiferencia','Callback','pantalla3(''fftgraf'')');


   espectro= uimenu('Parent',densi, ...
      'Callback','pantalla2', ...
      'interruptible', 'off', 'busyaction', 'cancel',  ...
      'Label','&Espectrograma', 'Tag','espectro','Separator', 'on');

   densidad=uimenu('Parent',densi,...
   'Callback','pantalla3', ...
   'interruptible', 'off', 'busyaction', 'cancel',  ...
   'Label','&Densidad Espectral', 'Tag','densidad');

   mnuSalir = uimenu('Parent',densi, ...
      'interruptible', 'off', 'busyaction', 'cancel',  ...
      'Label','&Salir', 'Tag','mnudiferencia','Callback','close(gcf)');



   y=wavread('estimacion.wav');
   x=wavread('referencia.wav');
   y=y/max(abs(y));
   x=x/max(abs(x));

   Ty=size(y,1);
   Tx=size(x,1);
   corr=zeros(Ty-Tx,1);
   x=[x;corr];
   zp=x-y;
   subplot(3,1,1)
   plot(x,'y');
   title(['Señal de refrencia']);
   ylabel('Amplitud');
```

```
    %xlabel('Interacción');
    set(gca,'Color','k'); set(gca,'Xcolor','w'); set(gca,'YColor','w');

    subplot(3,1,2)
    plot(y,'y');
    title(['Señal estimada']);
    ylabel('Amplitud')
    %xlabel('Interacción');
    set(gca,'Color','k'); set(gca,'Xcolor','w'); set(gca,'YColor','w');

    subplot(3,1,3)
    plot(abs(zp),'y');
    title(['Error entre achivos']);
    xlabel('Interacción');ylabel('Amplitud')
    set(gca,'Color','k'); set(gca,'Xcolor','w'); set(gca,'YColor','w');

    drawnow;

end
```

```
function pantalla51(opc)
global x y fm densi;
global xlong;
global SD_PLOTCMD;
global SD_DISP ;
global plotSpec ;
global xl x x3 xs;
global yp yp1 yps1 yps;
global z zs zp tiempo tiempo2  ;
global dato frec dato2 frec2;


if nargin<1
   opc='inicio';
end

switch opc
   case 'inicio'
  % colordef black
   densi = figure('Units','normalized', ...
   'MenuBar','none', ...
   'Name','Error', ...
   'NumberTitle','off', ...
   'PaperPosition',[18 180 576 432], ...
   'PaperUnits','normalized', ...
   'Resize','off',...
   'color','b',...
   'ToolBar','figure',...
   'Position',[0 0.05 1 0.84], ...
   'Tag','Fig9');


%*******************************
%** creación de menus *********


   psdgraf=uimenu('parent',densi,...
      'interruptible', 'off', 'busyaction', 'cancel',  ...
      'Label','&Transformada de Fourier',
'Tag','mnudiferencia','Callback','pantalla31(''fftgraf'')');


   espectro= uimenu('Parent',densi, ...
      'Callback','pantalla21', ...
      'interruptible', 'off', 'busyaction', 'cancel',  ...
      'Label','&Espectrograma', 'Tag','espectro','Separator', 'on');

   densidad=uimenu('Parent',densi,...
      'Callback','pantalla31', ...
      'interruptible', 'off', 'busyaction', 'cancel',  ...
      'Label','&Densidad Espectral', 'Tag','densidad');


   mnuSalir = uimenu('Parent',densi, ...
      'interruptible', 'off', 'busyaction', 'cancel',  ...
      'Label','&Salir', 'Tag','mnudiferencia','Callback','close(gcf)');



est=wavread('estimacion.wav');

ref=wavread('referencia.wav');

load datos1
Tx=size(est,1);
Ty=size(ref,1)
   corr=zeros(Tx-Ty,1);
   ref=[ref;corr];


   e=est-ref;
```

```
    subplot(2,1,1)
    plot(20*log10(abs(e)),'y')
    title('Bloque LMS Curva del error residual')
    xlabel('Iteraciones')
    ylabel('Amplitud potencia del error residual (dB)')
    set(gca,'Color','k'); set(gca,'Xcolor','w'); set(gca,'YColor','w');


    load coef a;
    subplot(2,1,2)
    plot(a,'y')
    title('Coeficientes del filtro')
    xlabel('Coeficintes')
    ylabel('Amplitud')
    set(gca,'Color','k'); set(gca,'Xcolor','w'); set(gca,'YColor','w');

    drawnow;

