01170

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE ESTUDIOS DE POSGRADO

Tesis:

Algoritmos de Filtrado Adaptable: Implementación, Evaluación, Comparación y Aplicaciones en Telecomunicaciones

Presentada por:

Larry Hipólito Escobar Salguero

Para Obtener el Grado de:

MAESTRO EN INGENIERÍA
(ELÉCTRICA)

Dirigida por:

Dr. Rogelio Alcántara Silva

Ciudad Universitaria, noviembre de 1997

Dedicatoria

A la vida, porque me lo ha dado todo, [Violeta Parra].

A la alegría, porque por ella vivo, [Julius Fusick].

A mis familiares Elena, Ada, Paty, Manolo, Fredy y Rolando, porque siempre han sido una fuente de inspiración para lograr mis metas.

A mis entrañables amigos Ing. José Luís Guerrero y familia, Dr. René Crocker y familia, Dra. Adriana Sandoval e hijos.

A México un país generoso y amable que me ha brindado la oportunidad de prepararme profesionalmente.

A los que compartieron mis sueños y no pudieron ver la luz de esperanza que se vislumbra en el infinito.

A la Universidad Nacional Autónoma de México, porque desde el valle del Tezulutlán hasta Tenochtitlán el ave bicéfala me ha cobijado en su ser.

Agradecimientos

A mis profesores de la Facultad de Ingeniería, al departamento de Electrónica de la Facultad de Ingeniería UNAM y a todas las personas que hicieron posible la culminación de este trabajo.

A los profesores **Dr. Boris Escalante, Dr. Francisco García Ugalde, Dr. Bohumil Psenicka** y al **Dr. Victor García Garduño** por sus comentarios y la revisión final de este trabajo.

Un agradecimiento muy especial al **Dr. Rogelio Alcántara Silva** por su apoyo incondicional, su valioso aporte y tiempo dedicado a la asesoría de este trabajo y su amistad.

RESUMEN

Con las crecientes necesidades en nuestro mundo actual de transmitir grandes volúmenes de información a altas tasas en tiempo real, la tecnología y la investigación se han empeñado por desarrollar nuevos dispositivos y algoritmos para lograr la comunicación entre nuestra sociedad. Para alcanzar esto, se requieren transmisiones a tasas mayores que la permitida por una canal de comunicación telefónica, así como también realizar la cancelación de interferencias no deseadas, el filtrado y estimación de señales. Por la naturaleza cambiante en el tiempo de los medios de transmisión, en la actualidad el filtrado adaptable ha tenido una gran aplicación, ya que no requiere un conocimiento a priori del canal o de las señales a tratar. En el presente trabajo se realiza el análisis, la implementación, la evaluación y comparación de los algoritmos adaptables en general, sus formas de implementación, sus formas rápidas, la evaluación de sus dinámicas, su convergencia y comportamientos numéricos. La utilización de los filtros adaptables la enfocamos a los problemas de comunicaciones digitales como la anulación de eco, la eliminación de ruido y la igualación de canal. La solución de estos problemas lleva consigo el procesamiento de grandes volúmenes de información y lo ideal es efectuar el procesamiento en tiempo real, por lo que con los avances tecnológicos de los procesadores de señales digitales resulta atractivo implementar la solución de los problemas de comunicaciones en arquitecturas modulares y específicas. La implementación de los algoritmos de filtrado adaptable (AFA) se realizó tanto en aritmética de punto flotante como en simulaciones de aritmética de punto entero, y se presentan los resultados de la comparación de la robustez numérica y de los tiempos de cálculo de los algoritmos AFA en el procesador de señal digitales el TMS320C50 de Texas Instruments. A partir de los resultados obtenidos en nuestro trabajo, los algoritmos AFA evaluados pueden ser integrados en un sistema de comunicación en tiempo real.

INDICE

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

Capítulo I

INTRODUCCION

INTRODUCCION

En general un filtro es un dispositivo utilizado para extraer información de interés de una señal con rasgos conocidos a priori o a estimar. Por otro, lado un filtro adaptable es capaz de seguir o ajustar su desempeño a los cambios de una señal que varía sus características con el tiempo, esta posibilidad hace que estos filtros tengan muchas aplicaciones en campos como las comunicaciones, el control, la identificación de sistemas, el radar, la sismología, el procesamiento de imágenes y el reconocimiento de patrones.

Los filtros adaptables han sido ampliamente utilizados en aplicaciones donde los filtros digitales fijos no tienen un buen comportamiento, por ejemplo: en control, igualación de canal, cancelación de ruido, cancelación de eco, etc. En los últimos años los filtros adaptables se han convertido en un componente estándar en telefonía, comunicación de datos, detección de señales y seguimiento de sistemas, [ALE86].

El estudio de los algoritmos adaptables involucra extensos desarrollos matemáticos, el conocimiento de sistemas discretos, el comportamiento de las señales, amplias bases sobre el procesamiento de señales, simulaciones en lenguaje de alto nivel para aritmética de punto flotante y aritmética de punto fijo, conocimiento de procesadores de señales digitales (DSP), arquitecturas paralelas, diseño de circuitos integrados con tecnología VLSI. Todo este conjunto de elementos hacen posible la implementación de algoritmos adaptables y sus aplicaciones.

En este trabajo se realiza una evaluación y comparación de los algoritmos adaptables en general, sus formas de implementación, su optimización, sus formas rápidas, la evaluación de sus dinámicas, su convergencia y comportamientos numéricos, se hacen una serie de simulaciones y análisis de posibilidades de agregar mejoras a algún algoritmo. Todo esto con la finalidad de facilitar su utilización, mejorar su desempeño y seleccionar el más adecuado para una aplicación particular. Posteriormente, se considera la problemática de su implementación en arquitecturas con Procesadores de Señales Digitales (DSP).

Como área de aplicación principal, el uso de los filtros adaptables se enfoca a los problemas de comunicaciones telefónicas de larga distancia cuando se acoplan impedancias tales como: la anulación de eco, la eliminación de ruido y la igualación de canal. Sin embargo, ya que las áreas de aplicación son extensas, se describirán, a manera de síntesis otras áreas como: identificación de sistemas, filtrado de señales, eliminación de frecuencias de 60 hz, etc. Sin duda, la solución de estos problemas lleva consigo el procesamiento de gran cantidad de información y lo ideal es efectuar el procesamiento en tiempo real, por lo que con los avances tecnológicos de los procesadores de señales digitales resulta atractivo implementar la solución de los problemas de comunicaciones en arquitecturas modulares.

Además, en este trabajo se presenta la implementación de los algoritmos analizados en una arquitectura de aritmética de punto fijo, específicamente en el procesador de señales digitales TMS320C50 de Texas Instruments (TI). En este punto se hace una evaluación de los problemas de implementación, las arquitecturas de procesamiento digital de señales, los problemas de robustez numérica y se hace una comparación de los tiempos de cálculo para la realización de algunos algoritmos en el DSP utilizado.

1.1 ESTADO DEL ARTE

Los primeros estudios de estimación de los mínimos cuadrados en señales aleatorias fueron hechos por Kolmogorov, Krein y Wiener en la década de los años 30 y 40. Kolmogorov desarrolló una solución a los problemas de predicción lineal para señales aleatorias en el tiempo discreto. Krein observó la relación de los resultados de Kolmogorov y utilizando polinomios ortogonales extendió su trabajo al tiempo continuo por medio de una transformación bilineal, [HAY91].

Por otro lado, Wiener, independientemente, formuló el problema de predicción lineal en tiempo continuo obteniendo la fórmula para predicción óptima, conocida como ecuación de Wiener-Hopf. Wiener también consideró el problema de filtrado en procesos corrompidos por ruido.

En 1947, Levinson formuló el problema de filtrado de Wiener en tiempo discreto, para el caso de señales estacionarias, donde la matriz de correlación **R** es simétrica y del tipo Toeplitz. Levinson desarrolló un algoritmo recursivo que explota las propiedades de la matriz de Toeplitz para resolver la ecuación de Wiener-Hopf. En 1960 Durbin redescubre el procedimiento de Levinson como un esquema conveniente para un modelo autorregresivo (AR), [HAY91].

Durante los años 50's, surgieron algunas generalizaciones de la teoría de filtrado Wiener-Kolmogorov para la estimación de procesos estacionarios dados por un intervalo de observación finita y cubriendo así la estimación de procesos no estacionarios, sin embargo, los resultados no fueron del todo satisfactorios por su dificultad de actualización cuando se incrementaba el intervalo de observación.

Swerling fue uno de los primeros en atacar el problema presentando algunos algoritmos recursivos en 1958. Por otro lado, Kalman formuló originalmente el problema de filtrado lineal derivado de procesos discretos y desarrolló independientemente un algoritmo más restringido que el de Swerling, Kalman publicó sus resultados que lo hicieron famoso, a pesar de lo reclamos de Swerling. Este algoritmo se le atribuye a Kalman, [HAY91].

En 1959 Widrow y Hoff desarrollaron el famoso algoritmo del gradiente estocástico aproximado (LMS) que es muy utilizado en la actualidad. Posteriormente, en los años 60 surgieron una serie de publicaciones que hicieron aproximaciones innovadoras, [HAY91],[SIB87].

Durante las últimas tres décadas se ha generado una gran cantidad de literatura sobre la solución de problemas relacionados con procesamiento adaptable de señales. la motivación principal de los investigadores en esta área han sido las necesidades prácticas del procesamiento de señales. [SIB87].

Los primeros investigadores en el área de filtrado adaptable estuvieron interesados en la adaptación de antenas e igualación de canal en sistemas de transmisión digital. En marzo de 1964 la IEEE publica una primera emisión especial sobre adaptación de antenas. [IEEE64].[SIB87].

En 1965 Paul Howells obtiene una patente para un cancelador de lóbulos laterales de radiación. siendo ésta la primera aplicación exitosa en técnicas de anulación. En este mismo año en Francia, H. Mermoz publica su disertación de Ph. D. titulada "Adaptive Filtering and Optimal Utilization of an Antenna".

El año de 1965 fue el inicio de un secuencia de publicaciones sobre igualación adaptable de sistemas digitales de transmisión. Lukcy, de los laboratorios Bell, publicó un artículo clásico sobre igualación adaptable. Más tarde sus ideas fueron aplicadas a la eliminación de ruido y cancelación de eco, [SIB87].

En 1967 B. Widrow y otros publican su clásico artículo sobre sistemas de adaptación de antenas. en esta publicación la minimización del error cuadrático medio fue completado por el algoritmo LMS. [WID67].

En 1976, la IEEE publicó una segunda emisión especial sobre adaptación de antenas. [IEEE76].

A finales de los 70's una tercera emisión especial de sistemas de procesamiento adaptable de antenas fue editado por William E. Gabriel, esta edición ilustra el crecimiento y el progreso en el procesamiento adaptable en ésa década, [SIB87].

En la década de los 80's hubo un aumento en el número de publicaciones sobre algoritmos adaptables, es decir, que los algoritmos adaptables y sus aplicaciones son relativamente recientes. aunque su base matemática y sus principios datan desde inicios de este siglo. Además, el futuro de estos algoritmos es muy amplio, sobre todo con el crecimiento de los sistemas de comunicaciones a altas velocidades, [SIB87].

1.2 ORGANIZACION DEL TRABAJO

En el capítulo **II**, se presenta un visión generalizada sobre lo que son los sistemas y filtros adaptables, las disciplinas y áreas de aplicación, sus estructuras de implementación y algunas características.

En el capítulo **III**, se hace un estudio de los algoritmos adaptables iniciando desde las ideas básicas, los algoritmos para señales estacionarias, la solución de Wiener-Hopf, el algoritmo del error cuadrático medio (MSE), algoritmos de pasos descendentes, el algoritmo LMS, el algoritmo recursivo de los mínimos cuadrados (RLS), el algoritmo rápido RLS, y los algoritmos recursivos en orden. Se hace un desarrollo presentando la deducción de estos algoritmos, mostrando sus características de convergencia y robustez. Como una prueba de la utilidad de estos algoritmos se hace una evaluación de las aplicaciones de estos algoritmos en la eliminación de eco, igualación de canal, cancelación ruido y predicción de señales, efectuando simulaciones de los diversos algoritmos y presentando los resultados que se obtienen.

En el capítulo **IV** se presentan algunas versiones estabilizadas de los algoritmos adaptables, ya que en la implementación de los algoritmos se pueden presentar divergencias debido a condiciones de inicialización, o cuando la señal de entrada no es excitada persistentemente, o bien debido a la propagación de errores numéricos.

En el capítulo **V**, se hace una evaluación de la robustez numérica de los algoritmos, se hace un resumen de la metodología de implementación empleada en este trabajo, se aborda la problemática de la implementación de algoritmos adaptables en aritmética de precisión finita, la generación de errores, se propone una arquitectura de cálculo que sea capaz de resolver el problema de la eliminación de eco en aplicaciones en tiempo real, además se presentan simulaciones y resultados. Se presentan un conjunto de pruebas y resultados de los diversos algoritmos utilizados, se hacen comparaciones y se evalúan los resultados.

En el capítulo **VI**, para finalizar, se presentan las conclusiones globales del trabajo desarrollado haciendo énfasis en los resultados obtenidos, y se presenta una visión general de las perspectivas de ésta área de investigación.

Como complemento al desarrollo de los capítulos anteriores se agrega una sección final de anexos. En el anexo **A** se hacen algunas deducciones y desarrollos que se citan en algunos capítulos del documento. Esto se hace con el objetivo de tener los orígenes y formas de llegar a los algoritmos presentados. En el anexo **B** se agrega un breve resumen de la arquitectura de del DSP TMS320C50 y algunos programas en lenguaje ensamblador.

Capítulo II

GENERALIDADES DEL FILTRADO ADAPTABLE

GENERALIDADES DEL FILTRADO ADAPTABLE

Durante las últimas tres décadas los algoritmos de filtrado adaptable han logrado un importante impulso y han tenido una diversidad de aplicaciones, sin embargo, no todas las personas que están cercanas a los campos de aplicación tienen un conocimiento extenso de estos algoritmos. Por lo tanto, en este trabajo se introduce en este capitulo una visión general de diversos sistemas y algoritmos adaptables conocidos, sus características sobresalientes, las estructuras de sistemas digitales y sus aplicaciones a manera de introducir el tema. Estas generalidades nos permiten sentar ciertas bases para el análisis que se presenta en capítulos posteriores.

II.1 LOS SISTEMAS ADAPTABLES

Un sistema adaptable se asemeja a las características de los sistemas biológicos que ante cualquier alteración en su estructura o funcionamiento de su organismo o de sus partes, como resultado de la selección natural, el organismo se vuelve más conveniente para sobrevivir y multiplicarse en ese ambiente.

La propiedad esencial de un sistema adaptable es autoajustar su desempeño a variaciones en el tiempo. Si una señal es aplicada a la entrada de un sistema adaptable, el sistema se adapta a la señal de entrada en relación a los cambios que presenta la entrada, esto significa que un sistema adaptable es difícil caracterizarlo en términos convencionales. En general, un sistema adaptable es aquel que se adecúa a los cambios del medio donde interactúa para efectuar correctamente su desempeño, [WID85].

En el procesamiento adaptable de señales existe un número finito de parámetros que son ajustables por los algoritmos al optimizar un criterio de desempeño, [SIB87].

Por su naturaleza, los sistemas adaptables son no lineales y variables en el tiempo. Generalmente, la forma, la estructura o el ajuste de los sistemas adaptables será diferente para dos entradas diferentes. Es decir, que no cumple con el principio de superposición, por lo tanto, los sistemas son no lineales, [WID85].

Dentro de los sistemas no lineales, un sistema adaptable no puede distinguirse si pertenece a un subconjunto claro de ellos, sin embargo, éstos tienen dos rasgos distintivos que los diferencian de otros sistemas no lineales [WID85]:

a) Los sistemas adaptables son ajustables, y este ajuste usualmente depende de las características promedio de una señal en un tiempo finito, en lugar de los valores instantáneos de las señales o los valores instantáneos del estado interno del sistema.

b) El ajuste del sistema adaptable cambia en el sentido de optimizar medidas específicas de su desempeño.

