

UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERIA Y ARQUITECTURA
ESCUELA DE INGENIERIA ELECTRICA



**“Diseño de un filtro digital adaptativo
como cancelador de ruido
basado en el algoritmo LMS”**

TRABAJO DE GRADUACION
PRESENTADO POR:

WALTER LEOPOLDO ZELAYA CHICAS

PARA OPTAR AL TITULO DE:

INGENIERO ELECTRICISTA

SAN SALVADOR, ENERO 2004

UNIVERSIDAD DE EL SALVADOR

RECTORA:

Dra. María Isabel Rodríguez

SECRETARIA GENERAL:

Licda. Lidia Margarita Muñoz Vela

FACULTAD DE INGENIERIA Y ARQUITECTURA

DECANO:

Ing. Mario Roberto Nieto Lovo

SECRETARIO:

Ing. Oscar Eduardo Marroquín Hernández

ESCUELA DE INGENIERIA ELECTRICA

DIRECTOR:

Ing. Luis Roberto Chevéz Paz

**UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERIA ELECTRICA Y ARQUITECTURA
ESCUELA DE INGENIERIA ELECTRICA**

Trabajo de Graduación previo a la opción al grado de:

INGENIERO ELECTRICISTA

Título:

**“Diseño de un filtro digital adaptativo como cancelador de ruido
basado en el algoritmo LMS”**

Presentado por:

Br. Walter Leopoldo Zelaya Chicas

Trabajo de Graduación aprobado por:

Coordinador y Asesor:

Ing. Carlos Eugenio Martínez Cruz

San Salvador, enero de 2004

Trabajo de Graduación aprobado por:

Coordinador y Asesor:

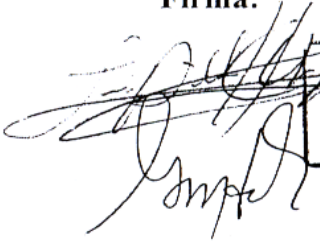
Ing. Carlos Eugenio Martínez Cruz

ACTA DE CONSTANCIA DE NOTA Y DEFENSA FINAL

En esta fecha, 9 de diciembre de 2003, en el local de Sala de Lectura de la Escuela de Ingeniería Eléctrica, a las diecisiete horas, en presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:

1. Ing. Luis Roberto Chévez Paz
Director
2. Ing. Gerardo Marvin Hernández
Secretario

Firma:

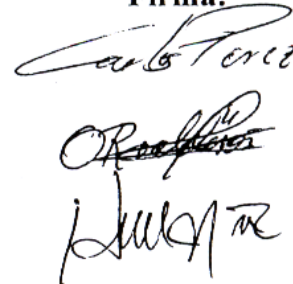


ESCUELA DE INGENIERIA ELÉCTRICA
FACULTAD DE INGENIERIA
Y ARQUITECTURA
Universidad de El Salvador

Y, con el Honorable Jurado de Evaluación integrado por las personas siguientes:

1. Ing. Carlos Ernesto Pérez Ramos
2. Ing. Octavio Rafael Calderón Escobar
3. Ing. Hugo Miguel Colato

Firma:



Se efectuó la defensa final reglamentaria del Trabajo de Graduación:

“Diseño de un filtro digital adaptativo como cancelador de ruido basado en el algoritmo LMS”

A cargo del Bachiller:

ZELAYA CHICAS, WALTER LEOPOLDO

Habiendo obtenido el presente Trabajo una nota final, global de: 7.9

(Siete punto nueve)

AGRADECIMIENTOS

- A Jehová Dios, por haberme acompañado durante toda mi vida, así como en mis estudios y poner a tantas personas que contribuyeron para que pudiera concluir esta tarea.
- A mis padres, por su esfuerzo y sacrificio, haciendo posible el éxito en mi carrera y a mis hermanos por darme su ayuda incondicional.
- A los padres de mi mamá, mi abuelo José y mi abuela Mercedes, por su ayuda económica y estímulo.
- A mi tío y mis tías maternos, por su ayuda en diversas circunstancias.
- A la familia González Salguero, por haberme mostrado su aprecio en mis últimos años de estudio.
- A la Universidad de El Salvador, por brindarme la oportunidad de obtener mi formación profesional en sus recintos.
- Al ingeniero coordinador y asesor, Carlos Eugenio Martínez Cruz, por la ayuda y conducción de este Trabajo de Graduación.
- A mis profesores, tanto de la Facultad Multidisciplinaria Oriental como de la Escuela de Ingeniería Eléctrica, por compartir sus conocimientos.
- Al personal del Laboratorio de la Escuela de Ingeniería Eléctrica, los señores Salvador Posada y Juan Olano, por la colaboración prestada en el desarrollo de este trabajo.
- A mis compañeros de estudio, por su apoyo y palabras de ánimo.
- A todas aquellas personas que no se mencionan por nombres, pero que de una u otra forma colaboraron conmigo en la realización de este proyecto.

Walter Zelaya

DEDICATORIA

Dedico este Trabajo de Graduación a las personas que siempre han estado a mi lado, en las buenas y en las malas:

- A MI MADRE:
EGDOMILIA CHICAS
- Por su gran amor y cariño mostrado, por ser una fuente de guía y estímulo, por enseñarme a tener fortaleza para continuar sin importar las dificultades que se presenten, por estar ahí cuando más la necesitaba, por siempre tener palabras de aliento y por todo el excelente trabajo como madre.
- A MI PADRE:
LEOPOLDO ZELAYA
- Por apoyar la decisión de que continuara con mi formación académica y por el aporte hecho para que culminara mi carrera.
- A MI HERMANO:
RAFAEL ZELAYA
- Por ser fuente de ánimo y alegría en los momentos difíciles, por haber sido el principal sostén económico en los últimos años y por darme la confianza y apoyo cuando lo necesitaba.
- A MI HERMANO:
NATAN ZELAYA
- Por toda una vida de compañía y amistad, por compartir las alegrías y las penas de la vida, los logros y las frustraciones en mis años de estudio y por el apoyo incondicional.
- A MI TIA:
EMMA CHICAS
- Por su gran ejemplo de superación personal ante circunstancias adversas, por su amor y apoyo abnegado a la familia, por su espíritu de lucha insistente y por su arduo trabajo.
- A MIS PRIMOS Y PRIMAS
MATERNOS:
- René, Leslie, Ricardo, Rafael, Rocío, Samuel, René, Rebeca, Yadira, Carlos, Andersson, Eliana.

Walter Zelaya

Contenido

Introducción ix

Capítulo 1 Introducción a filtros adaptativos 1

- 1.1 El problema del filtrado 2
- 1.2 Filtros adaptativos 3
- 1.3 Estructuras de los filtros lineales 6
- 1.4 El desarrollo de algoritmos utilizados en filtros adaptativos lineales 8
- 1.5 Aplicaciones 9
- 1.6 Cancelador adaptativo de ruido 13
- 1.7 Conclusiones 21
- 1.8 Referencias 21

Capítulo 2 Algoritmo Least Mean Square (LMS) 22

- 2.1 Introducción 23
- 2.2 Filtros adaptativos lineales 23
- 2.3 Apreciación general de la estructura y operación del algoritmo LMS 26
- 2.4 Algoritmo de adaptación del LMS 31
- 2.5 El algoritmo LMS en la práctica 34
- 2.6 Conclusiones 35
- 2.7 Referencias 37

Capítulo 3 Simulación del algoritmo LMS, utilizando Matlab 38

- 3.1 Introducción 39
- 3.2 El LMS en Matlab 40

- 3.3 Aplicación: Cancelador de ruido 44
- 3.4 Conclusiones 54
- 3.5 Referencias 54

Capítulo 4 Implementación del LMS usando la DSP56L811EVM 55

- 4.1 Introducción 56
- 4.2 Descripción general de la DSP56L811EVM 56
- 4.3 Tratamiento digital de señales 62
- 4.4 Preparación para el uso de la DSP56L811EVM 67
- 4.5 Programa LMS en lenguaje ensamblador 70
- 4.6 Conclusiones 78
- 4.7 Referencias 79

Anexo A Características de Matlab 80

Anexo B Uso de la DSP56L811EVM 85

Anexo C Filtros Wiener 99

Anexo D Programas para simulación 108

Anexo E Programas en lenguaje ensamblador 117

Introducción

En la actualidad los filtros adaptativos constituyen una importante parte del procesamiento digital de señales. Donde se requiere procesado de señales que resultan de operaciones en un ambiente desconocido estadísticamente, el uso de filtros adaptativos ofrece una atractiva solución al problema que usualmente no pueden solventar los filtros de coeficientes fijos, obtenidos por métodos convencionales.

Además, el uso de filtros adaptativos provee una nueva herramienta para procesado de señales que ni se consideraba posible realizar con filtros de coeficientes fijos.

Como tema principal de este trabajo de graduación, es el filtrado adaptativo lineal realizado en una estructura transversal y con una respuesta al impulso de duración finita (FIR). La aplicación a la que se dedica este documento es a la cancelación de ruido de señales periódicas acotadas en frecuencia mediante filtrado adaptativo, necesaria para muchas y variadas situaciones reales. Para ello, se ha diseñado un sistema de Control Activo de Ruido, (CAR o ANC del inglés Active Noise Control) utilizando un algoritmo con características lineales y un filtro de estructural transversal.

Aunque ésta no es la única línea de trabajo que soluciona el problema de filtrado adaptativo lineal (sin considerar los no lineales), este trabajo de graduación será un primer paso al amplio mundo del filtrado adaptativo lineal.

Finalmente, se escogió el algoritmo LMS (Least Mean Square) como herramienta matemática para conseguir una solución al problema de cancelación de ruido.

Organización del presente trabajo

El trabajo de graduación comienza en el capítulo 1 con una introducción sobre lo que es un filtro adaptativo lineal en términos prácticos. Además, muestra las bondades de usar una estructura transversal. Finalmente hace la presentación oficial de la aplicación sobre la que versa este trabajo de graduación: cancelación de ruido.

Luego en el capítulo 2, se presenta el LMS, algoritmo miembro de la familia de gradiente estocástico, que se deduce a partir de los filtros Wiener. Este algoritmo esta relacionado estrechamente con el método de máxima pendiente, situación que se aprovecha para explicar el LMS usando la superficie de error.

El capítulo siguiente, el 3, describe todas las características matemáticas y matriciales del algoritmo LMS para su codificación en lenguaje de Matlab. Ahí mismo se presenta el código de programa usado, así como los resultados obtenidos de la simulación. Todos los programas a los que se hace referencia, se encuentran escritos en los anexos, pero también, en una 'toolbox' o grupo de programas para simular filtrado adaptativo en el CD que acompaña este trabajo.

Finalmente el capítulo 4, muestra la implementación del algoritmo LMS en la tarjeta de evaluación DSP56L811EVM, fabricada por Motorola. Se explica segmentos de código, se describe la manera de compilar (ensamble) y enlazar (link) los programas para correrlos en la DSP56L811EVM y se muestran los resultados obtenidos tanto con una señal almacenada en memoria como en tiempo real.

CAPITULO

1

Introducción a filtros adaptativos

1.1 EL PROBLEMA DEL FILTRADO

El término *filtro* a menudo es usado para describir un circuito eléctrico (hardware) o un programa de computadora (software) que se aplica a un conjunto de datos ruidosos (o contaminados) para extraer una cantidad de información previamente definida como la respuesta deseada o la señal de interés. El ruido podría aparecer de una variedad de fuentes. Por ejemplo, los datos se pueden haber obtenido por medio de sistemas altamente sensitivos que junto con la señal de interés agregan ruido, o el ruido podría representar una parte útil de la señal como cuando ha sido modulada con otra frecuencia para transmitirla a través de un canal de comunicaciones. En todo caso, lo que se puede hacer es valerse de un filtro para realizar tres tareas básicas de procesamiento sobre la señal:

1. *Filtrado*, Consiste en extraer la información o datos de interés en un tiempo t de una señal contaminada y emplear esos datos al mismo tiempo con el fin de actualizar la salida ya sin ruido.
2. *Suavizado*, Difiere del filtrado porque la información o datos de interés no están a veces disponibles, y se tiene que usar los datos obtenidos en otro tiempo t . Esto significa que en el caso del suavizado, se produce un retardo en la salida.
3. *Predicción*, Este es un tratamiento preventivo que se le hace a la señal que contiene la información de interés. El objetivo aquí es obtener la cantidad de información de interés de la señal que se enviará algún momento $t+\tau$ en el futuro, para algún $\tau > 0$.

Cuando se busca una aproximación para resolver el problema de filtrado lineal, se dispone de ciertos parámetros estadísticos (es decir, funciones como media aritmética y correlación) de la señal deseada y el ruido aditivo no deseado, y el objetivo es diseñar un filtro lineal con los datos ruidosos como entrada, luego minimizar los efectos de dicho ruido a la salida del filtro según algún criterio estadístico. Una solución útil a este problema de optimización del filtro es minimizar el valor cuadrático medio de la señal de error que se define como la diferencia entre alguna respuesta deseada y la salida

actual del filtro. Para las entradas estacionarias, una buena solución son los filtros Wiener, que son óptimos al lograr reducir el error cuadrático medio. La gráfica del valor cuadrático medio de la señal de error contra los coeficientes ajustables de un filtro lineal es llamada superficie de comportamiento del error o solamente *superficie de error*. El punto mínimo de esta superficie representa la solución óptima de Wiener.

Cuando existen señales dinámicas no estacionaria como la voz, es necesaria la adaptación del filtro a los bruscos cambios de la señal de entrada. Mientras que un filtro transversal de coeficientes fijos (tipo Wiener) solo se aplicaría a señales que son procesos estacionarios para los cuales se conocen las propiedades estadísticas, el propósito de un filtro adaptativo es justamente no depender de ésta hipótesis, generalmente falsa en la realidad, para funcionar. Un filtro adaptativo es aquel cuyos coeficientes son actualizados mediante un algoritmo que cumple con un criterio predefinido, que puede ser minimizar el error cuadrático medio, como es el caso del LMS. La velocidad de esta adaptación puede variar según la implementación y el tipo de señales que se manejen. Es evidente que una actualización de los coeficientes con mayor frecuencia o velocidad permite obtener una mejor respuesta del filtro, por eso es común que se calculan de nuevo los coeficientes con cada muestra. Sin embargo, recalculer los coeficientes con cada nueva muestra aumenta el número de cálculos matemático a realizar. Por otro lado, el desempeño del filtro mejora cuando se aumenta el número de coeficientes.

Los filtros en general podemos clasificarlos en lineales y no lineales. Se dice que un filtro es lineal si el filtrado, suavizado, o predicción a la salida, es una función lineal de las muestras individuales aplicadas a la entrada del filtro. De otra manera, el filtro no es lineal.

1.2 FILTROS ADAPTATIVOS

El diseño de un filtro Wiener requiere información *previa* sobre la situación estadística de los datos para poder ser procesados. El filtro sólo es óptimo cuando las características de los datos de entrada se conocen previamente. Cuando esta información no es conocida completamente, no es posible diseñar filtros Wiener y el resto del diseño

ya no puede ser óptimo. Una forma con la que se puede encontrar la solución a ese problema, es realizar el proceso de “estimación y actualización”. Este es un proceso que consta de dos fases por medio de las cuales el filtro, primero “estima” los parámetros estadísticos de la señal (esto es, los coeficientes) y entonces “actualiza” los resultados obtenidos en la fórmula no recursiva (la salida del FIR) para calcular el error y luego de nuevo comenzar la estimación. Para el funcionamiento en tiempo real, este procedimiento tiene la desventaja de requerir circuitos electrónicos excesivamente detallados y costosos. Un método eficaz y menos costoso es usar un filtro adaptativo implementado en un Procesador Digital de Señales (DSP, por sus siglas en inglés).

El algoritmo que se decida emplear empieza fijando un conjunto predeterminados de condiciones iniciales. En un ambiente estacionario, aun es posible encontrar que el algoritmo converja a la solución óptima de Wiener. Pero, en un ambiente no estacionario, el algoritmo debe ofrecer una capacidad de rastreo, eso significa rastrear variaciones en el tiempo de los datos estadísticos a la entrada. Como una consecuencia directa de la aplicación de un algoritmo recursivo, se actualizan los parámetros de un filtro adaptativo en cada iteración, los parámetros se vuelven datos dependientes de la entrada y del error. Esto por lo tanto significa que un filtro adaptativo es en realidad un dispositivo no lineal, en el sentido de que no obedece el principio de superposición. A pesar de esta propiedad, los filtros adaptativos son clasificados normalmente como lineales o no lineales. Se dice que un filtro adaptativo es lineal si la estimación de una cantidad de interés se calcula adaptativamente (a la salida del filtro) como una combinación lineal del conjunto de muestras disponible a la entrada del filtro [Haykin, 1996].

A final de cuentas lo que determina cual algoritmo es el que debe usarse, son los siguientes factores:

- *Velocidad de convergencia.* Esto se define como el número de iteraciones requeridas por el algoritmo para alcanzar la solución óptima de Wiener. Una tasa o velocidad alta permite que la convergencia se alcance rápidamente, pero sacrificando el ajuste más óptimo.

- *Desajuste.* Mide la diferencia entre la solución de Wiener y la obtenida con el algoritmo adaptativo que se está usando. Generalmente, el desajuste es inversamente proporcional a la velocidad de convergencia.
- *Rastreo.* Cuando un algoritmo de un filtro adaptativo opera en un ambiente no estacionario, el algoritmo tiene que rastrear las variaciones estadísticas en el ambiente. El comportamiento de rastreo del algoritmo, sin embargo, está influenciada por dos características opuestas: (a) la velocidad de convergencia, y (b) la fluctuación del estado estable debido al ruido del algoritmo.
- *Robustez.* Para que un filtro adaptativo sea robusto, debe cumplirse que pequeños disturbios (es decir, perturbaciones con poca energía) sólo causen errores de estimación pequeños. Las perturbaciones pueden originarse por una variedad de factores, internos o externos al filtro.
- *Requisitos computacionales.* Aquí lo que preocupa es (a) el número de operaciones (es decir, multiplicaciones, divisiones, y sumas/restas) requeridas en cada iteración completa del algoritmo y (b) el tamaño de la memoria disponible para guardar los datos y el programa.
- *Estructura.* Esto se refiere a la ruta a seguir por el flujo de información dentro del algoritmo y está determinada por los criterios de diseño del programa. Por ejemplo, un algoritmo podría ser de estructura de programación modular (con subrutinas), o lineal (procedimientos seguidos, uno después del otro), operaciones en paralelo, etc.
- *Propiedades numéricas.* Cuando se ejecuta un algoritmo, numéricamente se producen inexactitudes debido a los errores de cuantificación. Los errores de cuantificación se deben a la conversión analógico-digital de los datos de entrada y la manipulación digital de los cálculos interiores. Comúnmente, es la última fuente de errores que se evalúa, pero significa un problema muy serio de diseño si no se toma en cuenta.

Estos factores, también se toman en cuenta para el diseño de filtros adaptativos no lineales, con la particularidad que ya no se tiene un marco bien definido de referencia como lo son los filtros Wiener.

1.3 ESTRUCTURAS DE LOS FILTROS LINEALES

El funcionamiento de un algoritmo para filtrado adaptativo lineal involucra dos procesos básicos: (1) un proceso de *filtrado* diseñado para producir una salida correspondiente a los datos de entrada, y (2) un proceso *adaptativo*, con el propósito de mantener un mecanismo de control adaptativo en un conjunto ajustable de parámetros que se usarán en el proceso de filtrado. Estos dos procesos trabajan interactivamente entre sí. Naturalmente, la opción de una estructura para el proceso de filtro tiene un efecto profundo en conjunto con el funcionamiento del algoritmo.

Los filtros FIR (Respuesta Finita al Impulso) son normalmente usados en aplicaciones de filtrado adaptativo para la ecualización de sistemas de comunicación y sistemas de control de ruido. Hay varias razones por las que se han hecho populares los filtros adaptativos FIR. Primero, estabilidad y fácil control gracias a que los coeficientes del filtro son limitados. Segundo, simplicidad y eficiencia del algoritmo para ajustar los coeficientes del filtro. Tercero, el desempeño de estos algoritmos es predecible en términos de convergencia y estabilidad [Hayes, 1996].

El algoritmo adaptativo debe disponer del error $e(n)$ para actualizar los coeficientes, ya que $e(n)$ permite definir las mejoras del filtro y determinar la forma en que han de modificarse dichos coeficientes.

La eficiencia de un filtro adaptativo lineal depende de una serie de factores, como el tipo de filtro (IIR o FIR), la estructura del mismo (transversal, de celosía o sistólico), o la función de costo usada como criterio de adaptación (error cuadrático medio, mínimo error cuadrático, etc.).

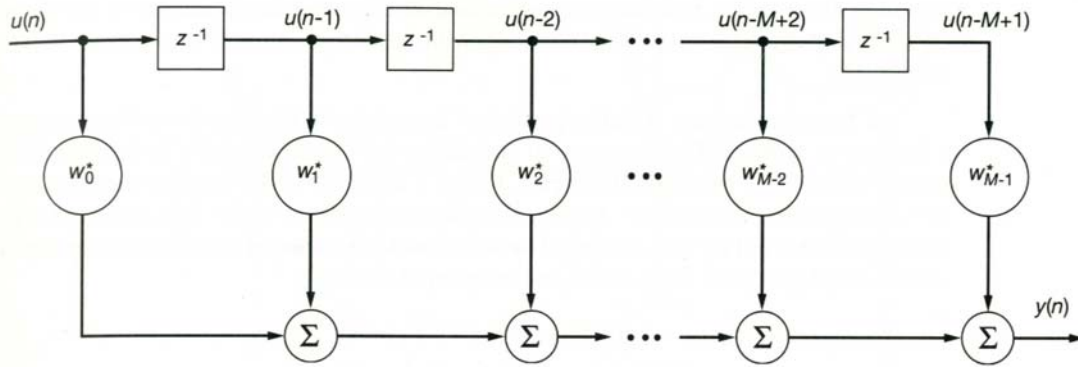


Figura 1.1 Filtro transversal

Nuestro estudio se va a centrar 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 de error es un paraboloide con sólo un mínimo, y por ello, la búsqueda del error cuadrático medio mínimo es relativamente sencilla.
- ◆ Dado que los coeficientes del filtro son limitados, se puede controlar fácilmente la estabilidad del filtro.
- ◆ Existen algoritmos para la actualización de los coeficientes que con filtros FIR son mucho más simples y eficientes.
- ◆ Las prestaciones de estos algoritmos son perfectamente conocidas en términos de convergencia y estabilidad.

De los tres tipos de estructuras de filtro que se distinguen en el contexto de un filtro adaptativo lineal, vamos a conceder toda la atención al tipo transversal.

Los filtros transversales, también llamados filtros directos de pesos retardados, consisten en tres elementos básicos, como se describió en la Figura 1.1: (a) *elementos de unidad de retardo*, (b) el *multiplicador*, y (c) la *sumatoria*. El número de elementos de retardo usados en el filtro, determinan la respuesta finita al impulso (FIR). El número de elementos de retardo, mostrado como $M - 1$ en Figura 1.1, normalmente está referido al

orden del filtro.¹ En esta figura, los elementos de retardo son identificados por el operador de la unidad de retraso z^{-1} . En particular, cuando z^{-1} opera en la entrada $u(n)$, la salida resultante es $u(n - 1)$. El papel de cada multiplicador en el filtro es realizar el producto del *valor de entrada* por un *coeficiente del filtro*. Así un multiplicador conectado k veces a entrada retardadas $u(n - k)$ produce la versión del escalar de que es el *producto interno*, $w_k^* u(n - k)$, donde w_k va desde $k=1$ a M . El asterisco denota la *conjugación compleja* que asume la entrada y por consiguiente también los pesos actualizable son valores complejos. El papel combinado de las sumatorias en el filtro es sumar los resultados de los productos individuales y producir una salida total del filtro. Para el filtro transversal descrito en la Figura 1.1, la salida está dada de la siguiente manera:

$$y(n) = \sum_{k=0}^{M-1} W_k \cdot u(n - k) \quad (1.1)$$

La ecuación (1.1) es llamada *sumatoria de convolución finita*, en el sentido que la respuesta convolucionada el impulso de duración finita del filtro, w_n^* , con la entrada $u(n)$ del filtro, luego cada producto individual se suma para dar como resultado $y(n)$.

La estructura transversal es la más sencilla de implementar, conduciendo a algoritmos igualmente sencillos. La estructura de celosía, presenta mejores propiedades, pues ofrece mayor robustez frente a errores de redondeo y una mayor eficiencia computacional. Sin embargo, aumenta la complejidad de los algoritmos. Por lo tanto en el presente trabajo de graduación se adopta la primera estructura debido a su sencillez pero aún así con buen desempeño [DET, 2003].

1.4 EL DESARROLLO DE ALGORITMOS UTILIZADOS EN FILTROS ADAPTATIVOS LINEALES

No existe una solución única al problema de filtros adaptativos lineales. Más bien, se tiene un “conjunto de herramientas” representado por una variedad de

¹ En otros casos, los coeficientes son enumerados desde W_1 hasta W_M .

algoritmos recursivos, donde cada uno de los cuales ofrecen ventajas sobre las desventajas de los demás. El desafío que enfrenta el usuario de filtros adaptativos es, primero, poder entender las capacidades y limitaciones de un algoritmo y segundo, usar esta información para hacer la mejor selección del algoritmo más apropiado para la aplicación.

Dado la gran variedad de algoritmos para utilizarlos en filtros adaptativos lineales, todo diseñador se enfrenta a tomar una decisión. Claramente, cualquier opción, tiene que ser rentable. Con esto como meta, se puede identificar tres problemas importantes que requieren la atención: el costo computacional, desempeño, y robustez. El uso de simulación en computadora proporciona un buen primer paso para la investigación detallada de estos problemas. Nosotros podemos empezar usando el algoritmo LMS como una herramienta del filtrado adaptativo a manera de estudio. Dado que el algoritmo LMS es relativamente simple de entender. Sin embargo es bastante poderoso para evaluar los beneficios prácticos que pueden ser alcanzados en una aplicación adaptativa. Es más, es muy buena referencia para evaluar cualesquier otro algoritmo de filtrado adaptativo.

Las aplicaciones prácticas de filtrado adaptativo son muy diversas, y cada aplicación tiene sus propias necesidades. La solución para una aplicación no puede ser conveniente para otra. En nuestro caso las condiciones bajo las que desarrollaremos este proyecto ya están seleccionadas. Además de ser un filtro adaptativo lineal FIR, con estructura transversal, usando el algoritmo LMS, nuestra aplicación es **cancelación de ruido**. A continuación mostramos brevemente las otras aplicaciones y luego la que merece nuestra atención.

1.5 APLICACIONES

La habilidad de un filtro adaptativo de operar de manera satisfactoria en un ambiente desconocido rastreando las variaciones estadísticas en el tiempo de una entrada, hace a los filtros adaptativos poderosos dispositivos para aplicaciones de procesamiento de señales y control. Es por eso que los filtros adaptativos han sido abundantemente aplicados en campos diversos tales como las comunicaciones, radar,

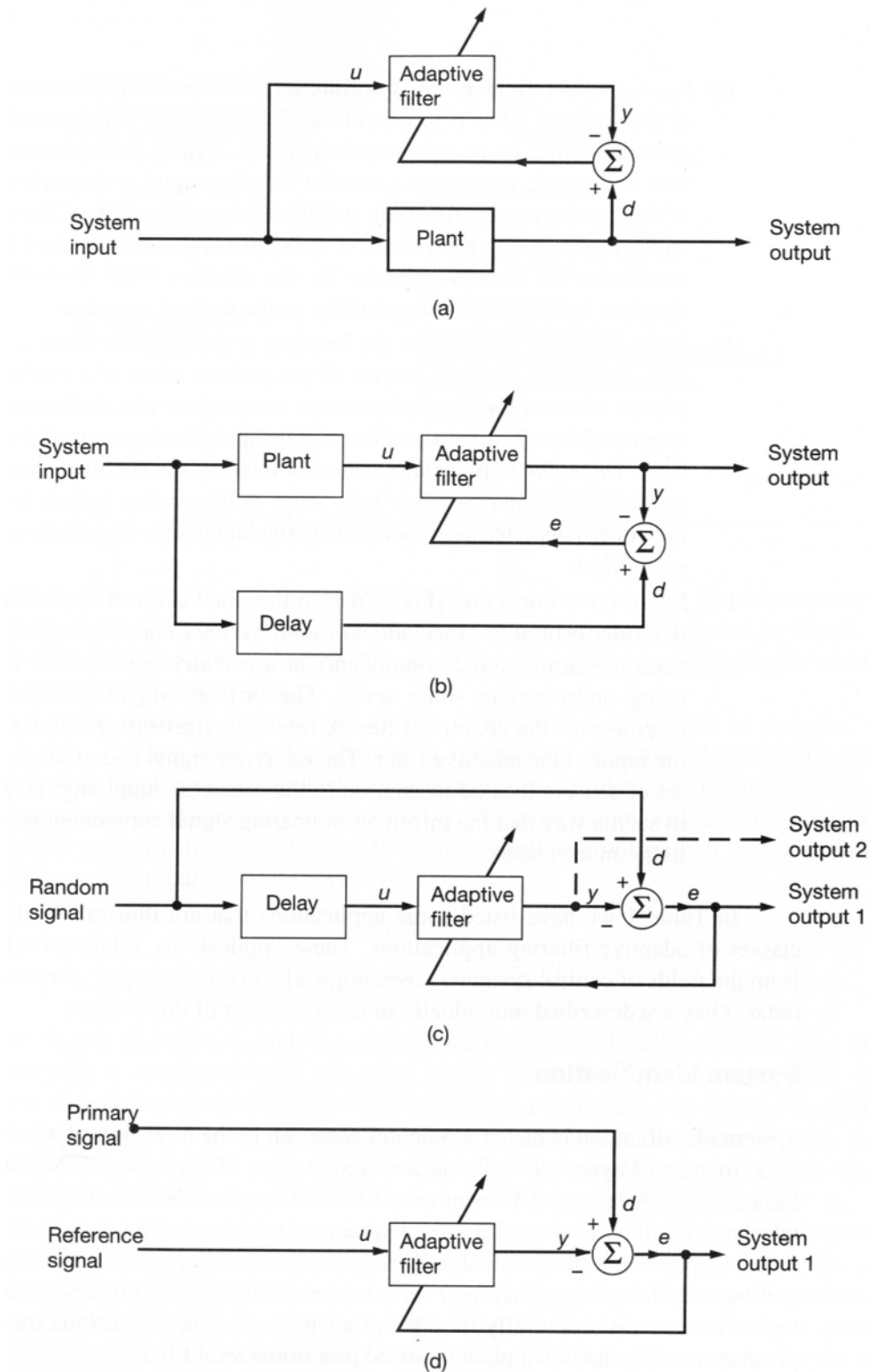


Figura 1.2 Cuatro clases de aplicaciones de filtros adaptativos (a) Clase I: Identificación; (b) Clase II: Modelado; (c) Clase II: Predicción; (d) Clase IV: Cancelación de interferencias

sonar, sismología, y la ingeniería biomédica. Aunque estas aplicaciones son de hecho bastante diferentes en naturaleza, no obstante, tienen un rasgo común básico: un vector de entrada y una respuesta deseada se usan para calcular el error de estimación, que a su vez se usa para controlar los valores de un conjunto de coeficientes ajustables. Los coeficientes ajustables pueden tomar la forma de pesos regulables (tap, del inglés), coeficientes de reflexión, parámetros de rotación, o pesos sinápticos, dependiendo de la estructura del filtro empleada. Sin embargo, la diferencia esencial entre las varias aplicaciones de filtrado adaptativo comienza con la manera como se extrae la respuesta deseada. En este contexto, podemos distinguir cuatro clases básicas de aplicaciones de filtros adaptativos, como se describen en la Figura 1.2. Para la conveniencia de la presentación, las siguientes anotaciones se usan en esta figura:

u = la entrada aplicada al filtro adaptativo²

y = la salida del filtro adaptativo

d = respuesta deseada

$e = d - y$ = error de estimación.

Las funciones de las cuatro clases básicas de aplicaciones de filtros adaptativos descritas aquí son como sigue:

I. Identificación [Figura 1.2(a)]. La notación de *un modelo matemático* es fundamental para la ciencia e ingeniería. En la clase de aplicaciones que tratan con identificación, un filtro adaptativo se usa para proporcionar un modelo lineal que represente el que mejor se adecua (en algún sentido) con una planta o *sistema desconocido*. La planta y el filtro adaptativo poseen la misma entrada. La salida del filtro es un conjunto de coeficientes que caracterizan el sistema desconocido. Si la planta es de naturaleza dinámica (sistema o señal no estacionaria), el modelado será variante en el tiempo.

² Otros autores utilizan X en lugar de U . En este trabajo de graduación se usan ambas representaciones de la señal de entrada.

II. Modelado inverso [Figura 1.2(b)]. En esta segunda clase de aplicaciones, la función del filtro adaptativo es proporcionar un *modelo inverso* que representa el que mejor se adapta (hasta cierto grado) a una *planta ruidosa desconocida*. Idealmente, en el caso de un sistema lineal, el modelo inverso tiene una función de transferencia igual al *recíproco (inverso)* de la función de transferencia de la planta. Una versión retardada de la planta (sistema) es la señal de entrada al filtro adaptativo que constituye la respuesta deseada. En algunas aplicaciones, la entrada de la planta se usa sin retardo como la respuesta deseada.

III. Predicción [Figura 1.2(c)]. Aquí la función del filtro adaptativo es proporcionar la mejor *predicción* (hasta donde sea posible) del valor presente de una señal aleatoria. El valor presente de la señal es la respuesta deseada para el filtro adaptativo. Los valores pasados de la señal son aplicados a la entrada del filtro adaptativo. Dependiendo de la aplicación de interés, la salida del filtro adaptativo o la estimación (predicción) del error, podría ser la salida que se busca. En el primer caso, el sistema opera como un predictor; en el último caso, opera como un filtro de predicción-error. Estas dos posibilidades existen, gracias a que el predictor lo que ha hecho, en realidad, es separar dos señales.