end
```

# ANEXO F

# GLOSARIO DE TÉRMINOS

**Amplitud**

Cantidad de una señal

**Atenuación**

Disminución de la amplitud de una señal

**Banda**

Rango de frecuencias entre dos limites definidos

**Causal**

Un sistema es **causal** si su salida en cualquier instante depende sólo de los valores de la entrada en el instante actual o en instantes anteriores.

**Canal**

Vía (canalización) de telecomunicaciones con una determinada capacidad (velocidad) entre dos ubicaciones de una red

**Central Telefónica**

Es un dispositivo que distribuye el uso compartido de recursos telefónicos comunes.

**Coeficientes de ponderación**

Coeficientes del filtro adaptivo.

**Combinador lineal adaptivo**

Estimador del tipo no recursivo, donde la salida se define en términos de una combinación lineal de la señal de entrada.

**Complejidad** (computacional)

Número de operaciones que han de realizarse en cada instante para implementar el algoritmo adaptivo

**Conmutación**

Función de elegir y concretar un camino interno que interconecte dos líneas, en base a la información que se recibe de la fuente (abonado u otra central).

**Convergencia**

Proceso de elaboración de un modelo de eco que se utilizará en el estimador de eco para obtener la estimación del eco del circuito.

**Correlación**

Es una operación entre dos secuencias, cuyo objetivo es medir el parecido que existe entre dos señales y así extraer la información que dependerá de la aplicación concreta considerada.

**Curva de aprendizaje**

Es la representación entre el módulo del error frente al número de interacciones.

**Densidad espectral de Potencia**

Es al representación de la magnitud de la potencia en el dominio de la frecuencia.

**Desajuste**

Mide la diferencia entre la solución de Wiener y la obtenida con el algoritmo adaptivo.

**Discontinuidad**

Instante en el cual la señal no esta definida.

**Dispersión**

La dispersión es el fenómeno por el cual un pulso se deforma a medida que se propaga a través de un medio de comunicación.

**DSP**

Procesamiento Digital de Señales.

**DTFT**

Transformada Discreta de Fourier.

**Efecto Gibbs**

Es el aparecimiento de un pico en puntos de discontinuidad cuando se reconstruye señales discontinuas a partir de las series de Fourier.

**ERL**

Pérdida de retorno de eco.

**MSE**

Error cuadrático medio.

**Espectro**

Conjunto de frecuencias determinadas.

**Espectrograma**

Gráfica que muestra la frecuencia en función del tiempo.

**Estabilidad**

Un sistema estable es aquel en que entradas pequeñas producen salidas que no divergen, es decir, salidas acotadas.

**Estructura Lattice**

Forma de implementar un filtro FIR.

**Factor de convergencia**

Es un escalar que controla la estabilidad y la velocidad de convergencia.

**FIR**

Finite Impulse Response: característica de la función de transferencia de solo ceros.

**Frecuencia**

Número de ciclos por unidad de tiempo de una onda sonora.

**Frecuencia de muestreo**

Es el número de muestras por segundo que se toman de una señal en particular.

**Teorema de Nyquist**

Se lo utiliza para la digitalización de señales que señala que a frecuencia mínima de muestreo debe ser igual o mayor a dos veces la máxima frecuencia de la señal a tratar.

**Función de transferencia**

Ecuación que sirve para representar un modelo discreto.

**Gradiente**

Diferenciación de una función, la cual nos sirve para encontrar puntos máximos o mínimos de esta función.

**Híbrida**

Se utilizan para desacoplar los circuitos de habla, pasando de 2 hilos a 4 hilos y viceversa.

La transmisión se hace a 4 hilos, lo que permite utilizar dispositivos (amplificadores, etc) de naturaleza unidireccional.

**IDFT**

Transformada discreta Inversa de Fourier.

**IIR**

Infinite Impulse Response: característica de la función de transferencia de solo polos o de polos y ceros.

**Interferencia**

Señal no deseable que se introduce en un sistema a tratar.

**ISC**

Centro de Conmutación Internacional.

**LMS**

Algoritmo adaptivo de mínimos cuadrados medios.

**MATLAB**

Programa interactivo para computación numérica y visualización de datos o de mínimos cuadrados medios.

**Muestra**

Amplitud de una señal en un instante determinado.

**NLP**

Procesador no lineal.

**PCM**

Pulse Code Modulación

**Pesos**

Coeficientes de ponderación.

**PSTN**

Red publica conmutada.

**Secuencias simétricas**

Representa la respuesta a impulso de un filtro ideal.

**Ventanas espectrales**

Utilizadas para reducir el efecto de un truncamiento abrupto.

**Velocidad de convergencia**

Es la taza a la que le filtro adaptivo converge a la solución de Wiener.