Ciertas formas de sistemas adaptables se convierten en sistemas lineales cuando sus ajustes se mantienen constantes después de la adaptación.

II.2 CARACTERISTICAS DE LOS SISTEMAS ADAPTABLES

En general los sistemas adaptables tiene las siguientes características, [WID85]:

1) Se adaptan automáticamente para enfrentar los cambios del ambiente y los cambios requeridos por el sistema.

2) Estos pueden ser entrenados para efectuar un filtrado específico y tomar decisiones en línea. Es decir, que los sistemas adaptables pueden ser programados por un proceso de entrenamiento.

3) Pueden extrapolar un modelo de comportamiento al tratar con nuevas situaciones, después de haber sido entrenados sobre una cantidad finita de señales o un patrón frecuente de entrenamiento.

4) Estos pueden corregirse asimismo, es decir, que ellos pueden adaptarse alrededor de cierta clase de defectos internos.

5) Pueden ser descritos como un sistema no lineal con parámetros variables en el tiempo.

6) Usualmente son más complejos que los sistemas no adaptables, pero ofrecen la posibilidad de un incremento en el desempeño del sistema, cuando las características de la señal de entrada es desconocida o variable en el tiempo.

La propiedad principal y esencial de un sistema adaptable es que este es variable en el tiempo y autoajusta su desempeño. En cambio, en un *sistema fijo*, el diseñador tiene un panorama de las posibles entradas, sus estadísticas, y conoce qué se debe hacer bajo cada una de estas circunstancias. El diseñador tendrá que elegir un criterio específico para juzgar la cantidad de error entre la salida del sistema real y el modelo seleccionado del sistema.

Cuando se desconocen las condiciones de entrada, y sus estadísticas o las condiciones pueden cambiar en el tiempo, en estos casos los *sistemas adaptables* buscan constantemente el desempeño óptimo dentro de todas las posibilidades permitidas.

Una forma de clasificación de los esquemas adaptables es pensar en términos de lazo abierto y lazo cerrado de adaptación, [WID85].

Un proceso de adaptación en lazo abierto, involucra mediciones de las entradas o las características del ambiente, aplica esta información a un algoritmo y usa los resultados como conjunto de ajuste para adaptar el sistema. El criterio de adaptación es una característica de la señal de entrada y quizás de otros datos.

Una adaptación en lazo cerrado, involucra una experimentación automática con los ajustes y el conocimiento de la salida en el sentido de optimizar el desempeño del sistema. Es decir, que la adaptación depende de la señal de salida.

II.3 APLICACIONES DE ADAPTACION EN LAZO CERRADO

En la figura 2.1, se tiene un diagrama de lazo cerrado de un sistema adaptable, donde para una señal de entrada **x(k)** y una señal de respuesta deseada **d(k)** (**k** es el tiempo discreto), se tiene una señal de error **e(k)** que es la diferencia entre la señal de salida deseada **d(k)** y la señal de salida verdadera **y(k)**.

Utilizando la señal de error, un algoritmo adaptable ajusta la estructura del sistema adaptable (el proceso en figura 2.1), alterando su respuesta característica para minimizar alguna medida del error, y por tanto cerrando el lazo. La diferencia esencial entre las diversas aplicaciones de filtrado adaptable es la manera en que la respuesta deseada es excitada. Bajo esta idea es posibles tener diferentes estructuras de sistemas adaptables.

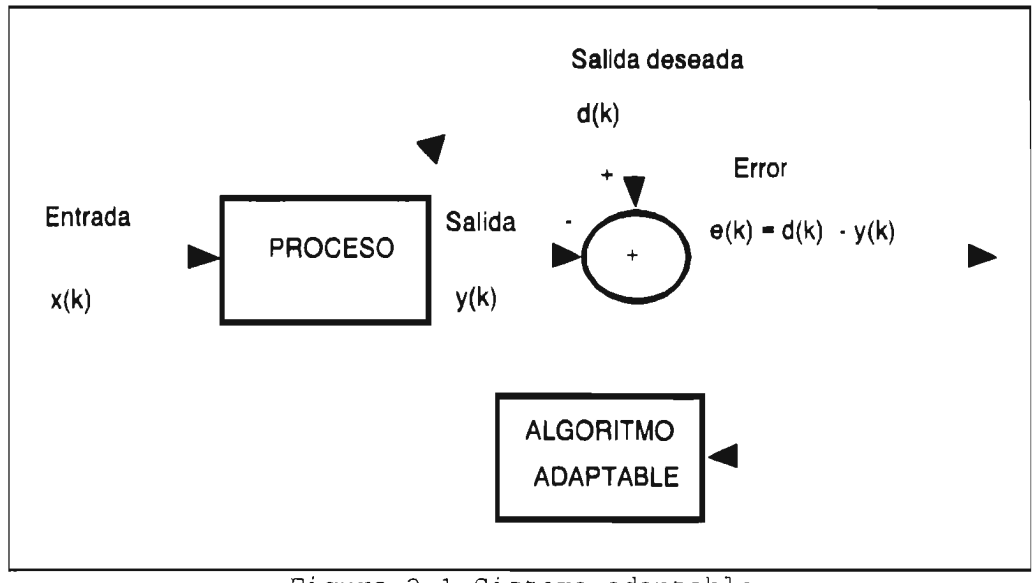


Figura 2.1 Sistema adaptable

A continuación se describe algunas de las aplicaciones más típicas del filtrado adaptable:

II.3.1 Predicción

Tal vez es la aplicación más simple, la señal de entrada x(k) y su versión retrasada es enviada al proceso adaptable (figura 2.2.a), éste debe de tratar de predecir la entrada actual de la señal x(k) de manera que las señales x(k) y x^(k) se cancelen para hacer tender el error a cero. La predicción es utilizada en la codificación o compresión de señales y la reducción de ruido. La señal deseada es la muestra actual y debe ser la respuesta del filtro. [WID85],[HAY91],[MAK75].

II.3.2 Identificación de sistemas

En este caso la señal transmitida x(k) es la señal de entrada tanto al proceso adaptable como a una planta desconocida. La reducción del error e(k) en el proceso adaptable trata de emular la función de transferencia característica de la planta desconocida. Después de la adaptación la planta es "identificada" en el sentido que la función de transferencia de la planta pueda ser especificada como la misma del proceso adaptable. Si la planta es de naturaleza dinámica el modelo variará en el tiempo (fig. 2.2b). Los sistemas adaptables de identificación pueden ser utilizados para modelar las variaciones lentas de una planta cuyas entradas y salidas son disponibles, tal es el caso de estudios de vibraciones mecánicas, [WID85],[HAY91],[WID96].

II.3.3 Modelo inverso

El proceso adaptable intenta recuperar una versión retardada de la señal de entrada x(k) (esta x(k) retardada sería la señal deseada), la cual se asume que ha sido alterada por las variaciones lentas de la planta y el contenido de ruido (fig. 2.2c). La igualación adaptable es utilizada para la deconvolución de los efectos de un transductor, un canal de comunicación o la producción de un modelo inverso de una planta. También es aplicable en el diseño de filtros digitales, [WID85],[HAY91],[WID96].

II.3.4 Cancelación de interferencias

En este caso la señal x(k) está corrompida por ruido n(k), y se dispone de una versión correlacionada de ruido $n_1(k)$ que entra al proceso adaptable. El objetivo del proceso adaptable es producir una salida y(k) parecida a x(k). Es decir, minimizar el error cuadrático medio de e(k) (figura 2.2.d). [WID85],[HAY91].

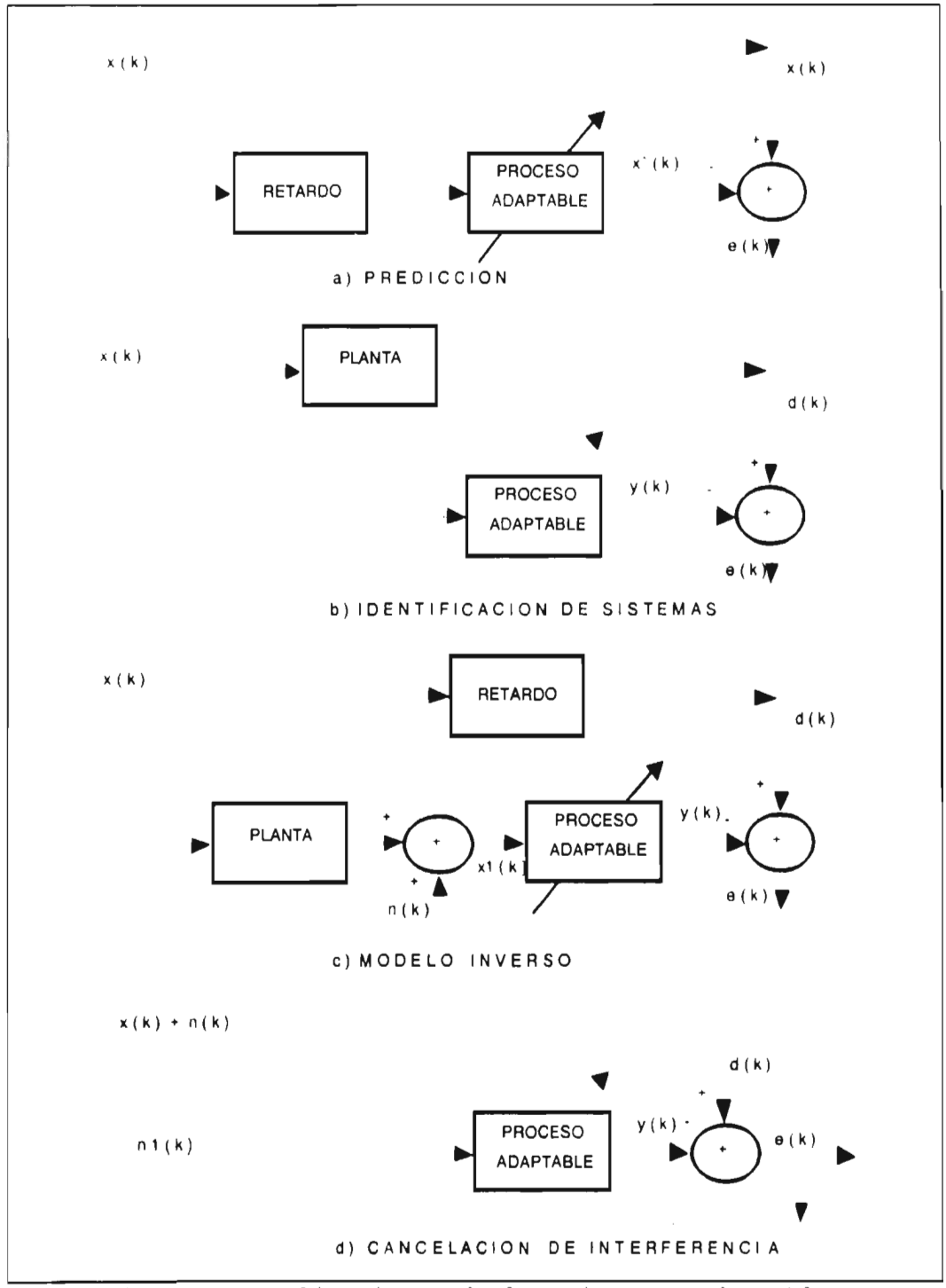


Figura 2.2 Aplicaciones de los sistemas adaptables

II.4 FACTORES PARA SELECCIONAR UN FILTRO ADAPTABLE

En la actualidad, han sido desarrollados una gran cantidad de algoritmos recursivos para la operación de filtros adaptables, la elección de uno u otro está determinada por varios factores [HAY91], unos algoritmos tienen ciertas ventajas sobre otros, o su comportamiento es mejor en la evaluacion de alguno de estos factores, en su momento se harán comparaciones de estos factores.

1) Razón de Convergencia
Es el número de iteraciones requeridas por el algoritmo en respuesta a las entradas para obtener una solución o una respuesta óptima. Una razón rápida de convergencia permite al algoritmo adaptarse rápidamente a un ambiente de estadísticas desconocidas o cambio del modelo.

2) Ajuste
Para el algoritmo de interés, estos parámetros dan una medida cuantitativa de la cantidad del valor final del error medio cuadrático. En general, cuando se tiene un conocimiento a priori del problema se pueden estimar valores o intervalos teóricos.

3) Seguimiento
Cuando un algoritmo de filtrado adaptable opera en un ambiente no estacionario, se requiere que el algoritmo siga las variaciones estadísticas del ambiente. El desempeño del seguimiento de un algoritmo está influenciado por dos rasgos:
a) la razón de convergencia
b) las fluctuaciones en estado estable del algoritmo o del algoritmo mismo.

4) Robustez
Se refiere a la capacidad del algoritmo de operar satisfactoriamente con datos mal acondicionados, en otras palabras, cuando alguna suposición hecha sobre el algoritmo se sale de lo especificado. El término robustez también es usado en el contexto del comportamiento numérico. Un algoritmo de filtrado adaptable se dice que es robusto numéricamente cuando es insensible a cambios en la longitud de palabra numérica en su implementación.

5) Requerimientos Computacionales
Incluye el número de operaciones que utiliza el algoritmo y la cantidad de memoria. Así, tenemos algoritmos rápidos, lentos o con poco o mucho consumo de memoria.

6) Estructura
La estructura se refiere al flujo de información en el algoritmo y es lo que determina la manera de implementarlo en hardware. Las estructuras más utilizadas son la transversal y la "lattice".

7) Propiedades numéricas
Se refiere a las imprecisiones que son producidas debidas a errores de cuantización (en un convertidor A/D), las propiedades numéricas del algoritmo adaptable están estrechamente ligadas a la robustez numérica. Aquí se tiene conceptos con significado diferente:

Estabilidad numérica: es una característica inherente de un algoritmo de filtrado adaptable.

Precisión numérica: está determinada por el número de bits utilizados en la representación de las muestras de una señal o los coeficientes del filtro.

II.5 FORMA GENERAL DE LOS FILTROS ADAPTABLES

La operación de un algoritmo de filtrado adaptable involucra dos procesos básicos, el filtrado y la adaptación:

1) *Proceso de filtrado*
   Está diseñado para producir una respuesta o secuencia de salida deseada a una secuencia de entrada dada.

2) *Proceso adaptable*
   Provee un mecanismo para el control de un conjunto de parámetros de ajuste usados en el proceso de filtrado.

Estos dos procesos trabajan iterativamente. La elección de una estructura para el proceso de filtrado tiene profundos efectos en la operación del algoritmo como un todo. Existen tres tipos de estructuras que se distinguen entre ellas en el contexto de filtros adaptables.

II.5.1 Filtros transversales

Consiste básicamente de bloques de retardos ("taps"), multiplicadores y sumadores. El número de bloques de retardo (p-1) determinan la duración de la respuesta al impulso, y el número p se refiere como el orden del filtro.