IV. Cancelación de Interferencia [Figura 1.2(d)]. En esta clase final de aplicaciones, el filtro adaptativo se usa para cancelar una *interferencia desconocida* contenida (sumada a una señal con información de interés) en una *señal primaria*. Con la cancelación se comienza optimizar la salida hasta cierto grado. La señal primaria sirve como la respuesta deseada para el filtro adaptativo. Una *señal de referencia (auxiliar)* es empleada como la entrada al filtro adaptativo. La señal de referencia se deriva de un sensor o conjunto de sensores localizados muy cerca del sensor que proporciona la señal primaria, solo que la componente del ruido en la señal primaria es débil o esencialmente indetectable y la componente del ruido en la señal de referencia resulta ser una proporción de la señal interferente desconocida.

En la Tabla 1.1 se han listado algunas aplicaciones que son ilustrativas de las cuatro clases básicas de filtros adaptativo. Estas aplicaciones ascienden a doce y son utilizadas en áreas de sistemas de control, sismología, electrocardiografía, comunicaciones y radar. En la siguiente sección se describen el caso particular del cancelador de ruido, que es el tema principal de esta Tesis.

TABLA 1.1 APLICACIONES DE FILTROS ADAPTATIVOS

Clase de filtro adaptativo	Aplicación
I. Identificación	Identificación de Sistemas Modelado de capas subterráneas
II. Modelado Inverso	Deconvolución Predictiva Ecuación Adaptativa Ecuación Ciega
III. Predicción	Codificación por Predicción Lineal (LPC) Codificación Diferencial Adaptativa (ADPCM) Análisis Espectral Autorregresivo Detección de Señal
IV. Cancelación de Interferencias	Cancelación Adaptativa de Ruido Cancelación de Eco Formas de haz Adaptativas

1.6 CANCELADOR ADAPTATIVO DE RUIDO

Como el nombre lo indica, el cancelador de ruido adaptativo confía en el uso de un *cancelador de ruido* para eliminar el ruido de una señal recibida. Ordinariamente, es imprudente quitar ruido de una señal recibida, debido a que semejante operación pudiera producir resultados desastrosos, causando un incremento en la potencia promedio del ruido de la salida. Sin embargo, cuando se hacen previsiones apropiadas y el control del filtrado y eliminación del ruido se realiza por un proceso adaptativo, es posible realizar un sistema superior de operación comparado al filtrado directo de la señal recibida [Haykin, 1996].

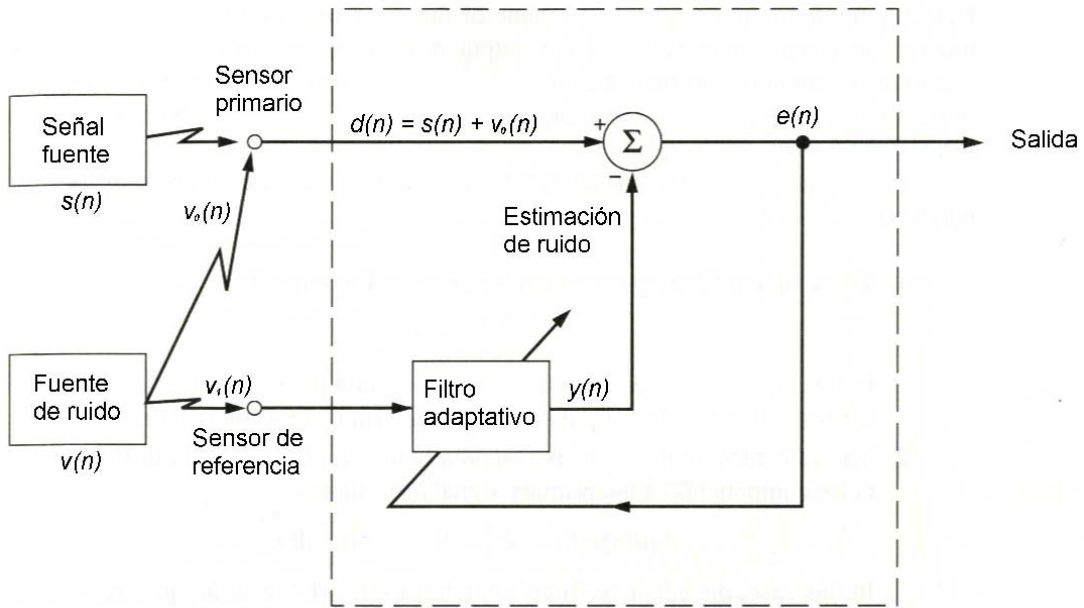


Figura 1.3 Cancelador de ruido adaptativo

Básicamente, un cancelador de ruido adaptativo es un arreglo de *entrada doble de lazo cerrado* del sistema de realimentación adaptativo como ilustra en la Figura 1.3. Se derivan las dos entradas del sistema de un par de sensores: un sensor primario y un sensor de referencia (auxiliar). Específicamente, nosotros tenemos lo siguiente:

1. El sensor primario recibe una señal que lleva información $s(n)$ adulterada por un ruido aditivo $v_0(n)$, como se muestra en la ecuación (1.2).

$$d(n) = s(n) + v_0(n) \quad (1.2)$$

La señal $s(n)$ y el ruido $v_0(n)$ están sin correlación una de la otra; esto es:

$$E[s(n)v_0(n-k)] = 0 \quad \text{para todo } k \quad (1.3)$$

Donde $s(n)$ y $v_0(n)$ se suponen que son valores reales.

2. El sensor de referencia recibe un ruido $v_1(n)$ que no tiene correlación con la señal $s(n)$ pero si tiene correlación con el ruido $v_0(n)$, el cual es obtenido con el sensor primario junto a la señal que se desea recuperar; esto es:

$$E[s(n)v_1(n-k)] = 0 \quad \text{para todo } k \quad (1.4)$$

y

$$E[v_0(n)v_1(n-k)] = p(k) \quad (1.5)$$

Donde, como antes, la señal es un valor real y $p(k)$ es una correlación cruzada desconocida para el retardo k .

La señal de referencia $v_1(n)$ es procesada por un filtro adaptativo para producir la salida:

$$y(n) = \sum_{k=0}^{M-1} \hat{w}_k(n)v_1(n-k) \quad (1.6)$$

Donde los $\hat{w}_k(n)$ son los valores de coeficientes (reales) ajustables del filtro adaptativo. La salida del filtro $y(n)$ es restada de la señal principal $d(n)$, conocida como la “respuesta deseada” para el filtro adaptativo. La señal de error se define por

$$e(n) = d(n) - y(n) \quad (1.7)$$

Así, sustituyendo la (1.2) en (1.7), se obtiene:

$$e(n) = s(n) + v_0(n) - y(n) \quad (1.8)$$

La señal de error es la que se utiliza para ajustar los valores de coeficientes del filtro adaptativo y el lazo de control alrededor de las operaciones de filtrado y sustracción están relacionados. Note que la señal que lleva la información es cierta parte de la señal $e(n)$, como se indica en ecuación (1.8).

La señal de error $e(n)$ constituye la salida total del sistema. De la ecuación (1.8) se puede ver, que la componente de ruido en la salida del sistema es $v_0(n) - y(n)$. Ahora comienza su labor el filtro adaptativo para minimizar el valor cuadrático medio de la señal de error $e(n)$. La señal esencial $s(n)$ que lleva la información, no es afectada por el cancelador de ruido adaptativo.

Así que, minimizando el valor cuadrático medio de la señal de error $e(n)$ es equivalente a minimizar el valor cuadrático medio del ruido de salida $v_0(n) - y(n)$. Con la señal $s(n)$ que permanece esencialmente constante, esto lleva a que *la minimización*

del valor cuadrático medio de la señal de error es ciertamente la misma como la maximización de la razón señal a ruido de la salida del sistema.

La operación de tratamiento de la señal descrita aquí, tiene ciertas características que son notables:

1. La operación de filtrado adaptativo es perfecta cuando

$$y(n)=v_0(n) \quad [e(n)=0]$$

En este caso, la salida del sistema esta libre de ruido y la cancelación de ruido es perfecta. Correspondientemente, la razón de señal a ruido de la salida es infinitamente grande.

2. La señal de referencia $v_1(n)$ esta completamente sin correlacionar tanto con la señal como con la componente de ruido de la señal primaria $d(n)$; esto es:

$$E[d(n)v_1(n-k)]=0 \quad \text{para todo } k \quad (1.9)$$

En este caso, el filtro adaptativo “se interrumpe el mismo”, resultando en un valor cero para la salida $y(n)$. Así que, el cancelador de ruido no tiene efecto en la señal primaria $d(n)$ y la razón señal a ruido de la salida permanece sin cambiar.

El uso efectivo del cancelador de ruido adaptativo requiere que se coloque el sensor de referencia en el campo de ruido del sensor primario con un par de especificaciones objetivas en mente. Primero, la componente de la señal que lleva la información de la salida del sensor de referencia. Segundo, la salida del sensor de referencia es altamente correlacionada con la componente del ruido a la salida del sensor primario.

A continuación se describen tres útiles aplicaciones de la operación de cancelación de ruido adaptativo:

1. *Cancelación de interferencia de 60 Hz en electrocardiografía.* La electrocardiografía (ECG, por sus siglas en inglés) comúnmente se utiliza para monitorear el corazón de los pacientes, con una descarga eléctrica se radia energía a través del tejido humano y el resultado de la salida es recibido por un electrodo. Este electrodo es usualmente posicionado de tal forma que la energía recibida sea máxima.

Sin embargo, típicamente la descarga eléctrica involucra potenciales muy bajos. Correspondientemente, la energía recibida es muy pequeña. Así que, el cuidado extra ha de ser ejercitado en la minimización de la degradación de la señal, debido a interferencia externa. Por lejos que este la fuerza de la interferencia es de una forma periódica de 60Hz recogida por el electrodo de recepción (actuando como una antena) del equipo eléctrico más cercano. Es innecesario decir que, esta interferencia tiene efectos indeseables en la interpretación del electrocardiograma; pero, ha quedado demostrado que el uso de cancelación de ruido adaptativo (basado en el algoritmo LMS), es indispensable como un método para la reducción de esta interferencia. Específicamente, la señal primaria es tomada del preamplificador ECG, y la señal de referencia es tomada de un lugar exterior con una apropiada atenuación. La Figura 1.4 muestra un diagrama de bloques del cancelador de ruido utilizado por Widrow (1975b). El filtro adaptativo tiene un par de coeficientes ajustables, $\hat{w}_0(n)$ y $\hat{w}_1(n)$. El coeficiente $\hat{w}_0(n)$ es alimentado directamente del punto de referencia. El otro coeficiente $\hat{w}_1(n)$, es alimentado de una versión desplazada 90° en fase de la entrada de referencia. La suma del par de versiones de coeficientes de la señal de referencia es entonces restada de la salida del ECG para producir una señal de error. Esta señal de error y los dos coeficientes son aplicados al algoritmo LMS. En esta aplicación el cancelador de ruido adaptativo, es como un filtro “notch” variable. La frecuencia de la interferencia sinusoidal en la salida del ECG es presumiblemente la misma que de la señal de referencia. Sin embargo, la magnitud y fase de la interferencia sinusoidal en la salida del ECG son conocidas. El par de coeficientes $\hat{w}_0(n)$ y $\hat{w}_1(n)$ proporcionan el grado de libertad requerido, tanto para controlar la amplitud y fase de la señal de referencia sinusoidal, como para cancelar la interferencia de 60Hz contenida en la salida del ECG.

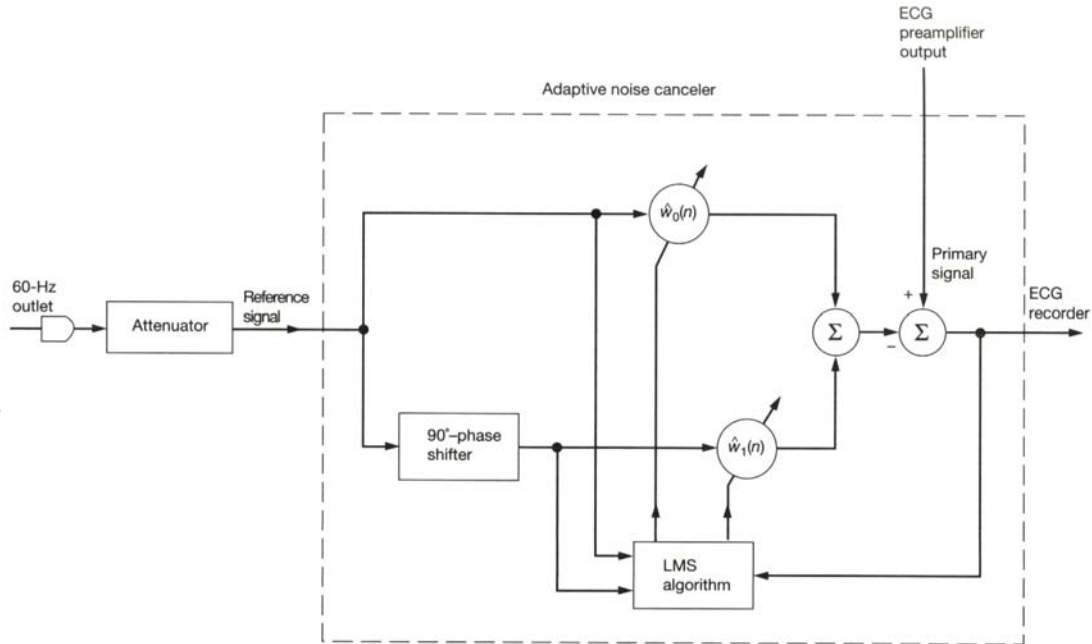


Figura 1.4 Cancelador de ruido para suprimir la interferencia de 60 Hz en un electrocardiograma.

2. *Reducción del ruido acústico en voz.* En un lugar donde hay ruido (por ejemplo, la cabina del piloto de una nave militar), la comunicación de voz es afectada por la presencia de ruido acústico. Este efecto es particularmente serio cuando la codificación predictiva lineal (LPC) es utilizada para la representación digital de la señal de voz a una razón baja de bits. Para especificar el ruido acústico de alta frecuencia, severamente afecta la estimación del espectro LPC tanto en las regiones de baja y alta frecuencia. Consecuentemente, la inteligibilidad para digitalizar la voz usando LPC cae a veces un nivel mínimo aceptable.

Kang y Fransen (1987) describen el uso de un cancelador de ruido adaptativo basado en el algoritmo LMS para la reducción de ruido acústico en voz. La voz descompuesta y ruidosa es utilizada como señal primaria para proporcionar una señal de referencia (solo ruido), un micrófono de referencia es colocado en el lugar donde hay suficiente aislamiento de la fuente de voz (muy lejos de la boca, por ejemplo en la parte trasera del casco). En el experimento descrito por Kang y Fransen, una reducción de 10 a 15 dB en el ruido acústico se realiza, sin degradar la calidad de voz. Es tanto significativa

una reducción del nivel de ruido en el mejoramiento de la calidad de voz que será inaceptable en otros casos.

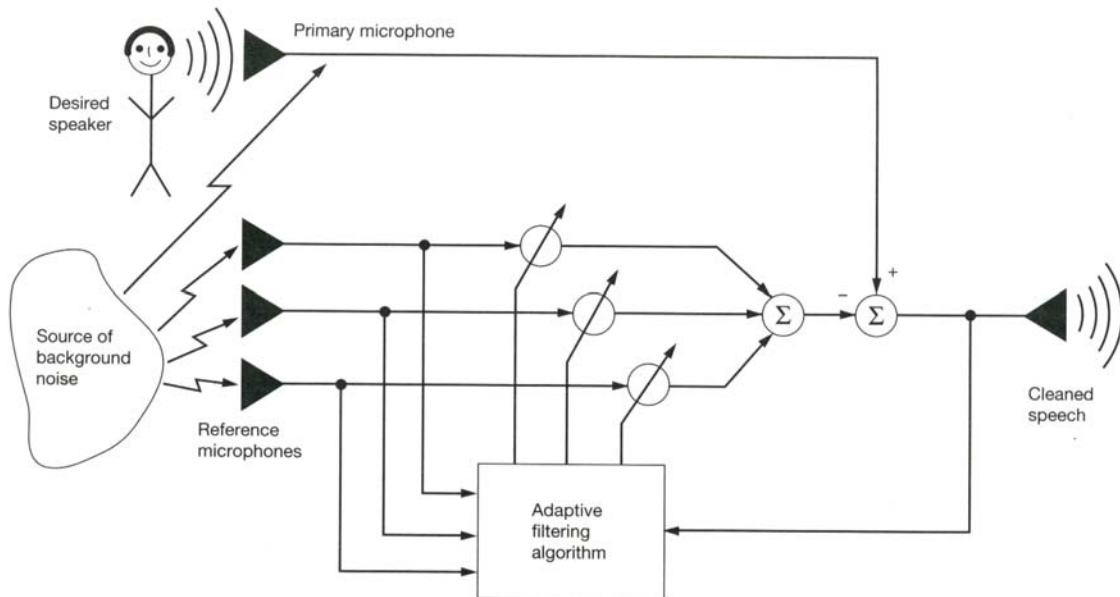


Figura 1.5 Cancelación adaptativa de ruido

3. *Mejoramiento adaptativo de voz.* Considere la situación descrita en la Figura 1.5. El requerimiento es el de oír la voz desde el parlante o bocina en la presencia de un ruido de fondo, el uso del cancelador de ruido adaptativo puede satisfacer el requerimiento. Específicamente, micrófonos de referencia son adheridos en lugares lo suficientemente lejos de la bocina deseada, tal que la salida contenga solo el ruido. Como se indica en la Figura 1.5, la suma del producto de los coeficientes con la salida del micrófono auxiliar, es restada de la salida del contenido de voz del micrófono principal y un algoritmo de filtrado adaptativo es utilizado para ajustar los coeficientes que minimizan la potencia promedio de salida. Una aplicación útil de la idea descrita aquí esta en la cancelación de ruido adaptativo para ayudar a oír a las personas con deficiencia auditiva³ (Chazan, 1988). El llamado “efecto de combinación de bandas” realmente limita la utilidad de ayudar a oír. El fenómeno del efecto de combinación de bandas se refiere a la habilidad de una persona con oído normal, para ‘enfocarse’ en una conversación, mientras se encuentra en una habitación concurrida. Esta habilidad no la pose alguien con problemas

³ Esta idea es similar a la del filtrado espacial adaptativo en el contexto de antenas.

en el oído, ya que es extrema la sensibilidad para la presencia de ruido de fondo. Esta sensibilidad es atribuida a un par de factores: (a) la pérdida de la pista de la dirección u orientación y (b) la limitada capacidad de canal para poder ‘oír’ por la reducción tanto del rango dinámico y la respuesta en frecuencia. Chazan (1988) describe una técnica de cancelador de ruido adaptativo para ayudar a vencer este problema. La técnica involucra el uso de un arreglo de micrófonos que explota la diferencia en características espaciales entre la señal deseada y el ruido en una habitación concurrida. El enfoque tomado por Chazan esta basado en el hecho que cada salida del micrófono puede verse como la suma de las señales producidas por las bocinas individuales ocupadas en la conversación en la habitación. Cada contribución de señal en una salida de un micrófono particularmente es esencial en el resultado de una señal de voz de una bocina, teniendo que pasar a través del filtro de la habitación. En otras palabras cada bocina (incluida la bocina deseada) produce una señal en la salida del micrófono que es la suma de la transmisión directa la señal de voz y su reflexión por las paredes de la habitación. El requerimiento es para reconstrucción de la señal de la bocina deseada, incluyendo el eco de la habitación, mientras se cancela la fuente de ruido. En general la transformación experimentada por la señal de voz de la bocina deseada no se conoce. También, las características del ruido de fondo son variables. Así que se tiene un problema de procesamiento de señal en donde el cancelador de ruido adaptativo ofrece una factible solución.

1.7 CONCLUSIONES

En este trabajo de graduación, vamos a centrar nuestro estudio en los *filtros digitales adaptativos*, 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 ajustándose según la señal de entrada es necesario cuando no se conocen por anticipado las características estadísticas de la señal a filtrar, o cuando se conocen y se sabe que son cambiantes con el tiempo.

La cancelación activa de ruido (CAR, en inglés ANC: Active Noise Control) es una técnica de procesamiento de señales digitales de gran utilidad para la recuperación de la información contenida en una señal contaminada cuando se ven afectadas por ruidos aditivos requiriendo un costo computacional relativamente bajo, fácil de implementarse con las tecnologías actuales y devolver resultados altamente satisfactorios.

1.8 REFERENCIAS

HAYKIN, SIMON (1996), “Adaptive Filter Theory”, 3ra. Edition, Prentice Hall

HAYES, MONSON H. (1996), “Statistical Digital Signal Processing and Modeling”,
JohnWiley & Sons

DEPARTAMENTO DE ELECTRÓNICA Y TELECOMUNICACIONES [DET] (2003),

“Filtrado Adaptativo”, Universidad del País Vasco

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

CAPITULO

2

Algoritmo Least Mean Square (LMS)

2.1 INTRODUCCIÓN

Dentro de este capítulo se desarrolla la teoría de un algoritmo ampliamente utilizado, el cual se conoce con el nombre de algoritmo de Mínimos Cuadrados Medios (LMS, por sus siglas en inglés) este algoritmo fue creado por Widrow y Hoff en el año de 1960. El algoritmo LMS pertenece a la familia de algoritmos de gradiente estocástico. El término “gradiente estocástico” pretende distinguir el algoritmo LMS del método de máxima pendiente que usa un gradiente determinista en un cálculo recursivo de los filtros Wiener para entradas estocásticas. Una característica muy importante y a la vez atractiva del algoritmo LMS es su simplicidad. No requiere medición de funciones de correlación, tampoco necesita de inversión de matrices. Es tan simple el algoritmo que lo utilizan como el estándar, con respecto a los otros algoritmos de filtrado adaptativo que existen, para evaluar un ordenador o tarjeta de evaluación (tal como una tarjeta de evaluación DSP56L811EVM).

Para comenzar con el capítulo se presenta una apreciación general de la estructura y operación del LMS. Pero antes, se expondrá una idea general acerca de filtros adaptativos lineales y cancelación de ruido.

2.2 FILTROS ADAPTATIVOS LINEALES

Los primeros trabajos acerca de filtros adaptativos se remontan al año 1950, durante esa época un cierto número de investigadores, se encontraban trabajando en diferentes aplicaciones de filtros adaptativos. De esos primeros trabajos emergió el algoritmo LMS como un simple pero efectivo algoritmo para la operación de filtros transversales adaptativos. El algoritmo fue ideado por Widrow y demostrado por Hoff en 1959. Todo comenzó con el estudio de un esquema de reconocimiento de un patrón, conocido como elemento lineal adaptativo (lógica de umbral) comúnmente referido dentro de la literatura científica como ADALINE (del inglés: ADaptive LINEar Element) [Haykin, 1996].

Algoritmo de gradiente estocástico

El algoritmo LMS es un algoritmo de gradiente estocástico en el que se hacen iteraciones de cada valor de los coeficientes de un filtro transversal en la dirección del gradiente con la magnitud al cuadrado de la señal de error con respecto al valor del coeficiente. Así que, el algoritmo LMS está estrechamente relacionado con el concepto de aproximación estocástica desarrollada por Robbins y Monro (1951) que se encuentran en estadística para resolver ciertos problemas de estimación de parámetros secuenciales. La primera diferencia entre ellos, es que el algoritmo LMS utiliza un parámetro de tamaño de paso fijo para controlar la corrección aplicada a cada valor de los coeficientes de una iteración a la próxima, mientras que en el método de aproximación estocástica el parámetro del tamaño de paso es inversamente proporcional al tiempo o la potencia de n (n es la variable independiente de tiempo discreto).

Otro algoritmo de gradiente estocástico, estrechamente relacionado con el algoritmo LMS es el algoritmo de gradiente de lattice adaptativo (GAL, por sus siglas en inglés); la diferencia entre ellos es la estructura en que se basan el algoritmo GAL se basa en estructura de celosía (lattice), mientras que el algoritmo LMS usa un filtro transversal. En 1981, Zame introdujo el *criterio H^∞* (o *minimax*) como un índice de robustez del comportamiento del algoritmo LMS para resolver problemas de estimación y control.

Cancelador de ruido adaptativo

El mejoramiento lineal adaptativo fue originado por Widrow y sus colaboradores en la Universidad de Stanford. Una primera versión de este mecanismo fue construida en 1965 para cancelar una interferencia de 60Hz en la salida de un registrador y amplificador de electrocardiografía. Este trabajo es descrito en el artículo por Widrow (1975). El mejoramiento lineal adaptativo y su aplicación como un detector adaptativo fueron patentados por McCool (1980).

El cancelador adaptativo y el mejoramiento lineal adaptativo, promete ser útil para diferentes aplicaciones, podría verse el ejemplo del cancelador de ruido adaptativo

discutido por Widrow (1975). Este esquema opera con la salida de los dos sensores; un primer sensor que suministra una señal deseada de interés soterrada de ruido y un sensor de referencia que suministra solo el ruido, como se ilustra en la Figura 2.1. Se asume que (1) la señal y el ruido a la salida del primer sensor esta sin correlación y (2) el ruido a la salida del sensor de referencia esta correlacionada con la componente de ruido de la salida del sensor primario.

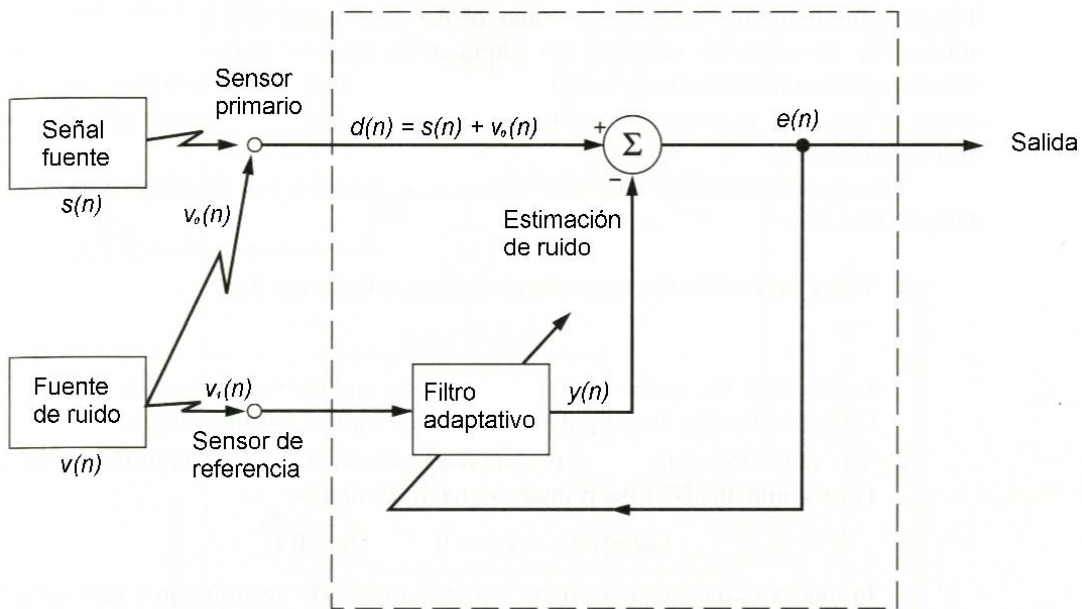


Figura 2.1 Cancelador de ruido adaptativo.

El cancelador de ruido aleatorio consiste en un filtro adaptativo que opera sobre la salida del sensor de referencia para producir una estimación del ruido, que se ha sumado a la señal deseada en el sensor primario. La salida total del cancelador es utilizada para controlar el ajuste aplicado a los valores de los coeficientes en el filtro adaptativo. El cancelador adaptativo tiende a minimizar el valor cuadrático medio de la salida total, por tanto causa que la salida sea la mejor estimación de la señal deseada en el sentido del mínimo error cuadrático medio.

2.3 APRECIACIÓN GENERAL DE LA ESTRUCTURA Y OPERACIÓN DEL ALGORITMO LMS

El algoritmo LMS es un algoritmo de filtrado lineal adaptativo que consiste de dos procesos básicos:

Un proceso de filtrado, que involucra (a) calcular la salida de un filtro transversal, la cual es producida por un conjunto de valores de entrada y (b) generar una estimación del error por la comparación de la salida y la respuesta deseada.

Un proceso adaptativo, el cual involucra el ajuste automático de los valores de los coeficientes del filtro, de acuerdo con la estimación del error.

Así, que la combinación de estos dos procesos constituyen un lazo de realimentación alrededor del algoritmo LMS, así como se ilustra en la Figura 2.2.

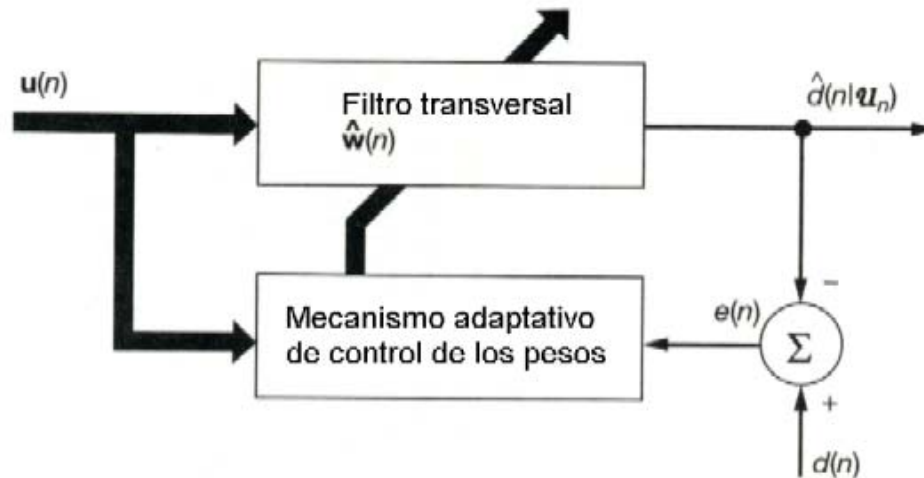


Figura 2.2 Diagrama del bloque de la estructura LMS

Observando la Figura 2.2, tenemos un filtro transversal, alrededor del cual se construye el algoritmo LMS, el cual es responsable de desempeñar el proceso de filtrado. Luego se tiene un mecanismo que desempeña el proceso de control adaptativo de los valores de los coeficientes del filtro transversal.

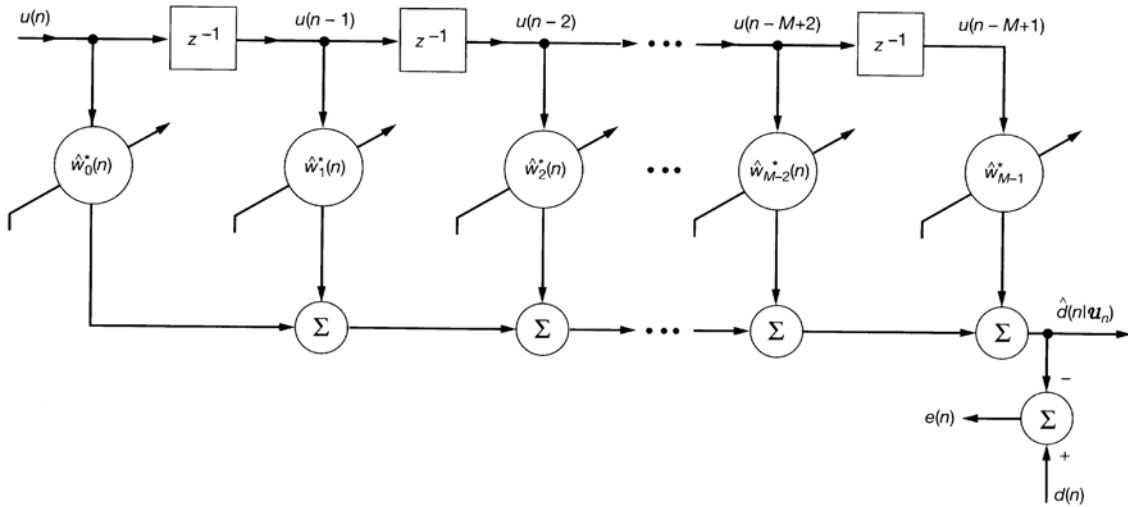


Figura 2.3 Diagrama del filtro transversal

Los detalles de los componentes del filtro transversal son presentados en la Figura 2.3. La entrada es representada por $u(n)$ ¹, $u(n-1)$, ..., $u(n-M+1)$ que son los elementos del vector $\mathbf{U}(M,1)$, donde $M-1$ es el número de elementos de retardo, estas entradas variantes en el tiempo, están representadas en el espacio multidimensional por \mathbf{U} . Correspondientemente, los valores $\hat{w}_0(n)$, $\hat{w}_1(n)$, ..., $\hat{w}_{M-1}(n)$ son los elementos del vector de coeficientes $\hat{\mathbf{w}}(M,1)$. El valor calculado para el vector de coeficientes $\hat{\mathbf{w}}(n)$ utilizando el algoritmo LMS representa una estimación del valor que se espera se aproxime a la solución de Wiener w_0 (para un entorno estacionario esto es óptimo) con un número de n iteraciones que se acercan al infinito.

Luego, durante el proceso de filtrado la respuesta deseada $d(n)$ es suministrada para procesarla, al lado del vector de entrada $u(n)$. Obtenida esta entrada, el filtro transversal produce una salida $\hat{d}(n|u_n)$ utilizada como una estimación de la respuesta deseada $d(n)$. Consecuentemente se define una estimación de error $e(n)$ como la diferencia entre la respuesta deseada y la salida del filtro, como se indica la salida en la Figura 2.3. La estimación del error $e(n)$ y los valores del vector de entrada $u(n)$ son

¹ En algunos libros en lugar de $u(n)$ se utiliza $x(n)$. También dentro de este trabajo de graduación se utilizan indistintamente.

aplicados al mecanismo de control y el lazo de realimentación alrededor de los valores de los coeficientes es cerrado consecuentemente.

La Figura 2.4 presenta los detalles del mecanismo de control adaptativo para los coeficientes, específicamente, una versión escalar del producto interno de la estimación del error $e(n)$ y la entrada $u(n-k)$ es calculada para $k=0,1,2,\dots, M-2, M-1$. El resultado así obtenido define la correlación $\delta\hat{w}_k(n)$ aplicada a los valores de los coeficientes $\hat{w}_k(n)$ en la iteración $n+1$. El factor de escala utilizado en este cálculo se designa por μ en la Figura 2.4. Es conocido como parámetro del tamaño de paso de adaptación o velocidad de adaptación.

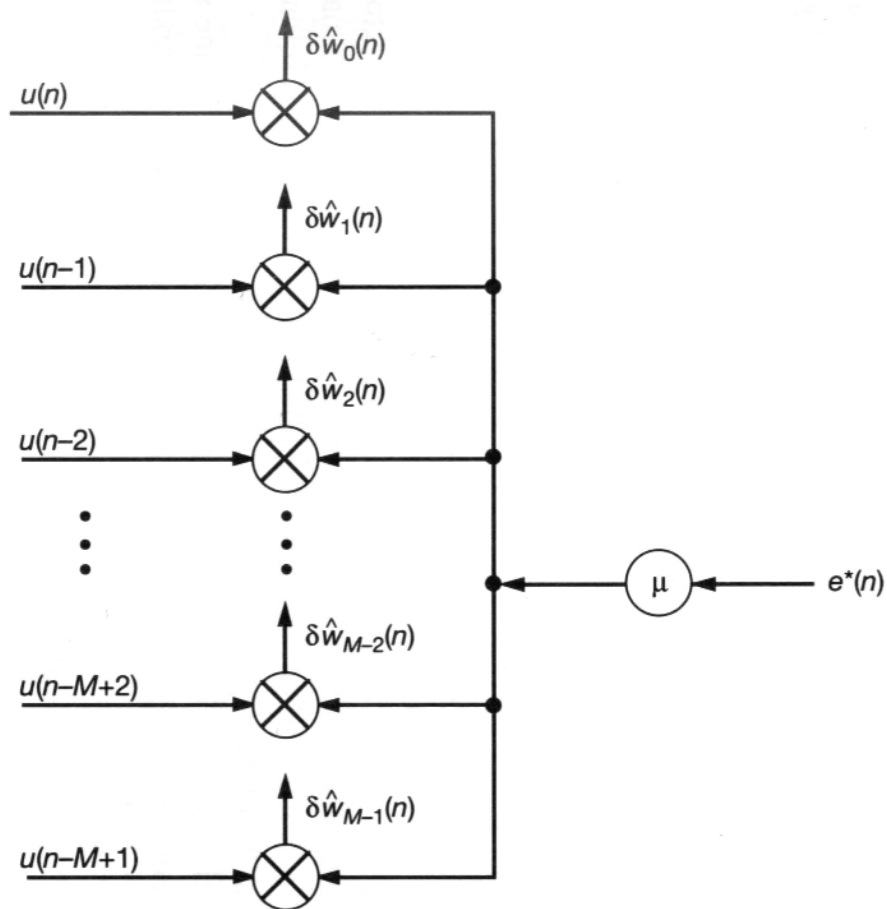


Figura 2.4 Detalles de la estructura del mecanismo de control adaptativo para los coeficientes

Comparando el mecanismo de control de la Figura 2.4 para el algoritmo LMS con el método de máxima pendiente, se ve en el LMS que el producto $u(n-k) \cdot e^*(n)$ es una estimación del elemento k dentro del gradiente $\nabla J(n)$ que caracteriza el método de máxima pendiente. En otras palabras, las expectativas del operador se pierden en todas las rutas de la Figura 2.4. En consecuencia, el cálculo recursivo de cada valor de coeficiente es afectado por un gradiente de ruido.

A través de este capítulo se asume que el vector de entrada $\mathbf{u}(n)$ y la respuesta deseada $d(n)$ son representados en un ambiente estacionario común. Para semejante ambiente se conoce que el método de pendiente pronunciada calcula un vector de coeficientes $\mathbf{w}(n)$ que se mueve hacia abajo de la superficie de funcionamiento de error que ensambla el error promedio a lo largo de una trayectoria determinista, la cual termina en la solución de Wiener w_0 (ver Figura 2.5). El algoritmo LMS, en otras palabras se comporta diferente, ya que esta en presencia de un gradiente de ruido.

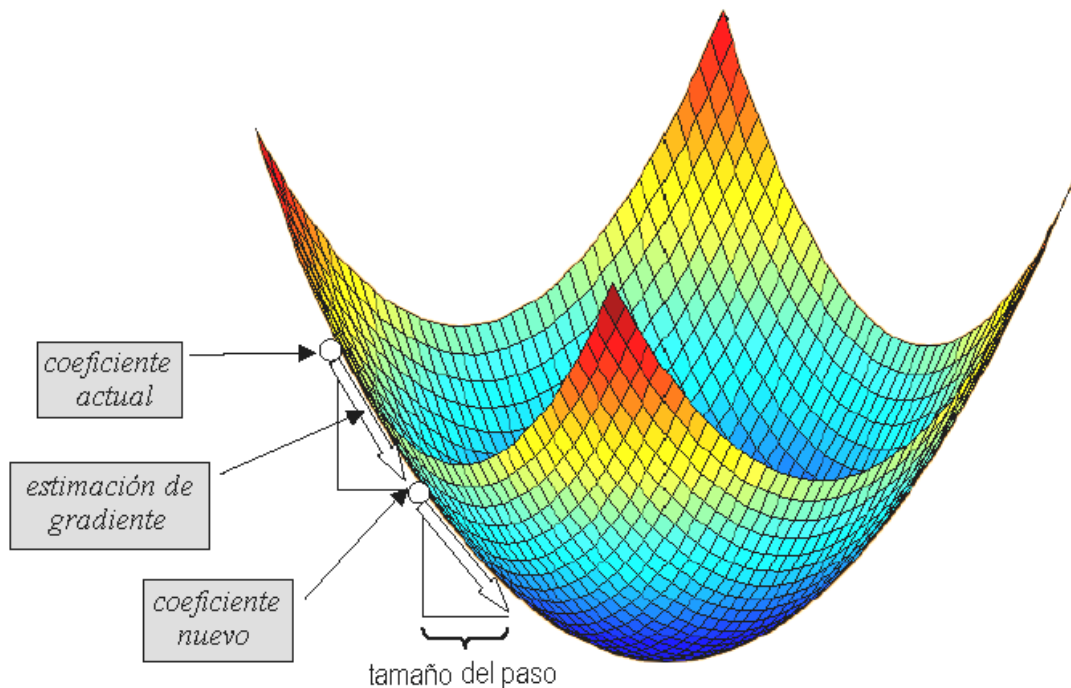


Figura 2.5 Superficie de error

Antes que terminemos en la solución de Wiener el vector de coeficientes $\hat{w}(n)$ (diferente de $w(n)$) calculado por el algoritmo LMS, se ejecuta un movimiento aleatorio alrededor del punto mínimo de la superficie de error. Al principio se indicó que el algoritmo LMS involucra una realimentación en su operación que por consiguiente eleva el resultado de la estabilidad. En este contexto, un criterio significativo requiere que:

$$J(n) \rightarrow J(\infty) \text{ como } n \rightarrow \infty$$

Donde $J(n)$ es el error cuadrático medio producido por el algoritmo LMS en el tiempo n , y su valor final $J(\infty)$ es una constante. Un algoritmo que satisface este requisito dice ser convergente en el valor cuadrático medio.

Para que el algoritmo LMS satisfaga este criterio, el parámetro del tamaño del paso μ tiene que satisfacer una condición relacionada con la estructura del valor propio² de la matriz de correlación de los valores de entrada.

La diferencia entre el valor final $J(\infty)$ y el valor mínimo J_{min} alcanzado por la solución de Wiener se conoce como ‘exceso del error cuadrático medio $J_{ex}(\infty)$ ’. Esta diferencia representa el precio a pagar por utilizar el mecanismo adaptativo (estocástico) para controlar el valor de los coeficientes en el algoritmo LMS en lugar del enfoque determinista como en el método de la máxima pendiente. La relación de $J_{ex}(\infty)$ a J_{min} se conoce como “desajuste” el cual es una medición de que tan lejos esta el cálculo de la solución de estado permanente por el algoritmo LMS de la solución de Wiener. Es importante que μ este bajo el control del diseñador. En particular, la acción del lazo de realimentación alrededor de los coeficientes, se comporta como un filtro pasa bajo, en el cual la constante de tiempo promedio es inversamente proporcional al parámetro del tamaño de paso μ . Así, que por la asignación de un valor pequeño de μ el proceso adaptativo se realiza progresivamente y el efecto del gradiente del ruido en los valores de los coeficientes es grande para cuando se filtra. Esto conduce al efecto de reducción del desajuste. Por consiguiente se justifica diciendo que el algoritmo LMS es simple en

² Sea A una matriz cuadrada de orden n. Se denominan valores propios reales de A los números reales λ que son raíces de su polinomio característico.

la implementación, además es capaz de manejar alto desempeño por la adaptación a su ambiente externo.

Así, que se debe poner particular atención para seleccionar un valor adecuado para el parámetro μ . Una consecuencia estrechamente relacionada con lo anterior es la especificación de un valor adecuado para el orden del filtro es $M-1$. Con el propósito de manejar el algoritmo LMS en la siguiente sección se construye el algoritmo a partir de nuestro conocimiento previo del método de máxima pendiente.

2.4 ALGORITMO DE ADAPTACIÓN DEL LMS

Si fuera posible hacer una medición exacta del gradiente $\nabla J(n)$ en cada \mathbf{n} iteración y si el parámetro del tamaño de paso μ es seleccionado adecuadamente, entonces el vector de valores de coeficientes, calculado usando el algoritmo de máxima pendiente podría verdaderamente converger a la solución óptima de Wiener.

Sin embargo, realmente la medición exacta del gradiente no es posible, ya que podría requerir un conocimiento previo, tanto de la matriz de correlación \mathbf{R} de la entrada y el vector \mathbf{p} de correlación cruzada entre la entrada y la respuesta deseada. Consecuentemente, el gradiente debe ser estimado del dato disponible [Haykin, 1996].

Para desarrollar una estimación del gradiente $\nabla J(n)$, la más obvia estrategia es sustituir la estimación de la matriz de correlación \mathbf{R} y el vector de correlación cruzada \mathbf{p} en la ecuación (2.1):

$$\nabla J(n) = -2\mathbf{p} + \mathbf{R}\mathbf{w}(n) \quad (2.1)$$

La solución simple de las estimaciones de \mathbf{R} y \mathbf{p} es utilizada para una estimación instantánea del vector de entrada y la respuesta deseada como se define respectivamente:

$$\hat{\mathbf{R}}(n) = \mathbf{u}(n)\mathbf{u}^H(n) \quad (2.2)^3$$

$$\hat{\mathbf{p}}(n) = \mathbf{u}(n)d^*(n) \quad (2.3)$$

³ H denota la *transposición Hermitian*; que es la operación de transposición combinada con conjugación compleja.

Correspondientemente la estimación instantánea del gradiente es:

$$\nabla J(n) = -2\mathbf{u}(n)d^*(n) + 2\mathbf{u}(n)\mathbf{u}^H(n)\hat{\mathbf{w}}(n) \quad (2.4)$$

Generalmente, esta estimación es parcial, porque la estimación del vector de coeficientes $\hat{\mathbf{w}}(n)$ es un vector aleatorio que depende del vector de entrada $\mathbf{u}(n)$. Note que la estimación $\nabla J(n)$, también podría verse como el operador de gradiente ∇ aplicado al error instantáneo cuadrado $|e(n)|^2$; sustituyendo la estimación de la ecuación (2.4) para el gradiente $\nabla J(n)$ en el algoritmo de la máxima pendiente descrito por la ecuación (2.5), obtenemos una nueva relación recursiva para actualización del vector de coeficientes.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{2}\mu[-\nabla J(n)] \quad (2.5)$$

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu \cdot \mathbf{u}(n) [d^*(n) - \mathbf{u}^H(n)\hat{\mathbf{w}}(n)] \quad (2.6)$$

Aquí se tiene que utilizar de nuevo el símbolo “^” para el vector de coeficientes, con el objetivo de hacer la distinción del valor obtenido por la utilización del algoritmo de pendiente máxima (es decir, un valor aproximado). Equivalentemente, se escribe el resultado en tres relaciones básicas:

Salida del Filtro:

$$y(n) = \hat{\mathbf{w}}^H(n)\mathbf{u}(n) \quad (2.7)$$

Estimación de error:

$$e(n) = d(n) - y(n) \quad (2.8)$$

Adaptación de los valores de los coeficientes:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*(n) \quad (2.9)$$

Las ecuaciones (2.7) y (2.8) son la estimación del error $e(n)$, el cálculo del cual se basa la estimación actual del vector de coeficientes $\hat{w}(n)$. Note también que el segundo término, $\mu u(n) \cdot e^*(n)$, en el lado derecho de la ecuación (2.9) representa la corrección que se aplica a la estimación actual del vector de coeficientes $\hat{w}(n)$. El procedimiento iterativo es iniciado con una aproximación inicial $\hat{w}(0)$.

El algoritmo descrito por las ecuaciones (2.7) a (2.9) es la forma del algoritmo de mínimos cuadrados medios (LMS) ⁴. En cada iteración o tiempo de actualización, también se requiere de valores más recientes de $u(n)$, $d(n)$ y $\hat{w}(n)$. El algoritmo LMS opera sobre entradas estocásticas; el conjunto permitido a lo largo del “paso” de un ciclo de iteración a la siguiente es aleatorio y no se sabe cuál será la dirección verdadera de los gradientes. La Figura 2.6 muestra una gráfica de flujo de señal de la representación del algoritmo LMS en la forma del modelo de realimentación. La gráfica de flujo de señal de la Figura 2.5 claramente ilustra la simplicidad del algoritmo LMS. En particular, vemos en la Figura 2.6 que el algoritmo requiere solamente $2M+1$ multiplicaciones complejas y $2M$ sumas complejas por iteraciones, donde M es el número de coeficientes utilizados en el filtro transversal adaptativo.

La estimación instantánea de \mathbf{R} y \mathbf{p} de la ecuación (2.2) y (2.3) respectivamente, tienen relativamente grandes variaciones. A primera vista, podría verse que el algoritmo LMS es incapaz de realizar un buen funcionamiento, ya que el algoritmo utiliza estas estimaciones instantáneas, sin embargo, debemos recordar que el algoritmo LMS es de naturaleza recursiva con el resultado del algoritmo en sí mismo efectivamente pues promedia esas estimaciones, en cierto sentido dentro del curso de la adaptación.

⁴ La forma compleja del algoritmo LMS, con el propósito original de Widrow (1975) difiere ligeramente de las ecuaciones (2.7) a (2.9).

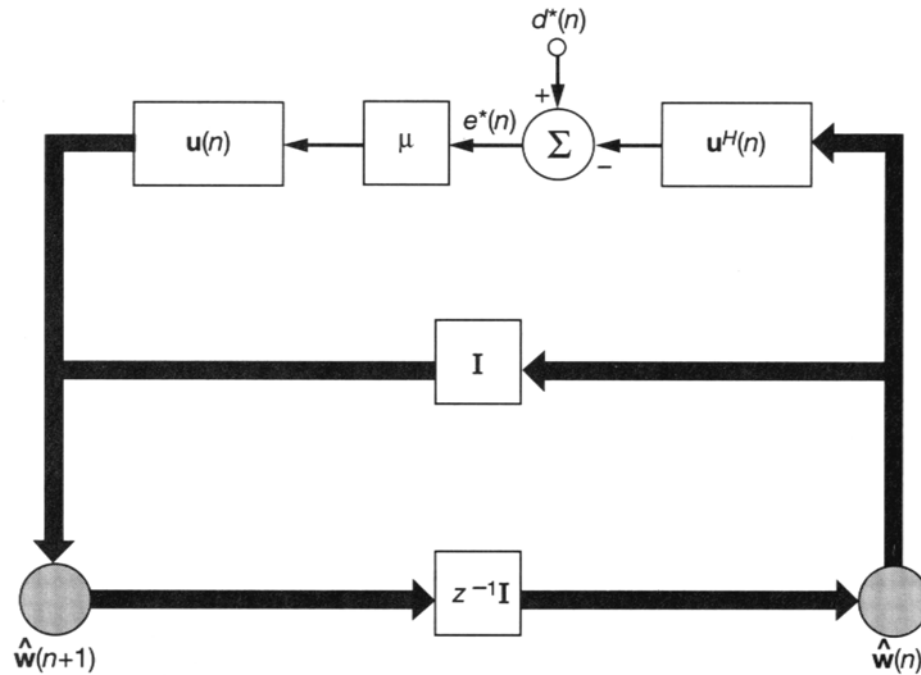


Figura 2.6 Gráfica de flujo de señal del algoritmo LMS

2.5 EL ALGORITMO LMS EN LA PRÁCTICA

El algoritmo de LMS es simple y todavía capaz de lograr actuación satisfactoria bajo las condiciones correctas. Sus limitaciones mayores son una proporción relativamente lenta de convergencia y una sensibilidad a las variaciones en el número de la condición de la matriz de la correlación de las entradas; el número de la condición de una matriz de Hermitian se define como la proporción de su valor propio más grande al valor propio más pequeño. No obstante, el algoritmo de LMS es muy popular y ampliamente usado en una variedad de aplicaciones.

En un ambiente no estacionario, la orientación de la superficie de error varía continuamente con el tiempo. En este caso, el algoritmo de LMS tiene la tarea agregada de rastrear el fondo de la superficie del error continuamente. De hecho, en el rastreo de los datos de entrada variarán comparado a la proporción del lento aprendizaje del algoritmo de LMS.

Una discusión introductoria de filtros adaptables lineales estaría incompleta sin decir algo sobre su comportamiento de rastreo. En este contexto, nosotros notamos algoritmos de pendiente estocástica como el LMS que es un modelo independiente; generalmente, se esperaría que estos algoritmos exhibieran muy buen comportamiento de rastreo, de hecho eso es lo que hacen. En contraste, el algoritmo RLS es un modelo dependiente; esto significa que su comportamiento de rastreo puede ser inferior y pésimo comparado con un miembro de la familia de pendiente estocástica.

A pesar de la sencillez de este algoritmo, actualmente ocupa un lugar muy amplio en las nuevas tecnologías de rastreo y detección. Por ejemplo, usando el LMS en forma más compleja, se aplica a sistemas de radar para la navegación submarina. La aviación militar y las comunicaciones satelitales, sobre todo en el campo militar [Etter Delores M., Steinhardt Allan O., And Stoner Susan L. (2002)].

2.6 CONCLUSIONES

A lo largo de este capítulo se ha pretendido presentar un estudio detallado del algoritmo de Mínimos Cuadrados Medios (LMS), que no es más, sino sólo una introducción al mundo del filtrado digital adaptativo lineal. La importancia práctica del algoritmo LMS se debe grandemente a dos atributos:

- ✓ Simplicidad de implementación.
- ✓ Modelo independiente y consecuentemente funcionamiento robusto.

La principal limitación del algoritmo LMS es su lenta velocidad de convergencia. Hay dos factores principales que inciden en el comportamiento del algoritmo LMS: el parámetro de tamaño de paso μ y el valor propio de la matriz de correlación \mathbf{R} del vector de entrada. Dentro de la perspectiva del análisis del algoritmo LMS, que utiliza una teoría independiente, sus efectos de manera individual se resumen así:

La convergencia del algoritmo LMS dentro del cuadrado medio de la entrada es asegurada por la selección del parámetro de tamaño de paso μ de acuerdo con la

condición práctica: $0 < \mu < \frac{2}{\text{potencia de entrada}}$ donde la potencia de entrada es la suma de los valores del cuadrado medio de todos los valores de entrada dentro del filtro transversal.

Cuando un valor pequeño se asigna a μ , la adaptación es lenta, ya que es equivalente a tener un algoritmo LMS con bastante “memoria”. Respectivamente, el exceso del error cuadrático medio después de la adaptación es pequeño en promedio, por la gran cantidad de datos utilizados por el algoritmo para la estimación del gradiente. En otras palabras, cuando μ es grande, la adaptación es relativamente rápida, pero hay un costo de incremento dentro del promedio del exceso del error cuadrático medio después de la adaptación. En este caso, menos datos entran para calcular la estimación, dentro de un desempeño de estimación de error degradado. Así, el recíproco del parámetro μ debe verse como memoria del algoritmo LMS.

Cuando el valor propio de la matriz de correlación \mathbf{R} es ampliamente esparcida, el exceso del error cuadrático medio producido por el algoritmo LMS es primeramente determinado por valores propios grandes y a la vez tomado por el promedio de los coeficientes $E[\hat{w}(n)]$ para que ocurra la convergencia, se limita a un valor propio pequeñísimo. Sin embargo, la velocidad de convergencia del error cuadrático medio, $J(n)$ es afectada por una dispersión de los valores propios de \mathbf{R} de menor magnitud que la convergencia de $E[\hat{w}(n)]$. Cuando la dispersión de los valores propios es grande (por ejemplo, cuando la matriz de correlación de los valores de entrada es una mala condición), la convergencia del algoritmo LMS es de baja velocidad. Sin embargo no es siempre así, ya que el comportamiento de la convergencia del algoritmo LMS toma una dirección natural bajo entradas contaminadas. Es apropiado, que para mejorar el proceso de convergencia de los coeficientes, la inicialización del algoritmo LMS puede optimizarse por otros métodos de filtrado.

Una limitación básica de la teoría de independencia es el hecho que ignora la dependencia estadística entre la dirección del gradiente, tal como el procedimiento del algoritmo de una iteración a la próxima. Otro enfoque del estado permanente del

algoritmo LMS es presentado, en el cual se confía en el uso de la solución de la serie de potencias para el vector de coeficientes $\epsilon(n)$ sin apelar a la suposición de independencia.

Definitivamente existen otros puntos de vista y análisis para el algoritmo LMS, pero dentro de este capítulo se ha desarrollado una visión muy resumida del análisis del algoritmo Least Mean Square (LMS).

2.7 REFERENCIAS

HAYKIN, SIMON (1996), "Adaptive Filter Theory", 3ra. Edition, Prentice Hall

ETTER DELORES M., STEINHARDT ALLAN O., AND STONER SUSAN L. (2002),

"Least Squares Adaptive Processing in Military Applications", IEEE
Signal Processing Magazine

CAPITULO

3

*Simulación del algoritmo LMS,
utilizando Matlab*

3.1 INTRODUCCIÓN

Todos los sistemas avanzados de comunicaciones de voz, imagen y datos a alta velocidad requieren del uso de sistemas de cancelación de interferencias más o menos sofisticados. En esta línea de trabajo se estudian y desarrollan algoritmos de adaptación capaces de gobernar eficientemente los coeficientes y parámetros de estos sistemas. Los requisitos deseados incluyen alta velocidad de convergencia, estabilidad asegurada, buen comportamiento en situaciones estáticas y robustez.

Con el fin de analizar cada uno de estos algoritmos antes de construir o adquirir instrumentación apropiada para el manejo de señales reales, es necesario contar con una herramienta que permita simular, es decir probar de manera virtual el comportamiento del algoritmo, modificando la frecuencia, amplitud y otras características de la señal a la que se desea hacerle algún tratamiento digital. Además, deberá ser posible ver los resultados obtenidos luego del tratamiento.

En nuestro caso, el tratamiento digital consiste en filtrado adaptativo lineal. Cuya función consiste en separar la señal deseada, que puede ser voz, imágenes o datos, del ruido (el ruido puede ser una señal de frecuencia fija, ruido aleatorio estacionario, con varianza y media constantes o ruido aleatorio no estacionario). El algoritmo a utilizar, como ya fue expuesto antes, es el LMS.

Entre los lenguajes de programación que pueden utilizarse están Fortran, Basic, C, etc. Pero ninguno de estos posee la capacidad para generar gráficos de manera rápida y fácil con solo escribir unas cuantas instrucciones. El programa utilizado, para generar cualquier señal, generar ruido aleatorio, aplicar el algoritmo LMS y ver los resultados a la salida, fue MATLAB¹.

¹Esta herramienta guarda semejanza con OCTAVE, el cual se utiliza en la mayoría de computadoras que usan el Sistema Operativo LINUX.

3.2 EL LMS EN MATLAB

Dada la característica de Matlab de manejar matrices, es necesario llevar las expresiones obtenidas en el capítulo anterior del algoritmo LMS a su forma matricial. Pero antes de escribir cualquier matriz se va a repetir el uso particular de algunas letras. Primero, M será el número de coeficientes del filtro y N , el total de muestras de una señal.

Para familiarizarse con la nomenclatura de las señales en un filtro adaptativo podemos fijarnos en la Figura 3.1, aquí $x(n)$ es la señal de entrada la cual después de ser retardada es operada por cada uno de coeficientes. Esta señal $x(n)$ es muy distinta para un cancelador de ruido con referencia que para uno sin referencia: En el primero es el ruido de referencia, y en el segundo es la misma señal contaminada retardada. Luego tenemos una señal $d(n)$ que se denomina señal deseada. La señal $d(n)$ es la señal de interés sumada al ruido; es decir, $d(n)$ es la señal contaminada para un cancelador de ruido sea con referencia o sin ella. El error $e(n)$ resulta ser la variable de control del sistema, pues $e(n)$ interviene en el algoritmo de actualización de los coeficientes del filtro [Haykin, 1996].

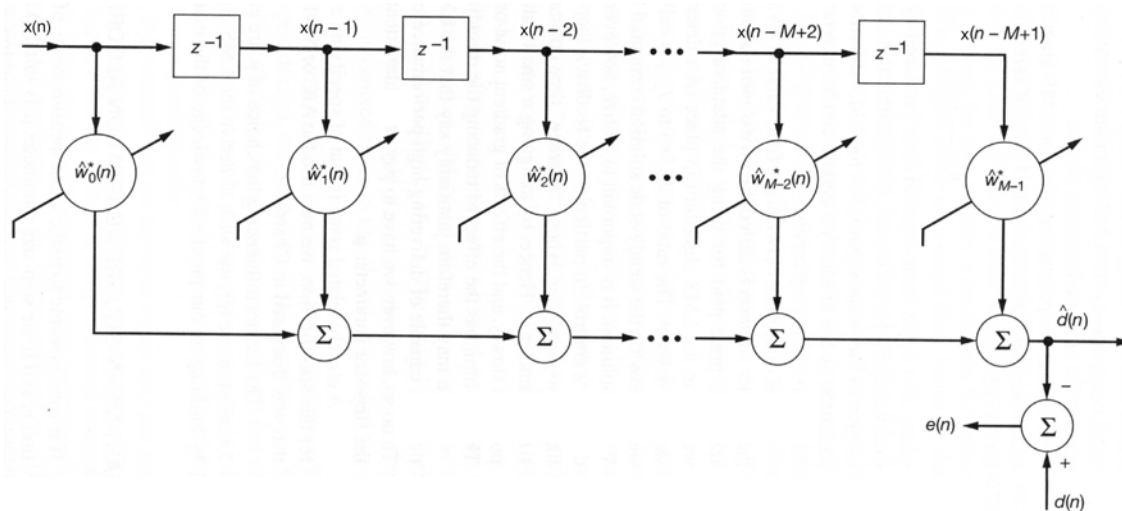


Figura 3.1 Estructura transversal adaptativa (FIR adaptativo)

Programación en Matlab de un filtro FIR

La ejecución de un filtro FIR en Matlab es muy fácil de realizar, pues solo requiere el uso de instrucciones tales como *conv*², que calcula la convolución de dos señales. Sin embargo, cuando se trata de un filtro adaptativo, ocurre que para cada muestra tomada se realiza la convolución de todos los coeficientes con todas las muestra ya existentes, luego se actualizan los coeficientes y en la siguiente iteración se vuelve a repetir lo mismo. Esto requiere un mejor control de los límites inferiores y superiores de la señal de entrada que se tomarán en cada iteración o ciclo; con ese fin, se tratan de manera matricial.

Es fácil comprender esto si se supone que deseamos realizar un filtro de longitud 3, es decir 3 coeficientes y tomar 7 muestras de una señal. En este caso la expresión para el filtro FIR queda de la siguiente manera:

$$y(n) = \sum_{i=0}^{M-1} W(i) \cdot x(n-i) = \sum_{i=1}^M W(i) \cdot x(n-i) \quad (3.1)$$

$$y(n) = \sum_{i=1}^3 W(i) \cdot x(n-i) \quad (3.2)$$

donde:

- y es la señal de salida del filtro FIR
- \mathbf{W} son los coeficientes o pesos del filtro
- \mathbf{x} es la entrada del filtro

desarrollando la ecuación 3.2 tenemos:

$$\begin{aligned} y(1) &= W(1) * x(0) + W(2) * x(-1) + W(3) * x(-2) \\ y(2) &= W(1) * x(1) + W(2) * x(0) + W(3) * x(-1) \\ y(3) &= W(1) * x(2) + W(2) * x(1) + W(3) * x(0) \\ y(4) &= W(1) * x(3) + W(2) * x(2) + W(3) * x(1) \\ y(5) &= W(1) * x(4) + W(2) * x(3) + W(3) * x(2) \\ y(6) &= W(1) * x(5) + W(2) * x(4) + W(3) * x(3) \\ y(7) &= W(1) * x(6) + W(2) * x(5) + W(3) * x(4) \end{aligned}$$

² La utilización de estas instrucciones puede saberse fácilmente si se escribe en MATLAB:
>> help conv

Al expresar lo anterior de forma matricial, es posible notar que la convolución ocurre entre todos los coeficientes y un subvector de la entrada x de dimensiones $(M, 1)$. Ese subvector se denominará \mathbf{X} ; es decir, \mathbf{X} no representa todos los datos de entrada, sino solo un subconjunto de estos en cada iteración. El resultado se muestra a continuación:

$$y(n) = \begin{bmatrix} W \end{bmatrix}^T \cdot \begin{bmatrix} X \end{bmatrix} = \begin{bmatrix} W \end{bmatrix} \cdot \begin{bmatrix} X \end{bmatrix} \Rightarrow y(n) = \mathbf{W}^T * \mathbf{X} \quad (3.3)$$

donde definimos a \mathbf{W} , y \mathbf{X} como vectores de $(M, 1)$ para la convolución en cada iteración y un vector resultante \mathbf{Y} de $(1, N)$ cuando se han completado todos los ciclos.

Programación en Matlab del LMS

El algoritmo LMS consta de una ecuación muy simple, desde el punto de vista matemático. Esta ecuación también puede ser escrita en forma matricial, desarrollada de la siguiente manera:

$$W_i(n+1) = W_i(n) + \mu e(n) x^*(n - i) \quad (3.4)$$

$$\begin{aligned} W_1(2) &= W_1(1) + \mu e(1) x^*(0) \\ W_2(2) &= W_2(1) + \mu e(1) x^*(-1) \\ W_3(2) &= W_3(1) + \mu e(1) x^*(-2) \end{aligned}$$

Donde $W_i(n)$ representa el valor del coeficiente i en el tiempo n .

Si lo anterior se hace para cada iteración, entonces se tiene,

$$\begin{bmatrix} W_{n+1} \end{bmatrix} = \begin{bmatrix} W_n \end{bmatrix} + \mu e(n) \cdot \begin{bmatrix} X \end{bmatrix} \Rightarrow \mathbf{W}_i(n+1) = \mathbf{W}_i(n) + \mu e(n) \cdot \mathbf{X}^* \quad (3.5)$$

Programación en Matlab para calcular el error

El cálculo del error es necesario para medir las mejoras que se están obteniendo en los coeficientes. La ecuación que se utiliza es:

$$e(n) = d(n) - y(n) \quad (3.6)$$

El resultado de esta ecuación luego de concluir todos los ciclos es de un vector \mathbf{E} de longitud $(1, N)$. Sobre \mathbf{E} se aplican funciones de costo, tales como, el error cuadrático y el error cuadrático medio, como criterios para minimizar el error.

En conclusión, todo filtro adaptativo, en forma recursiva (repetidas veces) realiza las operaciones que se listan a continuación:

- $\mathbf{Y} = \mathbf{W}^T * \mathbf{X}$
- $e(n) = d(n) - y(n)$
- $\mathbf{W}_i(n+1) = \mathbf{W}_i(n) + \mu e(n) \cdot \mathbf{X}^*$

El número total de repeticiones o ciclos necesarios es igual a \mathbf{N} y al ver las operaciones anteriores vemos que por cada ciclo requerirá un valor de \mathbf{M} multiplicaciones y \mathbf{M} sumas para encontrar \mathbf{Y} , \mathbf{M} multiplicaciones y \mathbf{M} sumas para actualizar los coeficientes (en total, $2\mathbf{M}+1$ multiplicaciones y $2\mathbf{M}$ sumas).

El algoritmo LMS normalizado (NLMS)

El algoritmo LMS Normalizado tiene por objetivo independizar la convergencia de la potencia de la señal de entrada. Es por ello, más robusto que el algoritmo LMS.