La salida y(k) del sistema se puede expresar como la suma de convolución entre la p valores retardados de la señal de entrada x(k) con los coeficientes w(i) del filtro, ver figura 2.3.

$$y(k) = \sum_{i=0}^{p-1} W(i)\ x(k-i) \qquad 2.1$$

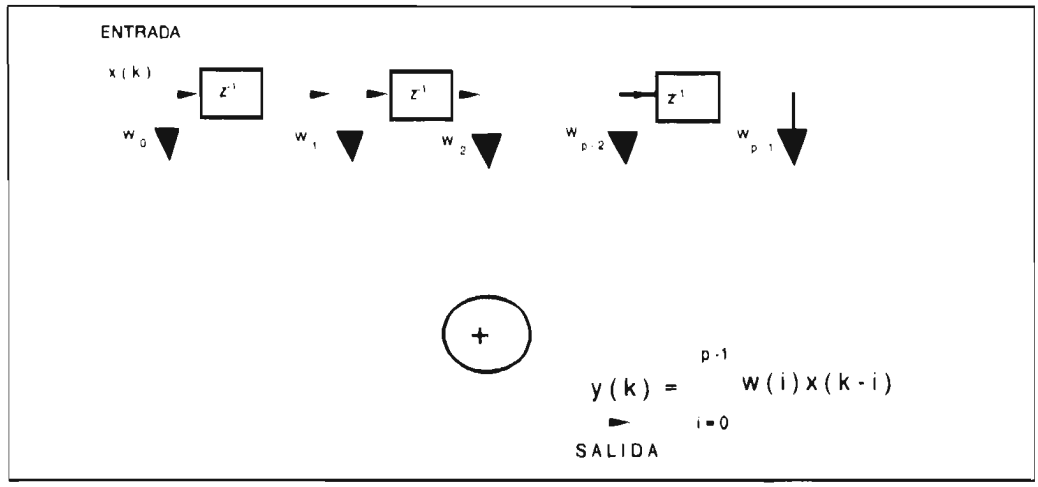


Figura 2.3 Estructura transversal

II.5.2 Filtro "lattice"

Una estructura "lattice" (de su nombre en inglés) es la estructura de un filtro similar a una rejilla de líneas horizontales de retardos y sumas, y líneas cruzadas de coeficientes K_i. Un predictor "lattice" es una estructura modular que consiste de un número individual de etapas, cada una de ellas tiene apariencia de "lattice". El número "p", se refiere al orden del predictor. La etapa p-ésima es descrita por las relaciones (asumiendo entradas en el sentido estacionario) [FRI82],[HAY91]:

$$e_p^f(k) = e_{p-1}^f(k) + K_p\, e_{p-1}^b(k-1) \qquad 2.2$$

$$e_p^b(k) = e_{p-1}^b(k-1) + K_p\, e_{p-1}^f(k) \qquad 2.3$$

$$\therefore \quad k = 1,2 \ldots M-1$$

Los coeficientes K_i son llamados coeficientes de reflexión (fig. 2.4).

El error "forward" (hacia adelante), $e_p^f(k)$, es definido como la diferencia entre la entrada y(k) y un valor de predicción pasado, donde este valor de predicción está basado en "p" muestras anteriores de las entradas y(k-1)...y(k-p).

El error "backward" (hacia atrás), $e_p^b(k)$, es definido como la diferencia entre la entrada y(k-m) y un valor de predicción hacia atrás, donde este valor de predicción está basado en m muestras futuras de las entradas y(k)...y(k-p+1).

También se cumple que las condiciones iniciales son :

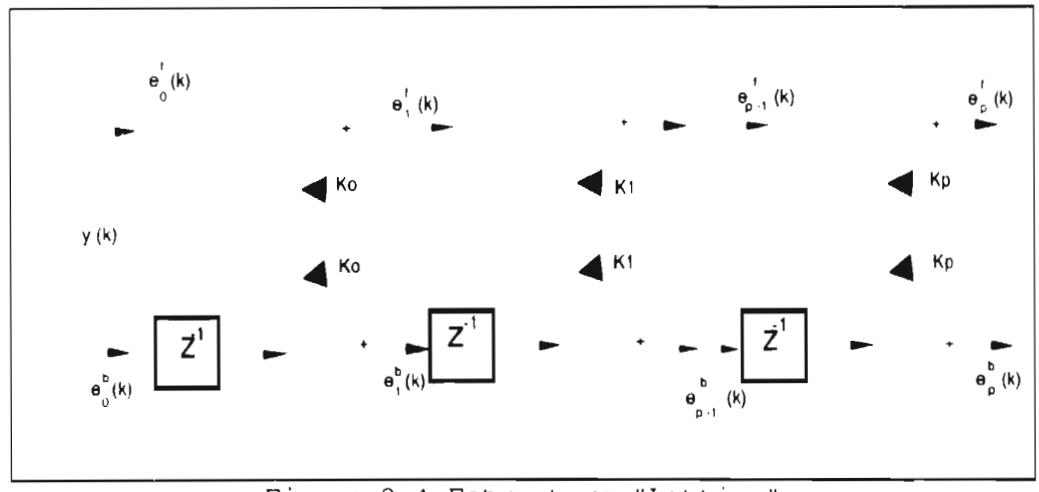


Figura 2.4 Estructura "Lattice"

$$e_0^f(k) = e_0^b(k) = y(k) \qquad 2.4$$

como se observa en la figura 2.4, donde y(k) es la señal de la entrada al instante k.

Partiendo de las condiciones iniciales y un conjunto de coeficientes de reflexión K_1, K_2... K_{p-1} se pueden encontrar el par de salidas $e_{p-1}^f(k)$ y $e_{p-1}^b(k)$ desplazándose a través del predictor "lattice" etapa a etapa.

Para un proceso estacionario, las secuencias y(k), y(k-1)...y(k-p+1) y los errores de predicción hacia atrás $e_0^b(k), e_1^b(k) \ldots e_{p-1}^b(k)$ forman una secuencia de variables aleatorias no correlacionadas, [PRN92].

Una de las ventajas de la estructura "lattice" sobre la transversal es que las etapas son desacopladas, es decir, que si se quiere aumentar o disminuir el orden del predictor, agregando o disminuyendo etapas, los coeficientes anteriores permanecen sin cambio. Además, los coeficientes K_i siempre son de magnitud menor que uno, lo que garantiza la estabilidad del filtro. En una estructura transversal cuando se incrementa o disminuye el orden es necesario recalcular todos los coeficientes.

La relación matemática entre los pesos de los retardos (w_i) de la estructura transversal y los coeficientes de reflexión k_i está dada por el algoritmo Levinson-Durbin, [HAY91].

II.5.3 Filtro "Lattice-ladder"

Es un tipo de estructura que se utiliza en el proceso de estimación conjunta. Si se utilizan los errores de predicción hacia atrás $e_0^b(k), e_1^b(k) \ldots e_{p-1}^b(k)$ de la estructura lattice como entradas a

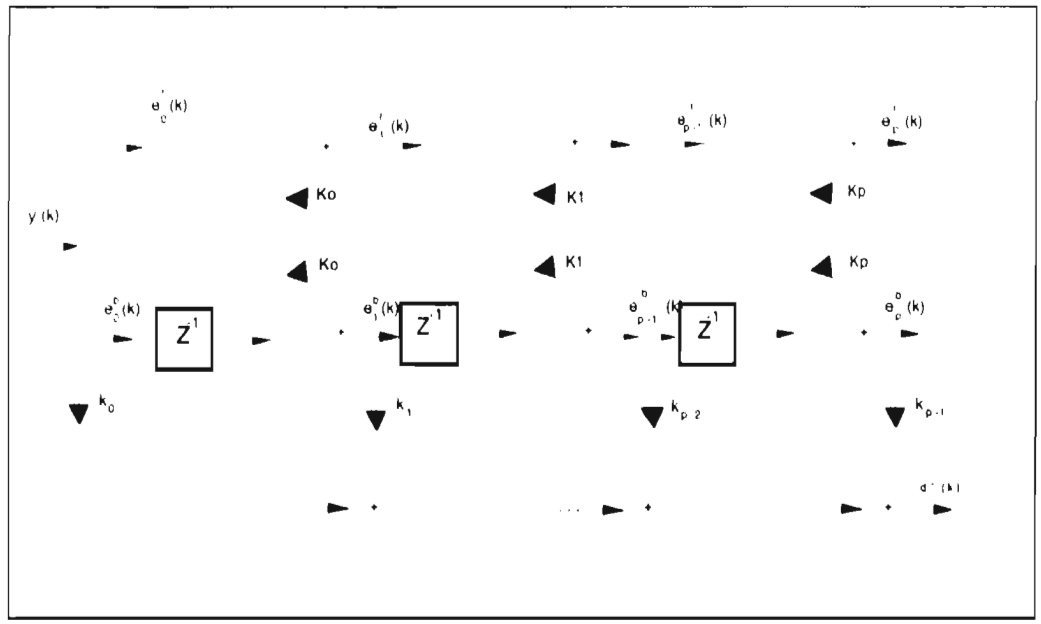


Figura 2.5 Estructura "Lattice-ladder"

un filtro transversal cuyo vector de coeficientes está dado por la regresión: κ_0, κ_2, ... κ_p (fig. 2.5), se obtiene un estimador de la respuesta deseada d^(k). A esta estructura se le conoce como estimador conjunto dado que resuelve el problema de estimar el proceso deseado d(k) a partir de las observaciones del proceso de entrada y(k), es decir, se tiene un proceso conjunto {d(k),y(k)}, [ALC86],[HAY91].

II.6 CLASIFICACION DE LOS ALGORITMOS ADAPTABLES

Cuando se diseña un filtro en general se puede partir de la idea de un *filtro fijo*, sin embargo, éstos se basan en un conocimiento previo de las señales de interés, es decir, su diseño es válido sólo para las característica previstas.

Otra técnica de diseño es el *filtrado de Wiener*, sin embargo, para su diseño se requiere un conocimiento previo de las propiedades estadísticas de la señal de entrada, la señal deseada y las relaciones estadísticas entre estas. Por lo tanto, el filtrado de Wiener es el diseño de un filtro fijo, [WID96].

En el diseño utilizando ***filtros adaptables***, no se requiere un conocimiento previo de la señal de entrada y la señal a estimar. Estos convergen en forma asintótica a la solución de Wiener.
En los problemas de filtrado adaptable no existe una solución única ya que en la actualidad se conocen una variedad de algoritmos, sin embargo, cada uno ofrece sus propios rasgos o características.

En el estudio de filtros adaptables pueden surgir varios criterios para la clasificación de los algoritmos, dentro de estas están los factores para seleccionar un filtro y que se mencionaron en el punto II.4. Sin embargo, no se puede hacer una clasificación tan excluyente para agrupar a los algoritmos, es decir, que una clasificación de los mismos puede involucrar varios criterios. A continuación se hace una clasificación bastante general con base al criterio de minimización del error, la velocidad de convergencia y la estructura de los algoritmos, además en el capítulo III se sigue una secuencia similar para el desarrollo de algunos algoritmos.

En la obtención de un algoritmo de filtrado adaptable, como se explicó anteriormente se tiene un error de predicción, y uno de los objetivos es minimizar este error para obtener la estimación optima de los coeficientes. Uno de los criterios más utilizados para la minimización de este error es el de los mínimos cuadrados (MC o LS del inglés least square), y dependiendo cómo se pondere el criterio se pueden obtener varios algoritmos. A continuación se listan varios algoritmos de filtrado adaptable.

II.6.1 Algoritmos tipo gradiente

Estos algoritmos son de tipo estocásticos, es decir, que se basan en la derivación de propiedades estadísticas de las señales. La principal medida estadística utiliza la esperanza matemática del error cuadrático, frecuentemente a esto se le llama error cuadrático medio (MSE).

Los algoritmos del gradiente estocástico se basan en la mínimización del error cuadrático (llamada por algunos autores función de costo). Estos algoritmos hacen una estimación recursiva del gradiente del error cuadrático medio para ajustar los coeficientes del filtro. Dentro de estos están:

Algoritmo de los pasos descendentes ("steepest descent")

En este algoritmo se hace una minimización de la esperanza del error cuadrático medio (MSE) para obtener la solución óptima. Este es un algoritmo recursivo, ya que empieza de un valor inicial arbitrario, el valor final del vector de coeficientes converge a la solución de Wiener, un hecho importante del algoritmo es que encuentra el mínimo de una superficie de error sin ningún conocimiento a priori de la misma, [HAY91].

Algoritmo del gradiente estocástico aproximado (LMS)

Este algoritmo hace una minimización sobre promedios temporales del error cuadrático medio (MSE del inglés mean square error). Como se verá en el capítulo III este algoritmo sustituye los promedios probabilísticos por promedios temporales, y se le llama aproximado porque hace un cálculo aproximado de los coeficientes.

II.6.2 Aproximación basada en la teoría de filtrado de Kalman

El problema de Kalman para un sistema dinámico lineal es formulado en términos de dos ecuaciones básicas, o modelo por variables de estado [HAY91]:

a) Ecuación de proceso: que describe la dinámica del sistema en términos de los vectores de estado.

b) Ecuación de medición: que describe la medida del error incurrido en el sistema.

La solución del problema es expresado como un conjunto de recursiones de una matriz actualizada en el tiempo. La teoría requiere que se postule un modelo de las condiciones de operación óptima, que sirve como una referencia para el filtro de Kalman o el seguimiento. También se puede utilizar una solución recursiva utilizando algoritmos adaptables para la actualización de los pesos de un filtro transversal.

Los algoritmos de Kalman pueden tratar con ambientes estacionarios y no estacionarios, su limitación básica es su complejidad de cálculo, que es una consecuencia directa de la formulación matricial que da la teoría de Kalman a la solución del problema de filtrado.

II.6.3 Método de mínimos cuadrados (LS del inglés least square)

Este método involucra el uso de promedios en el tiempo, de acuerdo a este método se minimiza un índice de desempeño que consiste en la suma de los errores cuadrados, donde el error residual es definido como la diferencia entre la respuesta deseada y la respuesta de salida actual del filtro [HAY91].

El método LS se puede realizar por medio de una estimación de bloques o una estimación recursiva.

En la estimación *por bloques* los datos de entrada son arreglados en forma de segmentos de la misma duración finita y el filtrado se procesa sobre un bloque base, una vez procesado un bloque se espera otro nuevo bloque de datos para continuar con el proceso.

La estimación *recursiva* de los coeficientes del filtro es actualizada muestra a muestra. La estimación recursiva requiere menos recursos de almacenamiento que la estimación por bloques, debido a que no espera obtener un bloque de datos para seguir procesando.

Algoritmo Recursivo de los Mínimos Cuadrados (RLS)

El algoritmo RLS asume el uso de estructuras de filtros transversales como la base de los filtros adaptables. La derivación de este algoritmo se basa en el lema de inversión matricial (ver anexo A.1). El algoritmo RLS es un caso especial del algoritmo de Kalman para filtrado adaptable.

Algoritmo de mínimos cuadrados de descomposición basado en descomposición Q-R (QR-RLS)

Esta clase de algoritmos se basa en las descomposición matricial Q-R. Esta transformación involucra triangularización ortogonal de los datos de la matriz de entrada.

II.6.4 Algoritmos rápidos

El algoritmo RLS que utiliza estructuras transversales y su contraparte la descomposición QR, tienen una complejidad computacional que se incrementa al cuadrado de p, donde p es el número de pesos ajustables o coeficientes de filtro adaptable. Tales algoritmos son frecuentemente referidos como algoritmos $O(p^2)$, donde O(.) denota "orden de" o número de operaciones aritméticas, [CAR84],[ALC86].

Un algoritmo rápido tiene como propósito explotar la estructura de la matriz de covarianza de los datos. Entre los algoritmos rápidos se pueden identificar:

Algoritmo RLS rápido (FRLS) o Algoritmo rápido de Kalman (ARK)

Como su nombre lo indica es un algoritmo que proviene del RLS, éste explota la redundancia en el algoritmo LS y para su realización efectúa la predicción forward (hacia adelante) y backward (predicción hacia atrás), [CAR84],[FAL78]. Su nombre proviene de un vector de ganancia similar a la aproximación del filtrado de Kalman para la formulación en variables de estado, este se desarrollará en el capítulo III.

Algoritmo de filtrado transversal rápido (FTF)

Este algoritmo involucra una combinación de cuatro filtros transversales, cada uno tiene una tarea asignada a ejecutar. Este algoritmo reemplaza algunas operaciones vectoriales del algoritmo FRLS por operaciones escalares, reduciendo el número de operaciones respecto del ARK [CIO84],[HAY91],[ALE86].

Algoritmo recursivo de los mínimos cuadrados Lattice (LSL del inglés Least square lattice)

Utiliza estructuras "lattice" para efectuar el filtrado, y la actualización de los parámetros del mismo se basa en la estimación de los mínimos cuadrados de los predictores "forward" y "backward", [FRI82].

Algoritmos lattice de mínimos cuadrados basados en la descomposición QR

En este algoritmo las formas de predicción "forward" y "backward" son explotadas con el propósito de reducir la complejidad computacional en la ejecución de la descomposición QR de la matriz de datos de entrada, [K&T93].

II.6.5 Algoritmos estabilizados

Es importante notar que la remoción de la redundancia de los algoritmos LS pueden llevar a la vulnerabilidad de los efectos de precisión numérica. Debido a que los algoritmos RLS rápidos presentan problemas de estabilidad numérica, los algoritmos estabilizados agregan algunas redundancia en los cálculos que intentan revertir esta problemática [SLO91],[BOT89]. Sin embargo, éstos parten de algún algoritmo existente y su objetivo es mantenerlo estable. En este trabajo se estudian algoritmos estabilizados cuya estructura proviene del algoritmo rápido de Kalman.