En el algoritmo LMS, la corrección aplicada al vector de pesos $\mathbf{w}(n)$ es proporcional al vector de entrada $\mathbf{x}(n)$. Por tanto, si $\mathbf{x}(n)$ es elevado al cuadrado, el algoritmo LMS experimenta un problema de amplificación de ruido del gradiente. Con la normalización del parámetro de convergencia, este problema se reduce, de igual manera que se evita un aumento desmedido de la corrección del vector $\mathbf{w}(n)$ cuando la entrada disminuye drásticamente.

El algoritmo queda así:

$$\mathbf{W}_i(n+1) = \mathbf{W}_i(n) + \beta \frac{\mathbf{X}^*}{\|\mathbf{X}\|^2} e(n) \quad (3.7)$$

3.3 APLICACIÓN: CANCELADOR DE RUIDO

Antes de implementar en el DSP56L811 el cancelador de ruido, es necesario realizar la simulación que nos permita ver cuales son las características de la señal que debemos colocar a la entrada del filtro, ejecutar el algoritmo LMS y obtener la salida. Con ese objetivo, planteamos un ejemplo de un cancelador de ruido con referencia [Hayes, 1996].

Planteamiento del problema

Si tenemos las señales $d(n)$, $v_1(n)$, y $v_2(n)$ en donde la señal que deseamos recuperar es $d(n)$; sin embargo, no la conocemos, más bien se saben las características de la señal ya sumada con el ruido blanco gaussiano, la cual es:

$$x(n) = d(n) + v_1(n)$$

Claramente sin ninguna información sobre $d(n)$ o $v_1(n)$ no es posible separar la señal del ruido. Sin embargo, si obtenemos una señal de referencia, $v_2(n)$, la cual está en correlación con $v_1(n)$, entonces esta señal de referencia puede usarse para estimar el ruido $v_1(n)$, y esta estimación puede substraerse entonces de $x(n)$ para formar una estimación de $d(n)$,

$$\hat{d}(n) = x(n) - \hat{v}_1(n)$$

Por supuesto que $d(n)$, $v_1(n)$ y $v_2(n)$ son procesos conjuntos estacionarios en sentido amplio, y si la autocorrelación $r_{v_2}(k)$ y la correlación cruzada $r_{v_1v_2}(k)$ son conocidas, entonces podemos diseñar un filtro Wiener para encontrar el mínimo cuadrático medio de la estimación de $v_1(n)$. En la práctica, sin embargo, una suposición de estacionariedad no es generalmente apropiada y, aun si lo fuera, los requerimientos estadísticos de $v_2(n)$ y $v_1(n)$ son generalmente desconocidos. Por consiguiente, como una alternativa de los filtros Wiener, consideremos un cancelador de ruido adaptativo mostrados en la Figura 3.2. Si la señal de referencia $v_2(n)$ está incorrelada con $d(n)$, entonces habrá que minimizar el error cuadrático medio $E\{|e(n)|^2\}$ que es equivalente a minimizar $E\{|v_1(n) - \hat{v}_1(n)|^2\}$. En otras palabras, la salida del filtro adaptativo es la

estimación del mínimo cuadrático medio de $v_1(n)$. Básicamente, si no hay información sobre $d(n)$ en la señal de referencia $v_2(n)$, entonces es mejor que el filtro adaptativo pueda minimizar $E\{|e(n)|^2\}$ al remover la parte de $e(n)$ que puede estimarse de $v_2(n)$ que está correlacionada con $v_1(n)$. Puesto que la salida de un filtro adaptativo es el mínimo cuadrático medio estimado de $v_1(n)$, entonces con $e(n)$ se procede a estimar el mínimo medio cuadrático $d(n)$.

Como un ejemplo específico, veamos como estimar una señal sinusoidal que deseamos recuperar:

$$d(n) = \sin(n\omega_o + \varphi)$$

con $\omega_o = 2\pi * 0.025$ y el ruido blanco $v_1(n)$ y $v_2(n)$ es generado por las ecuaciones de diferencia de primero orden

$$v_1(n) = 0.8v_1(n-1) + g(n)$$

$$v_2(n) = -0.6v_2(n-1) + g(n)$$

donde el $g(n)$ es ruido blanco de media cero ($\mu=0$) y varianza uno ($\sigma^2=1$), que está incorrelada con $d(n)$. En la Figura 3.3a se grafican 1000 muestras de la sinusoidal y en la Figura 3.3b se ve la señal ruidosa $x(n) = d(n) + v_1(n)$. La señal de referencia $v_2(n)$ es usada para estimar $v_1(n)$ y se muestra en la Figura 3.3c. Usando un cancelador de ruido adaptativo de 12 coeficientes y μ igual a 0.025, se puede usar el algoritmo LMS. Más adelante se presentan los resultados.

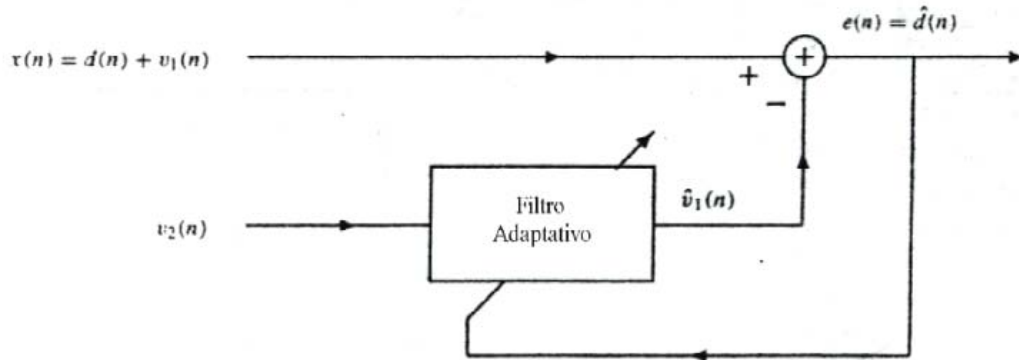
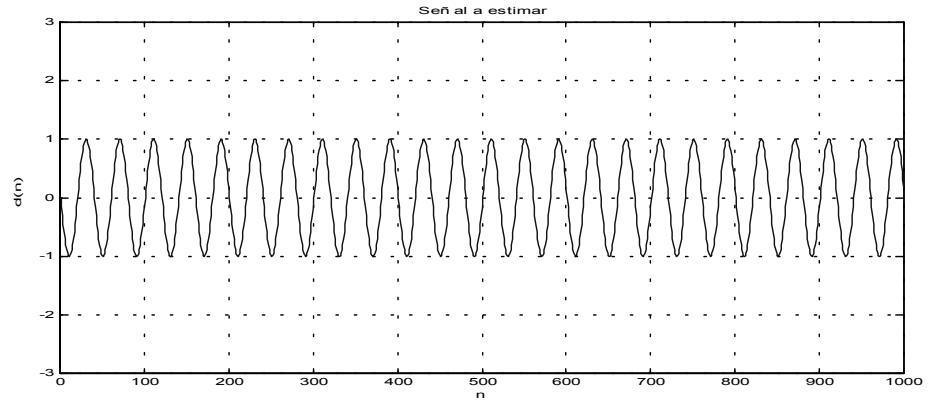
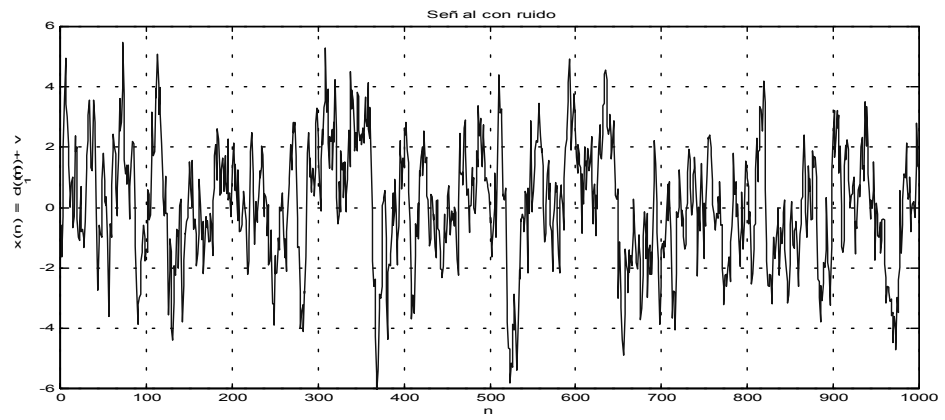


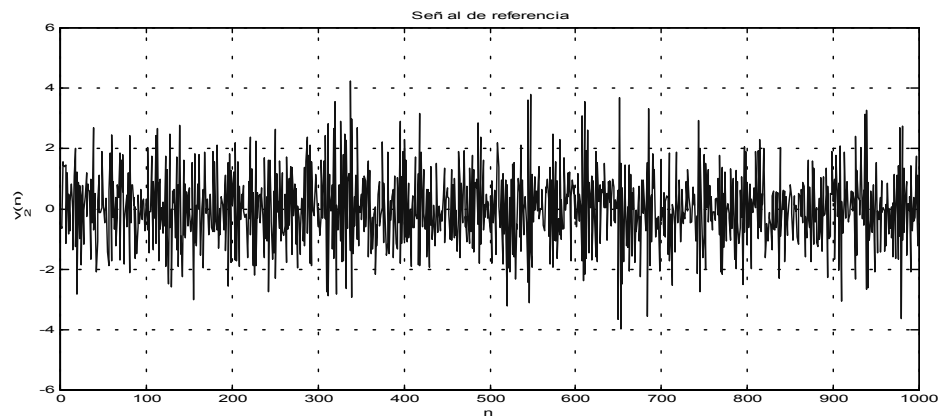
Figura 3.2 Cancelador de ruido usando un filtro adaptativo con referencia



(a)



(b)



(c)

Figura 3.3 Ejemplo de cancelador de ruido. (a) Señal sinusoidal a estimar (b) Ruido añadido a sinusoidal (c) Señal de referencia usada en el segundo sensor

Solución utilizando el LMS

Con las características anteriores se escribió un código en lenguaje de Matlab para generar estas señales.

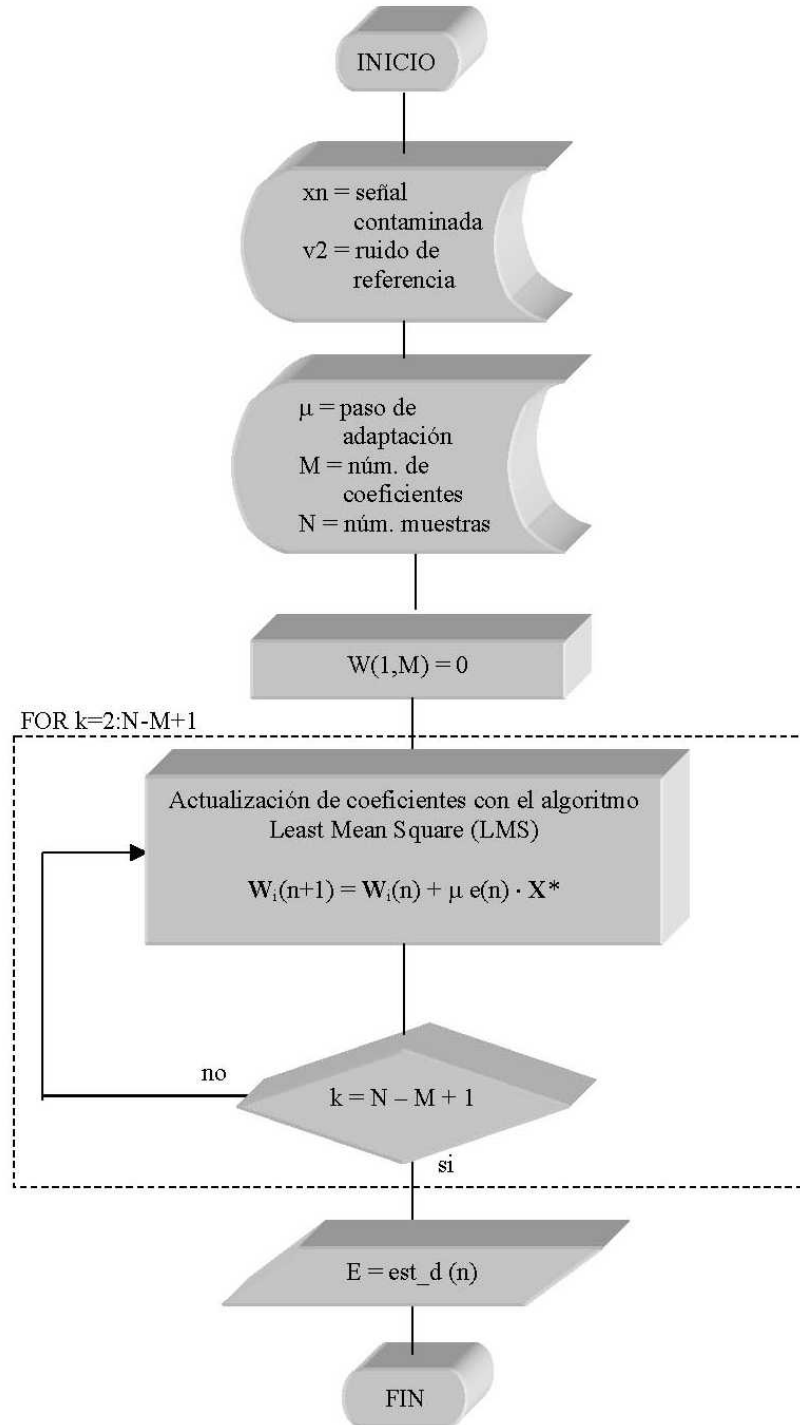


Figura 3.4 Diagrama de flujo del algoritmo LMS usando Matlab

Recuadro 3.1 Generación de señales

```

%señal deseada
t = (0:1:1000-1);
d = sin(2*pi*.025*t+pi);
d = d(:).'; %forma un vector de 1*n sea el original n*1 o 1*n

%interferencia
sigma=sqrt(1); %sigma=desviacion standart
               %sigma^2=varianza

g =sigma*randn(size(t));
v1 = filter(1,[1,-0.8],g);
v1 = v1(:).'; %forma un vector de 1*n sea el original n*1 o 1*n
v2 = filter(1,[1,0.6],g);
v2 = v2(:).'; %forma un vector de 1*n sea el original n*1 o 1*n

%señal deseada + interferencia
x = d + v1;
    
```

Recuadro 3.2 Algoritmo LMS

```

X=convm(x,nord);

[M,N] = size(X); %N=nord, es decir numero de coeficientes

if nargin < 5, a0 = zeros(1,N); end

a0 = a0(:).'; %forma un vector de 1*n sea el original n*1 o 1*n

E(1) = d(1) - a0*X(1,:).';

A(1,:) = a0 + mu*E(1)*conj(X(1,:));

if M>1

    for k=2:M-nord+1;

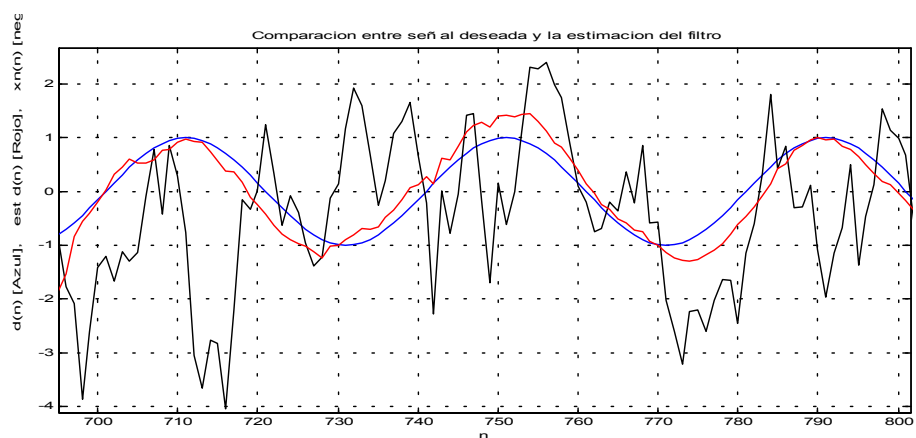
        E(k) = d(k) - A(k-1,:)*X(k,:).';

        A(k,:) = A(k-1,:) + mu*E(k)*conj(X(k,:));

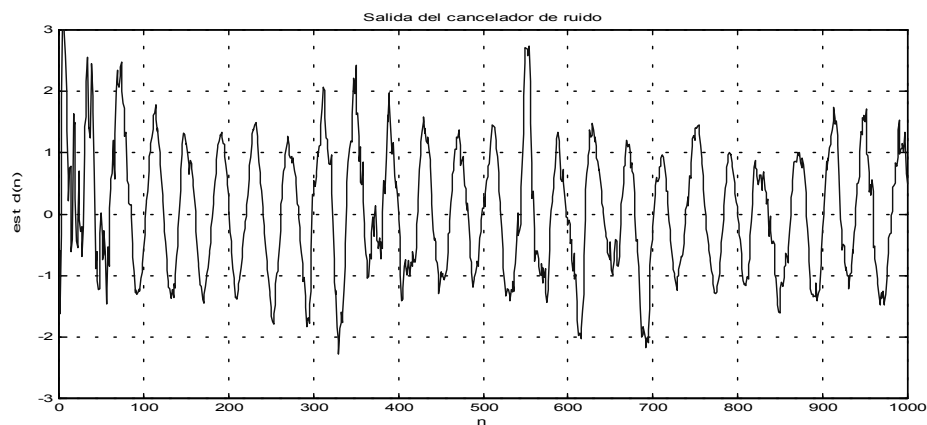
    end;

end;
    
```

En los recuadros anteriores, se muestran segmentos del programa *cancelwz_22.m*. En el Recuadro 3.1, se ve que la señal generada es de 1000 muestras, con amplitud 1 y frecuencia igual a 0.025Hz. Las características del ruido blanco gaussiano de media cero ($\mu=0$) y varianza 1 ($\sigma^2=1$), se obtienen con la instrucción *randn*, que se incluye en las librerías de Matlab. El Recuadro 3.2 muestra la simplicidad del algoritmo LMS. Se requiere tan solo un lazo *for* para realizar las iteraciones y calcular la salida $y(n)$, calcular el error $e(n)$ y actualizar los coeficientes. Con el programa completo *cancelwz_22.m* se obtuvieron los siguientes resultados.



(a)



(b)

Figura 3.5 Salida del filtro de orden 12 (LMS). (a) Comparación con otras señales
(b) Señal filtrada

En la Figura 3.5a, la señal de color azul es la señal que se desea recuperar. Es evidente en la Figura 3.2 que en el filtro adaptativo entra una señal que es la suma de la señal a recuperar más el ruido; es decir, no se conoce la señal sin contaminación, pero para fines didácticos, podemos hacer el gráfico para compararla. Lo anterior significa que la señal de entrada al filtro es la de color negro, que es la señal contaminada y la señal filtrada la representamos con el color rojo.

Solución utilizando el NLMS

Utilizando las mismas entradas del Recuadro 3.1 (Figura 3.3). A continuación se presentan los resultados obtenidos utilizando el algoritmo NLMS, con el cual es evidente el mejor desempeño de este con respecto al LMS. Se usa un valor de $\beta=0.25$ y el código NLMS descrito en el Recuadro 3.3.

Recuadro 3.3 Algoritmo NLMS

```
X=convm(x,nord);
[M,N] = size(X);
if nargin < 5, a0 = zeros(1,N); end
a0 = a0(:).';
E(1) = d(1) - a0*X(1,:).';
DEN=X(1,:)*X(1,:)' + 0.0001;
A(1,:) = a0 + beta/DEN*E(1)*conj(X(1,:));
if M>1
    for k=2:M-nord+1
        E(k) = d(k) - A(k-1,:)*X(k,:).';
        DEN=X(k,:)*X(k,:)' + 0.0001;
        A(k,:) = A(k-1,:) + beta/DEN*E(k)*conj(X(k,:));
    end
end
```

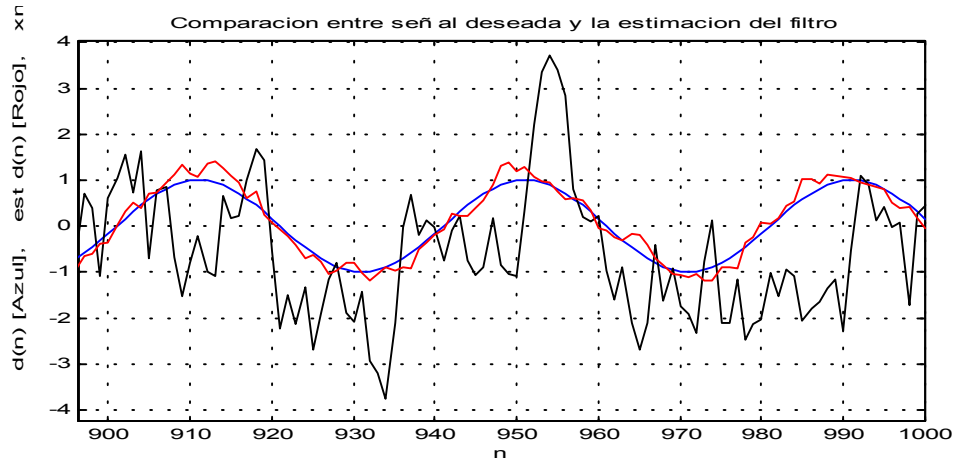


Figura 3.6 Salida del filtro de orden 12 (NLMS)

Al igual que en la Figura 3.5, aquí en la Figura 3.6, la señal de color azul es la señal que se desea recuperar, la de color negro es la señal contaminada y la señal filtrada es de color rojo.

Convergencia del algoritmo LMS

Con el objetivo de evaluar la eficiencia del algoritmo, pueden verse algunos parámetros estadísticos. Esto se hace con el fin de medir el rendimiento del algoritmo LMS.

Para comenzar, en el algoritmo LMS un valor crítico y decisivo para obtener buenos resultados, es el valor de μ^3 . En la Figura 3.7 se ilustran la evolución de los coeficientes para 1000 muestras tomadas de una señal. Este gráfico se realizó con el programa *coef_evol.m* que se encuentra en los anexos de este trabajo de graduación. En este gráfico es posible ver que si se quiere que la convergencia se alcance pronto, el valor de μ es grande, pero se sacrifica la estabilidad. No obstante, cuando el valor de μ es demasiado pequeño (tendiendo a cero), los coeficientes se toma bastante tiempo en alcanzar los valores más eficientes [Solano, 2003].

³ Paso o velocidad de adaptación o factor de convergencia

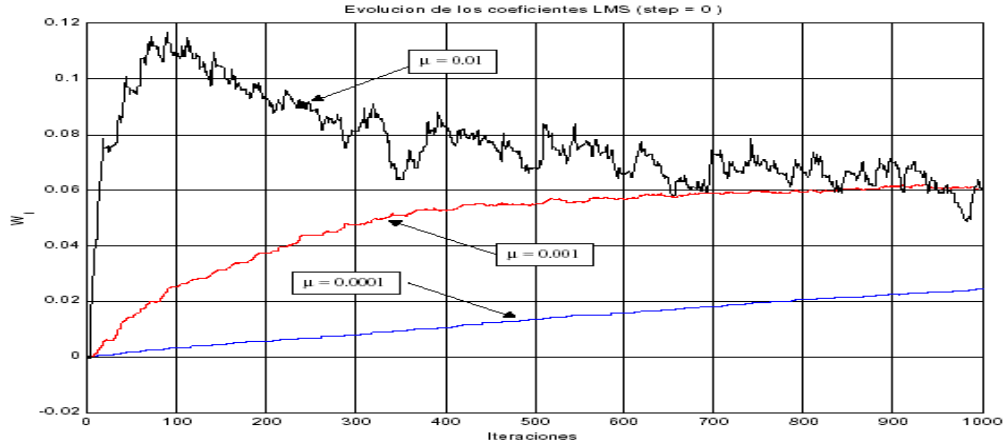


Figura 3.7 Evolución de los coeficientes del algoritmo LMS

En nuestro caso los coeficientes se comportaron de manera muy parecida a la curva que corresponde a $\mu = 0.001$, tal como se ve en la Figura 3.8.

Esta figura muestra el rango de variación de los coeficientes. Se observa que para w_1 el rango a partir de 700 muestras cambia de 1.2 a 1.4; o sea, 0.2 de diferencia. El w_2 tiene un rango un poco menor a 0.2, lo que lo hace más estable. Dado que son 12 pesos o coeficientes, donde algunos son más estables que otros, es posible obtener la salida que se ve en Figura 3.5.

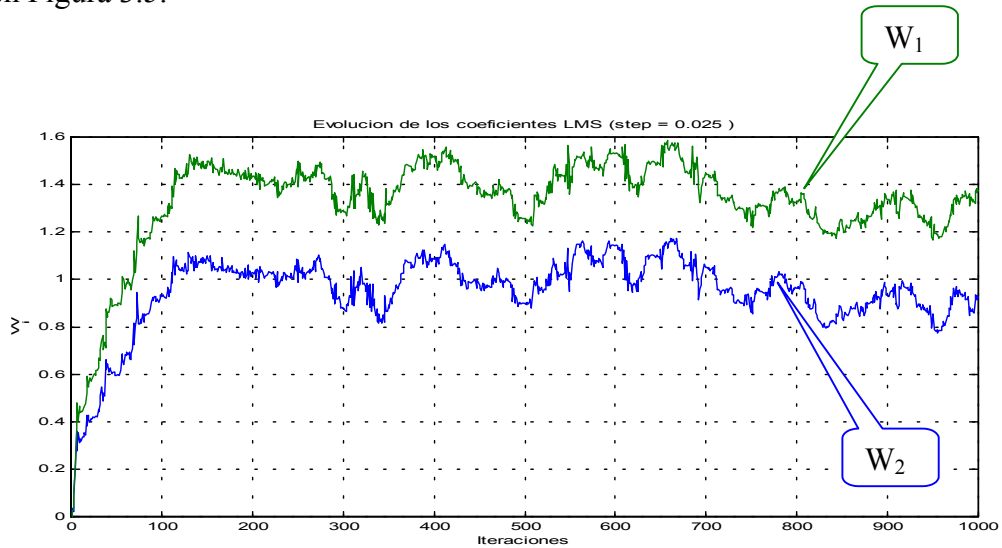


Figura 3.8 Evolución de los coeficientes del algoritmo LMS con $\mu = 0.0025$

Además se puede evaluar el error cuadrático medio (o Ensemble Averaged Squared Error) para distintos pasos de convergencia y comparar los resultados experimentales obtenidos con los teóricos.

Puesto que la curva de aprendizaje es una representación de $\xi(n) = E\{|e(n)|^2\}$ frente a n , podemos aproximar esta curva de aprendizaje promediando los valores de $|e(n)|^2$ obtenidos en distintas realizaciones. Supongamos que implementamos el predictor K veces, y en cada realización k tenemos el error cuadrático en el instante n denotado por $|e_k(n)|^2$. Así tenemos los resultados en la Figura 3.5. En este caso se eligió $K = 50$, y un vector de pesos inicializado a cero. El valor del paso de adaptación fue $\mu = 0.025$.

Este es otro parámetro estadístico, pero es evidente a simple vista que el error va reduciéndose en cada iteración, lo que garantiza que el LMS ha alcanzado la convergencia. Por lo visto en la Figura 3.8, que el rango dentro del que varían los coeficientes es de 0.2 y lo que se ve en la Figura 3.9 que la curva descende pero no toca el eje cero sino que se detiene en 0.025, queda demostrado que los coeficientes no alcanzan los valores óptimos, pero son bastante buenos.

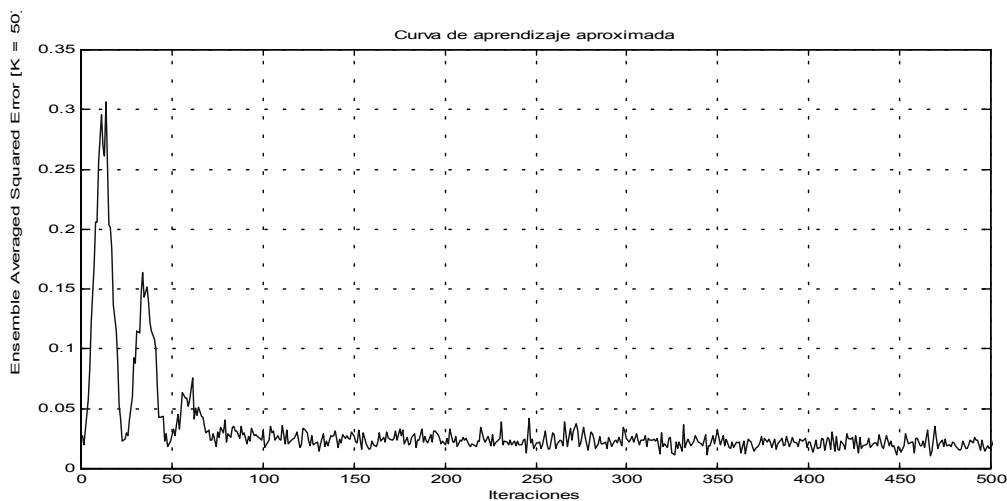


Figura 3.9 Curva de aprendizaje, con $K = 50$.

3.4 CONCLUSIONES

- Dado que Matlab está basado en un sofisticado software de matrices para el análisis de sistemas de ecuaciones, esto permite resolver complicados problemas numéricos sin necesidad de escribir un programa. El algoritmo LMS se resuelve escribiendo muy pocas instrucciones y con tiempos de ejecución muy rápidos.
- El diseño de un filtro adaptativo no requiere las técnicas tradicionales para tratamiento digital de la señal, tal como definir la frecuencia de corte, definir la clase de ventana que se utilizará o manipular el ancho de la banda de transición.
- El uso de ruido blanco para un sistema de filtro lineal adaptativo, se debe a que es una función idealizada que se considera una excelente entrada a un sistema adaptativo por su semejanza a muchos casos reales.
- En las figuras 3.5 y 3.6, vemos que después de aproximadamente 100 iteraciones el filtro adaptativo está produciendo una estimación bastante exacta de $d(n)$ y, después de aproximadamente 200 iteraciones el filtro adaptativo parece haber alcanzado la estabilidad en su comportamiento. Aunque es obvio que el NLMS tiene un mejor comportamiento.

3.5 REFERENCIAS

HAYKIN, SIMON (1996), “Adaptive Filter Theory”, 3ra. Edition, Prentice Hall

HAYES, MONSON H. (1996), “Statistical Digital Signal Processing and Modeling”,
JohnWiley & Sons

SOLANO, EDUARDO (2003), “Tratamiento Digital de la Señal ”, Centro Politécnico
Superior Universidad de Zaragoza

<http://www.gtc.cps.unizar.es/~eduardo/docencia/tds/Temario.html#introduccion>

CAPITULO

4

*Implementación del LMS,
usando la DSP56L811EVM*

4.1 INTRODUCCIÓN

Existen diferentes fabricantes de dispositivos de procesamiento digital de señales (DSP, por sus siglas en inglés) que permiten implementar el algoritmo LMS. Este trabajo de graduación utiliza el DSP56L811, fabricado por Motorola Inc., para implementar el algoritmo LMS y con ello, comprobar la eficiencia del mismo algoritmo y del dispositivo.

Con el fin de promover un mayor grado de información y conocimiento del dispositivo en cuestión, se realiza dentro de este capítulo una descripción general del hardware utilizado. Pero para aquellas personas que deseen conocer un poco más, existen muchos otros documentos que pueden leerse, además de lo que se incluye en los anexos [Blanco y Calderón, 2001; Motorola, 1996].

Luego, se presenta una descripción del procedimiento realizado para la obtención del archivo ejecutable. Este programa calcula los pesos o coeficientes necesarios para la implementación eficiente del algoritmo LMS en el DSP56L811 y luego realiza la convolución de la señal de entrada por los coeficientes. Al final se muestran los resultados obtenidos y sus respectivas conclusiones.

La elaboración de este capítulo, tiene la visión de que los resultados hablen por sí mismos sobre las potencialidades del DSP56L811, aunque se incluye una descripción básica de la tarjeta de evaluación y detalles del propio Procesador Digital de Señales (DSP56L811).

4.2 DESCRIPCIÓN GENERAL DE LA DSP56L811EVM

La DSP56L811EVM (ver Figura 4.1) pertenece a la familia DSP56800, la cual tiene un núcleo (CPU) de 16 bit, que consta básicamente de una Unidad Microcontroladora (MCU) y un DSP. De entre sus muchas características se tienen:

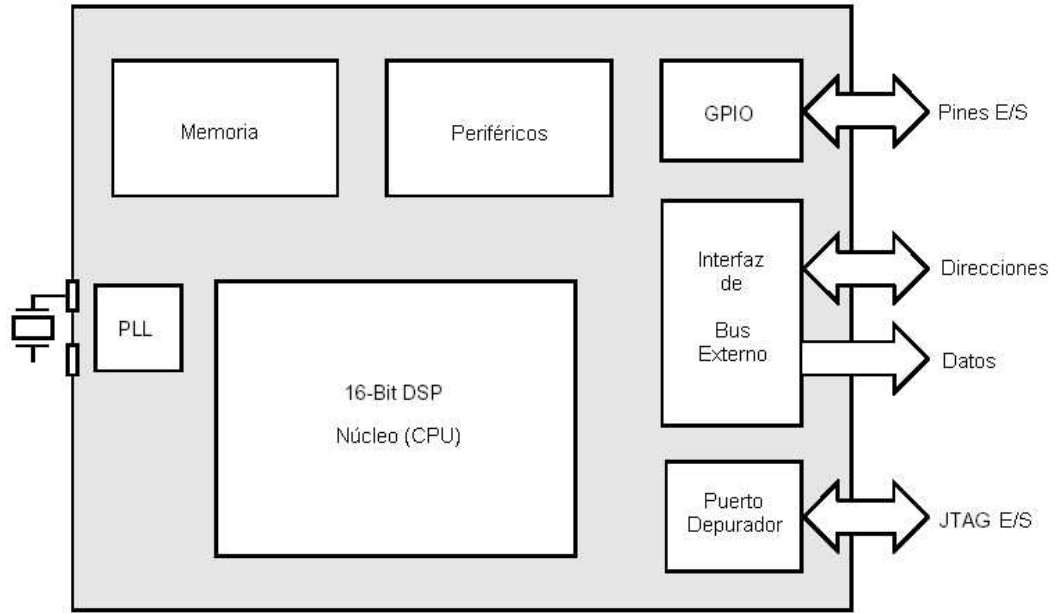


Figura 4.1 Diagrama de bloques de la familia DSP56800

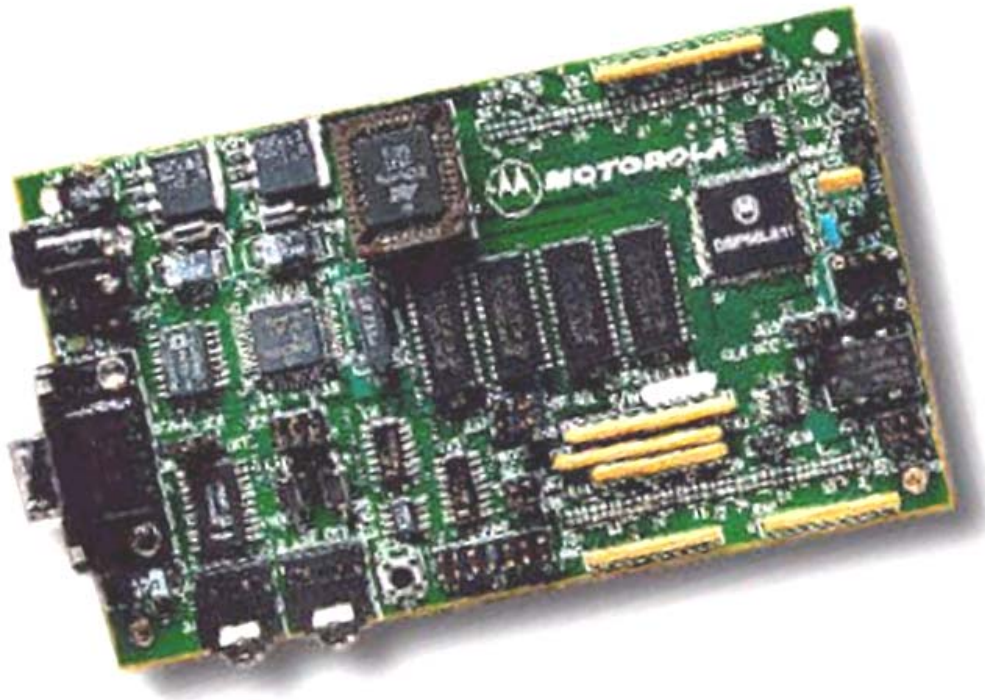


Figura 4.2 Fotografía de la tarjeta de evaluación DSP56L811EVM

Primero, un amplio conjunto de instrucciones y modos de direccionamiento, el diseño de un código eficiente para las tareas de la MCU, instrucciones de manipulación de bits, una pila de software y otra de hardware. Permite interrupciones y anidar programas, soporta programación estructurada. Además, las instrucciones del programa pueden ordenar la ejecución de operaciones aritméticas y movimientos en paralelo desde la sección de memoria de datos, esto ocurre cuando dentro de una operación aritmética también se puede realizar lectura y escritura a la memoria de datos. En la Figura 4.2 también se puede observar los diferentes componentes que constituyen el núcleo de la DSP56800. Lo más notable son los puertos de entrada y salida de propósitos generales (GPIO, por sus siglas en inglés), el cronómetro de propósito general, el cronómetro de tiempo real y el cronómetro de seguridad (watchdog), las memorias RAM y ROM. Cada interfaz de periféricos llega al núcleo del DSP56800 a través de un bus de periféricos, el cual permite el montaje relativamente fácil de diseños estándares de periféricos.

Generalidades del núcleo

En la Figura 4.3 se muestra un diagrama de bloques del núcleo del DSP56800. Este es un DSP CMOS de 16 bit, el cual está compuesto por una unidad de generación de direcciones de 16 bit (AGU, por sus siglas en inglés), un decodificador de programas, un emulador interno (OnCE, por sus siglas en inglés), tres tipos de bus; el bus de control, datos y direcciones y un conjunto de instrucciones.

Entre las características principales del núcleo se encuentran:

- ◆ El núcleo del Procesador Digital de Señales (DSP)
 - Eficiente dispositivo DSP de 16 bit, miembro de la familia DSP56800
 - Hasta 20 millones de instrucciones por segundo (MIPS) a 40MHz
 - En un solo ciclo multiplicación acumulación en paralelo de 16 x 16 bit
 - Dos acumuladores de 36 bit, incluyendo los bit de extensión
 - Conjunto de modos de direccionamiento para instrucciones en paralelo, muy propio de la DSP
 - Lazos de hardware DO y REP

- Lazos DO anidados con lazos de software
 - Bus de direcciones:
 - Un bus de 16 bit de direccionamiento de memoria interna (XAB1)
 - Un bus de 16 bit de direccionamiento de memoria interna (XAB2)
 - Un bus de 19 bit de direccionamiento de programa interno (PAB)
 - Un bus de 16 bit de direccionamiento externo (EAB)
 - Bus de datos:
 - Un bus de datos bidireccional de 16 bit de memoria interna (CGDB)
 - Un bus de datos unidireccional de 16 bit de memoria interna (XDB2)
 - Un bus de datos bidireccional de 16 bit de periféricos dedicados (PGDB)
 - Un bus de datos bidireccional de 16 bit para programas internos (PDB)
 - Un bus de datos bidireccional de 16 bit externo (EDB)
 - Conjunto de instrucción que controlan tanto el DSP como al MCU
 - Modo de direccionamiento similar al MCU e instrucciones de código compacto
 - Un compilador de C eficiente y soporta variables locales
 - Subrutinas e interrupciones, de longitud limitada
- ◆ Memoria
- Arquitectura Harvard que permite hasta tres accesos simultáneos a memoria de programa y datos
 - Memoria RAM de 1K x 16 para programa (aquí se almacenan el código de programa)
 - Memoria bootstrap ROM de 64 x 16 (es un sistema de arranque, encargado de realizar el chequeo de los componentes antes de pasar el mando al usuario)
 - Memoria RAM de 2K x 16 para datos
 - Los programas pueden correr fuera de la memoria de datos
- ◆ Periféricos y circuitos de soporte
- Interfaz de memoria externa (EMI, External Memory Interface)
 - Dieciséis pines dedicados para propósitos generales entradas/salidas (GPIO, General Purpose Input/Output) (ocho pines programable para interrupciones)

- Interfaz serial para periféricos (SPI, Serial Peripheral Interface) soporta: Dos puertos configurables de 4 pines (SPI0 y SPI1) (u ocho líneas adicionales para GPIO)
 - Puede manejar LCD, subsistemas de A/D, y sistemas MCU
 - Soporta comunicación dentro del procesador en un sistema master múltiple
 - Manejo de demanda de dispositivos como maestro o esclavo con altas tasas de procesamiento de datos
 - La interfaz serial síncrona (SSI, Synchronous Serial Interface) soporta: Un puerto de 6 pines (o seis líneas adicionales GPIO)
 - Maneja dispositivos serial con uno o más estándar de la industria, tales como codecs, otros DSP, microprocesadores, y otros periféricos
 - Transmisión asíncrona o síncrona y secciones de recepción con relojes internos/externos separados o compartidos y tramas de sincronismo
 - Modo redes usando tramas de sincronismo y hasta 32 secciones de tiempo (time slots)
 - Longitud palabra de datos 8 bit, 10 bit, 12 bit, y 16 bit
 - Tres cronómetros programables (acceso usando dos pines de E/S que pueden también programarse como dos líneas adicionales para GPIO)
 - Dos líneas de control externas interrupción/modo
 - Un reset externo para aplicar reset por hardware
 - Puerto emulador de 5 pines usando JTAG/On-Chip (OnCE) nada complicado (transparente al usuario), con proceso de depuración a una velocidad independiente del DSP
 - Sintetizador de frecuencias PLL (del inglés, Phase Locked Loop) programable a través de software
 - Cronómetros de operaciones de cálculo apropiadamente (COP, Computer Operating Properly) e interrupción de tiempo real (RTI, Real Time Interrupt)
- ◆ Diseño eficiente para el uso de energía
- Ahorro de energía por los modos disponible de ‘espera’ (wait) y ‘detenida’ (stop)

- Totalmente estática, tecnología HCMOS diseñada para operar frecuencias desde 40MHz hasta DC (0.0Hz)
- 100 pines agrupados en paquete plástico TQFP (Thin Quad Flat Pack)
- Voltaje requerido de 2.7V a 3.6V

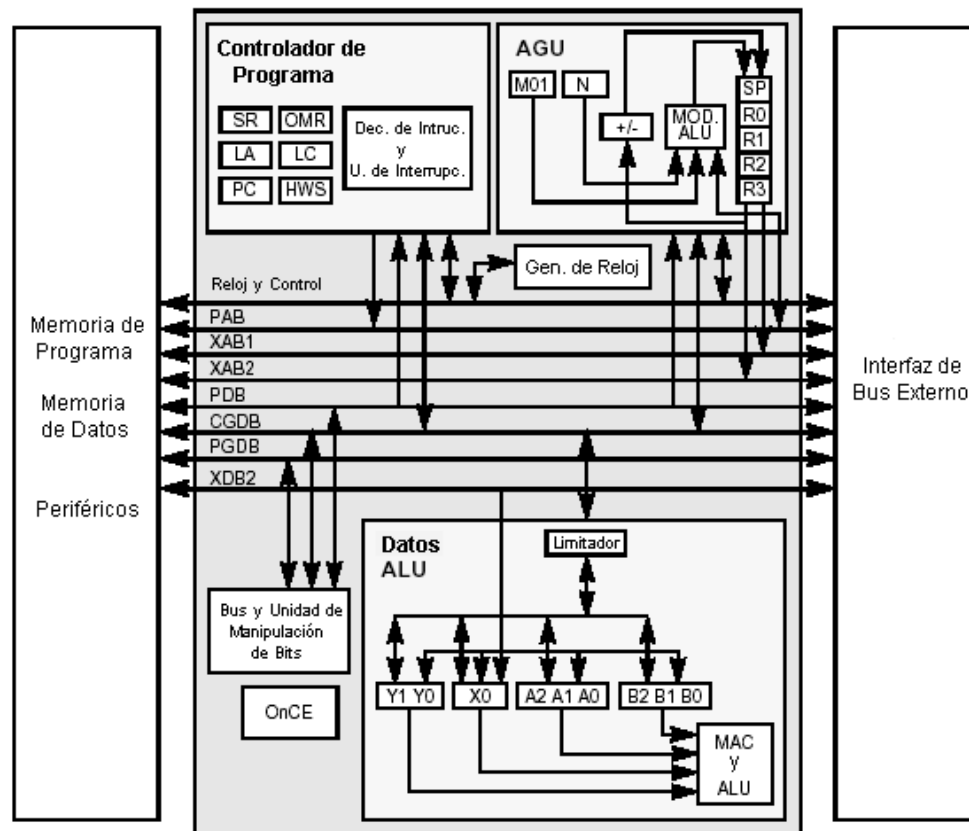


Figura 4.3 Diagrama de bloques del núcleo de DSP56800

Este dispositivo posee muchas otras características, todas diseñadas con el objetivo de procesar señales digitales, pero para descubrirlas es necesario involucrarse más de lleno con el desarrollo de aplicaciones de DSP, utilizando el módulo DSP56L811EVM [Motorola, 1996]. Algo fundamental para utilizar este dispositivo es conocer la memoria con que se cuenta (ver Figura 4.4) para poder aprovechar tanto su memoria de datos, como de programa.

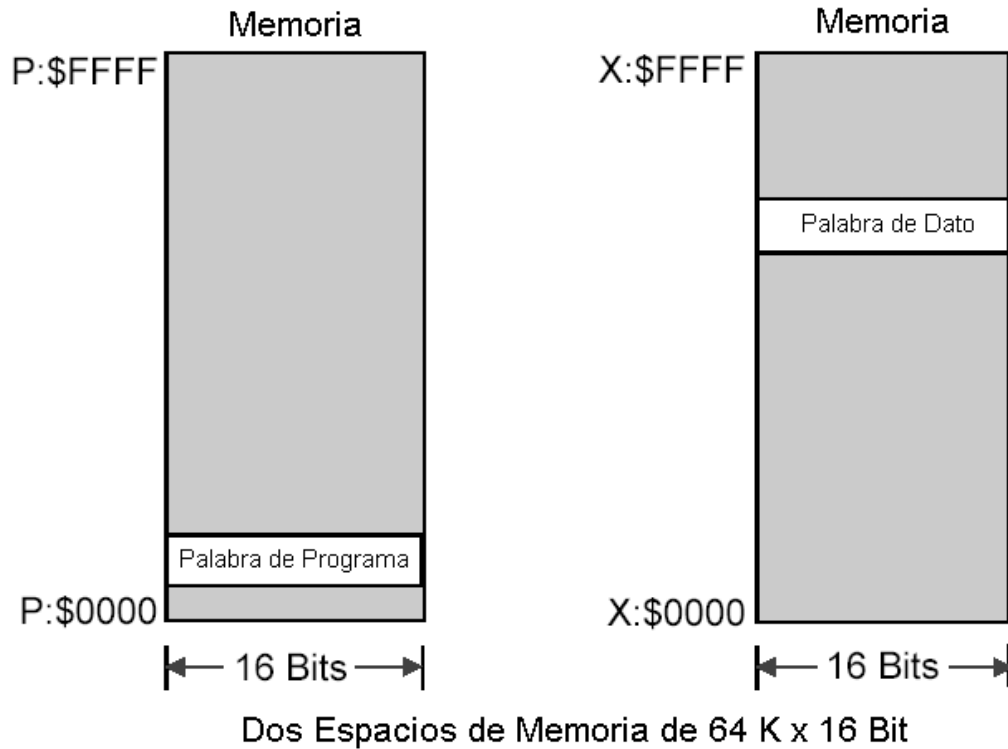
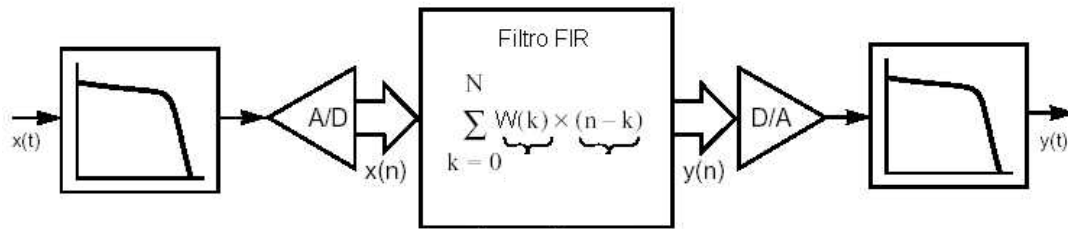


Figura 4.4 Diagrama del mapa de memoria de programas (P) y datos (X)

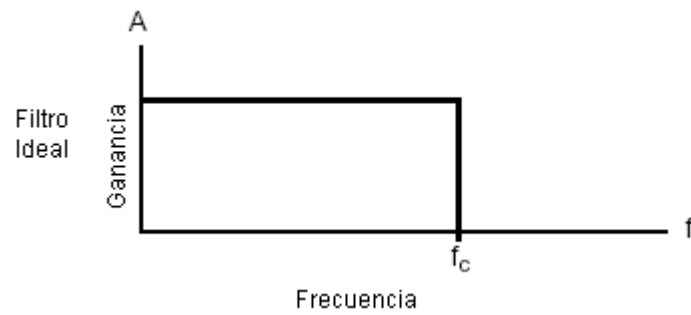
4.3 TRATAMIENTO DIGITAL DE SEÑALES

El DSP56L811 es un procesador aritmético, pero también es capaz de realizar el muestreo de una señal en tiempo real a tiempos regulados y digitalizados. Ejemplos de aplicaciones con el DSP56L811 son:

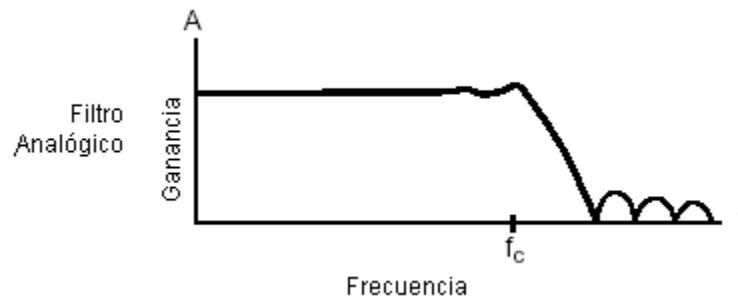
- Filtrado Digital
- Convolución (mezcla de dos señales)
- Correlación (comparación de dos señales)
- Rectificación, amplificación, y/o transformación, etc.



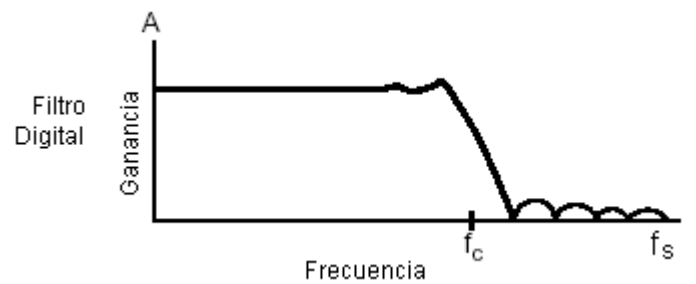
(a)



(b)



(c)



(d)

Figura 4.5 Tratamiento Digital de Señales, (a) Esquema de Entrada/Salida y Proceso en la DSP, (b) Respuesta en frecuencia de un Filtro Ideal, (c) Respuesta en frecuencia de un Filtro Analógico, (d) Respuesta en frecuencia de un Filtro Digital en la DSP

Procesado digital de señales analógicas y filtros FIR

En aplicaciones de ingeniería es frecuente encontrarse con la tarea de manipular también señales discretas, además de las analógicas. La Figura 4.5 muestra un ejemplo de señales analógicas procesadas digitalmente utilizando el DSP56L811. Estos procedimientos requieren un convertidor analógico-digital (A/D) y un convertidor digital-analógico (D/A), además del DSP.

El procesamiento en este circuito comienza con la limitación en banda de la señal con un filtro anti-aliasing. Eliminando el solapamiento causado por el muestreo. La señal muestreada se digitaliza con un convertidor A/D y es enviada al DSP. Ahora, la implementación del filtro en el DSP es estrictamente materia de programación.

La DSP posee un codificador-decodificador (**codec**, por sus siglas en inglés) con la característica de modular por codificación de pulso (PCM, por sus siglas en inglés). Este se encarga de realizar la conversión de A/D y D/A, de tal manera, que la entrada y la salida analógica son procesadas por el *codec*.

Para el caso del filtro adaptativo lineal, el **codec** es utilizado tanto para digitalizar la señal contaminada, como para después del procesamiento digital volverla una señal analógica. Estas acciones entrada/salida de la DSP56L811EVM, se relacionan con el procedimiento de recepción y transmisión, manejado por software. La frecuencia de muestreo (f_s) se establece a través de software, y cumpliendo el criterio de Nyquist, se ajustó a un valor 8kHz.

En resumen las ventajas de utilizar el DSP56L811 son:

- Pocos componentes
- Desempeño y estabilidad determinada
- Amplio rango de aplicaciones
- Filtros con una tolerancia más cerrada
- Alta inmunidad al ruido
- Mejor rechazo a variaciones de potencia

Operación multiplicación–acumulación (MAC)

La familia DSP56800 no es un IC para aplicaciones particulares, sino para aplicaciones de propósitos generales. Los atributos fundamentales para realizar el tratamiento digital de la señal son:

- Operación de multiplicación acumulación (MAC)
- Captura dos operaciones en un simple ciclo de instrucción
- Control de programas para operaciones versátiles
- Entrada/Salida para mover datos dentro y fuera de la DSP.

La operación multiplicación–acumulación (MAC, por sus siglas en inglés) es una operación fundamental usada en la DSP. Como se muestra en la Figura 4.6, los operandos $W(k)$ y $X(n-k)$ se direccionan a la memoria de datos, para que luego se lleve a cabo la operación de multiplicación y el resultado es sumado o agregado al acumulador, este proceso es realizado por la instrucción MAC.

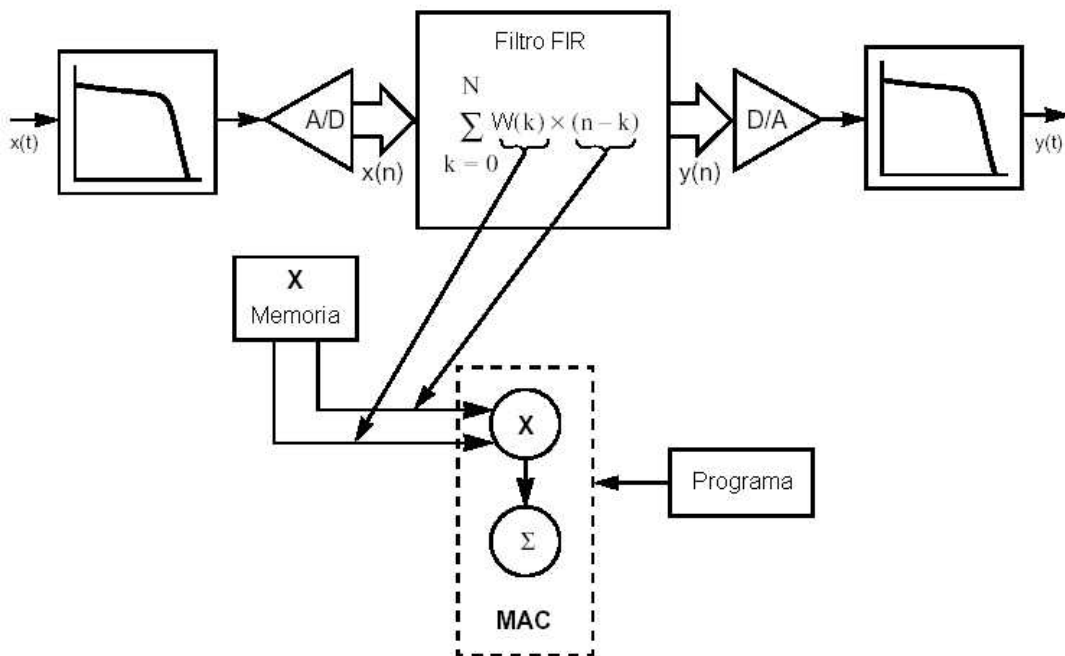


Figura 4.6 Diagrama de la instrucción MAC y movimientos paralelos (FIR)

Ya que la memoria y la instrucción MAC son independientes, la DSP puede desempeñar dos movimientos de memoria, una multiplicación y una acumulación y dos direccionamientos de actualización en una simple operación. Además, por medio de la interfaz SSI, el codec y el control de interrupciones puede recibir y transmitir datos desde el DSP56L811.

Aritmética Modular

La DSP soporta dos tipos de aritmética, requiriendo que la AGU genere direcciones de manera especial. Para ello se cuenta con el registro **m01** que determina el tipo de aritmética de direccionamiento que se va usar. Las opciones existentes son [Motorola, 1996]:

- Aritmética lineal
- Aritmética modular (módulo n, donde $n < 16K$)

En nuestro caso, fijamos un buffers para los datos de entrada de igual tamaño al buffers donde se almacenan los coeficientes del filtro FIR.

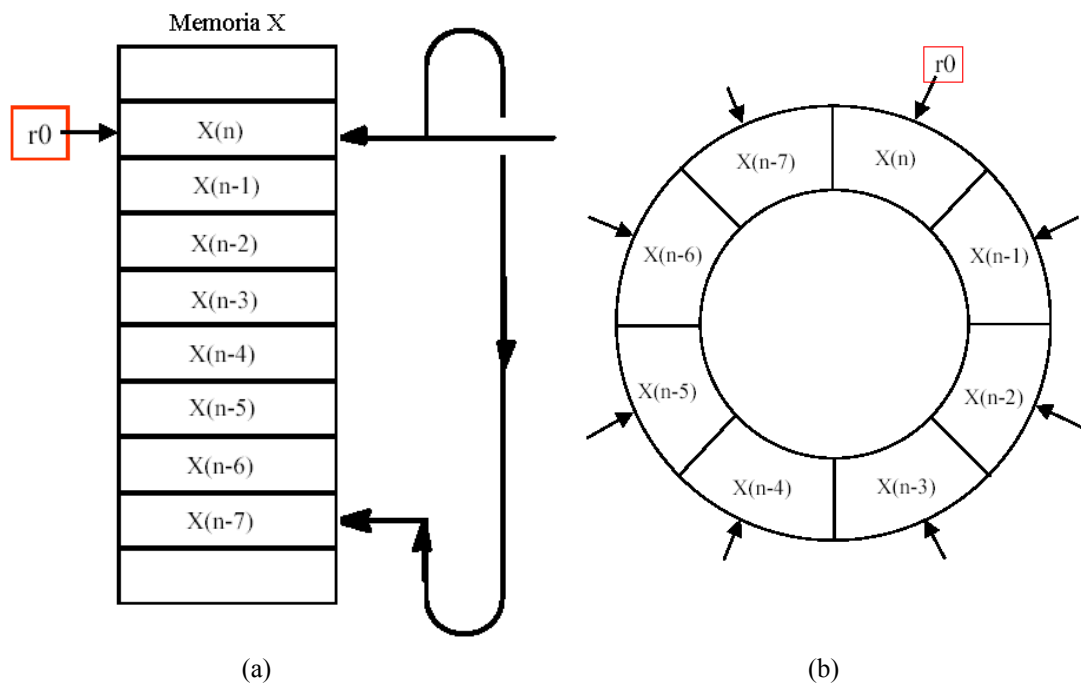


Figura 4.7 Aritmética modular. (a) Representación física; (b) Representación funcional

La aritmética modular, se implementa para el acceso a buffers circulares (memoria temporal), que pueden ser útiles en el manejo de ciertas estructuras de datos y algoritmos de acceso. Por ejemplo, la utilización de buffers circulares cuando llega una señal muy grande, la cual no puede almacenarse en su totalidad. En las figuras 4.7a y 4.7b se representa el concepto de buffer circular. Los apuntadores que pueden ser usados solo son r0 y r1 [Universidad de Chile, 2003].

4.4 PREPARACIÓN PARA EL USO DE LA DSP56L811EVM

El proceso de implementación de un algoritmo de tratamiento digital de señales tal como un filtro adaptativo en la DSP56L811EVM, consta básicamente de tres etapas, las cuales se describen a continuación [Motorola, 1997]:

1. Crear un archivo fuente, conteniendo el código en lenguaje ensamblador del algoritmo a implementar. Este archivo debe tener extensión “asm”. Dicho archivo puede ser escrito en cualquier editor de texto.
2. Compilar y enlazar el archivo fuente usando los programas **asm56800.exe** como compilador y el de enlace **dsplnk.exe**, de la siguiente manera:

Estando en la pantalla de comandos del MS-DOS, se escribe lo siguiente:

```
...\asm56800 -b -l lms.asm, y luego se pulsa ENTER.
```

Con esta instrucción se crean los siguientes archivos:

lms.cln; archivo objeto que se convertirá en ejecutable al enlazarlo.

lms.lst; archivo de listado, puede quitar esta opción al no colocar -l.

Si el archivo fuente es correcto, se crearán los archivos antes mencionados, de lo contrario, se presentarán llamadas de atención (advertencias) o errores; el programa compilador es capaz de especificar la línea del programa fuente donde se encuentra la advertencia o error, con lo que se debe proceder a corregir el error o la advertencia.

Si todo resulta sin errores, entonces se deberá digitar lo siguiente, siempre en ambiente de MS-DOS:

...`\dsplnk -b lms.cln`, y luego se pulsa ENTER.

Con esta instrucción se crea el archivo `lms.cld` que es el archivo ejecutable en la tarjeta.

3. Cargar el archivo ejecutable en la tarjeta por medio de la comunicación del puerto serie utilizando la interfaz gráfica del programa EVM56811, para lo cual se recomienda primero leer el Manual incluido con la tarjeta de evaluación [Domain Technologies, 1996].

Todo lo anterior es posible hacerlo mucho más rápido utilizando la *utilidad para compilar y enlazar* diseñada en Matlab¹. En la Figura 4.8 se observa el aspecto que tiene y se deduce lo fácil que es de usar. En la línea de comando de Matlab se digita `interfaz2.m` y la caja de diálogo mostrada ahí es abierta, busca los archivo `*.asm` en una carpeta previamente asignada y con solo presionar los botones respectivos se completan los paso 1 al 3 antes mencionados.

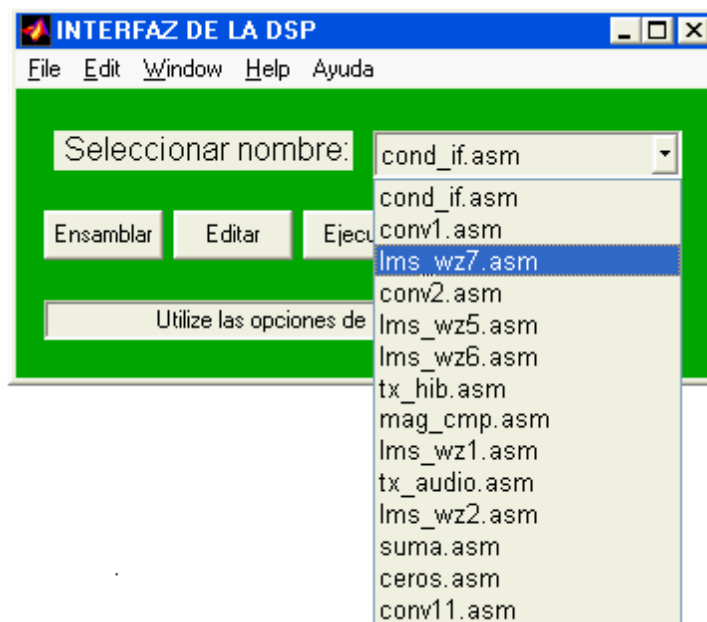


Figura 4.8 Utilidad para compilar y enlazar archivos *.asm

¹ El programa está escrito para Matlab 5.0. El mismo en su forma más básica fue diseñado en el curso de Tratamiento Digital de la Señal en el año 2002 y posteriormente, en esta Tesis, fue ampliado.

Cuando ya se tienen los archivos generados con las instrucciones anteriores, se procede a utilizar el programa de la interfaz gráfica EVM56811 (dicha interfaz se describe de manera amplia en los anexos).

Una vez obtenido el archivo con la extensión **.cld* se carga el archivo que se desea ejecutar en la tarjeta DSP56L811EVM, esto se hace en la pantalla mostrada en la Figura 4.9, a la cual se accede desde Windows.

Al observar la pantalla de la interfaz gráfica se pueden apreciar una serie de menús, botones y ventanas, algunos de los cuales se describen a continuación con el propósito de conocer el manejo de la interfaz.

Lo primero que se hace es acceder al menú **Load**, y se escoge el archivo de interés, buscando la ruta en donde éste se encuentra almacenado, luego se oprime **Aceptar**. Con esto se carga el archivo en la memoria de la tarjeta DSP56L811EVM.

Ahora el archivo está listo para ser ejecutado, lo cual se hace con solo oprimir el botón de **Start**. Si se desea detener la ejecución se presiona el botón **Stop**.

La ventana **DATA** se usa para desplegar la memoria de la DSP, tanto la memoria interna como la memoria externa. Además, pueden abrirse simultáneamente hasta 10 ventanas, en donde se pueden seleccionar los segmentos de datos que se desean ver. La memoria de datos X: puede desplegarse y editarse en esta ventana.

En esta ventana se tiene al lado izquierdo una columna de direcciones, con valores en base Hexadecimal. En el borde superior de la ventana aparece el nombre de la ventana, la base numérica y la etiqueta con que se distingue el grupo de datos (si se ha designado con un nombre).

El contenido de la ventana DATA son los valores que ocupan cada localización de memoria. Aquí se puede mostrar y editar la memoria en base Hex, Frac, Bin y ASCII. También, pueden desplegarse gráficamente en Hex, Dec y Frac.

La ventana **REGISTERS**, es usada para desplegar y editar los registros internos de la DSP. Los registros X0, Y, Y0, Y1, A y B pueden ser mostrados y editados en Frac, Dec y Hex. Los otros registros que aparecen en esta ventana sólo pueden verse y editarse en Hex.