En este capítulo se ha hecho una exposición general de los algoritmos de filtrado adaptable, sus características, sus aplicaciones, forma general, las estructuras de implementación y una clasificación muy general. Como se observa existe una gran cantidad de algoritmos adaptables, sin embargo sería muy extenso abordarlos todos detalladamente en este trabajo, y dependiendo de la aplicación y de las características deseadas se elegirá uno u otro. Por lo tanto, bajo el esquema anterior se desarrollarán en el siguiente capítulo algunos de los algoritmos mencionados y se presentan sus características y secuencia de implementación.

Capítulo III

ALGORITMOS DE FILTRADO

ADAPTABLE

ALGORITMOS DE FILTRADO ADAPTABLE

Al final del capitulo II se hizo una clasificación general de los algoritmos de filtrado adaptable, en este capitulo se desarrollan algunos de esos algoritmos partiendo del criterio de optimizacion utilizado para llegar a una forma resumida del algoritmo llamada organigrama del algoritmo, ademas, se presenta la forma de implementarlos, se explican sus caracteristicas, su convergencia y su robustez. Para no ser muy extenso en los desarrollos y demostraciones, las partes intermedias o muy largas están en el anexo A del trabajo. Se presentan algunos resultados parciales de su convergencia y al final del capitulo se hace un análisis de las aplicaciones de éstos algoritmos, presentando resultados al efectuar las aplicaciones. Se hace notar que por la gran cantidad de algoritmos existentes no se abordan todos, sin embargo, se abordan los algoritmos más utilizados y que revelan mejores características.

Los algoritmos adaptables para el procesamiento de señales generalmente prueban la optimización de medidas que son función de parámetros desconocidos a identificar. La mayoria de los metodos se basan en la estimación del error cuadrático, aunque el error específico de estimación depende del algoritmo en particular, [ALE86].

Como se observa más adelante algunas de las propiedades de los filtros MSE y las superficies MSE son derivadas utilizando técnicas básicas de algebra lineal (ver anexo A.5).

III.1 ERROR CUADRÁTICO MEDIO (MSE) Y SUPERFICIES MSE

Si se parte de la idea de predicción, se tiene una señal de entrada y(k), una señal deseada d(k)=y(k) y una señal estimada y^(k) de salida del proceso adaptable. La señal y^(k) se obtiene como una combinación lineal de los coeficientes estimados y las entradas retrasadas de y(k-i), es decir, que se tiene una combinación lineal. Idealmente la salida y^(k) debería de ser igual a la señal deseada d(k) para tener un error de predicción cero. Sin embargo, existe un error e(k) que es la diferencia entre la señal deseada y la señal y^(k).

La combinación lineal adaptable o filtrado adaptable no recursivo, aparece en una u otra forma en la mayoría de filtros y sistemas adaptables, en general este es el elemento simple más importante a aprender en sistemas y procesos adaptables, [WID85]. En este caso una señal de salida y^(k) puede representarse como la combinación lineal de las muestras de retardo y(k-i) multiplicadas por los respectivos pesos de un filtro transversal.

$$\hat{y}(k) = \sum_{i=1}^{p} W_i \, y(k-i) \qquad 3.1$$

expresándolos en forma vectorial:

$$\mathbf{w} = [w_1, w_2 \ldots w_p]^T \qquad 3.2$$

$$\mathbf{y} = [y(k-1), y(k-2) \ldots y(k-p)]^T \qquad 3.3$$

la señal de salida $y\hat{}(k)$ al instante k es un escalar, entonces se puede escribir como un producto interno entre los vectores **y** y **w**:

$$y\hat{}(k) = \mathbf{y}^T\mathbf{w} = \mathbf{w}^T\mathbf{y} \qquad 3.4$$

El valor de $y\hat{}(k)$ es una combinación lineal de las muestras pasadas de la señal de entrada y(k) multiplicadas por los valores actuales correspondientes de w(k).

Si se requiere obtener una señal deseada $d(k)=y(k)$, entonces al hacer la comparación con la salida $y\hat{}(k)$ se tiene un error e(k), que es la diferencia entre estas dos señales:

$$e(k) = d(k) - y\hat{}(k) = y(k) - \mathbf{y}^T\mathbf{w} = y(k) - \mathbf{w}^T\mathbf{y} \qquad 3.5$$

Para que el error sea $e(k) = 0$, es decir, que $y\hat{}(k) = d(k)$, la señal de error es minimizada por algún criterio en el sentido de encontrar los valores de los pesos de w(k).

En la mayoría de los sistemas un valor de error grande negativo como positivo van en detrimento del sistema. Por lo tanto, una medida del desempeño deseable es que existan pesos negativos iguales a los positivos, [ALE86]. Una proposición al respecto es la evaluación del valor absoluto del error $|e(k)|$, sin embargo, esta medida lleva a resultados matemáticos intratables cuando se pretende analizar el resultado de los algoritmos. La medida del error cuadrático $e^2(k)$ lleva un mejor análisis y por lo tanto es ampliamente utilizado.

Desarrollando la expresión del error cuadrático

$$e(k)^2 = (y(k) - y\hat{}(k))^2 = y(k)^2 + \mathbf{w}^T\mathbf{y}\mathbf{y}^T\mathbf{w} - 2d(k)\mathbf{y}^T\mathbf{w} \qquad 3.6$$

asumiendo que e(k) e **y** son estadísticamente estacionarias y tomando el valor esperado del error cuadrático:

$$E\{e(k)^2\} = E\{y(k)^2\} + \mathbf{w}^T E\{\mathbf{y}\mathbf{y}^T\}\mathbf{w} - 2E\{y(k)\mathbf{y}^T\}\mathbf{w} \qquad 3.7$$

Los coeficientes **w** son constantes con respecto a la esperanza matemática.

definiendo: La matriz de autocorrelación de la señal de entrada **y**

$$\mathbf{R} = E\{\mathbf{y}\mathbf{y}^T\} \qquad 3.8$$

donde:

$$R = E[y_k y_k^T] \quad \begin{bmatrix} R_0 & R_1 & \cdots & R_{n-1} \\ R_1 & R_0 & \cdots & \cdots \\ \cdot & \cdot & \cdot & \cdot \\ R_{n-1} & \cdot & \cdot & R_0 \end{bmatrix} \qquad 3.9$$

donde los elementos de la matriz se puede calcular para señales estacionarias y ergódicas como:

$$R_i = \frac{1}{p} \sum_{j=0}^{p-(i)-1} y(k-j)\, y(k-i-j) \qquad 3.10$$

Donde R_i es un estimador de la función de autocorrelación de y(k), y definiendo el vector de intercorrelación entre la respuesta deseada y los componentes de entrada

$$\mathbf{p} = E\{y(k)\mathbf{y}^T\} = E\,[y_k y_{1k} \quad \cdots \quad y_k y_{pk}]^T \qquad 3.11$$

El error cuadrático medio MSE se escribe:

$$\text{MSE} \triangleq E\{e(k)^2\} = \xi = E\{y(k)^2\} + \mathbf{w}^T \mathbf{R}\, \mathbf{w} - 2\, \mathbf{p}^T \mathbf{w} \qquad 3.12$$

Para señales estacionarias:

$$R_0 = E\{y(k)^2\} = \sigma^2 > 0 \quad \text{que es la energía de las señal y(k).} \qquad 3.13$$

El error medio cuadrático ξ es precisamente una función cuadrática de las componentes del vector de peso **w**, donde la señal de entrada y la señal deseada son variables estocásticas estacionarias.

Al graficar el error ξ para dos coeficientes (w_1, w_2), modelo de orden dos, se obtiene un paraboloide cóncavo hacia arriba, el punto que está en el fondo del paraboloide corresponde a las coordenadas **w*** (solución óptima). A esta superficie en el espacio tridimensional se le llama *superficie del error cuadrático medio*. Una de las propiedades más importantes de las superficies MSE es que tienen un sólo extremo y éste es el punto mínimo, [ALE86], es decir, que la derivada de $\xi(k)$ respecto de los parámetros **w** es cero.

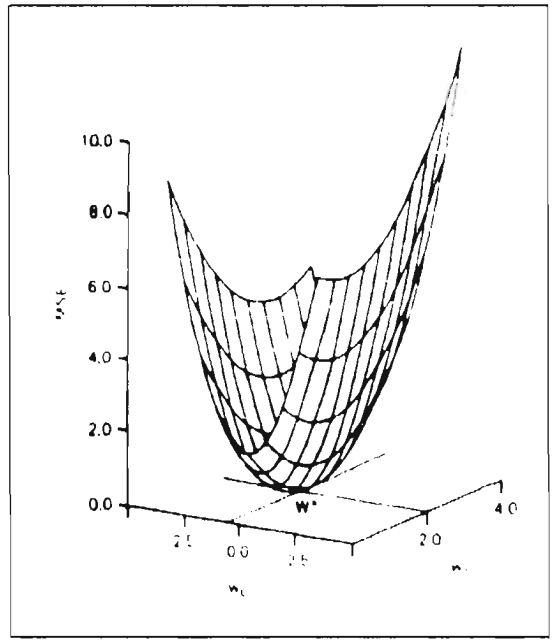


Figura 3.1 Superficie de error

III.1.1 El gradiente y el error cuadrático medio mínimo

Los métodos de gradiente son utilizados en procesos adaptables para buscar el vector de coeficientes que corresponde al mínimo de la superficie del error cuadrático medio MSE. [WID85].

Obteniendo el gradiente del error MSE:

$$\nabla\xi \triangleq \frac{\partial\xi}{\partial w} = \left[\frac{\partial\xi}{\partial w_1} \quad \frac{\partial\xi}{\partial w_2} \quad \cdots \quad \frac{\partial\xi}{\partial w_p}\right]^T \qquad 3.14$$

$$\nabla\xi = 2\mathbf{R}\mathbf{w} - 2\mathbf{p} \qquad 3.15$$

igualando a cero el gradiente se obtiene la solución óptima

$$\nabla = 0 = 2\mathbf{R}\mathbf{w}^* - 2\mathbf{p} \qquad 3.16$$

$$\mathbf{w}^* = \mathbf{R}^{-1}\mathbf{p} \qquad 3.17$$

A esta última ecuación se le conoce como ecuación de *Wiener-Hopf* (W-H), también es llamada ecuación normal. [WID85],[HAY91],[PRN92]. En aplicaciones prácticas, raras veces se efectúa la inversion de la matriz R, [HAY91], existiendo métodos más eficientes para la solución de esta matriz. Es decir, que la importancia de esta matriz es sólo de carácter teórico. Afortunadamente, en la mayoría de aplicaciones la matriz R es no singular, y por lo tanto existe su inversa.

Existen dos soluciones para la ecuación de W-H, una dada por el algoritmo de Levinson-Durbin y el otro por el algoritmo de los pasos descendentes, éste último conduce a tecnicas de procesamiento adaptable. El algoritmo de Levinson-Durbin conduce a estructuras tipo lattice adaptables. [PRN92].

Sustituyendo esta solución en el error cuadrático medio ξ :

$$\begin{aligned} \xi_{min.} &= E\{y(k)^2\} + \mathbf{w}^{*T}\,\mathbf{R}\,\mathbf{w}^* - 2\,\mathbf{p}^T\,\mathbf{w}^* \\ &= E\{y(k)^2\} + [\mathbf{R}^{-1}\mathbf{p}]^T\,\mathbf{R}\,[\mathbf{R}^{-1}\mathbf{p}]^T - 2\,\mathbf{p}^T\,[\mathbf{R}^{-1}\mathbf{p}]^T \end{aligned} \qquad 3.18$$

por propiedades de las matrices

$$\xi_{min.} = E\{y(k)^2\} - \mathbf{p}^T\,\mathbf{R}^{-1}\mathbf{p} = E\{d(k)^2\} - \mathbf{p}^T\,\mathbf{w}^* \qquad 3.19$$

Si las señales son estacionarias y tienen propiedades estadísticas invariantes, entonces las superficies cuadráticas permanecen fijas y rígidas en sus coordenadas del sistema. El proceso adaptable consiste en empezar desde algún punto de la superficie hasta llegar al mínimo.

La predicción óptima $\mathbf{w}^*$ de un filtro remueve las porciones de la señal deseada $y(k)$ que están correlacionadas con los datos $\mathbf{y}(k)$.

Cuando las señales son no estacionarias con propiedades estadísticas que cambian lentamente, la superficie tiene un comportamiento difuso u ondulatorio (cambia con el tiempo) o se mueve en las coordenadas del sistema. El proceso adaptable consiste no solo en desplazarse hacia el mínimo, sino también en hacer un seguimiento del mínimo tal como éste se mueva en las coordenadas del sistema, esta es una característica fundamental de un sistema adaptable. [WID85].

III.1.2 La ecuación normal

La ecuacion normal (3.17) provee una expresión muy importante en la predicción del error cuadrático medio y en filtrado, extendiéndose a diversos campos de procesamiento de señales, voz, comunicaciones, radar, economía, acústica, etc. Sin embargo, la teoría de la ecuación normal permite calcular el mínimo de un filtro MSE, pero aun existe la dificultad del cálculo de la matriz inversa. [ALE86].

La ecuacion normal puede obtenerse utilizando una propiedad del mínimo en la solucion MSE.

conocido como condicion de ortogonalidad, minimizando el MSE con respecto a **w** e igualando a cero:

$$\partial\, E\{\, e^2(k)\,\}\,/\partial \mathbf{w} = E\{\, \partial\{\, e^2(k)\,\}\,/\partial \mathbf{w}\,\} = E\,\{\, 2\, e(k)\, \partial e(k)/\partial \mathbf{w}\,\} = 0 \qquad 3.20$$

$$\partial e(k)/\partial \mathbf{w} = \partial\,\{\, y(k) - \mathbf{w}^T\mathbf{y}\,\}\,/\partial \mathbf{w} = -\,\mathbf{y} \qquad 3.21$$

$$E\,\{\; e(k)\, y(k)\,\} = 0$$

$$E\,\{\; e(k)\, \mathbf{y}(k-i)\,\} = 0 \qquad 3.22$$

Estas dos últimas expresiones son conocidas como *la condición de ortogonalidad* y son el resultado directo de la minimización del MSE. La condición de ortogonalidad establece que el producto escalar del error con los datos en la predicción son cero. La condición de ortogonalidad debe ser mantenida por el mínimo del predictor MSE. Una inmediata implicación es que si la media de e(k) o y(k) es cero, entonces la condición de ortogonalidad implica que la secuencia del error no está correlacionada con los datos (anexo A.3).

El principio de ortogonalidad establece que la longitud de $\xi = E\{e(k)^2\}$ es mínima cuando e(k) es perpendicular al subespacio de datos, es decir, que e(k) es ortogonal a cada dato y(k). Una interpretación geométrica se observa en la figura 3.2 para un espacio de dos datos, [PRN92]. Donde se observa que la señal estimada $\hat{y}(k)$ se obtiene al proyectar la señal deseada y(k) en el espacio de los datos y(1) e y(2) ponderados por sus respectivos pesos w(1) y w(2), como se y el error e(k) es perpendicular a los datos (ver anexo A.3).

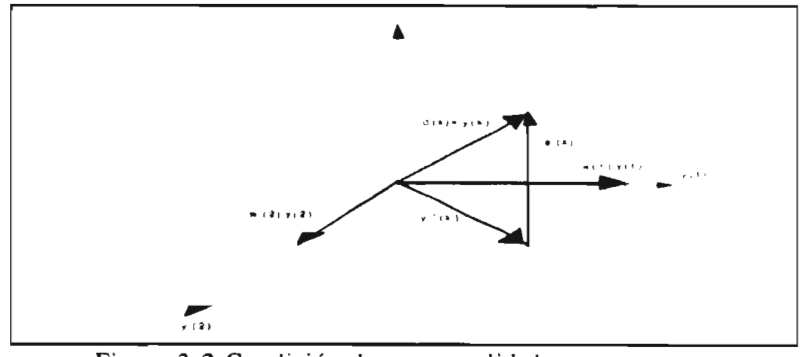


Figura 3.2 Condición de ortogonalidad

La ecuación normal es una consecuencia natural de las condiciones de ortogonalidad y de esta propiedad derivan su nombre.