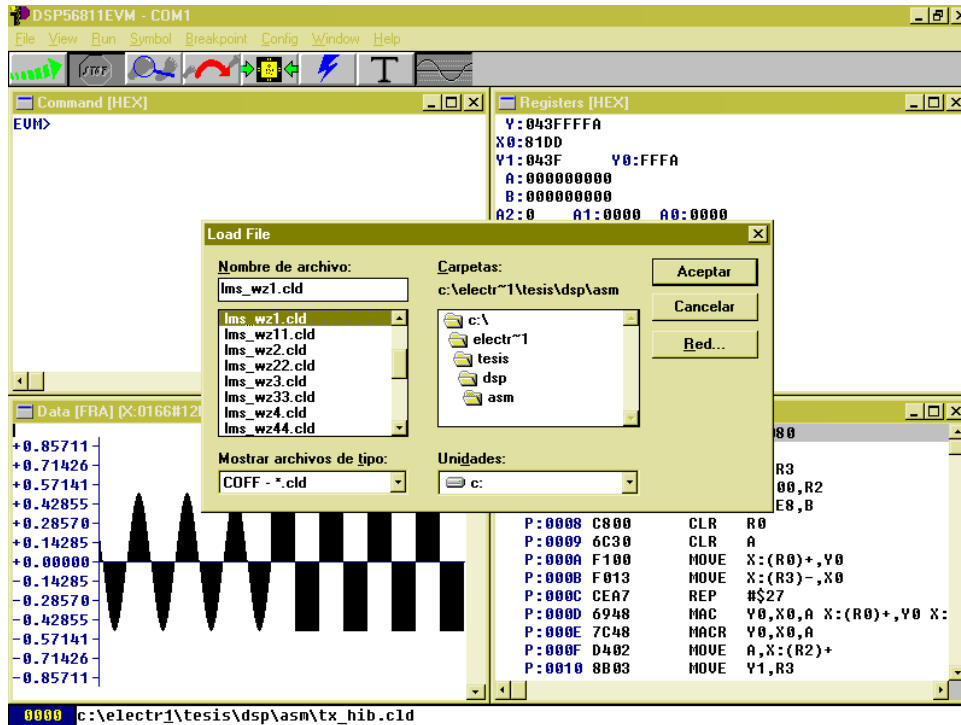


Figura 4.9 Interfaz gráfica EVM56811 y menú LOAD

4.5 PROGRAMA LMS EN LENGUAJE ENSAMBLADOR

La ecuación para el algoritmo LMS mostrada en los capítulos anteriores, debe ser implementada, utilizando el lenguaje ensamblador de la tarjeta. El código del Recuadro 4.1 no es más que el algoritmo LMS visto matricialmente:

$$\begin{bmatrix} W_{n+1} \end{bmatrix} = \begin{bmatrix} W_n \end{bmatrix} + \mu e(n) \cdot \begin{bmatrix} X \end{bmatrix}$$

donde se ha reservado una posición de memoria (en la memoria X para datos) para los coeficientes, señalados por un registro apuntador y otra posición para los datos de entrada.

DIAGRAMA DEL ALGORITMO LMS, IMPLEMENTADO EN LA DSP56L811

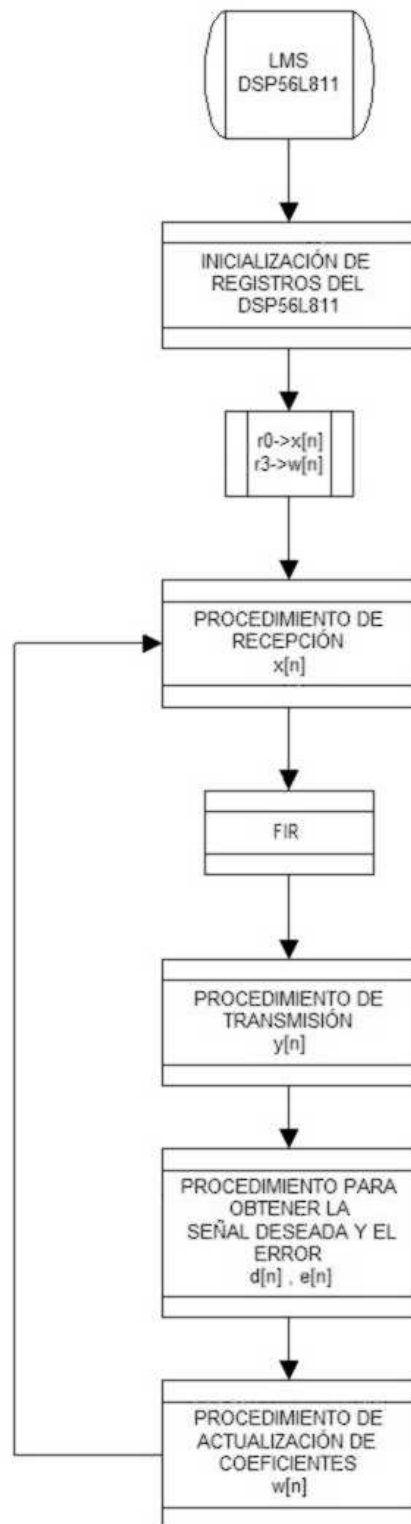


Figura 4.10 Diagrama de flujo del algoritmo LMS ejecutado en tiempo real con la DSP

escribir el nombre que se le dio al dato precedido por el numeral (es como definir constantes). Luego con **org** se define el lugar donde se desea que sean escritos los datos, dentro de la memoria de datos, los que en un principio son cero. En la Figura 4.11 se muestran los resultados obtenidos al ejecutar el *lms_wz11.asm* con señales semejantes a las del ejemplo *cancelwz_22.m* del capítulo anterior [Motorola, 1996].

Procedimiento de transmisión

El proceso de transmisión consiste en enviar una señal muestreada a una frecuencia que cumple el criterio de Nyquist, a través de la interfaz SSI y del *codec*, hacia el *jack* de salida *estéreo*. El código fuente completo se muestra en los anexos, aquí se presenta fragmentos junto a una breve explicación de dicho programa:

Recuadro 4.3 Definición de registros generales y de Transmisión

```

;DEFINICION DE REGISTROS
;-----

;Registros del Bus, PLL, Puerto C, Prioridad de Interrupción y SSI

bcr      equ      $fff9      ; registros control de bus interno
pcr1     equ      $fff3      ; registro de control 1
pcr0     equ      $fff2      ; registro de control 2
pcc      equ      $ffed      ; registro de control del puerto c
ipr      equ      $fffb      ; registro de prioridad de interrupción
scrrx    equ      $ffd4      ; registro de control de transmisión
scrtx    equ      $ffd3      ; registro de control de recepción
scr2     equ      $ffd2      ; registro de control de transmisión y recepción
ssrtsr   equ      $ffd1      ; registro de status de transmisión y recepción
stxrx    equ      $ffd0      ; registro tx/rx
output   equ      $ffd0
input    equ      $ffd0
    
```

El código de arriba muestra todos los registros que se deben inicializar. Estas direcciones se encuentran en el mapa de memoria, las cuales se definen en el CD titulado *DSP Technical Documentation* en la librería DSP56800. Cada uno de esos registros se debe direccionar, ya que en el programa se configuran y utilizan para habilitar ciertos elementos dentro de la tarjeta. Pues para poder utilizar una frecuencia adecuada de transmisión se debe habilitar el PLL y configurar los registros PCR1 y PCR0.

Procedimiento de recepción

El proceso de recepción es similar al proceso de transmisión, solo que los vectores y servicios de interrupción son diferentes, también se debe tomar en cuenta que el *jack* de entrada esta conectado al *codec* y esta señal de entrada debe estar en el rango de [1.2 V a ($V_{DD} - 1.2$ V)] y dicha señal será muestreada al configurar el *Clock* como se definió en el proceso de transmisión. En nuestro caso la frecuencia de recepción es la misma de transmisión.

Recuadro 4.4 Recepción

```

;recepción de datos
;lectura

rssi
    brclr    #0080,x:ssrtsr,rssi    ; espera por RDF
    lea     (r0)-                    ; decrementa r0
    movep   x:input,y0              ; almacena dato en y0
    bfclr   #2000,x:scr2            ; deshabilita receptor
    move    y0,x:(r0)+              ; almacena y0 en xn_entrada
    
```

Estos servicios de interrupción se encargan de que el valor leído se coloque en un espacio de memoria, y esto se repite por medio de un *loop* circular, el cual facilita el almacenamiento de datos.

Implementación del FIR

La convolución de dos señales almacenada en la memoria de la tarjeta es posible gracias a los accesos paralelos a memoria.

Recuadro 4.5 Filtro FIR

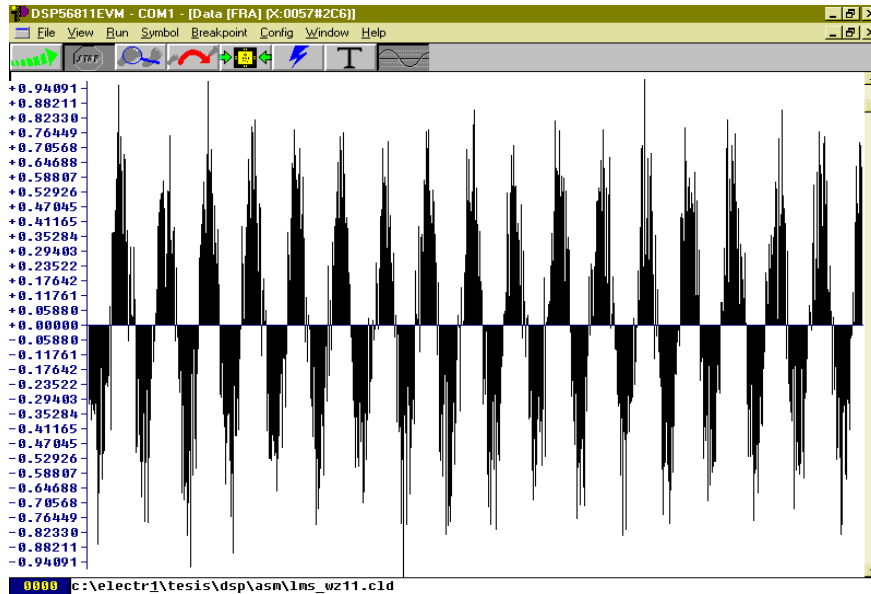
```

    move    x:(r3)+,x0              ; x0 = W_coef(n)
    move    x:(r0)+,y0              ; y0 = xn_entrada(n-1)

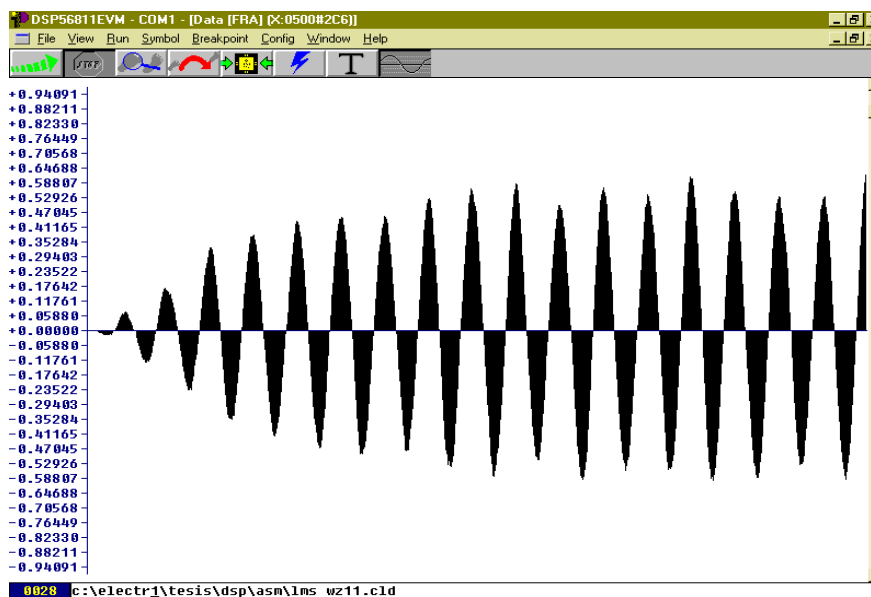
;----- filtro fir -----
FIR    rep     #M_M-1
        mac     y0,x0,a    x:(r0)+,y0    x:(r3)+,x0
        macr    y0,x0,a    ; y(n) = SUM(I=0,..,N-1) { W(I)*xn(n-I) }
    
```

Con la ayuda de la instrucción **move** se logra que el registro apuntador **r0** contenga el valor inicial de la señal y el registro apuntador **r3** contenga el valor inicial de los coeficientes; entonces, el registro **x0** contiene el primer valor de la señal y **y0** el primer valor de los coeficientes.

Por medio de las instrucciones **rep**, **mac** y **macr** se logra efectuar la operación que realiza un filtro FIR.



(a)



(b)

Figura 4.11 Resultados del algoritmo en la DSP56L81EVM (a) Señal contaminada y (b) Señal filtrada

Resultados obtenidos con señales en tiempo real

Aunque no fue posible generar una señal de ruido blanco gaussiano con un circuito electrónico y luego agregarla a una señal de interés, se pudo probar el programa en tiempo real al grabar esta señal como archivo de sonido, con la instrucción *wavwrite.m* que se incluye en las librerías de Matlab. El paso siguiente es introducirla en el jack de entrada de la DSP56L811EVM para poder ver la salida en el osciloscopio. En la Figura 4.12 se puede ver la forma en la que se está utilizando la tarjeta. Después, en la Figura 4.13 se ven los resultados al ejecutar el programa *lms_wz7.asm* que opera en tiempo real.

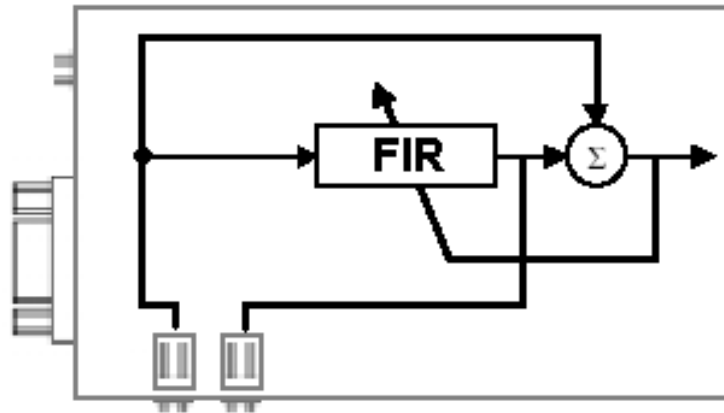


Figura 4.12 Diagrama del filtrado adaptativo con la tarjeta

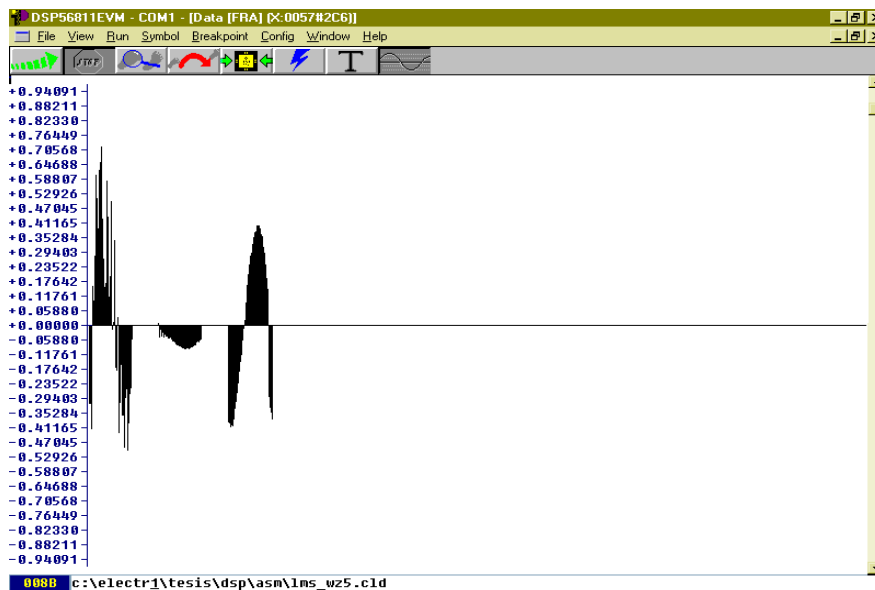


Figura 4.13 Resultados vistos en la interfaz gráfica para procesado en tiempo real

Esta última figura representa la respuesta del filtro en tiempo real. Lo desplegado en la interfaz gráfica es la señal de entrada, los coeficientes y la señal de salida. Para las señales de entrada y salida se han definido dos buffer de igual tamaño, esto se hizo así debido al poco espacio disponible para el usuario en la memoria de datos.

En la siguiente fotografía se observa como se conectó la DSP a la PC para controlarla a través de la interfaz gráfica. Usando la salida de audio de la misma PC se envió la señal contaminada con ruido a la entrada de la DSP, para luego obtenerla filtrada. Ambas señales podían verse en el osciloscopio.



Figura 4.14 PC, DSP56L811EVM y osciloscopio

Las señales vistas en el osciloscopio, tanto la contaminada como la filtrada, se muestra en las fotografías que están a continuación.



(a)



(b)

Figura 4.15 Resultados vistos en el osciloscopio (a) Señal contaminada y (b) Señal filtrada

Todo filtro adaptativo debe saber adaptarse a cualquier señal recibida y a cualquier cambio, ya sea en fase o en magnitud, actualizar los pesos o coeficientes del filtro y continuar filtrando la señal. Si la señal de entrada sufriera cambios profundos en sus características estadísticas, esta es conocida como una señal no estacionaria. Sin embargo, el programa con el cual se obtuvieron los resultados que aparecen en la figura anterior, el *lms_wz7.asm*, aunque es funcional, solo realiza un excelente filtrado adaptativo para señales estacionarias. Esta deficiencia no es por las características del LMS, sino fue una limitante al momento de implementarlo en la DSP.

4.6 CONCLUSIONES

- La utilización de un DSP para ejecutar algoritmos como los requeridos para un filtro adaptativo, es posible gracias a dos características del DSP56L811: el uso de **buffers circulares**, que son necesarios en este tipo de aplicaciones iterativas en tiempo real; y la **multiplicación-acumulación (MAC)**. Podemos agregar otros rasgos novedosos tales como el **redondeo** y los **movimientos paralelos** en una sola instrucción, que ayuda a ahorrar líneas de código, haciendo a los filtros más rápidos.
- El filtrado adaptativo realizado con el programa *lms_wz7.asm* es funcional para señales estacionarias solamente. Esto se debe a que en el momento de implementar el LMS en la DSP, se tuvo problemas con la convergencia de los coeficientes (pues estos divergen). La solución por la que se optó fue: después de un número de iteraciones, se detuvo la actualización de los pesos y el filtro se convierte en uno de coeficientes fijos, funcional solo para la señal a la que se adaptó desde un principio. Para que el filtrado sea aceptable para otra señal, deben de volverse a calcular los coeficientes. Es decir no se adapta a los cambios drásticos, estadísticamente.
- Se podrían reducir los errores de truncamiento, si se ajusta la DSP en punto flotante. También, se podrían revisar los rangos dinámicos de las señales de trabajo, para evitar la saturación del *codec*.

4.7 REFERENCIAS

BLANCO G. E. y CALDERON E. O. (2001), “Diseño de Filtros FIR utilizando la tarjeta DSP56L811EVM,” Proyecto de Ingeniería UES

MOTOROLA INC. (1996), “DSP56L811 16-bit Digital Signal Processor User’s Manual”

MOTOROLA INC. (1997), “DSP56L811EVM User’s Manual”

DOMAIN TECHNOLOGIES INC. (1996), “EVM56811 Debugger for the Motorola DSP56811EVM Card”

UNIVERSIDAD DE CHILE (2003), Facultad de Ciencias Físicas y Matemáticas,
Departamento de Ingeniería Eléctrica, “Diseño en Procesado Digital de la Señal”
<http://www.uchile.cl/investigacion2/index.html>
<http://gorrion.die.uchile.cl/~el652/catedras.html>

CD’S [cd:\dsp\development\tools](#) y [cd:\dsp\technical\documentation](#)

ANEXO

A

Características de Matlab

A.1 ORIGEN DE MATLAB

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. Al pasar de los años fue complementado y reimplementado en lenguaje C. Actualmente la licencia de Matlab es propiedad de MathWorks Inc.

Actualmente está disponible para un 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.

A.2 CARACTERISTICAS GENERALES

El nombre de Matlab proviene de la contracción de los términos **MAT**rix **LAB**oratory y fue inicialmente concebido para proporcionar fácil acceso a las librerías LINPACK y EISPACK, las cuales representan hoy en día dos de las librerías más importantes en computación y cálculo matricial.

Este es un programa interactivo para computación numérica y visualización de datos. Es ampliamente usado en el análisis y diseño de sistemas de control, 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. Permite resolver complicados problemas numéricos sin necesidad de escribir un programa.

Matlab es un entorno de computación y desarrollo de aplicaciones totalmente integrado orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos. Además, 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 manualmente, sin necesidad de hacer uso de la programación tradicional.

Matlab es un programa para realizar cálculos numéricos con vectores y matrices. Como caso particular puede también trabajar con números escalares, tanto reales como complejos. Una de las capacidades más atractivas es la de realizar una amplia variedad de gráficos en dos y tres dimensiones. Dado que Matlab tiene un lenguaje de programación propio, esto puede ser muy sencillo.

Matlab emplea matrices porque con ellas se puede describir infinidad de cosas de una forma altamente flexible y matemáticamente eficiente. Una matriz de pixeles puede ser una imagen o una película. Una matriz de fluctuaciones de una señal puede ser un sonido o 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 un avión a 40.000 pies de altura, o los coeficientes de un filtro digital adaptativo.

Desde sus primeras versiones, este programa es un sistema de trabajo interactivo cuyo elemento básico de trabajo son las matrices. El programa permite realizar de un modo rápido la solución numérica de problemas en un tiempo mucho menor que si se intentara resolver con lenguajes de programación tradicionales como pueden ser los lenguajes Fortran, Basic o C.

A.3 UTILIZACIÓN DE MATLAB

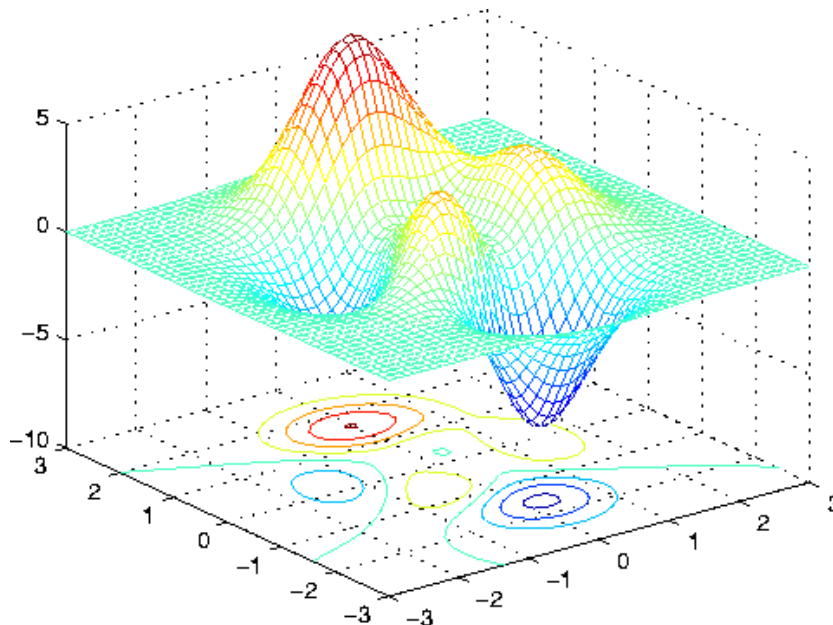
Matlab goza en la actualidad de un alto nivel de implementación en escuelas y centros universitarios, así como en departamentos de investigación y desarrollo de muchas compañías industriales nacionales e internacionales. En niveles universitarios, por ejemplo, Matlab se ha convertido en una herramienta básica, tanto para los profesionales e investigadores de centros docentes, como una importante herramienta para impartir cursos universitarios, tales como sistemas de control, álgebra lineal, procesado digital de imágenes, etc. En el mundo industrial, Matlab está siendo utilizado

como herramienta de investigación para la resolución de complejos problemas planteados en la realización y aplicación de modelos matemáticos en ingeniería.

Los usos más característicos de esta herramienta los encontramos en áreas de computación y cálculo numérico tradicional, prototipaje algorítmico, teoría de control automático, estadística, análisis de series temporales para procesamiento digital de señales. Todas estas aplicaciones son posibles gracias a que Matlab dispone en la actualidad de un amplio abanico de herramientas especializadas, denominadas Toolboxes, que extienden significativamente el número de funciones incorporadas en el programa principal. Estas Toolboxes cubren en la actualidad prácticamente todas las áreas principales en el mundo de la ingeniería y la simulación, destacando entre ellas el procesado de imágenes, control automático, estadística, análisis financiero, matemáticas simbólicas, redes neuronales, lógica difusa, identificación de sistemas, simulación de sistemas dinámicos, etc.

En este ejemplo se ve la manera sencilla de generar una gráfica en 3 dimensiones, con proyección en el plano X-Y.

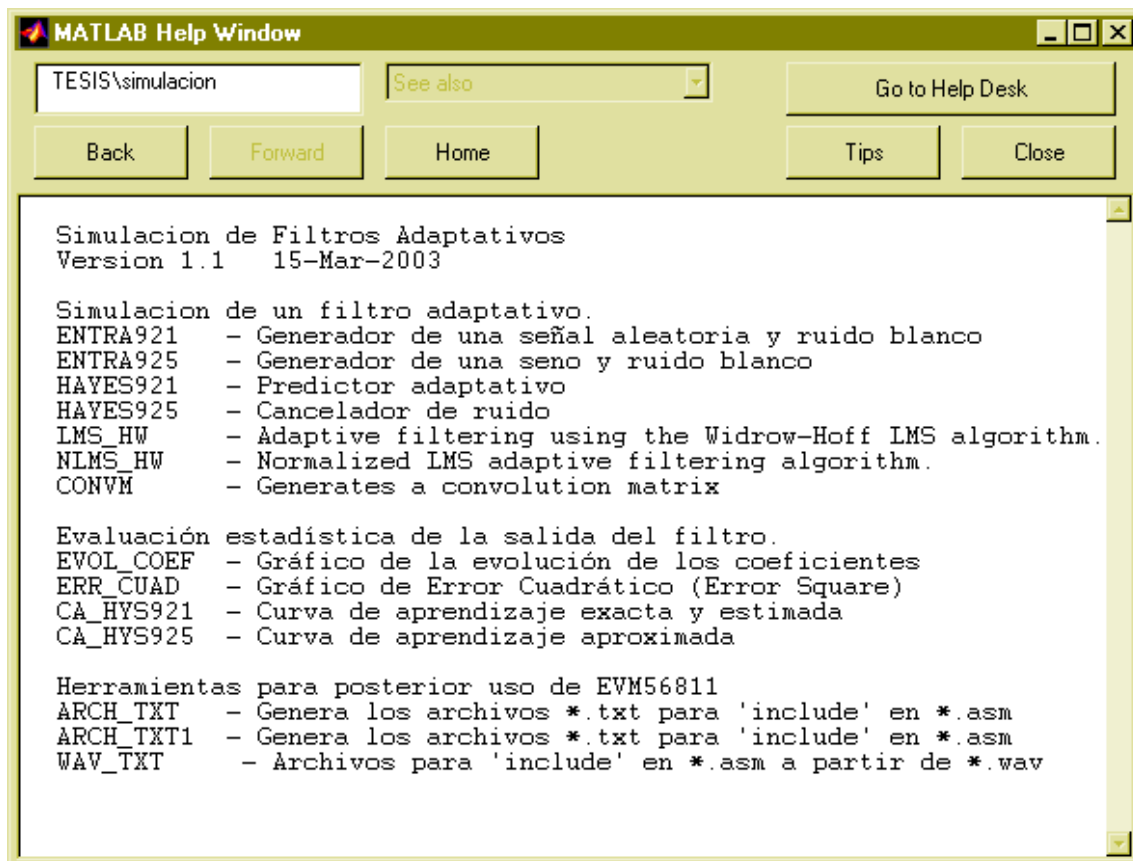
```
[X,Y] = meshgrid(-3:.125:3);
Z = peaks(X,Y);
meshc(X,Y,Z);
axis([-3 3 -3 3 -10 5])
```



A.4 HERRAMIENTAS EXTRAS

La empresa MathWorks ofrece Matlab como su principal producto para computación numérica, análisis y visualización de datos. También ofrece Simulink como un anexo a Matlab que interactúa con el lenguaje de Matlab y con lenguajes de bajo y medio nivel. Simulink es usado para simulación y modelado no lineal avanzado. Se ofrecen además numerosas herramientas especiales en “Toolboxes” para resolver problemas de aplicaciones específicas, por ejemplo control, procesamiento de señales, redes neuronales, etc. Estas herramientas son colecciones de rutinas escritas en Matlab.

Un ejemplo de una toolbox es la que se muestra a continuación. Esta contiene muchas funciones y programas escritos en lenguaje de Matlab para realizar filtrado digital adaptativo de señales.



ANEXO

B

Uso de la DSP56L811EVM

B.1 CONFIGURACION DEL MODULO DE EVALUACION DSP56L811EVM

El módulo de evaluación DSP56L811EVM ha sido diseñado para operar como un producto independiente o como un sistema avanzado de aplicaciones (ADS, por sus siglas en inglés). Cuando se utiliza como producto independiente, el módulo permite al usuario evaluar y probar las funciones básicas del DSP56L811. Cuando se utiliza como ADS, el módulo facilita la depuración y prueba de aplicaciones de software y hardware complejas. En nuestro caso hemos elegido la opción de producto independiente.

Equipo suministrado por el usuario

Para utilizar el módulo de evaluación como un producto independiente, el usuario deberá suministrar el siguiente equipo:

- Una fuente de potencia en el rango de 9-12 V dc, 500 mA, un conector de potencia hembra de 2.5mm (interior positivo).
- Un cable RS-232 con un conector DB9 macho y un DB9 hembra.
- Una computadora personal PC (clase 386 o mayor) corriendo con Windows 3.1 y DOS 6.0 (o mayor), o Windows9x, con un puerto serie RS-232 con capacidad de operar 9600-576000 bit/segundo, 4 Mbytes de RAM, una unidad de disco de 3.5", una unidad de CD-ROM, un disco duro con 4 Mbyte de espacio libre y un mouse.

Preparación de la tarjeta de evaluación DSP56L811EVM

Se deben localizar los doce jumpers JG1-JG12 en la tarjeta DSP56L811EVM, como se muestra en la Figura B.1. En la Tabla B.1 junto a la Figura B.2 describen la configuración sugerida para que la tarjeta opere en modo independiente.

B. Uso de la DSP56L811EVM

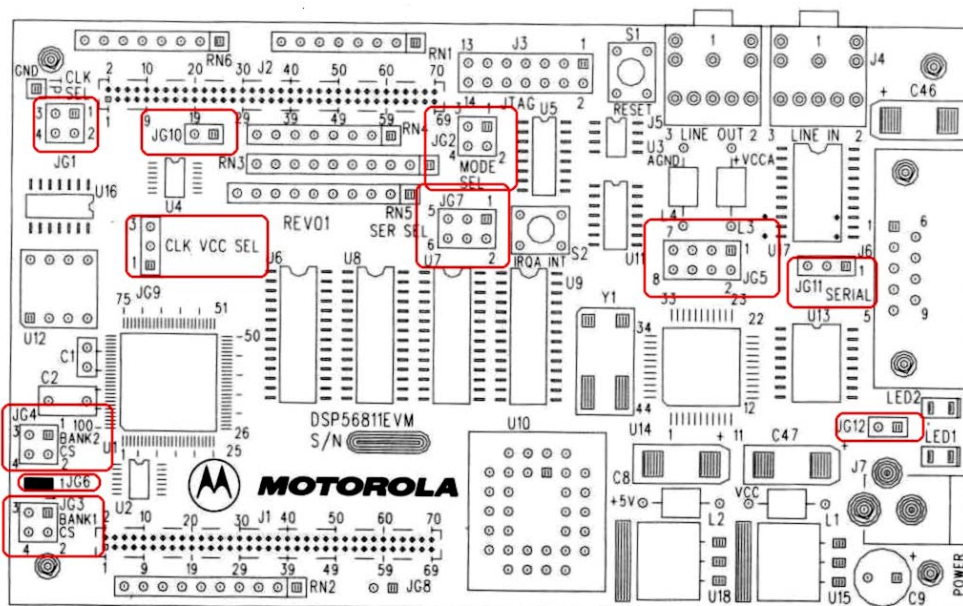


Figura B.1 Jumper en la tarjeta de evaluación DSP56L811EVM

Tabla B.1 Configuración de jumpers para operación independiente(stand-alone)

Jumper Group	Comentario	Stand-alone
JG1	Entrada de clock	1-2
JG2	Modo de operación 2	1-2
JG3	La memoria de programas con FLASH RAM	3-4
JG4	Memoria de datos	1-2
JG5	CODEC de la tarjeta	1-2, 3-4, 5-6, 7-8
JG6	Reservado	-
JG7	El convertidor RS-232 para Stand-alone	1-3,2-4
JG8	Reservado	-
JG9	Vcc = +5v para oscilador	2-3
JG10	Indica que PC15 no esta conectado a una resistencia pull-down	NC
JG11	La salida del CODEC a 1.7V pico a 300 ohm	1-2
JG12	PC14 selecciona función estándar	NC

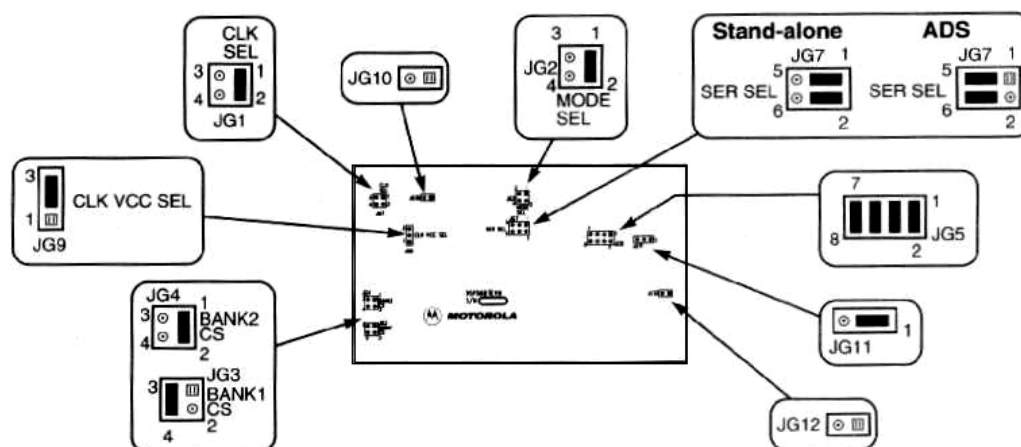


Figura B.2 Configuración de jumper sugerida

Instalación del módulo

La instalación del módulo depende de cómo desea usarse la tarjeta de evaluación, ya sea como, un módulo Stand-alone o como ADS. En nuestro caso se selecciona el módulo Stand-alone. La Figura B.3 muestra el diagrama de interconexión entre la PC y la tarjeta DSP56L811EVM. Se debe seguir el siguiente procedimiento para una correcta conexión:

- Conectar el terminal macho DB9 del cable de interfaz RS-232 al puerto serial de la PC
- Conectar la terminal hembra DB9 del cable RS-232 a J6 en la tarjeta DSP56L811EVM, como se muestra en la Figura B.3. Esto permitirá a la PC tener el control de la tarjeta
- Asegurarse que la fuente de potencia de 9-12V dc @ 500 mA no este energizada
- Conectar el plug de potencia de 2.5mm en el J7, como se muestra en la Figura B.3
- Energizar la fuente de potencia, entonces encenderá el LED de color verde cuando se aplique la potencia correcta

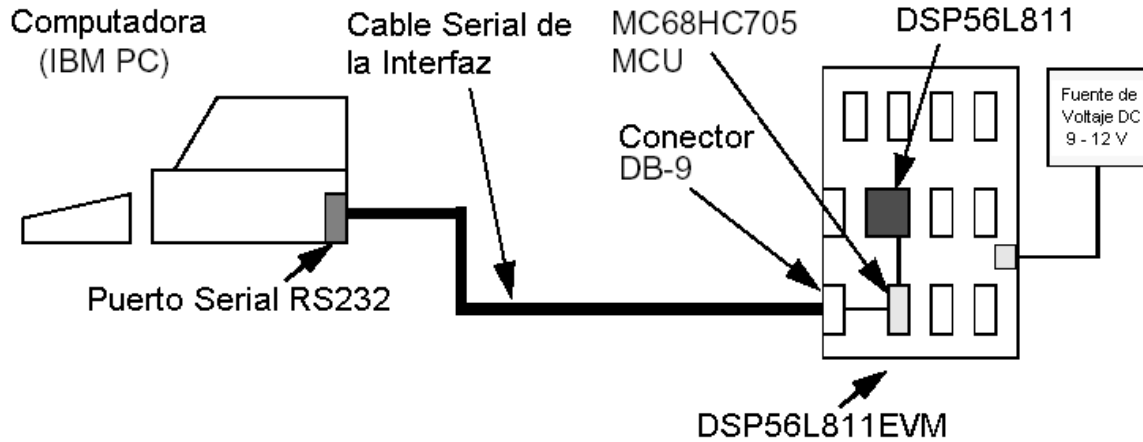


Figura B.3 Conexión de cables a la DSP56L811EVM

Instalación del software

La DSP56L811EVM usa un software diseñado por Domain Technologies Debugger para ser utilizado en operación Stand-alone en combinación con el Assembler y Linker (Compilador y Enlazador).

Estos programas están contenidos en el kit de la tarjeta en el CD titulado Motorola Development Tools, el cual contiene el Assembler y el Linker. También se proporciona un disquete de Domain Technologies el cual contiene el programa de interfaz gráfica para ejecutar los programas dentro de la memoria de la tarjeta.

El proceso de implementación comienza por la correcta instalación del programa EVM568XX, ya que de esto depende el éxito de la implementación, los requerimientos mínimos que se necesitan para la instalación de dicho programa son:

- a) Microprocesador 80486 en adelante.
- b) 5 MB de disco duro libres para instalar el software.
- c) DOS 3.1 o versión más avanzada.
- d) Microsoft Windows 9x o superior.

Luego se siguen las instrucciones que aparecen en pantalla, hasta finalizar dicha instalación.

B.2 USO DE LA INTERFAZ GRÁFICA PARA MANEJO DE DSP56L811EVM

Una vez se tengan los archivos *.cld, se procede a utilizar el programa de la interfaz gráfica, para lo cual se recomienda primero conocer la ayuda que incluye esta interfaz. Para familiarizarse con la interfaz, a continuación se muestran algunas de las pantallas de ayuda, explicando brevemente cada una de ellas.



Figura B.4 Ventana general de ayuda de EVM568XX

En la Figura B.4 se puede observar la primera pantalla de ayuda que proporciona la interfaz gráfica, se aprecia que posee cuatro hipervínculos que nos permiten conocer más de cada uno de esos temas. Si por ejemplo, se hace clic en el hipervínculo de USER INTERFACE, se despliega otra pantalla mostrada a continuación.

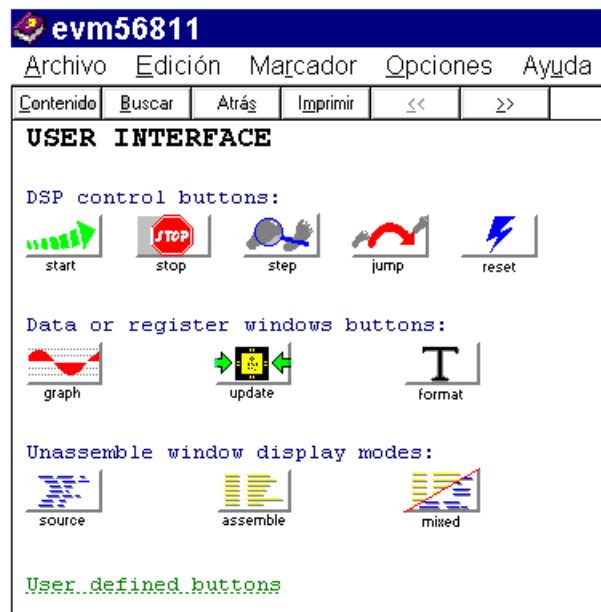


Figura B.5 Ventana de USER INTERFACE

En la Figura B.5 aparecen los botones que ofrece la interfaz para manejar y controlar la ejecución de un programa. Al igual que en el caso anterior, existen hipervínculos que permiten conocer las funciones de cada uno de los botones mostrados en esta figura. Si se hace clic sobre el icono de la flecha, llamado *start*, se presenta una descripción de lo que hace dicho botón a la hora de la ejecución de un programa, esta descripción se muestra en la Figura B.6

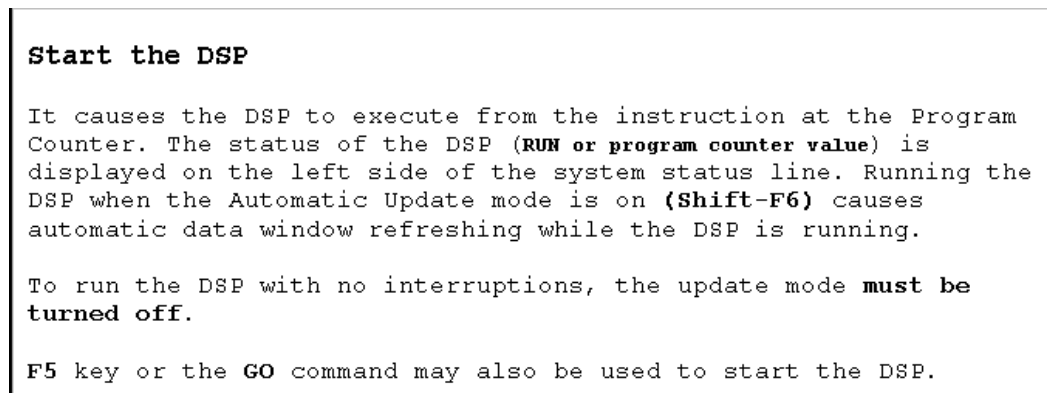


Figura B.6 Descripción de botón START

Así como aparecen las características para el botón de *start*, de la misma manera pueden conocerse las de los otros.

Si regresamos a la Figura B.4, observamos que el segundo hipervínculo se relaciona con los comandos que se pueden ejecutar en la interfaz gráfica, si exploramos este hipervínculo aparece, la siguiente pantalla.

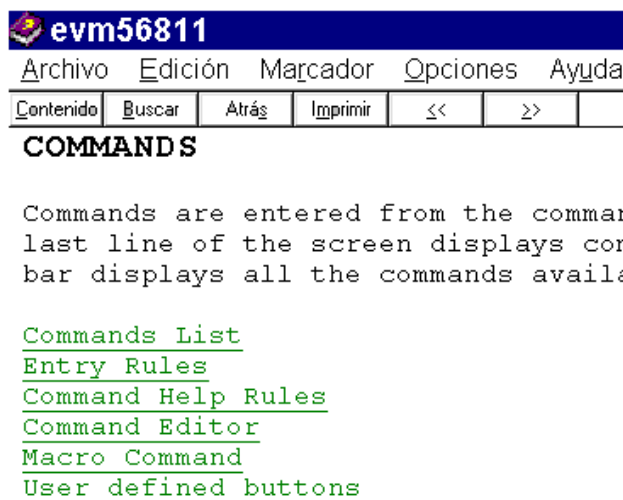


Figura B.7 Contenido de hipervínculo COMMANDS

Al igual que en las pantallas anteriores, se presenta la oportunidad de profundizar un poco más en el conocimiento de estos comandos, los cuales se explican si se hace un clic en COMMAND-LIST, apareciendo la lista mostrada en la Figura B.8

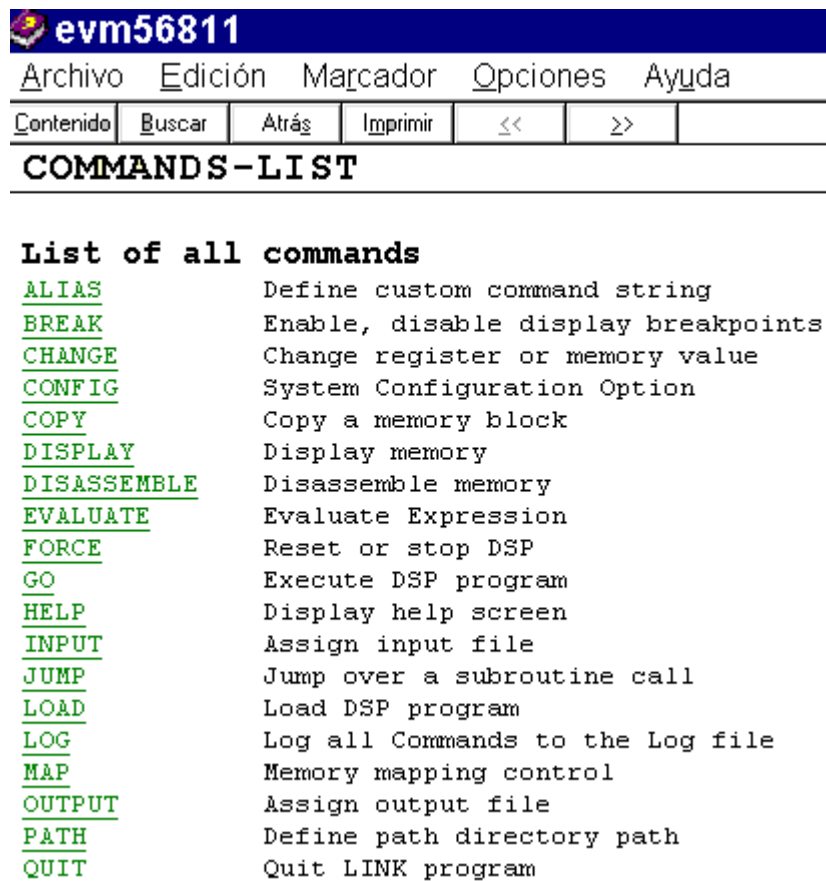
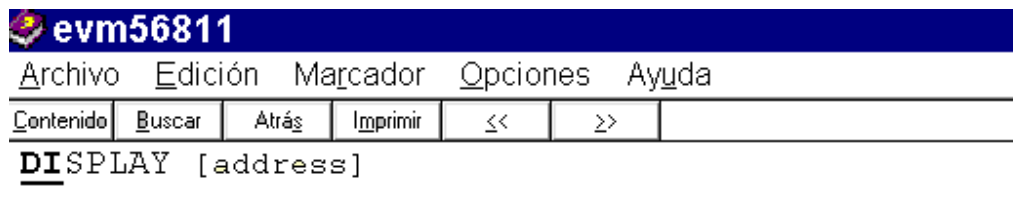


Figura B.8 Listado de comandos que se pueden ejecutar en la interfaz gráfica

En la Figura B.8 sólo se muestran unos de los posible comandos, pero existen otros, a los cuales se puede acceder en esta misma pantalla. Además se puede conocer más acerca de cada uno de estos comandos, solamente haciendo clic en el nombre del que nos interese. Por ejemplo si se desea conocer que hace el comando DISPLAY, se presenta la siguiente pantalla.



The **DISPLAY** command displays program or data memory

If more than one data window is open, the data is displayed in the window that "was last selected".

[**address**] is the starting address of the window. If [**address**] is scrolled down by one page.

Example 1

```
DISPLAY LABEL_1
Display from memory location LABEL_1
```

Example 2

```
DISPLAY P:100
Display from memory location P:100
```

Figura B.9 Descripción del comando DISPLAY

El tercer hipervínculo que aparece en la Figura B.4 se trata de FUNCTIONS KEYS, cuya pantalla se muestra en la Figura B.10.

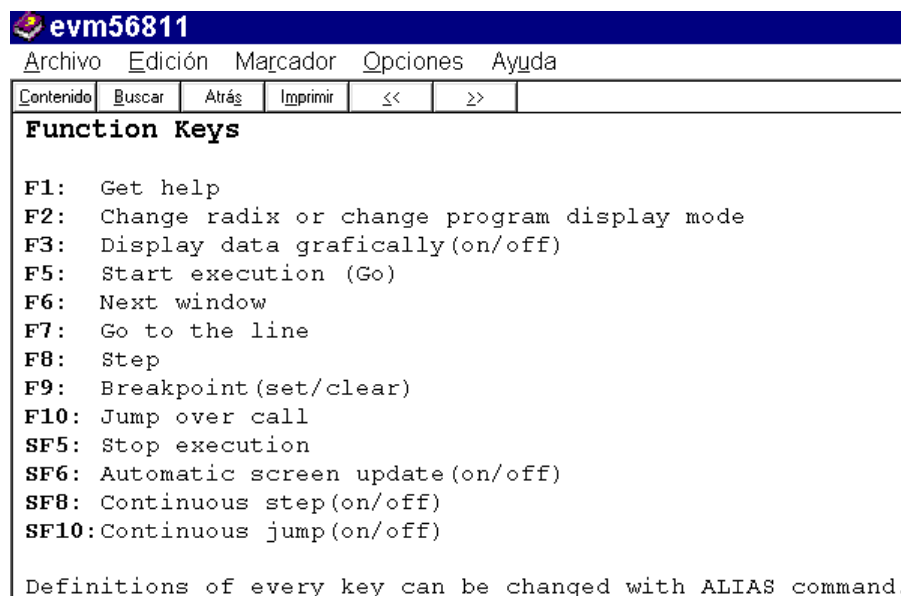
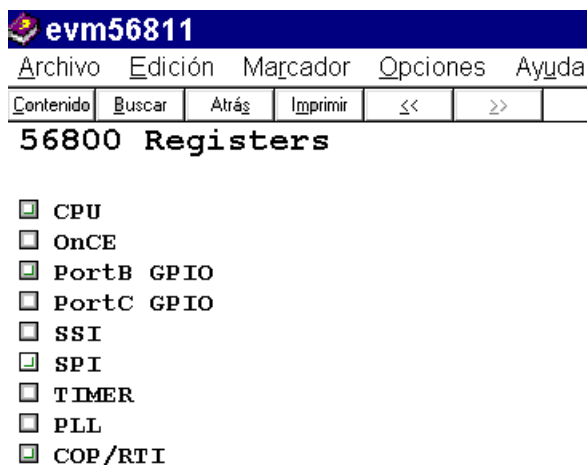


Figura B.10 Descripción de FUNCTION KEYS de la interfaz gráfica



[Alphabetical Registers List](#)

Figura B.11 Registros existentes en la DSP56800

El último hipervínculo mostrado en la Figura B.4 se refiere a los registros de la DSP56L811EVM, como se puede ver en a Figura B.11. Si se desea conocer más acerca de los registros tan sólo se hace clic al registro de interés con el mouse, con esto se despliega una pantalla conteniendo las principales características de esos registro. Por ejemplo en el caso de los registros de la CPU, la pantalla que describe los registros que la forman se muestran a continuación.

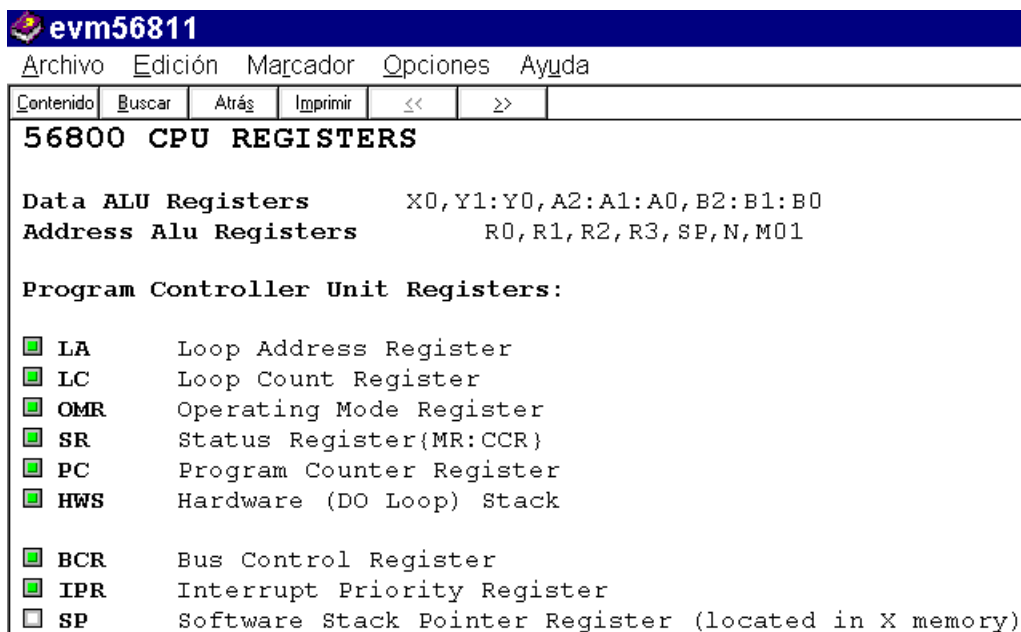
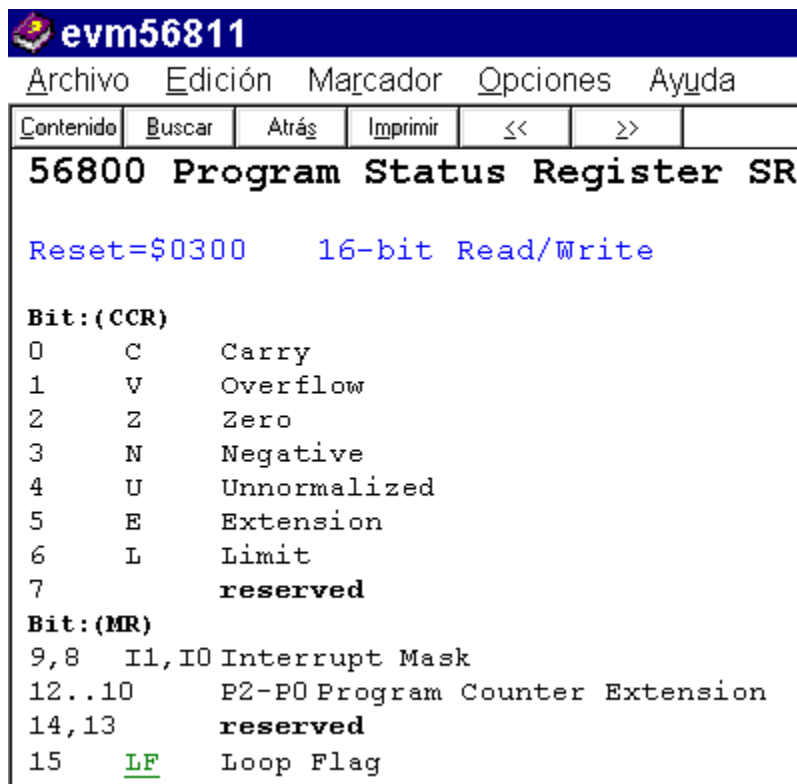


Figura B.12 Registros de CPU en la DSP56800

En la pantalla mostrada en la Figura B.12 se puede acceder a cada uno de lo registro que forman la CPU, por ejemplo si se desea conocer más acerca del registro SR, se presiona el hipervínculo y aparece la siguiente pantalla:



```

evm56811
Archivo Edición Marcador Opciones Ayuda
Contenido Buscar Atrás Imprimir << >>
56800 Program Status Register SR
Reset=$0300 16-bit Read/Write
Bit:(CCR)
0 C Carry
1 V Overflow
2 Z Zero
3 N Negative
4 U Unnormalized
5 E Extension
6 L Limit
7 reserved
Bit:(MR)
9,8 I1,I0 Interrupt Mask
12..10 P2-P0 Program Counter Extension
14,13 reserved
15 LF Loop Flag

```

Figura B.13 Registros de estado SR de CPU en la DSP56800

Con el conocimiento de esta interfaz es posible poder depurar los programas, rastrear los errores y conseguir ver el comportamiento de los registros. Cuando se hace un buen uso de estas herramientas pueden verse cuales son líneas de instrucción pueden sustituirse por una sola.

Una vez familiarizado con la interfaz gráfica se procede a cargar el archivo que se desea ejecutar en la tarjeta DSP56L811EVM.

B.3 PROGRAMACIÓN DE FRECUENCIA DE MUESTREO

Para que una señal analógica pueda ser representada en un sistema digital, tienen que tomarse muestras de esta y debe registrarse su valor de la forma más fiel posible. La frecuencia de muestreo indica el número de veces por segundo que se mide la señal analógica; y cuanto más grande sea la resolución en bit, mayor será el número de posibles valores que pueden utilizarse para representar esa señal.

La velocidad a la cual se realiza la actualización de las muestras se denomina frecuencia de muestreo (F_s). La teoría ha demostrado que para reconstruir las señales analógicas originales, sin pérdida apreciable de información, estas deben ser muestreadas por lo menos a una tasa de dos veces la frecuencia de la señal de interés (F_c) por N bit. Este resultado se conoce como Teorema del Muestreo de Nyquist. De acuerdo a este criterio:

$$F_s \geq 2 \cdot F_c \cdot N$$

Dado que se busca manejar archivos de audio y voz, cuya frecuencia esta alrededor de 4kHz. Se buscará entonces configurar los siguientes registros para fijar una frecuencia de muestreo de 8kHz.

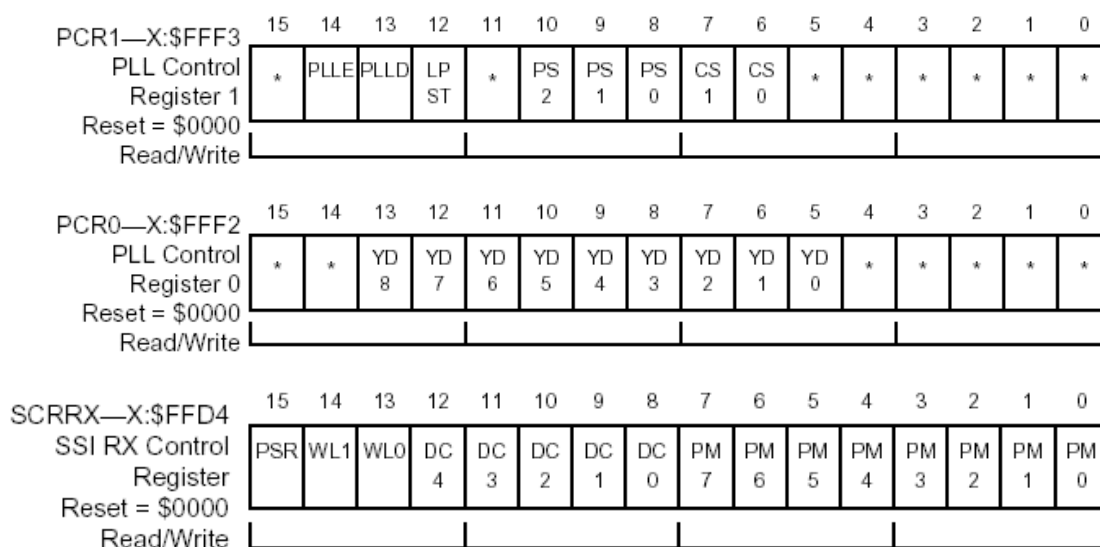


Figura B.14 Registros a ajustar para obtener una frecuencia de muestreo de 8kHz

Tabla B.2 SSI Data Word Lengths

WL1	WL0	Number of bits/word
0	0	8
0	1	10
1	0	12
1	1	16

Ecuacion para reloj de SSI

$$SCK = \text{Phi Clock Frequency} / (4 \times (7 \times \text{PSR} + 1) \times (\text{PM} + 1))$$

where PM = PM[7:0]

Donde:

Phi clock frequency: $20 \text{ MHz} * [\text{YD}] = 20\text{MHz} * 2 = 40 \text{ MHz}$.

PSR : Prescaler = 0.

PM : Prescaler modulus select = 19.

Por lo tanto $SCK = 40\text{MHz} / (4 * 1 * 20) = 500000$.

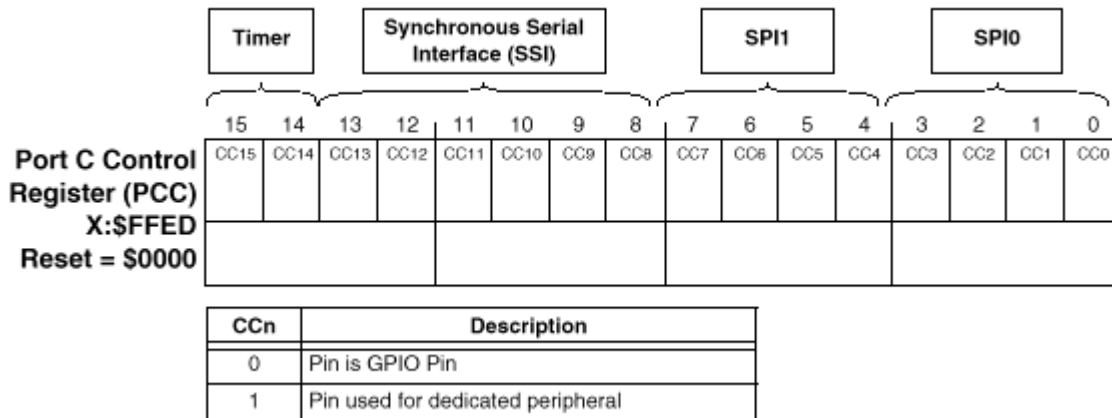
La frecuencia de muestreo viene dada por:

DC que aparece en un registros de los anteriores, se fija a un valor de 4.

$FS = SCK / (\text{DC} * \text{WL}) = 500000 / (4 * 16) = 7812.5\text{Hz} \approx 8000\text{Hz}$

Otro caso: DC de programa = 0, si introduce a la ecuacion; PM=71; PSR=0; WL=8 y un Phi clock frequency de 36.864MHz. Con lo anterior se genera SCK= 128 kHz y una FS de 16 KHz.

Otro registro utilizado para utilizar el puerto C es el que se muestra a continuación.



Generando un retardo

A = Valor almacenado en acumulador A = 50000 = \$C350.

B = Ciclos de instrucción = 8 ciclos.

A*B = 50000 lazos * 8 ciclos / lazo = 400000 ciclos.

400000 ciclos / 20000000 ciclos/ segundo = 0.02 seg. = 20 mseg.

Para mayor información sobre el uso de la tarjeta o el ajuste de estos registros ver las referencias siguientes:

- > MOTOROLA INC. (1996), “DSP56L811 16-bit Digital Signal Processor User’s Manual”
- > MOTOROLA INC. (1997), “DSP56L811EVM User’s Manual”
- > DOMAIN TECHNOLOGIES INC. (1996), “EVM56811 Debugger for the Motorola DSP56811EVM Card”
- > CD’S cd:\dsp\development\tools y cd:\dsp\technical\documentation

ANEXO

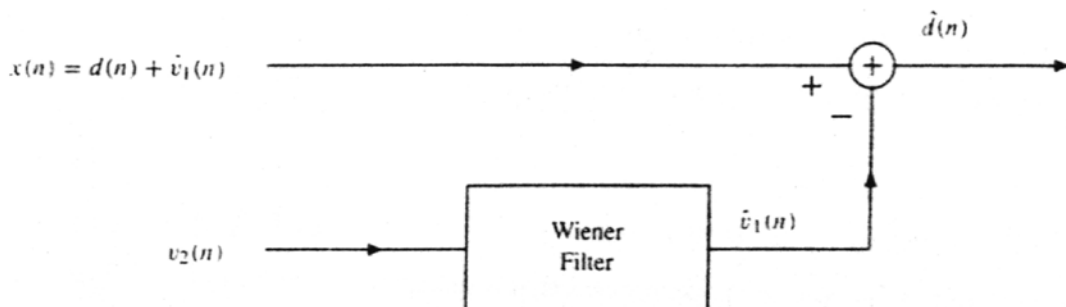
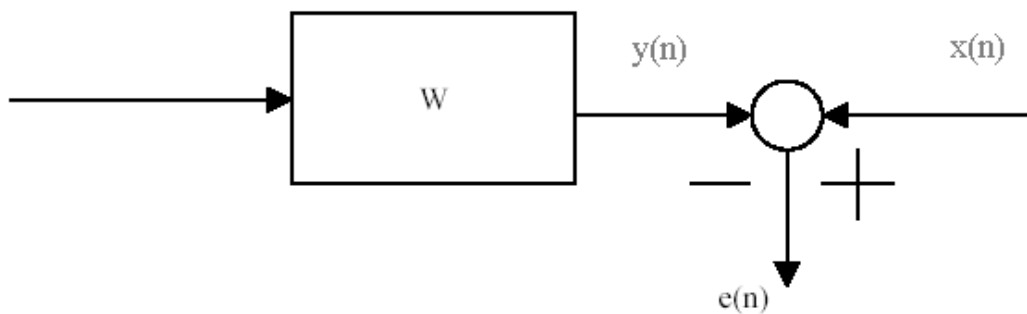
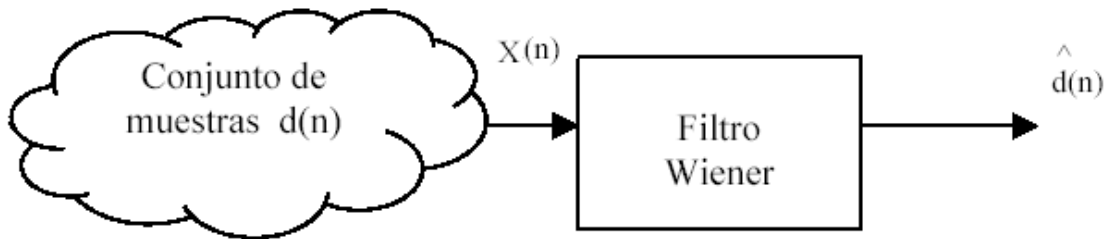
C

Filtros Wiener

C.1 INTRODUCCION A LOS FILTROS WIENER FIR

El filtro Wiener es uno de los filtros lineales óptimos más importantes. En su forma más general, consiste en una señal de entrada, $x(n)$, una respuesta deseada, $d(n)$, y un filtro lineal de respuesta al impulso $h(n)$. Este filtro es alimentado por $x(n)$ y produce a su salida $y(n)$. La diferencia entre la señal de salida del filtro, $y(n)$, y la señal deseada, $d(n)$, es el error de la estimación, $e(n)$.