Utilizando la condición de ortogonalidad:

$$E\,\{\; \mathbf{y}(k)\, e(k)\,\} = 0 \qquad 3.23$$

$$E\{\ y(k)(\ y(k) - \mathbf{w(k)}^T\mathbf{y(k)}\ )\ \} = 0 \qquad 3.24$$

$$E\{\ y(k)y(k)\ \} = E\{\mathbf{y(k)}\ \mathbf{w(k)}^T\mathbf{y(k)})\ \} \qquad 3.25$$

como $\mathbf{y(k)}^T\mathbf{w(k)} = \mathbf{w(k)}^T\mathbf{y(k)}$ 3.26

entonces

$$\mathbf{R}\ \mathbf{w}^* = \mathbf{p} \qquad 3.27$$

La solución de la ecuación normal es única si los datos $\mathbf{y}(k)$ en la estimación de $y\hat{}(k)$ son linealmente independientes, es decir, que la matriz de autocorrelación R es no singular.

III.1.3 Propiedades de la matriz de autocorrelación R

1) La matriz de autocorrelación **R** es simétrica o hermitica en el caso complejo, es decir, $\mathbf{R}=\mathbf{R}^T$, entonces sus vectores característicos son ortogonales, si **Q** es la matriz de vectores característicos de la matriz **R**, entonces [WID85] :
$\mathbf{Q}_m{}^T\mathbf{Q}_n = 0$ 3.28

2) La matriz **R** es real, y por ser simétrica, entonces todos sus valores característicos deben ser reales y mayores o iguales a cero, [WID85].

3) La matriz **Q** de vectores característicos puede ser normalizada, de tal forma que, [WID85]
$\mathbf{QQ}^T = \mathbf{I}$. 3.29

4) Para procesos estacionarios la matriz de autocorrelación R es una matriz de Toeplitz, es decir, que todos los elementos de cada diagonal son iguales, [HAY91].

5) La matriz R es siempre no negativa definida y casi siempre positiva definida, [WID85]
$\mathbf{y}^T\ \mathbf{R}\ \mathbf{y} > 0$ 3.30

6) Cuando los elementos de un vector del proceso estacionario en observación son arreglados en forma "backward", el efecto es equivalente a la transposición de la matriz de autocorrelación, [HAY91].

$$\text{si } \mathbf{y}^{bT}(k) = [y(k-p+1)\ y(k-p+2)\ \ldots\ y(k)\ ]^T \qquad 3.31$$

$$E\{\ \mathbf{y}^b(k)\ \mathbf{y}^{bT}(k)\ \} = \mathbf{R}^T \qquad 3.32$$

7) La matriz $\mathbf{R}_{p+1}$ de orden p+1 para un proceso estacionario se puede escribir en función

de la matriz $\mathbf{R}_p$ de orden p (es decir, del orden anterior), [HAY91].

III.1.4 Búsqueda de superficies de desempeño

El algoritmo de Newton y el algoritmo de los pasos descendentes envuelven el uso de estimadores tipo gradiente para indicar la dirección en la que tiende el mínimo de una superficie. Estos algoritmos son utilizados particularmente en superficies cuadráticas y en otros tipos de superficies, [WID85].

El algoritmo de *Newton* tiene una significado matemático fundamental, aunque es difícil de implementar en la práctica. Este es un algoritmo de búsqueda del gradiente causando que todos los componentes de los vectores de peso o ponderación sean cambiados en cada paso. Los cambios son siempre en la dirección del mínimo de la superficie de desempeño, [WID85].

El algoritmo de *Pasos descendentes* es un algoritmo de búsqueda de gradiente y causa que todas las componentes del vector de pesos de ponderación sean cambiadas en cada paso o ciclo de iteración. En este caso los cambios son en la dirección del gradiente negativo de la superficie de desempeño. No es necesario que esta dirección sea en la dirección del mínimo, dado que el gradiente negativo tiende hacia el mínimo sólo cuando el origen tiende a uno de los ejes principales de la superficie, [WID85],[HAY91].

III.2 ALGORITMO DE LOS PASOS DESCENDENTES ("STEEPEST DESCENT")

El algoritmo de los pasos descendentes provee un procedimiento iterativo para obtener la solución óptima $\mathbf{w}^*$ de la ecuación de W-H. Este procedimiento iterativo es la base de las técnicas actuales del procesamiento adaptable de señales. Se observará que éste conduce directamente al algoritmo adaptable de los mínimos cuadrados, [ALE86].

El algoritmo de los pasos descendentes es raramente utilizado, sin embargo, algunas aproximaciones a este se aplican frecuentemente en procesos reales, desde el punto de vista matemático se observa como una aproximación utilizando una superficie MSE en tres dimensiones, donde existe dos coordenadas que corresponden a los valores w_1 y w_2 (orden 2) y la tercera coordenada corresponde al error MSE (ver figura 3.3).

Suponiendo un vector inicial $\mathbf{w}(0)$ elegido arbitrariamente, entonces le corresponde un error $\epsilon(0)$. Existe una orientación específica de la superficie que puede ser descrita por la dirección de la derivada de la superficie en el punto. Esta derivada direccional cuantifica la razón de cambio de la superficie MSE con respecto a las coordenadas $\mathbf{w}_p$ de los ejes. Es decir, en el punto $[\mathbf{w}(0),\epsilon(0)]$ sobre la superficie existe una pendiente de la superficie, donde esta pendiente tiene los valores definidos por las derivadas direccionales $\partial\epsilon/\partial w_1$ y $\partial\epsilon/\partial w_2$, que es evaluada en el punto $\epsilon = \epsilon(0)$, y para cualquier vector $\mathbf{w}_p(k)$, $\epsilon = \epsilon(k)$, [WID85],[HAY91],[ALE86].

El gradiente de la superficie de error está definido como el vector de la derivada direccional:

$$\nabla_{w}[\varepsilon] \qquad \frac{\partial \varepsilon}{\partial w_{p}} \qquad 3.33$$

Evaluando el gradiente en el punto inicial $\epsilon = \epsilon(0)$ para el caso $p = 2$:

$$\frac{\partial \varepsilon}{\partial w_{p}}\Big|_{\varepsilon=\varepsilon(0)} = \begin{bmatrix} \frac{\partial \varepsilon}{\partial w_{1}} \\ \frac{\partial \varepsilon}{\partial w_{2}} \end{bmatrix}_{\varepsilon=\varepsilon(0)} \qquad 3.34$$

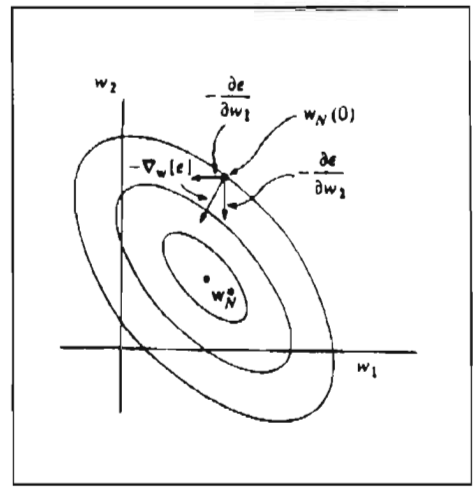


Figura 3.3 El gradiente

El gradiente apunta en la dirección de máximo cambio de la superficie en el punto [w(0),ϵ(0)], sin embargo, el punto mínimo buscado está en la dirección opuesta a este gradiente, la dirección equivalente es el de pasos descendentes (desciende de un punto de la superficie hacia el mínimo). El concepto de superficie MSE y el gradiente de la superficie MSE es para un entendimiento intuitivo de cómo los pasos descendente iterativos convergen a la solución w*. El algoritmo de pasos descendentes resuelve la ecuación normal vía un algoritmo matemático. [WID85],[HAY91].

Para una nueva posición en el tiempo discreto, se obtienen la posición en la superficie MSE al tiempo k (la iteración del algoritmo al tiempo k). Para la posición inicial [w(0),ϵ(0)], la siguiente posición es [w(1),ϵ(1)] y deben de estar relacionadas por:

$$\mathbf{w}_p(1) = \mathbf{w}_p(0) - \mu\nabla_w[\epsilon(0)] \qquad 3.35$$

donde μ es una constante, y en general para cualquier tiempo k, la ecuación 3.35 se puede escribir:

$$\mathbf{w}_p(k+1) = \mathbf{w}_p(k) - \mu\nabla_w[\epsilon(k)] \qquad 3.36$$

Esta es una expresión recursiva de los nuevos valores del vector solución **w** en función del vector pasado, mas un término de corrección que depende de las propiedades del error de la superficie en la posición del vector **w** pasado.

El mínimo MSE ocurre encontrando el gradiente de la superficie MSE como una función de la matriz de autocorrelación **R** y del vector **p**. En el algoritmo de los pasos descendentes, el vector de pesos **w** varía siempre en función de la iteración k, por lo que se denota $\mathbf{w}_p(k)$.

Como el valor de gradiente es:

$$\nabla_w[\epsilon(k)] = -2\mathbf{p}_N + 2\mathbf{R}_p\,\mathbf{w}_p(k) \qquad 3.37$$

sustituyendo en la ecuación 3.36

$$\mathbf{w}_p(k+1) = \mathbf{w}_p(k) + 2\mu\{\mathbf{p}_p - \mathbf{R}_p\,\mathbf{w}_p(k)\} \qquad 3.38$$

$$\mathbf{w}_p(k+1) = \{\mathbf{I} - 2\mu\mathbf{R}_p\}\mathbf{w}_p(k) + 2\mu\,\mathbf{p}_p \qquad 3.39$$

Si k = 0 $\qquad \mathbf{w}_p(1) = \{\mathbf{I} - 2\mu\mathbf{R}_p\}\mathbf{w}_p(0) + 2\mu\,\mathbf{p}_p \qquad 3.40$

si k = 1 $\qquad \mathbf{w}_p(2) = \{\mathbf{I} - 2\mu\mathbf{R}_p\}\mathbf{w}_p(1) + 2\mu\,\mathbf{p}_p \qquad 3.41$

continuando el proceso iterativo para toda k

$$w_p^*(k) = [I_p - 2\mu R_p]^k w_p(0) + 2\mu p_p \sum_{j=0}^{k-1} [I_p - 2\mu R_p]^j \qquad 3.42$$

La ecuación 3.42 es factible programarse para el cálculo del vector de pesos w, sin embargo, enfrenta una laboriosa tarea de computación, ya que hay que estar elevando matrices a exponentes, este cálculo y el conocimiento de $\mathbf{R}_p$ nos conduce a la solución óptima $\mathbf{w}^*$ (ecuación 3.17).

III.3 ALGORITMO DEL GRADIENTE ESTOCASTICO APROXIMADO (LMS)

Si fuera posible hacer una medida exacta del gradiente en cada iteración y con un tamaño de paso correctamente elegido μ, entonces el algoritmo de los pasos descendentes convergería a la solución óptima de Wiener. Sin embargo, una medida exacta del vector de gradiente no es posible debido a que se requiere un conocimiento previo de la matriz de correlación **R** y el vector de intercorrelación **p** entre la entrada y la respuesta deseada (ecuaciones 3.9, 3.11 y 3.42).

Esto implica que el vector gradiente se debe estimar de los datos disponibles, es decir, que los pesos del vector son actualizados con un algoritmo que adapta los datos de entrada, a este algoritmo se le llama del gradiente estocástico aproximado (LMS) o algoritmo de Widrow-Hoff, [WID85],[HAY91]. Una característica importante es su simplicidad y no requiere funciones de correlación e inversión de matrices.

Haciendo la estimación del gradiente por la sustitución del vector **p** y la matriz **R**:

$$\nabla_w[\epsilon(k)] = -2\mathbf{y}_p(k)y(k) + 2\mathbf{y}_p(k)\mathbf{y}_p^T(k)\mathbf{w}_p(k) \qquad 3.43$$

En general esta estimación es sesgada porque las etapas de estimación del vector $\mathbf{w}\hat{}(k)$ es un vector aleatorio que depende de las etapas del vector de entrada $\mathbf{y}(k)$.

Sustituyendo el gradiente en la ecuación de adaptación :

$$\mathbf{w}_p(k+1) = \mathbf{w}_p(k) - 1/2\ \mu\nabla_w[\epsilon(k)] \qquad 3.44$$

$$\mathbf{w}_p(k+1) = \mathbf{w}_p(k) + \mu[\ \mathbf{y}_p(k)d(k) - \mathbf{y}_p(k)\mathbf{y}_p^T(k)\mathbf{w}_p(k)\ ] \qquad 3.45$$

$$\mathbf{w}_p(k+1) = \mathbf{w}_p(k) + \mu\ e(k)\ \mathbf{y}_p(k) \qquad 3.46$$

donde: $$e_p(k) = y(k) - y\hat{}(k) \qquad 3.47$$

El término $\mu e(k)\mathbf{y}_p(k)$ representa una corrección aplicada a la estimación actual del vector $\mathbf{w}_p(k)$. Para el cálculo iterativo de $\mathbf{w}_p(k+1)$ se establecen las condiciones iniciales $\mathbf{w}_p(0) = \mathbf{0}$. El algoritmo LMS es de naturaleza recursiva y promedia las estimaciones durante la adaptación. Este algoritmo requiere 2p + 1 multiplicaciones complejas y 2p sumas complejas por iteración.

1) $\hat{y}(k) = \mathbf{y}^T\mathbf{w} = \mathbf{w}^T\mathbf{y}$	3.48
2) $e(k) = y(k) - \hat{y}(k)$	3.49
3) $\mathbf{w}_p(k+1) = \mathbf{w}_p(k) + \mu\, e(k)\, y_p(k)$	3.50

Tabla 3.1 Organigrama del algoritmo LMS. [WID85],[HAY91].

El parámetro μ se escoge de tal forma que la convergencia satisfaga [WID85],[HAY91],[WID96]:

1) La convergencia en la media, es decir, que el valor esperado del vector w(p) se aproximen a la solución óptima $\mathbf{w}^*$, en tanto que el número de iteraciones se aproximen al infinito.
2) Convergencia en la media cuadrada, es decir que el valor final esperado del error al cuadrado se aproxime a un valor finito cuando las iteraciones se aproximen a un valor finito.

Resumen del algoritmo LMS

p : número de "taps".
μ : parámetro de tamaño de paso.
$0 < \mu < 2$ /(potencia total de entrada o máximo valor característico de matriz R)
$\mu << 2 / \lambda_{máx}$
Condiciones iniciales: $\mathbf{w}(0) = \mathbf{0}$

datos : $\mathbf{y}(k)$ vector de entradas de p x 1 al tiempo k.
$\hat{y}(k)$ respuesta deseada al tiempo k.
calcular: $\hat{\mathbf{w}}(k+1)$ estimado al tiempo k + 1

para k = 0, 1, 2, ...

$$e_p(k) = y(k) - \hat{y}(k) \qquad 3.49$$

$$\mathbf{w}_N(k+1) = \mathbf{w}_N(k) + \mu\, e(k)\, \mathbf{y}_N(k) \qquad 3.50$$

Mientras que el algoritmo de los pasos descendentes realiza en cada iteración una medida exacta del vector gradiente, el algoritmo LMS hace una estimación ruidosa del vector gradiente, resultando así que el vector de pesos estimado $\hat{\mathbf{w}}(k)$ sólo se aproxima al valor óptimo $\mathbf{w}^*$. Después de un gran número de iteraciones produce pequeñas fluctuaciones alrededor de $\mathbf{w}^*$. En el algoritmo de los pasos descendentes la curva de aprendizaje está bien definida, obtenida por la graficación del error cuadrático medio contra el número de iteraciones, esta curva es una suma de exponenciales descendentes. En el algoritmo LMS, la curva de aprendizaje es de tipo exponencial descendente ruidosa, la amplitud del ruido se hace pequeña conforme μ se reduce, [WID85],[HAY91],[WID96].