C.2 BASE MATEMATICA DE LOS FILTROS WIENER

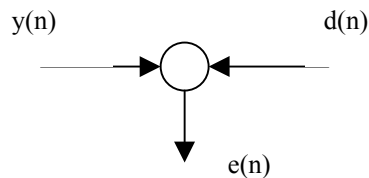


Caso real:

$$y(n) = \sum_{k=-\infty}^{\infty} w(k)u(n-k) \quad \text{para } n = 0,1,2,\dots$$

Caso complejo:

$$y(n) = \sum_{k=-\infty}^{\infty} w^*(k)u(n-k) \quad \text{para } n = 0,1,2,\dots$$



$$e(n) = d(n) - y(n)$$

Criterio de error cuadrático medio.

$$J = E \left[e(n)e^*(n) \right]$$

Buscamos minimizar J (derivando e igualando a cero), derivamos con respecto a los w_k 's pero como w_k es complejo introducimos el operador ∇_k :

$$\nabla_k = \frac{\partial}{\partial a_k} + j \frac{\partial}{\partial b_k}$$

$$\nabla_k J = \frac{\partial J}{\partial a_k} + j \frac{\partial J}{\partial b_k} = 0$$

$$\begin{aligned}\nabla_k J &= \nabla_k E[e(n)e^*(n)] = E[\nabla_k e(n)e^*(n)] \\ \nabla_k J &= E\left[e^*(n)\frac{\partial e(n)}{\partial a_k} + e(n)\frac{\partial e^*(n)}{\partial a_k} + je^*(n)\frac{\partial e(n)}{\partial b_k} + je(n)\frac{\partial e^*(n)}{\partial b_k}\right] \\ \frac{\partial e(n)}{\partial a_k} &= \frac{\partial}{\partial a_k} [d(n) - y(n)] = -\frac{\partial}{\partial a_k} y(n) \\ &= -\frac{\partial}{\partial a_k} \sum_{k=0}^{\infty} w_k^* u(n-k) = -\sum_{k=0}^{\infty} \frac{\partial}{\partial a_k} w_k^* u(n-k) \\ &= -\sum_{k=0}^{\infty} \frac{\partial}{\partial a_k} \{a_k - jb_k\} u(n-k) \\ &= -\sum_{k=0}^{\infty} \frac{\partial}{\partial a_k} a_k u(n-k) = -u(n-k) \quad k=0, \infty\end{aligned}$$

Procediendo de igual manera se obtuvieron las otras derivadas parciales:

$$\begin{aligned}\frac{\partial e^*(n)}{\partial a_k} &= -u^*(n-k) \quad k=0, \infty \\ \frac{\partial e(n)}{\partial b_k} &= ju(n-k) \quad k=0, \infty \\ \frac{\partial e^*(n)}{\partial b_k} &= -ju^*(n-k) \quad k=0, \infty\end{aligned}$$

$$\nabla_k J = E\left[e^*(n)\{-u(n-k)\} + e(n)\{-u^*(n-k)\} + je^*(n)\{ju(n-k)\} + je(n)\{-ju^*(n-k)\}\right]$$

$$\begin{aligned}\nabla_k J &= E\left[-u(n-k)e^*(n) - u^*(n-k)e(n) - u(n-k)e^*(n) + u^*(n-k)e(n)\right] \\ &= E\left[-2u(n-k)e^*(n)\right] \\ &= -2E\left[u(n-k)e^*(n)\right]\end{aligned}$$

$$\begin{aligned}\nabla_k J = 0 &= -2E[u(n-k)e_o^*(n)] \\ E[u(n-k)e_o^*(n)] &= 0 \quad k = 0,1,2,\dots \\ e_o(n) &= d(n) - \hat{d}(n|u_n) \\ e_o(n) &= d(n) - \sum_{k=0}^{\infty} w_{ok} u^*(n-k)\end{aligned}$$

C.3 DEDUCCION DE LA ECUACION DE WIENER-HOPF

El objetivo del filtrado de Wiener es determinar la respuesta al impulso $h(n)$ de forma que el error $e(n)$ sea, en un sentido estadístico, "lo más pequeño posible". El criterio elegido es la minimización del valor cuadrático medio del error

$$x = E\{|e(n)|^2\}$$

Esta función de costo posee una dependencia de segundo orden con los coeficientes del filtro, y tiene un mínimo que determina los coeficientes óptimos, es decir, el mejor diseño del filtro.

Consideremos un filtro causal de longitud p , orden $p-1$, de respuesta al impulso $w(n)$, y función de red:

$$W(z) = \sum_{n=0}^{p-1} w(n)z^{-n}$$

Si $x(n)$ es la entrada al filtro, la salida es la estimación de $d(n)$, obtenida como la convolución de $w(n)$ con $x(n)$

$$\hat{d}(n) = \sum_{l=0}^{p-1} w(l)x(n-l)$$

Debemos encontrar los coeficientes que minimizan la función de costo

$$J = E\{|e(n)|^2\} = E\{|d(n) - \hat{d}(n)|^2\}$$

Por tanto, derivamos ξ respecto a $w^*(k)$ e igualamos las derivadas a cero para los valores posibles de k ($k = 0, 1, \dots, p-1$).

$$\frac{\partial \xi}{\partial w^*(k)} = \frac{\partial}{\partial w^*(k)} E\{e(n)e^*(n)\} = E\left\{e(n) \frac{\partial e^*(n)}{\partial w^*(k)}\right\} = 0$$

Con

$$e(n) = d(n) - \sum_{l=0}^{p-1} w(l)x(n-l)$$

la derivada de $e^*(n)$ es

$$\frac{\partial e^*(n)}{\partial w^*(k)} = -x^*(n-k)$$

Y sustituyendo obtenemos el **principio de ortogonalidad o teorema de proyección**:

$$E\{e(n)x^*(n-k)\} = 0 \quad ; \quad k = 0, 1, \dots, p-1$$

Este teorema indica que ξ es mínimo y los coeficientes del filtro asumen sus valores óptimos cuando $e(n)$ es ortogonal a cada muestra de entrada $x(n)$ que es utilizada para el cálculo de la estimación. Como consecuencia, el error también es ortogonal a la salida del filtro. Este principio establece una **condición suficiente y necesaria** para la optimización.

Aplicando la expresión del error $e(n)$ tenemos

$$E\{d(n)x^*(n-k)\} - \sum_{l=0}^{p-1} w(l)E\{x(n-l)x^*(n-k)\} = 0$$

Con la suposición de que $x(n)$ y $d(n)$ son procesos estacionarios en sentido amplio (WSS),

$$E\{x(n-l)x^*(n-k)\} = r_x(k-l) \quad \text{y} \quad E\{d(n)x^*(n-k)\} = r_{dx}(k),$$

por tanto,

$$\sum_{l=0}^{p-1} w(l)r_x(k-l) = r_{dx}(k) \quad ; \quad k = 0, 1, \dots, p-1$$

Tenemos un conjunto de p ecuaciones lineales con p incógnitas $w(k)$, para $k = 0, 1, \dots, p-1$. En forma matricial, utilizando el hecho de que la secuencia de autocorrelación posee simetría conjugada, $r_x(k) = r_x^*(-k)$, podemos expresar las ecuaciones de Wiener-Hopf como

$$\begin{bmatrix} r_x(0) & r_x^*(1) & \dots & r_x^*(p-1) \\ r_x(1) & r_x(0) & \dots & r_x^*(p-2) \\ r_x(2) & r_x(1) & \dots & r_x^*(p-3) \\ \dots & \dots & \dots & \dots \\ r_x(p-1) & r_x(p-2) & \dots & r_x(0) \end{bmatrix} \begin{bmatrix} w(0) \\ w(1) \\ w(2) \\ \dots \\ w(p-1) \end{bmatrix} = \begin{bmatrix} r_{dx}(0) \\ r_{dx}(1) \\ r_{dx}(2) \\ \dots \\ r_{dx}(p-1) \end{bmatrix}$$

y en notación vectorial

$$\mathbf{R}_x \mathbf{w} = \mathbf{r}_{dx}$$

donde \mathbf{R}_x es la matriz de autocorrelación toeplitz hermítica $p \times p$, \mathbf{w} es el vector de coeficientes del filtro, y \mathbf{r}_{dx} es el vector de correlación cruzada entre la señal deseada $d(n)$ y la señal recibida $x(n)$.

El error cuadrático medio puede obtenerse de la siguiente manera:

$$\xi_{\min}^e = E\{|e(n)|^2\} = E\left\{e(n) \left[d(n) - \sum_{l=0}^{p-1} w(l)x(n-l) \right]^*\right\} = E\{e(n)d^*(n)\} - \sum_{l=0}^{p-1} w^*(l) E\{e(n)x^*(n-l)\}$$

Si los coeficientes \mathbf{w} son la solución de las ecuaciones de Wiener-Hopf, $E\{e(n)x^*(n-k)\} = 0$ y tenemos el error cuadrático medio mínimo

$$\xi_{\min}^e = E\{e(n)d^*(n)\} = E\left\{\left[d(n) - \sum_{l=0}^{p-1} w(l)x(n-l) \right] d^*(n)\right\}$$

Tomando el valor esperado,

$$\xi_{\min} = r_d(0) - \mathbf{r}_{dx}^H \mathbf{w}$$

Y en notación vectorial

$$\xi_{\min} = r_d(0) - \sum_{l=0}^{p-1} w(l) r_{dx}^*(l)$$

Como $\mathbf{w} = \mathbf{R}_x^{-1} \mathbf{r}_{dx}$, el mínimo error cuadrático medio puede expresarse en términos de la matriz de autocorrelación \mathbf{R}_x y el vector de correlación cruzada \mathbf{r}_{dx} :

$$\xi_{\min} = r_d(0) - \mathbf{r}_{dx}^H \mathbf{R}_x^{-1} \mathbf{r}_{dx}$$

De forma más general, este estudio puede realizarse considerando que los coeficientes del filtro pueden ser complejos, y así podemos expresar

$$\mathbf{R} \mathbf{w}_{opt} = \mathbf{p}$$

$$\xi_{\min} = \mathbf{r}_d(0) - \mathbf{p}^H \mathbf{w}_{opt} = \mathbf{r}_d(0) - \mathbf{w}_{opt}^H \mathbf{R} \mathbf{w}_{opt}$$

donde

$$\mathbf{x} = [x(n) \quad x(n-1) \quad \dots \quad x(n-p+1)]^T$$

$$\mathbf{R} = E[\mathbf{x}\mathbf{x}^H] = \begin{bmatrix} r(0) & r(1) & \dots & r(p-1) \\ r^*(1) & r(0) & \dots & r(p-2) \\ \dots & \dots & \dots & \dots \\ r^*(p-1) & r^*(p-2) & \dots & r(0) \end{bmatrix} = \begin{bmatrix} r(0) & r(1) & \dots & r(p-1) \\ r(-1) & r(0) & \dots & r(p-2) \\ \dots & \dots & \dots & \dots \\ r(-p+1) & r(-p+2) & \dots & r(0) \end{bmatrix}$$

$$\mathbf{p} = E[\mathbf{x}d^*] = [r_{xd}(0) \quad r_{xd}(-1) \quad \dots \quad r_{xd}(-p+1)]^T$$

En conclusión, se llega a la ecuación de Wiener-Hopf partiendo del simple hecho que en el proceso, se va a minimizar el error. Matemáticamente la forma de encontrar los mínimos de una función es derivándola y buscar los valores donde su resultado es cero. Entonces podemos ver desde esta óptica lo siguiente:

$$\begin{aligned}
 E\left[u(n-k)e_o^*(n)\right] &= 0 \\
 E\left[u(n-k)(d^*(n) - \sum_{k=0}^{\infty} w_{oi} u^*(n-i))\right] &= 0 \\
 E\left[u(n-k)d^*(n) - u(n-k) \sum_{k=0}^{\infty} w_{oi} u^*(n-i)\right] &= 0 \\
 E\left[u(n-k)d^*(n)\right] &= E\left[\sum_{k=0}^{\infty} w_{oi} u^*(n-i)u(n-k)\right] \\
 E\left[\sum_{k=0}^{\infty} w_{oi} u^*(n-i)u(n-k)\right] &= E\left[u(n-k)d^*(n)\right] \\
 \sum_{k=0}^{\infty} w_{oi} E\left[u^*(n-i)u(n-k)\right] &= E\left[u(n-k)d^*(n)\right] \\
 \sum_{k=0}^{\infty} w_{oi} r(i-k) &= p(-k) \\
 R w_o &= p
 \end{aligned}$$

ANEXO

D

Programas para simulación


```

for k=2:N
    y(k) = W(k-1,:) * X(k,:).';
end

error=E-d; %calculo del error

%algunas graficas
figure(1)
%subplot(2,2,1)
plot(d); grid
title('Señal a estimar')
xlabel('n'),ylabel('d(n)')
axis([0 1000 -3 3])

figure(2)
%subplot(2,2,2)
plot(xn); grid
title('Señal con ruido')
xlabel('n'),ylabel('x(n) = d(n) + v_1(n)')
axis([0 1000 -6 6])

figure(3)
%subplot(2,2,3)
plot(v2); grid
title('Señal de referencia')
xlabel('n'),ylabel('v_2(n)')
axis([0 1000 -6 6])

figure(4)
%subplot(2,2,4)
plot(E); grid
title('Salida del cancelador de ruido')
xlabel('n'),ylabel('est d(n)')
axis([0 1000 -3 3])

figure(5)
plot(d),hold on
plot(xn,'k')
plot(E,'r'),grid, zoom
title('Comparacion entre señal deseada y la estimacion del filtro')
xlabel('n'),ylabel('d(n) [Azul], est d(n) [Rojo], xn(n) [negro]')
%axis([0 1000 -3 3])

figure(6)
plot(error); grid
title('Error')
xlabel('Iteraciones'),ylabel('e(n) = d(n) - y(n)')

%otras graficas
figure(7),err_cuad(error);
%figure(8),evol_coef(W,b,2);
%figure(8),evol_coef2(W,b,2);
%figure(9),ca_hys925(M,b,50);

figure(8),evol_coef2(W,u,2);
figure(9),ca_hys925(M,u,50);

```

entra925.m

```

function [x,v2,d,v1]=entra925(W)

%ENTRA925  Generador de una seno y ruido blanco
%
% SINTAXIS
%   [x,v2,d,v1]=entra925
%
% ENTRADA
%   ninguna
%
% SALIDA
%   x = d + v1
%   v2 = ruido de referencia
%   d = señal deseada
%   v1 = ruido
%
% EJEMPLO
%   >>[x,v2,d,v1]=entra925;

%Creado          20.03.2003, 11:00 p.m.
%Modificado      04.04.2003, 12:10 a.m.
%
%Walter Zelaya / Tesis / Universidad de El Salvador
%                                     wlzelaya77@hotmail.com

%señal deseada
t = (0:1:1000-1);
d = sin(2*pi*.025*t+pi);
d = d(:).'; %forma un vector de 1*n sea el original n*1 o 1*n

%interferencia
sigma=sqrt(1); %sigma=desviacion standart
               %sigma^2=varianza

g =sigma*randn(size(t));
v1 = filter(1,[1,-0.8],g);
v1 = v1(:).'; %forma un vector de 1*n sea el original n*1 o 1*n
v2 = filter(1,[1,0.6],g);
v2 = v2(:).'; %forma un vector de 1*n sea el original n*1 o 1*n

%señal deseada + interferencia
x = d + v1;

```

lms_hw.m

```

function [A,E] = lms(x,d,mu,nord,a0)

%LMS Adaptive filtering using the Widrow-Hoff LMS algorithm.
%by M.H. Hayes

%---

%USAGE [A,E] = lms(x,d,mu,nord,a0)

%
%       x       : input data to the adaptive filter.
%       d       : desired output
%       mu      : adaptive filtering update (step-size) parameter
%       nord    : number of filter coefficients
%       a0     : (optional) initial guess for FIR filter
%               coefficients - a row vector.  If a0 is omitted
%               then a0=0 is assumed.
%
%       The output matrix A contains filter coefficients.
%       - The n'th row contains the filter coefficients at time n
%       - The m'th column contains the m'th filter coeff vs. time.
%       - The output vector E contains the error sequence versus time.
%
%       see also NLMS and RLS
%
%-----
% copyright 1996, by M.H. Hayes.  For use with the book
% "Statistical Digital Signal Processing and Modeling"
% (John Wiley & Sons, 1996).
%-----

```

D. Programas para simulación

```
X=convm(x,nord);

[M,N] = size(X); %N=nord, es decir numero de coeficientes

if nargin < 5, a0 = zeros(1,N); end

a0 = a0(:).'; %forma un vector de 1*n sea el original n*1 o 1*n

E(1) = d(1) - a0*X(1,:).';

A(1,:) = a0 + mu*E(1)*conj(X(1,:));

if M>1

    for k=2:M-nord+1;

        E(k) = d(k) - A(k-1,:)*X(k,:).';

        A(k,:) = A(k-1,:) + mu*E(k)*conj(X(k,:));

    end;

end;
```

nlms_hw.m

```

function [A,E] = nlms(x,d,beta,nord,a0)

%NLMS Normalized LMS adaptive filtering algorithm.
%by M.H. Hayes

%---

%USAGE   [A,E] = nlms(x,d,beta,nord,a0)

%
%       x       : input data to the adaptive filter   (v2)
%       d       : desired output   (xn)
%       beta    : adaptive filtering update (step-size) parameter
%       nord    : number of filter coefficients
%       a0     : (optional) initial guess for FIR filter
%               coefficients - a row vector.  If a0 is omitted
%               then a0=0 is assumed.
%
%       The output matrix A contains filter coefficients.
%       - The n'th row contains the filter coefficients at time n
%       - The m'th column contains the m'th filter coeff vs. time.
%       - The output vector E contains the error sequence versus time.
%
%       see also LMS and RLS
%
%-----
% copyright 1996, by M.H. Hayes.  For use with the book
% "Statistical Digital Signal Processing and Modeling"
% (John Wiley & Sons, 1996).
%-----

```

```

X=convm(x,nord);

[M,N] = size(X);

if nargin < 5,    a0 = zeros(1,N);    end

a0 = a0(:) .';

E(1) = d(1) - a0*X(1,:) .';

DEN=X(1,:)*X(1,:) + 0.0001;

A(1,:) = a0 + beta/DEN*E(1)*conj(X(1,:));

if M>1

    for k=2:M-nord+1

        E(k) = d(k) - A(k-1,:)*X(k,:) .';

        DEN=X(k,:)*X(k,:) + 0.0001;

        A(k,:) = A(k-1,:) + beta/DEN*E(k)*conj(X(k,:));

    end

end

```

ca_hys925.m

```

function ca2=ca_hys925(M,step,K)

%CA_HYS925 Curva de aprendizaje aproximada
%
% SINTAXIS
% ca_hys925(M,step,K,exact)
%
% ENTRADA
% M = orden del filtro (número de coeficientes)
% step = paso de la adaptacion
% K = veces sobre las que se promediará
%
% SALIDA
% gráfico
%
% EJEMPLO
% >>ca_hys925(12,0.25,50)
%
% Las señales son generadas con entra925.m
% Tiempo estimado de ejecucion para M= 12 y K=50 --> 5 min

```

```

%Referencias:
% ° "Adaptive Filte Theory"
%   Simon Haykin, Pretince Hall, 1996
% ° "Statistical Digital Signal Processing and Modeling"
%   Monson H. Hayes, John Wiley & Sons, 1996
%
%Creado          06.04.2003, 08:00 a.m.
%Modificado     07.06.2003, 09:00 a.m.
%
%Walter Zelaya / Tesis / Universidad de El Salvador
%              wlzelaya77@hotmail.com

%ecuacion de pag. 513, Hayes
apre=[];
for k=1:K
    [xn,v2,d] = entra925;           % para un cancelador de ruido, E es la
    [W,E] = nlms_hw(v2,xn,step,M); % estimacion de señal deseada d(n),
    apre(k,:)=E-d;                 % entonces, error = E - d
    barra(k,K,'Ejecutando ca_hys925 para encontrar error')
end

N=length(xn);
cal=(abs(apre)).^2;
for i=1:N
    ca2(i)=(sum(cal(:,i)))/K;
    barra(i,N,'Ejecutando ca_hys925 para graficar EASE')
end

%grafico
plot(ca2),grid,zoom
title('Curva de aprendizaje aproximada'),xlabel('Iteraciones')
ylabel(sprintf('Ensemble Averaged Squared Error [K = %g]',K))
%axis([0 500 1e-3 1e0])
set(gca,'xlim',[0 500])
%set(gca,'ylim',[1e-3 1e0])

```

ANEXO

E

*Programas en lenguaje
ensamblador*

in_out.asm

```

;IN_OUT Entrada - Salida
;

        page    137,80
        title   'IN_OUT Entrada - Salida'

;      programa que ingresa una señal a la DSP56L811EVM
;      y la envia al puerto de salida
;
;Referencias:
;      Proyecto de Ingeniería UES
;      Calderon, Octavio, 2001
;
;Creado          05.05.2003, 05:00 p.m.
;Modificado     18.05.2003, 09:30 a.m.
;
;Walter Zelaya / Tesis / Universidad de El Salvador
;                               wlzelaya77@hotmail.com

;definicion de registros

bcr      equ    $fff9
pcr1     equ    $fff3
pcr0     equ    $fff2
pcc      equ    $ffed
ipr      equ    $fffb
scrrx    equ    $ffd4
scrtx    equ    $ffd3
scr2     equ    $ffd2
ssrtsr   equ    $ffd1
stxrx    equ    $ffd0
output   equ    $ffd0
input    equ    $ffd0
start    equ    $0100

;inicio de direccion de programa

        org     p:$0000
        jmp     start

;inicio de tablas de vectores de interrupcion

        org     p:$0020
        jsr     leceisr
        jsr     lecisr
        jsr     esceisr
        jsr     escisr

;programa principal

        org     p:start
        move    #0,sp
        move    #0,x:bcr

```

E. Programas en lenguaje ensamblador

```
    bfclr    #$0200, sr
    movep   #$01c0, x:pcr1
    movep   #$0, x:pcr0
    clr     a
    move    #$61a8, a1
retar
    dec     a
    bne    retar

    bfset   #$4000, x:pcr1
    bfset   #$0f00, x:pcc
    movep   #$7301, x:scrtx
    movep   #$7301, x:scrrx
    movep   #$a1a0, x:scr2
    move    #0, r0
    move    #$aaaa, x:input
    bfset   #$0010, x:scr2
    movep   #$0200, x:ipr
    bra    *

;servicio de interrupcion de lectura sin excepcion

lecisr
    move    #$125, a

lazol
    dec     a
    bne    lazol

    movep   x:input, x0
    move    x0, x: (r0)
    movep   x:input, x0
    bfclr   #$a000, x:scr2
    bfset   #$5000, x:scr2
    rti

leceisr
    movep   x:input, y0
    move    x: (r0), x0
    movep   x0, x:output
    rti

esceisr
    movep   x:input, y0
    move    x: (r0), x0
    movep   x0, x:output
    rti

escisr
    movep   x:input, y0
    move    x: (r0), x0
    movep   x:ssrtsr, y1
    movep   x0, x:output
    bfclr   #$5000, x:scr2
    bfset   #$a000, x:scr2
    rti

end
```

lms_wz11.asm

```
;IMS Programa que realiza Least Mean Square (lms_wz1.asm)
;
```

```
page 137,80
title 'IMS Programa que realiza Least Mean Square'
```

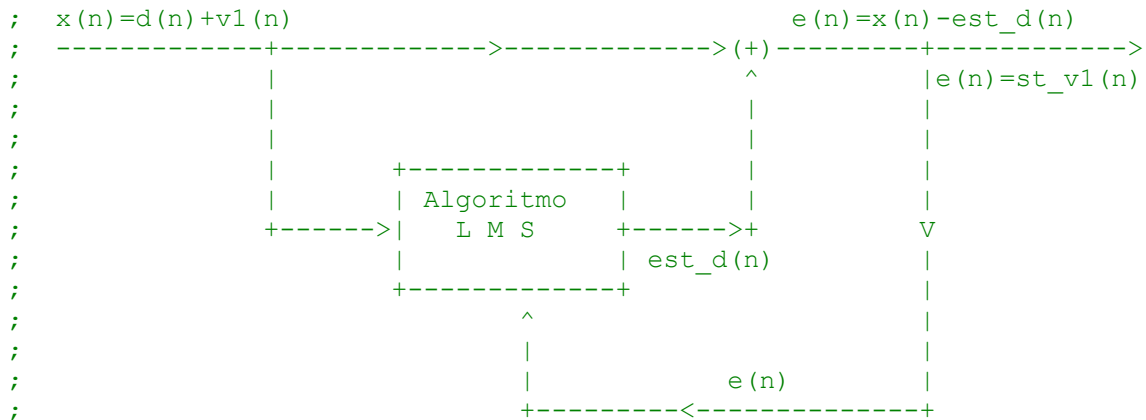
```
;*****
```

```
;Diferencia con los otros:
; No usa modulo para coeficientes ni señal
; 1000 muestras
;
```

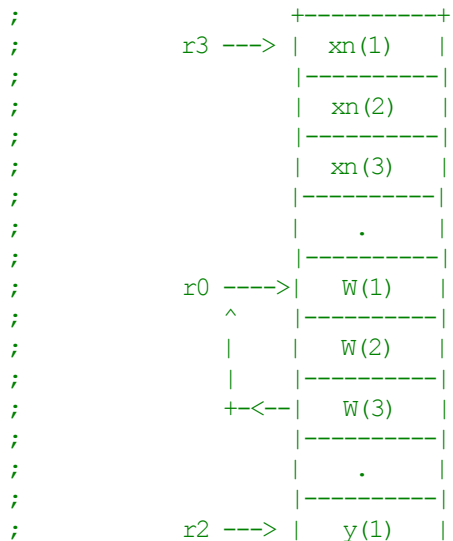
```
;Descripción:
; Toma la señal xn1000.txt y calcula los coeficientes,
; luego presenta la salida
;
```

```
;*****
```

CANCELADOR DE RUIDO USANDO EL ALGORITMO IMS



----- Mapa de memoria X -----



E. Programas en lenguaje ensamblador

```

;----- programa principal -----

        org     p:$0                ; vector de inicio de programa

        move    #xn_entrada,y1      ; apuntador a inicio de xn_entrada
        decw    y1                   ; desplazamiento de la señal de netrada
        move    y1,r3                ; apunta a primer valor de datos desplazado
        move    #y_salida,r2        ; apuntador para guardar y_salida

        move    #N_N,b              ; contador = numero de muestras
                                    ; para lazo exterior

lazo    move    #W_coef,r0          ; apunta a primer valor de W_coef
        clr     a
        move    x:(r0)+,y0          ; y0 = W_coef(n)
        move    x:(r3)-,x0          ; x0 = xn_entrada(n-1)

; salida del filtro
FIR     rep     #M_M-1
        mac     y0,x0,a             x:(r0)+,y0     x:(r3)-,x0
        macr    y0,x0,a
        move    a,x:(r2)+          ; guardar salida del fir [y(n)]
                                    ; luego, de aqui se enviara a salida

; obtener xn(n) y calcular e(n)
        move    y1,r3              ; apunta a primer valor de datos desplazado
        move    a,x0                ; x0 = [y(n)]
        move    x:(r3+1),a         ; a = [xn(n)]
        sub     x0,a                ; a = a - x0 [e(n) = xn(n) - y(n)]
        move    #miu_2,x0          ; x0 = 2*miu
        move    a,y0                ; y0 = e(n)

; actualizar coeficientes
        clr     a
        mpy    x0,y0,a              ; a = 2*miu*e(n)
        move    #W_coef,r0          ; apunta a primer valor de W_coef
        move    a,y0                ; y0 = 2*miu*e(n)
        move    x:(r3)-,x0          ; x0 = xn_entrada(n-1)
        move    x:(r0),a            ; a = W_coef(n)

        do     #M_M,actualizar
        mac    x0,y0,a              ; a = a + x0*y0
                                    ; [W_coef(new) = W_coef(old) +
                                    ;           2*miu*e(n)*xn_entrada(n-1)]

        move    x:(r3)-,x0
        move    a,x:(r0)+
        move    x:(r0),a

actualizar

        incw    y1                  ; apuntar al siguiente valor de xn_entrada
        move    y1,r3
        decw    b                   ; condicion para volver a lazo
        bgt    lazo                 ; SI salta si b no es cero

end

```


E. Programas en lenguaje ensamblador

```

;          |      | W(2) |
;          |-----|
;          +<---| W(3) |
;          |-----|
;          |      | . |
;          |-----|
;          r2 --->| y(1) |
;          |-----|
;          |      | y(2) |
;          |-----|
;          |      | . |
;          +-----+
;
;
;----- Seudocodigo -----
;
;-> Obtener muestras de entrada a FIR [xn(n)]
;-> Convolución de entrada con coeficientes
;-> Obtener señal deseada [xn(n)]
;-> Calcular error [e(n)]
;-> Actualizar coeficientes
;-> Enviar a la salida y(n) [est_d(n)]
;-> Desplazar vector x(n)
;
;
;Referencias:
;   - DSP Benchmarks - DSP56800 Family Manual
;     Motorola, 1996
;
;   - Trabajo intenso para conocer el algoritmo IMS
;
;   - Proyecto de Ingeniería UES
;     Calderon, Octavio, 2001
;
;
;Creado          15.10.2003, 09:00 a.m.
;Modificado      19.10.2003, 04:00 p.m.
;
;Walter Zelaya / Tesis / Universidad de El Salvador
;                wlzelaya77@hotmail.com
;
;
;DEFINICION DE REGISTROS
;-----
;Registros del Bus, PLL, Puerto C, Prioridad de Interrupción y SSI

bcr      equ      $fff9      ; registros control de bus interno
pcr1     equ      $fff3      ; registro de control 1
pcr0     equ      $fff2      ; registro de control 2
pcc      equ      $ffed      ; registro de control del puerto c
ipr      equ      $fffb      ; registro de prioridad de interrupción
scrrx    equ      $ffd4      ; registro de control de transmisión
scrtx    equ      $ffd3      ; registro de control de recepción
scr2     equ      $ffd2      ; registro de control de transmisión y recepción
ssrtsr   equ      $ffd1      ; registro de status de transmisión y recepción

```


E. Programas en lenguaje ensamblador

```

;PROGRAMA PRINCIPAL
;-----

; este programa corre en tiempo real un total de infinitas muestras

    org     p:INIC                ; vector de inicio de programa

    move    #$2000,sp             ; inicio de stack
    move    #0000,x:bcr           ; cero estado de espera en el bus x/p
    movep   #$0600,x:pcr1         ; configuracion de pll
    nop
    nop
    nop
    nop
    bfset   #$4000,x:pcr1         ; habilitado PLL
    bfclr   #$0200,sr             ; habilitando interrupciones
    bfset   #$0f00,x:pcc         ; configuracion de puerto C para SSI
    movep   #fsrx,x:scrtx        ; configuracion de clocks de transmisión
    movep   #$2180,x:scr2        ; configuracion de SSI, recepción

    move    #$8000+M_M,m01        ; modulo de tamaño M_M + 1
    move    #xn_entrada,r0        ; apuntador a inicio de xn_entrada
    move    #y_salida,r1          ; apuntador para guardar y_salida

    bfset   #$0010,x:scr2        ; habilita SSI
    movep   #$0200,x:ipr         ; habilita prioridad de interrupción

;opt     cc                      ;habilita contador de ciclos de intrucción

    move    #N_N,b

;----- lazo pricipal -----
lazo
    move    #W_coef,r3           ; apunta a primer valor de W_coef

;recepcion de datos
;lectura

rssi
    brclr   #$0080,x:ssrtsr,rssi ; espera por RDF
    lea     (r0)-                ; decrementa r0
    movep   x:input,y0           ; almacena dato en y0
    bfclr   #$2000,x:scr2        ; deshabilita receptor
    move    y0,x:(r0)+           ; almacena y0 en xn_entrada

    clr     a
    move    x:(r3)+,x0           ; x0 = W_coef(n)
    move    x:(r0)+,y0           ; y0 = xn_entrada(n-1)

;----- filtro fir -----
FIR    rep     #M_M-1
        mac     y0,x0,a x:(r0)+,y0 x:(r3)+,x0
        macr    y0,x0,a          ; y(n) = SUM(I=0,..,N-1){ W(I)*xn(n-I) }

```

E. Programas en lenguaje ensamblador

```

;transmision de datos
;escritura
    move    a,x:(r1)+          ; guardar salida del fir [y(n)]
    movep   a,x:output        ; envia dato a salida
    bfset   #$1000,x:scr2     ; habilita transmisi3n

wssi
    brclr   #$0040,x:ssrtsr,wssi ; espera por TDE
    bfclr   #$1000,x:scr2     ; deshabilita transmisor
    nop
    bfset   #$2000,x:scr2     ; habilita receptor de nuevo

;----- obtener xn(n) y calcular e(n) -----
    move    a,y0              ; y0 = [y(n)]
    move    x:(r0)+,a         ; a = [xn(n)]
    sub     y0,a              ; a = a - y0 [e(n) = xn(n) - y(n)]
    move    #miu_2,y0        ; y0 = 2*miu
    move    a,x0              ; x0 = e(n)

;----- actualizar coeficientes -----
    clr     a
    mpy    y0,x0,a           ; a = 2*miu*e(n)
    move    #W_coef,r3      ; apunta a primer valor de W_coef
    move    a,x0             ; x0 = 2*miu*e(n)

    do     #M_M,actualizar
    move    x:(r0)+,y0       ; y0 = xn_entrada(n-1)
    move    x:(r3),a         ; a = W_coef(n)
    mac    y0,x0,a          ; a = a + y0*x0
                                ; [W_coef(new) = W_coef(old) +
                                ;      2*miu*e(n)*xn_entrada(n-1)]
    move    a,x:(r3)+

actualizar

    decw   b                 ; condicion para volver a lazo
    bgt    lazo              ; SI salta si b no es cero
    ;jmp   lazo

;----- lazo principal 2 -----
lazo2
    move    #W_coef,r3      ; apunta a primer valor de W_coef

;repcion de datos
;lectura

rssi2
    brclr   #$0080,x:ssrtsr,rssi2 ; espera por RDF
    lea    (r0)-
    movep   x:input,y0      ; almacena dato en y0
    bfclr   #$2000,x:scr2   ; deshabilita receptor
    move    y0,x:(r0)+      ; almacena y0 en xn_entrada

    clr     a
    move    x:(r3)+,x0       ; x0 = W_coef(n)
    move    x:(r0)+,y0      ; y0 = xn_entrada(n-1)

```

E. Programas en lenguaje ensamblador

```
;----- filtro fir -----
FIR2  rep    #M_M-1
      mac    y0,x0,a x: (r0)+,y0 x: (r3)+,x0
      macr   y0,x0,a                ; y(n) = SUM(I=0,..,N-1){ W(I)*xn(n-I) }

;transmision de datos
;escritura
      move   a,x:(r1)+                ; guardar salida del fir [y(n)]
      movep  a,x:output                ; envia dato a salida
      bfset  #$1000,x:scr2            ; habilita transmisi3n

wssi2
      brclr  #$0040,x:ssrtsr,wssi2    ; espera por TDE
      bfclr  #$1000,x:scr2            ; deshabilita transmisor
      nop
      bfset  #$2000,x:scr2            ; habilita receptor de nuevo

      ;decw  b                        ; condicion para volver a lazo
      ;bgt   lazo                      ; SI salta si b no es cero
      jmp    lazo2

end
```