Otras formas del algoritmo LMS

LMS signum error

Los coeficientes se actualizan aplicando la función signum al error, [TI90] :

$$\mathbf{w}_p(k+1) = \mathbf{w}_p(k) + \mu \, \mathrm{sgn}(\, e(k) \,) \, \mathbf{y}_p(k) \qquad 3.51$$

donde sgn(e(k)) es 1 si $e(k) \geq 0$ y -1 si $e(k) < 0$

LMS signum dato

Los coeficientes se actualizan aplicando la función signum a los datos de entrada, [TI90] :

$$\mathbf{w}_p(k+1) = \mathbf{w}_p(k) + \mu \, e(k) \, \mathrm{sgn}(\, \mathbf{y}_p(k) \,) \qquad 3.52$$

LMS signum dato signum error

Los coeficientes se actualizan aplicando la función signum al los datos de entrada y al error, [TI90]:

$$\mathbf{w}_p(k+1) = \mathbf{w}_p(k) + \mu \, \mathrm{sgn}(\, e(k) \,) \, \mathrm{sgn}(\, \mathbf{y}_p(k) \,) \qquad 3.53$$

Este último caso es más conciso desde el punto de vista matemático porque no requiere operaciones de multiplicación. Sin embargo, desde el punto de implementación en arquitecturas para procesamiento digital de señales, los algoritmos LMS signum tienen bajo desempeño con respecto al LMS estándar, ya que la determinación del signo rompe las operaciones de pipeline reduciendo la velocidad de procesamiento, [TI90].

En los algoritmos signum el número de operaciones es similar al LMS, es decir, 2p, a excepción que se introduce una o dos decisiones sobre datos o el error, además la convergencia es aun más ruidosa que para el caso LMS.

LMS normalizado (NLMS)

En este caso el gradiente es normalizado con la norma del vector de retrasos de y(k)

$$\mathbf{w_p(k+1)} = \mathbf{w_p(k)} + \mu \, \mathbf{x}_p(k) \, e(k)/(a + \| y(k) \|^2) \qquad 3.54$$

donde "a" es una constante que asegura la no existencia de divisiones entre cero, [HAY91].
El algoritmo NLMS requiere del orden de 3p operaciones por iteración y una división. A pesar que mejora notablemente la convergencia respecto al LMS duplica el número de operaciones.

III.4 ALGORITMO DE LOS MINIMOS CUADRADOS

Aunque por el momento no se ha hecho una comparación de los algoritmos desarrollados, la idea fundamental en el desarrollo de algoritmos es acercarse a uno que ofrezca las condiciones optimas. En filtrado adaptable estas condiciones óptimas involucran que la convergencia del error cuadrático sea lo mas rápido y también que el numero de operaciones por iteración sea mínima. En seguida se desarrollan algunas algoritmos que se acercan a estos requerimientos, durante el desarrollo se presentan algunos resultados parciales y al final del capítulo se hace una comparación entre ellos.

El algoritmo de los mínimos cuadrados (MC o LS del inglés "least square") hace una estimación sobre una ventana de datos de longitud N. Si se quiere estimar el valor de una señal $\hat{y}(k)$ a un tiempo k, partiendo de p muestras anteriores conocidas de y, entonces se tiene un error e(k) de estimación al tiempo k dado por:

$$e_p(k) = y(k) - \hat{y}(k) \qquad 3.55$$

$$e_p^f(k) = y(k) + A_p^T(k)Y_p(k-1) \qquad 3.56$$

donde: $\hat{y}(k) = - A_p^T(k)Y_p(k-1)$

$e_p^f(k)$ se le conoce como error "forward", y $A_p[k]$ es el vector de los parámetros estimados dado por:

$$A_p^T(k) = [a_1\ a_2\ a_3\ \ldots\ a_p]^T \qquad 3.57$$

$Y_p(k-1)$ es el conjunto de muestras pasadas de la señal y(k) que se escriben como un vector de p elementos:

$$Y_p^T(k-1) = [y(k-1)\ y(k-2)\ \ldots\ y(k-p)] \qquad 3.58$$

Hacer una minimización en el sentido de los mínimos cuadrados, significa minimizar la sumatoria de los errores al cuadrado respecto a los parámetros $A_p(k)$:

$$E_p(k) = \sum_{i=0}^{k} \lambda^{k-i} [e_p^k(i)]^2 \qquad k=0,1,2,\ldots,N-1 \qquad 3.59$$

donde λ se conoce como factor de olvido, que pondera las muestras pasadas en la ventana.

$$[e_p^f(k)]^2 = e_p^f(k)\, e_p^{fT}(k) = (y(k) + A_p^T Y_p(k-1))\,(y(k) + A_p^T Y_p(k-1))$$

$$= y(k)^2 + 2A_p^T(k)Y_p(k-1)y_p(k) + A_p^T(k)Y_p(k-1)Y_p^T(k-1)A_p(k) \qquad 3.60$$

definiendo:

$$E_p(k) = r_y(0) + 2A_p^T(k)\, r_p^f(k) + A_p^T(k) R_p(k-1) A_p(k) \qquad 3.61$$

donde:

$$r_y(0) = \sum_{j=0}^{k} y^2(j) \qquad 3.62$$

$$r_p^f(k) = \sum_{j=0}^{k} \lambda^{k-j} Y_p[j-1]\, y(j) \qquad 3.63$$

$$R_p(k-1) = \sum_{j=0}^{k} \lambda^{k-j} Y_p[j-1]\, Y_p^T[j-1] \qquad 3.64$$

Derivando $[E_p(k)]^2$ respecto al vector de parámetros $A_p[k]$ y haciendo la derivada igual a cero para encontrar el mínimo, se obtiene la solución, [ALC86],[HAY91],[PRN92]:

$$\mathbf{A_p(k) = - R_p^{-1}(k-1)\, r_p^f(k)} \qquad 3.65$$

donde :

$$R_p(k) = \sum_{j=0}^{k} \lambda^{k-j} Y_p[j]\, Y_p^T[j] \qquad 3.66$$

Se observa que la ecuación 3.65 es similar a la ecuación de Wiener-Hopf, sólo que el vector **r** y la matriz **R** se definen como en 3.63 y 3.66 respectivamente.

La solución de los mínimos cuadrados se realiza en una ventana de datos, es decir, la solución es global. Se observa que es necesario encontrar la matriz **R** y el vector **r** en toda la ventana de longitud N, e invertir la matriz **R** para obtener la solución, este proceso involucraría una cantidad de operaciones del orden p^3, es decir, $0(p^3)$.

III.5 ALGORITMO RECURSIVO DE LOS MINIMOS CUADRADOS (RLS)

El modelo recursivo parte de la solución global del algoritmo LS (ecuación 3.64), el algoritmo RLS para cada muestra hace una actualización de los parámetros en el tiempo k tomando en cuenta p entradas anteriores, siendo esto la base de su recursividad.

Considerando la solución:

$$\mathbf{R_p(k-1)\ A_p(k) = - r_p^f(k)} \qquad 3.65$$

Para el modelo LS fue necesario hacer una inversión de la matriz R, esto tiene como consecuencia que la programación del algoritmo sea complejo, de orden $0(p^3)$, sobre todo cuando se tiene que implementar en una arquitectura donde hay que programarlo en lenguaje ensamblador.

Desarrollando las sumatorias que conforman a $R_p[k]$ y $r_p[k]$, estos pueden quedar en función de sus valores anteriores y si el factor de olvido λ se hace uno, como caso particular y para hacer más fácil las deducciones, se tiene:

$$r_p^f(k) = r_p^f(k-1) + Y_p(k-1)y_p(k) \qquad 3.67$$

$$R_p(k) = R_p(k-1) + Y_p(k)Y_p^T(k) \qquad 3.68$$

Un desarrollo de la matriz $R_p(k-1)$ y el vector $r_p(k)$ llevan a un algoritmo recursivo, donde también hay que resolver recursivamente la inversa de la matriz R, [ALC86],[HAY91],[PRN92].

Sustituyendo las ecuaciones 3.67 y 3.68 en 3.65:

$$\{ R_p(k-2) + Y_p(k-1)Y_p^T(k-1) \}\ A_p(k) = -\{ r_p^f(k-1) + Y_p(k-1)y_p(k) \}$$

despejando $A_p(k)$

$$A_p(k) = -R_p^{-1}(k-2)r_p^f(k-1) - R_p^{-1}(k-2)Y_p(k-1)\{y(k) + A_p^T(k)\ Y_p(k-1)\} \qquad 3.69$$

definiendo el vector ***ganancia de Kalman dual*** [ALE86],[ALC86],[HAY91],[PRN92] :

$$W_p(k-1) \triangleq -R_p^{-1}(k-2)Y_p(k-1) \qquad 3.70$$
o
$$W_p(k) \triangleq -R_p^{-1}(k-1)Y_p(k)$$

y recordando la definición del error de predicción forward (ecuación 3.56), la ecuación 3.69 se escribe:

$$A_p(k) = A_p(k-1) + W_p(k-1)e_p^f(k) \qquad 3.71$$

Por otro lado, escribiendo de nuevo la ecuación 3.65 y sustituyendo 3.67

$$R_p(k-1)A_p(k) = -\{ r_p^f(k-1) + Y_p(k-1)y_p(k) \} \qquad 3.72$$

sumando y restando a la ecuación 3.72 el término $Y_p(k-1)Y_p^T(k-1)\,A_p(k-1)$, reordenando y factorizando $Y_p(k-1)$, tenemos:

$$R_p(k-1)A_p(k) = -r_p^f(k-1) - Y_p(k-1)\{ y_p(k) + A_p^T(k-1)Y_p(k-1) \}$$

$$+ Y_p(k-1)Y_p^T(k-1)A_p(k-1) \qquad 3.73$$

donde

$$-r_p^f(k-1) = R_p(k-2)A_p(k-1) \qquad 3.74$$

$$e_p^f(k) = y(k) + A_p^T(k-1)Y_p(k-1) \qquad 3.75$$

factorizando $A_p(k-1)$

$$R_p(k-1)A_p(k) = \{R_p(k-2) + Y_p(k-1)Y_p^T(k-1)\}A_p(k-1) - Y_p(k-1)\{e_p^f(k)\} \qquad 3.76$$

donde el primer término entre llaves se puede sustituir por la forma recursiva de $R_p(k-1)$ (ecuación 3.68), en ecuación 3.76, entonces:

$$R_p(k-1)A_p(k) = R_p(k-1)A_p(k-1) - Y_p(k-1)\{e_p^f(k)\} \qquad 3.77$$

para despejar el vector de coeficientes $A_p(k)$, se premultiplica por la inversa de $R_p(k-1)$

$$A_p(k) = A_p(k-1) - R_p^{-1}(k-1)Y_p(k-1)\{e_p^f(k)\} \qquad 3.78$$

definiendo la ***ganancia de Kalman***, [ALE86],[ALC86],[HAY91],[PRN92]:

$$K_p(k-1) \triangleq - R_p^{-1}(k-1)Y_p(k-1) \qquad 3.79$$

se tiene finalmente:

$$A_p(k) = A_p(k-1) + K_p(k-1)*e_p^f(k) \qquad 3.80$$

donde $e_p^f(k)$ es un error forward.

Utilizando el *lema de inversión matricial* (ver anexo A.1), que se aplica en la obtención de la inversa de una matriz particionada,

$$(A + B\,D^{-1}\,C)^{-1} = A^{-1} + A^{-1}\,B(D + CA^{-1}B)^{-1}\,C\,A^{-1} \qquad 3.81$$

Entonces podemos evaluar:

$$R_p(k) = R_p(k-1) + Y_p(k)Y_p^T(k) \qquad 3.82$$

haciendo

$$A = R_p(k-1) \qquad B = Y_p(k) \qquad D = I$$

$$C = Y_p^T(k) \qquad R_p^{-1}(k) = P_p(k)$$

entonces si

$$P_p(k) = P_p(k-1) - \{ [P_p(k-1)Y_p(k)Y_p^T(k)P_p(k-1)]/[1 + Y_p^T(k)P_p(k-1)Y_p(k)] \} \qquad 3.83$$

la ganancia de Kalman también se puede escribir:

$$K_p(k-1) = W_p(k-1) + [W_p(k-1)Y_p(k)Y_p^T(k)W_p(k-1)]/[1 - Y_p^T(k)W_p(k-1)] \qquad 3.84$$

por divisor común en la parte derecha se deduce que:

$$K_p(k-1) = W_p(k-1)/[1 + W_p^T(k-1)Y_p(k)] \qquad 3.85$$

definiendo el factor escalar de verosimilitud γ, [CAR84],[CIO84],[HAY91]

$$\gamma(k) \triangleq 1/[1 + W_p^T(k-1)Y_p(k)] \qquad 3.86$$

entonces

$$K_p(k-1) = \gamma(k)W_p(k-1) \qquad 3.87$$

De las ecuaciones 3.83 a 3.87 se observa que la matriz inversa de R se puede resolver en diferente orden de operaciones. Una amplia variedad de algoritmos pueden ser establecidos si se cambia la secuencia de cálculo de operaciones dando lugar a que su desempeño numérico en punto entero sea diferente [ALE86], por ejemplo, si tomamos las siguientes expresiones:

$$P_p(k) = P_p(k-1) - K_p(k)W_p^T(k) \qquad 3.88$$

$$P_p(k) = P_p(k-1) - W_p(k)K_p^T(k) \qquad 3.89$$

$$P_p(k) = P_p(k-1) - \gamma(k)W_p(k-1)W_p^T(k) \qquad 3.90$$

La forma de implementar P determina una robustez del algoritmo en particular. Tomando en cuenta este orden se tiene tres modelos de implementación.

III.5.1 Modelo RLS1

En este modelo se tiene una forma mas resumida de implementar las ecuaciones del algoritmo RLS, es decir, que el algoritmo implica resolver para cada muestra de entrada y(k) la secuencia de ecuaciones:

1) $e(k) = y(k) + A_p^T(k-1)*Y_p(k-1)$	3.91
2) $A_p(k) = A_p(k-1) + K_p(k-1)*e(k)$	3.92
3) $W_p(k) = - P_p(k-1)*Y_p(k)$	3.93
4) $\gamma_p(k) = 1 / (1 - W_p^T(k)*Y_p(k))$	3.94
5) $K_p(k) = \gamma_p(k)*W_p(k)$	3.95
6) $P_p(k) = P_p(k-1) - K_p(k)*W_p^T(k)$	3.96

Tabla 3.2 Organigrama del algoritmo RLS1, [K&T93].

Partiendo de las condiciones iniciales cero para los vectores $A_p(k)$, $K_p(k)$, $P_p(k)$ y $W_p(k)$ y si la matriz $P_p(k)$ corresponde al número de parámetros $A_p(k)$ a encontrar.

Para cada iteración se encuentra un nuevo juego de parámetros $A_p(k)$, la matriz $P_p(k)$ y el vector $K_p(k)$ en el tiempo actual k y se actualizan para la siguiente iteración, [ALE86],[ALC86],[HAY91],[PRN92].

III.5.2 Modelo RLS2

Para la implementación de este modelo se parte de la solución recursiva de la matriz inversa $R_p(k)$, sin efectuar los pasos intermedios del cálculo de $W_p(k)$.

1) $e(k) = y(k) + A_p^T(k-1)*Y_p(k-1)$	3.91
2) $A_p(k) = A_p(k-1) + K_p(k-1)*e(k)$	3.92
3) $P(k) = P(k-1) - [P(k-1)Y_p(k)Y_p^T(k)P(k-1)]/[1 + Y_p^T(k)P(k-1)Y_p(k)]$	3.97
4) $K_p(k) = - P_p(k)*Y_p(k)$	3.98

Tabla 3.3 Organigrama del algoritmo RLS2.

Es decir, que se utiliza directamente la ganancia de Kalman, y la solución directa de la matriz inversa R, sin acudir a reducir pasos de implementación.

III.5.3 Modelo RLS3

Para este modelo se utilizan la relación directa entre los vectores $K_p(k)$ y $W_p(k)$, es decir, se tienen las ecuaciones:

1) $e(k) = y(k) + A_p^T(k-1)*Y_p(k-1)$	3.91
2) $A_p(k) = A_p(k-1) + K_p(k-1)*e(k)$	3.92
3) $W_p(k) = - P_p(k-1)*Y_p(k)$	3.93
4) $K_p(k) = W_p(k) + [W_p(k)Y_p^T(k)W_p(k)] / [1 - Y_p^T(k)W_p(k)]$	3.99
5) $P_p(k) = P_p(k-1) - K_p(k)*W_p^T(k)$	3.100

Tabla 3.4 Organigrama del algoritmo RLS3.

En este caso se reducen más las ecuaciones comparando con el modelo RLS1.

En las figuras 3.4, 3.5 y 3.6 se muestran los resultados de la convergencia del error para los modelos anteriores. Las operaciones efectuadas por el algoritmo RLS son $4p^2 + 4p$ multiplicaciones y divisiones y $3p^2 + p - 1$ sumas y restas, es decir efectúa operaciones del orden $O(p^2)$.

Resultados del Algoritmo RLS

En la figura 3.4 se observa que existen puntos totalmente divergentes cuando la matriz R inicial es no simetrica. En cuanto a la simetria de la matriz inicial P(k) se hicieron otras pruebas y se concluye que los valores más adecuados de P(k) inicial son $0.5\mathbf{I} < P(k) < \mathbf{I}$

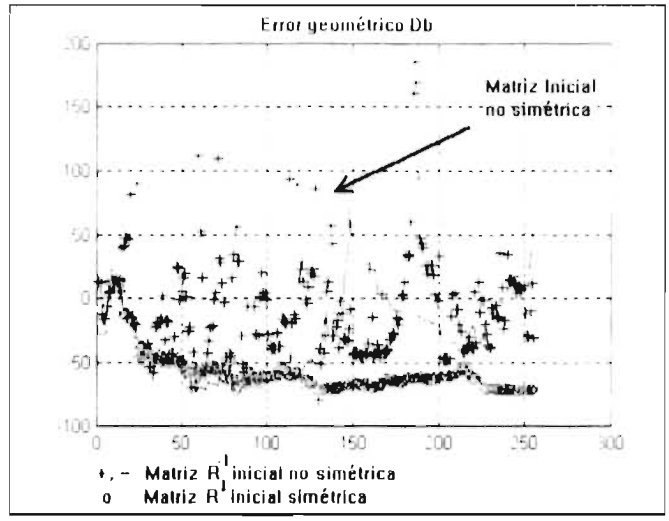


Figura 3.4 Convergencia del error cuadrático, RLS1.

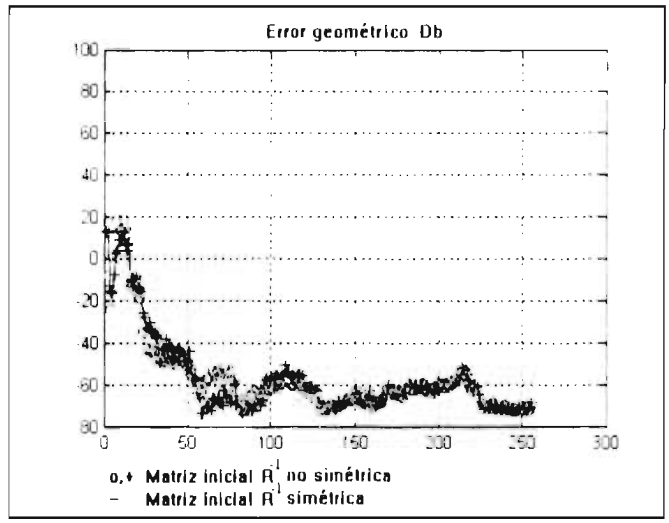


Figura 3.5 Convergencia del error cuadrático, RLS2

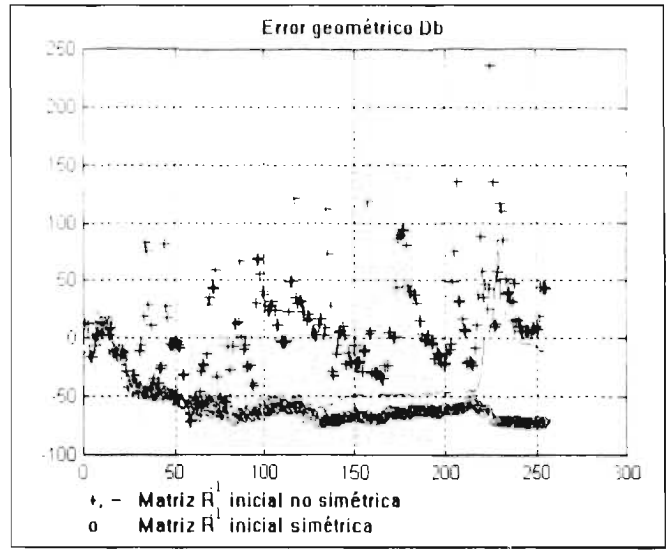


figura 3.6 Convergencia del error cuadratico, RLS3

En el *modelo RLS1*, y de la figura 3.4, se puede concluir que para obtener resultados satisfactorios, las condiciones iniciales de la matriz P debe ser estrictamente simétrica, ya que para alguna variación de 0.009 que la convierta en no simétrica, el algoritmo se hace inestable en la muestra 125, es decir que no converge. Para valores superiores a 0.0099 de antisimetría en la matriz P, el algoritmo converge cuando llega a la muestra 20, es decir, cuando se ha completado dos veces la longitud del vector de parámetros $A_p(k)$.

Para el *modelo 2* con la solución de las ecuaciones para, e(k), $A_p(k)$, $K_p(k)$ y la solución completa de la matriz recursiva $P_p(k)$, el algoritmo es de gran estabilidad, ya que para valores tan pequeños (0.0099 , 0.099 hasta 10.0) de antisimetría en la matriz inicial $P_p(k)$, el algoritmo siempre tiende a la estabilidad, para valores de 10 el algoritmo converge aproximadamente en la muestra 50, si se incrementan los valores de antisimetría (hasta 50.00) tal como se muestra en la figura 3.5 del modelo RLS2 el algoritmo tiende a converger. Estos resultados hablan de la robustez de esta implementación.

Para el *modelo RLS3* se tienen menos abreviaciones en la implementación de las ecuaciones, es decir, que la ganancia de Kalman involucra más términos en su determinación. Aunque para este caso, el algoritmo también diverge bajo condiciones iniciales no simétricas en la matriz $P_p(k)$, el algoritmo es más estable que para el algoritmo RLS1, ya que cuando el algoritmo uno diverge en la muestra 125, bajo las mismas condiciones el algoritmo RLS3 diverge hasta la muestra 220. Para otro caso (0.099) cuando el algoritmo RLS1 diverge en la muestra 5, el algoritmo RLS3 diverge en la muestra 25.

En general la robustez del algoritmo RLS depende de la forma como se implemente la recursion de la matriz inversa de $R_p(k)$. De los resultados, la forma mas robusta es cuando se implementa directamente la matriz inversa de $R_p(k)$ que se obtiene del lema de inversión matricial.

La literatura reporta que la convergencia para filtrado adaptable se logra del orden de 2p, esto es cuando la medida del error de convergencia es pequeña comparado con la señal deseada y(k), es decir, la relación señal a ruido es alta [HAY 91], es decir, que para nuestro caso deberia de darse la convergencia en la iteración numero diez. La ventaja del algoritmo recursivo es la facilidad de su implementación al no tener que resolver una matriz inversa.

Una caracteristica muy importante de los algoritmos LS y RLS es que son insensibles a las propiedades de correlación de los datos en contraste con algoritmos de los pasos descendentes tales como el LMS.

Siguiendo la evolución del desarrollo de nuestro trabajo, se tiene un tipo de algoritmo con la ventaja de converger rápidamente, y la forma de RLS2 es más robusta respecto de la divergencia del error cuadratico, sin embargo, todavia está el inconveniente que el orden de las operaciones a realizar son de razón cuadrática, lo que implica un costo computacional excesivo, es decir, que la velocidad de convergencia que se logra, puede traducir en contraproducente por la cantidad de operaciones a la hora de realizar los cálculos. Tal como se expresó anteriormente, la idea es buscar algoritmos de rápida convergencia y de un mínimo de operaciones. En la siguiente sección se trata con algoritmos que se fundamentan en el RLS, sin embargo, tienen la característica de ser mucho más rápidos.

III.6 ALGORITMOS RECURSIVOS RAPIDOS

Tal como se ha mostrado anteriormente, el algoritmo RLS mejora la velocidad de convergencia comparado con los algoritmos del gradiente estocastico como el LMS, sin embargo, aumentan el numero de operaciones a realizar. Bajo la idea de reducir la cantidad de operaciones, se han desarrollado una serie de algoritmos que ademas mantienen la velocidad de convergencia. A este tipo de algoritmos se les llama algoritmos recursivos rapidos, de los cuales se presentan algunos de ellos.

III.6.1 Algoritmo Rápido de Kalman (ARK) o RLS Rápido (FRLS)

El algoritmo rápido de Kalman (ARK) hace uso de la estimación hacia adelante y hacia atrás (forward y backward) obteniéndose una estimación de la ganancia de Kalman $K_p(k)$ al orden $p+1$. El desarrollo del algoritmo explota al maximo las relaciones que se derivan del lema de inversión matricial y es lo que permite llegar a expresiones simples para su implementacion. [CAR84],[ALC86],[HAY91],[PRN92].

Las operaciones que efectúa el algoritmo final son computacionalmente mas simples que los anteriores, ya que en ningún momento se utilizan matrices (como en el LS y el RLS), no hay productos entre matrices y vectores, y sólo se necesitan productos escalares, escalares por vectores, suma de vectores y divisiones entre escalares y es ahí donde radica la rapidez del algoritmo. Este algoritmo reduce el número de operaciones para el cálculo de los parámetros del vector $A_p(k)$ del orden $O(p^2)$ a orden $O(p)$ operaciones, haciéndolo más rápido que el algoritmo RLS, [CAR84],[ALC86],[HAY91],[PRN92]. Por esta razón se clasifica o conoce como algoritmo rápido.

Para su desarrollo se parte de la predicción forward y backward (hacia adelante y hacia atrás). Considerando la solución de la predicción forward (f) al orden p (fig. 3.7):

$$R_p(k-1)A_p(k) = -r_p^f(k) \qquad 3.101$$

con el vector de parametros $A_p(k)$ de orden p

$$A_p^T(k) = [a_1^p, a_2^p, \ldots, a_p^p] \qquad 3.102$$

la solución al orden $p+1$ sería:

$$R_{p+1}(k-1)A_{p+1}(k) = -r_{p+1}^f(k) \qquad 3.103$$

definiendo :

$$A_{p+1}^T(k) = [a_1^{p+1}, a_2^{p+1}, \ldots, a_p^{p+1}] \qquad 3.104$$

$$\ldots [k] = \sum \lambda \ldots \qquad 3.105$$

$$R_{p,1}[k-1] = \sum \lambda^{\ldots} Y_{p,1}[j-1] Y^{T}_{p,1}[j-1] \qquad 3.106$$

$y\hat{}(k) = y(k)$ predictor forward
$y\hat{}(k\ p) = y(k-p)$ predictor backward

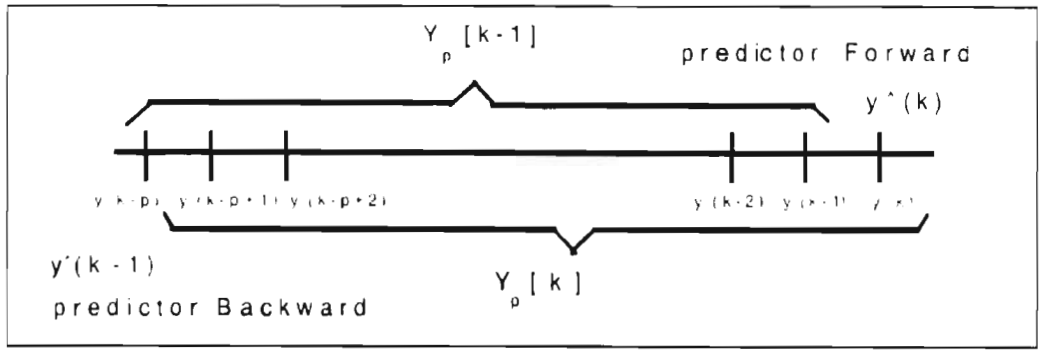


Figura 3.7 Predicción forward y backward

$$Y_p^T[k-1] = [y(k-1), y(k-2), y(k-3) \ldots y(k-p)] \qquad 3.107$$

$$Y_p^T[k] = [y(k), y(k-1), y(k-2) \ldots y(k-p+1)] \qquad 3.108$$

entonces el vector Y al orden p+1 se puede escribir en forma abreviada. [CIO84].[ALC86].[HAY91]:

$$Y_{p+1}^T[k] = [y(k), Y_p^T[k-1] = [Y_p^T[k], y(k-p)] \qquad 3.109$$

escribiendo la matriz $R_{p+1}[k-1]$ al instante k y si $l = j-1$

$$R_{p+1}[k] = \sum_{j=0}^{k-1} \lambda^{k-j-1} Y_{p+1}[j-1] Y_{p+1}^T[j-1] = \sum_{l=-1}^{k} \lambda^{k-l-1} Y_{p+1}[l] Y_{p+1}^T[l] \qquad 3.110$$

donde por condiciones iniciales $l = -1$, pero puede empezar de cero, entonces $l = 0$.
La solución general al tiempo k+1 es:

$$R_{p+1}[k]\ A_{p+1}[k+1] = -r_{p+1}[k+1] \qquad 3.111$$

sustituyendo en $R_{p+1}[k]$ la forma de $Y_{p+1}[k]$, [ALC86]:

$$R_{p+1}[k] = \sum_{j=0}^{k} \lambda^{k-j} \begin{bmatrix} y(j) \\ Y_p[j-1] \end{bmatrix} \begin{bmatrix} y(j) & Y_p^T[j-1] \end{bmatrix} \qquad 3.112$$

en forma más compacta:

$$R_{p+1}[k] = \sum_{j=0}^{k} \lambda^{k-j} \begin{bmatrix} Y_p[j] \\ y(j-1) \end{bmatrix} \begin{bmatrix} Y_p^T[j] & y(j-1) \end{bmatrix} \qquad 3.113$$

$$R_{p+1}[k] = \begin{bmatrix} r_y[0] & r_p^{fT}[k] \\ r_p^{f}[k] & R_p[k-1] \end{bmatrix} = \begin{bmatrix} R_p[k] & r_p^{b}[k] \\ r_p^{bT}[k] & r_y[p] \end{bmatrix} \qquad 3.114$$

donde se definen a $r_p^b[k]$ y $r_y[p]$ como:

$$r_p^b[k] = \sum_{j=0}^{k} \lambda^{k-j} Y_p[j]\, y(j-p) \qquad 3.115$$

$$r_{y+1}[p] = \sum_{j=0}^{k} \lambda^{k-j} y(j-p)\, Y(j-p) \qquad 3.116$$

$$R_p[k] = \sum_{j=0}^{k} \lambda^{k-j} Y_p[j]\, Y_p^T[j] \qquad 3.117$$

$r_y[0]$ se definió en ecuación 3.62. $R_p[k-1]$ en ecuación 3.64 y $r_p^f[k]$ en ecuación 3.67.

Como en la solución de $A_{p+1}[k]$ aparece la ganancia de Kalman al orden p+1:

$K_{p+1}[k] = - R_{p+1}^{-1}[k]\, Y_{p+1}[k]$

$$K_{p+1}[k] = - R_{p+1}^{-1}[k] \begin{bmatrix} y(k) \\ Y_p[k-1] \end{bmatrix} = - R_{p+1}^{-1}[k] \begin{bmatrix} Y_p[k] \\ y(k-p) \end{bmatrix} \qquad 3.118$$

sustituyendo la ecuación de $R_{p+1}[k]$ en las dos versiones de $K_{p+1}[k]$, [ALC86]:

3.119

Se debe encontrar la solución para $K_{p+1}[k]$ en función de $A_p[k]$, $K_p[k-1]$, $B_p[k]$ y $K_p[k]$. En el anexo A.5 se hacen los desarrollos necesarios para llegar al algoritmo de Kalman que se presenta en la tabla 3.5, [FAL78],[CAR84],[ALC86],[HAY91],[PRN92].

Algo importante en el desarrollo del algoritmo es ubicar correctamente el vector $M_p[k]$ y la constante $U_p[k]$ (estos se definen en la tabla 3.5), para efectuar la iteración. Los vectores A, B, K y M se inicializan con ceros. Los valores de energía del error hacia adelante se inicializan con valores alrededor de 1. El número de operaciones es del orden 8p.

De las pruebas realizadas, la inicialización de la energía forward debe tener un valor alrededor de 1 (entre 0.8 y 1.5) para alcanzar la estabilización más rápida, para valores abajo de 0.8 converge después de la iteración 25 y para valores mayores de 1.5 converge para iteraciones mayores de 30.

	Inicializar $\alpha_0(0) = \delta > 0$ y $\lambda < 1$	
1	$e_0(k) = y(k) + A_0^T(k-1)Y_0(k-1)$	3.120
2	$A_0[k] = A_0[k-1] + K_0[k-1]e_0^*(k)$	3.121
3	$e_0(k) = y(k) + A_0^T[k]Y_0[k-1]$	3.122
4	$\alpha_0(k) = \lambda\alpha_0(k-1) + e_0^*e(k)_0(k)$	3.123
5	$K_{p+1}[k] = \begin{bmatrix} M_p[k] \\ U_p[k] \end{bmatrix} = \begin{bmatrix} 0 \\ K_p[k-1] \end{bmatrix} + \begin{bmatrix} 1 \\ A_p[k] \end{bmatrix} \alpha_{p-1}^{-1}(k)\epsilon_p$	3.124
6	$e_p^b(k) = y(k-p) + B_p^T[k-1]Y_p[k]$	3.125
7	$K_p[k] = (M_p[k] - B_p[k-1]Up(k)) / (1 + e_p^b(k)U_p(k))$	3.126
8	$B_p[k] = B_p[k-1] + K_p[k]e_p^b(k)$	3.127
	Otra posibilidad es cambiando el orden de 7 y 8 y sus formas:	
7	$B_p[k] = (M_p[k]e_p^b(k) - B_p[k-1]) / (1 + e_p^b[k]U_p[k])$	3.126a
8	$K_p[k] = M_p[k-1] - B_p[k]U_p(k)$	3.127a
	Filtrado	
	$e_p(k) = d(k) + H_p^T(k-1)\, Y_p(k)$	3.128
	$H_p(k) = H_p(k-1) + K_p(k)\, e_p^T(k)$	3.129

Tabla 3.5 Organigrama del algoritmo rápido de Kalman. [LMF78],[ALC86].

III.6.2 Algoritmo Transversal Rápido (FTF)

El algoritmo transversal rapido (FTF) reemplaza algunas operaciones vectoriales del algoritmo rapido de Kalman por actualizaciones escalares, esto se logra por proyecciones en el espacio de los vectores (aproximaciones geometricas), [ALE86]. Este algoritmo reduce aproximadamente a 5p operaciones aritmeticas por iteración y ademas permite hacer una interpretacion geometrica. Sin embargo, no provee un mejoramiento en el seguimiento de la señal en todos lo casos.

El algoritmo FTF es visto como la combinacion de cuatro filtros transversales separados trabajando en conjunto, [CIO84],[ALE86],[HAY91],[PRN92]:

1) El predictor del error forward.
2) El predictor del error backward.
3) Calcula la ganancia del algoritmo RLS.
4) Filtro adaptable cuyo vector de parametros es la estimación deseada.

La combinación de estos cuatro filtros está diseñada para producir la solución exacta del problema RLS. La derivación del algoritmo FTF consiste en determinar en el tiempo la actualización de los cuatro filtros transversales y la forma en que estos interactúan.

Uno de estos filtros es la respuesta al impulso de un filtro adaptable, este filtro es equivalente al que aparece en le algoritmo LMS, los valores de sus pesos de ponderación son los parametros deseados en un sistema de identificacion. Cuando el mínimo del error cuadrático promedio no es cero, son necesarios tres filtros para completar la solución. Los tipos de cálculos son idénticos a los utilizados en el algoritmo LMS: la excitación del filtro transversal para el cálculo de la señal de error y la actualización del filtro por incremento de los valores anteriores por un escalar, [CIO84].

Definiciones para el algoritmo, de la notación del autor [CIO84]:

$$X_p(k) \triangleq [x(k)\ x(k-1)\ x(k-2)\ \ldots x(k-p+1)\ ]^t$$

Vector pinning

$$\sigma \triangleq [1\ \ 0\ \ 0\ \ \ldots\ \ 0]^t \quad \text{de px1} \qquad 3.130$$

Respuesta al impulso óptima del filtro de predicción forward que produce la mínima distancia de la muestra actual x(k) a la combinación lineal x(k-1), x(k-2) ... x(k-p)

$$A_p(k) = [1\ \ -x(k)^t K_p(k-1)] \qquad 3.131$$

Respuesta óptima al impulso del filtro de predicción backward que produce la mínima distancia de la muestra x(k-p) a la combinación lineal x(k), x(k-2) ... x(k-p+1)

$$B_p(k) = [-x(k-p)^t K_p(k)\ \ \ 1] \qquad 3.132$$

Donde $K_p(k)$ es la ganancia de Kalman, definida en ecuación 3.79.

$$C_p(k) = -\alpha_p(k)\, K_p(k) \qquad 3.133$$

en esencia $C_p(k)$ es el negativo de la ganancia de Kalman

$$W_p(k) = -d(k)^T K_p(k) \qquad 3.134$$

definendo los filtros de prediccion residual, donde los valores anteriores del filtro actuan sobre la nueva entrada del dato.

Error de prediccion forward

$$e_p(k) = e_p^f(k) \triangleq A_p(k)[x(k) \quad x(k-1) \quad \ldots \quad x(k-p)]^T = x(k)^T P_p(k-1)^{\perp} o \qquad 3.135$$

Error de predicción backward

$$r_p(k) = e_p^b(k) \triangleq B_p(k)[x(k) \quad x(k-1) \quad \ldots \quad x(k-p)]^T = x(k-p)^T P_p(k)^{\perp} o \qquad 3.136$$

$$\varepsilon_p(k) \triangleq d(k) + W_p(k)[x(k) \quad x(k-1) \quad \ldots \quad x(k-p+1)]^T = d(k)^T P_p(k)^{\perp} o \qquad 3.137$$

Se observa que cada cantidad asociada con el problema RLS puede ser interpretada como un filtro transversal.
$P_p(k)$ es una matriz de proyección de la muestra $x(k)$ sobre el espacio de datos anteriores $X_p(k)$, esta se define

$$P_p(k) = X_p(k)\,(X_p(k)X_p(k)^T)^{\#} X_p(k) \qquad 3.138$$

donde # significa la inversa generalizada.

$$\gamma_p(k) \triangleq 1 + C_p(k)[x(k) \quad x(k-1) \quad \ldots \quad x(k-p+1)]^T = o^T P_p(k)^{\perp} o \qquad 3.139$$

Lee & Mort y otros [LEE81] relaciona la constante de verosimilitud $\gamma_p(k)$ como el cuadrado del coseno del ángulo entre el subespacio $[x(k) \quad x(k-1) \quad \ldots \quad x(k-p+1)]$ y $[x(k-1) \quad x(k-1) \quad \ldots \quad x(k-p)]$. Así el filtro de $C_p(k)$ actúa sobre los datos de entrada para estimar la proximidad en términos del ángulo de los sucesivos espacios vectoriales p-dimensionales que están formados por vectores de orden p y p muestras del dato de entrada, [CIO84].

$C_p(k)$ y $\gamma_p(k)$ son instrumentadas en la actualización de otros filtros reduciendo la complejidad del algoritmo rápido de Kalman.

Energía del error forward

$$\alpha_p^b(k) \triangleq \min \| [X_{p-1}(k) \quad A_p(k)] \|^2 = x(k)^T P_p(k-1)^{\perp} x(k) \qquad 3.140$$

Energía del error backward

$$\beta_n^T(k) = \min \left[X_{p-1}(k) + B_n(k) \right] = x(k-p)^T P_n(k) - x(k-p) \qquad 3.141$$

Constante de verosimilitud

$$\gamma_n(k) = \min \left[\alpha + X_n(k)^T C_n(k) \right] = \alpha^T P_n(k) - \alpha \qquad 3.142$$

El símbolo — significa un vector o matriz normal.

Uno de los beneficios del algoritmos FTF es que reduce la cantidad de operaciones a 5p conservando la velocidad de convergencia de los algoritmos RLS. El organigrama del algoritmo FTF se muestra en la tabla 3.6.

Inicializar $\alpha_0^f(0) = \alpha_0^b(0) = \delta > 0$ y $\lambda < 1$	
$e_0^f(k) = A_0^T(k-1)X_{p-1}(k-1)$	3.143
$e_0^b(k) = \gamma_0(k-1)\, e_0^f(k)$	3.144
$\alpha_0^f(k) = \lambda\alpha_0^f(k-1) + e_0^f(k)\, e_0^f(k)$	3.145
$\gamma_{0-1}(k) = \lambda\alpha_0^b(k-1)\gamma_0(k-1)/\alpha_0^f(k-1)$	3.146
$K_{p-1}[k] = \begin{bmatrix} 0 \\ K_p[k-1] \end{bmatrix} + \frac{\lambda^{-1}(k)\,\varepsilon_p^f\, A_p[k]}{\alpha_p^f(k-1)}$	3.147
$A_p[k] = A_p(k-1) - \varepsilon_p^f(k)\begin{bmatrix} 0 \\ K_p[k-1] \end{bmatrix}$	3.148
$e_{p+1}^b(k) = \lambda\alpha_p^b(k-1)\, K_{p-1}^{p+1}(k) = r_p(k)$	3.149
$\gamma_p(k) = \gamma_{p+1}(k)/(1 - e_p^b(k)\gamma_{p-1}(n)K_{p-1}^{p+1}(k))$	3.150
$\varepsilon_p^b(k) = \gamma_p(k)\, e_p^b(k)$	3.151
$\alpha_p^b(k) = \lambda\alpha_p^b(k-1) + \varepsilon_p^b(k)\, e_p^b(k)$	3.152
$\begin{bmatrix} K_p[k] \\ 0 \end{bmatrix} = K_{p-1}(k) - K_{p+1}^{p+1}(k)\, C_p(k-1)$	3.153
$C_p[k] = C_p(k-1) - \varepsilon_p^b(k)\begin{bmatrix} K_p[k] \\ 0 \end{bmatrix}$	3.154
Filtrado	
$e_p(k) = d(k) + H_p^T(k-1)\, Y_p(k)$	3.155
$\varepsilon_p(k) = e_p(k)\gamma_p(k)$	3.156
$H_p(k) = H_p(k-1) + K_p(k)\, \varepsilon_p^T(k)$	3.157

Tabla 3.6 Organigrama del algoritmo FTF. [CIO84].

III.6.3 Algoritmo Rápido Secuencial o Kalman a Posteriori (FAEST) [CAR84]

El exito del algoritmo rapido de Kalman (ARK), es que reduce el número de operaciones del algoritmo RLS de $O(p^2)$ a $O(p)$ (donde p es el número de etapas de un filtro transversal), este algoritmo explota algunas propiedades de invarianza al corrimiento, logrando una actualizacion directa de la ganancia y se basa en la estimacion del error a priori, el error a posteriori y operaciones vectoriales para la actualizacion de los errores.

La clave del desarrollo del algoritmo secuencial a posteriori es la existencia de formulas simples para la actualizacion de parametros de la ecuación normal (ecuacion de Wiener). El algoritmo FAEST no efectua operaciones entre vectores y matrices, y reduce la cantidad de operaciones de tipo producto escalar que realiza el algoritmo ARK, [CAR84]. El algoritmo FAEST conduce a una cantidad de operaciones de $5p+8$, similar al FTF que efectua $5p$. En estos dos se efectúan operaciones escalares para aumentar el desempeño en cuanto al número de operaciones, sin embargo, el algoritmo FAEST se basa en la prediccion a posteriori.

Partiendo de las fórmulas recursivas:

$$R_p(k) = R_p(k-1) + Y_p(k)\, Y_p^{\dagger}(k) \qquad 3.158$$

$$r_p^{f}(k) = r_p^{f}(k-1) + Y_p(k)\, d_p(k) \qquad 3.159$$

$$r_p^{b}(k) = r_p^{b}(k-1) + Y_p(k-p)\, d_p(k) \qquad 3.160$$

y la ecuacion normal

$$R_p(k-1)A_p(k) = -r_p(k) \qquad 3.161$$

La reducción de la complejidad en el algoritmo FAEST es lograda a través de diferentes definiciones de la ganancia de Kalman, esta definición es una consecuencia natural de la introducción del error a posteriori. El algoritmo FAEST es una versión del ARK que utiliza la ganancia de Kalman alternativa o dual

$$W_p(k) = -R_p^{-1}(k-1)Y_p(k) \qquad 3.162$$

en vez de la ganancia de Kalman directa

$$K_p(k) = -R_p^{-1}(k)Y_p(k) \qquad 3.163$$

La ganancia de Kalman dual $W_p(k)$ de alguna forma contiene menos información en el dominio del tiempo que $K_p(k)$, los errores a priori y a posteriori incluyen igual información acerca de la respuesta deseada, pero el error a posteriori es más actualizado al modelo evolutivo con el tiempo.

Si:

$$e_p^{f}(k) = y(k) + A_p^{T}(k-1)Y_p(k-1) \qquad 3.164$$

$$\gamma_p(k) = \left(1 - Y_p(k)\, W_p^{T}(k)\right)^{-1} \qquad 3.165$$

$$0 \leq \gamma_p(k) \leq 1$$

esta condicion garantiza la inversión de la matriz $R_p(k+1)$

Se puede calcular el error a posteriori ($\varepsilon_p(k) = y(k) + A_p^t(k)Y_p(k)$) como

$$\varepsilon_p^f(k) = e_p^f(k)\,\gamma_p(k-1) \qquad 3.166$$

$$\varepsilon_p^b(k) = e_p^b(k)\,\gamma_p(k) \qquad 3.167$$

estas ecuaciones hacen una estimación del error al tiempo k antes que los parametros $A_p(k)$, $B_p(k)$ y $\Pi_p(k)$ del filtro se actualicen. Además los errores a posteriori son menores en magnitud que los errores a priori y la varianza de los errores a posteriori también es menor. Los errores a posteriori son menores en magnitud que los a priori y su varianza también es menor, es decir, que el algoritmo FAEST hace una mejor explotación de las relaciones entre el error a priori y a posteriori. En las ecuaciones 3.166 y 3.167 es evidente que en el cálculo de cada uno de estos errores a posteriori se reduce de una cantidad de operaciones p a 1.

El esquema seguido por el algoritmo es obtener la ganancia Kalman dual partiendo del valor anterior, luego incrementar el orden y después obtener esa ganancia al orden p en el tiempo actual, es decir:

$$W_p(k-1) \;==>\; W_{p+1}(k) \;==>\; W_p(k)$$

En la tabla 3.7 se muestra el organigrama del algoritmo FAEST.

Inicializacion Disponibles al tiempo k-1 $\alpha_p(k-1) = \delta > 0$; $\alpha_p^b(k-1) = \lambda^p\delta$ $\gamma_p(k-1) = 1$; $A_p(k-1) = B_p(k-1) = H_p(k-1) = W_p(k-1) = Y_p(k-1) = 0$	
Ganancia dual de Kalman (prediccion)	
$e_p(k) = x(k) + A_p^T(k-1)\, Y_p(k-1)$	3.168
$\varepsilon_p(k) = e_p(k)\, \gamma_p(k-1)$	3.169
$\alpha_p(k) = \lambda\alpha_p(k-1) + \varepsilon_p(k)\, e_p^T(k)$	3.170
$\gamma_{p+1}(k) = \gamma_p(k-1) - \varepsilon_p^T(k)\, \alpha_p^{-1}(k)\, \varepsilon_p(k)$	3.171
o en caso escalar $\gamma_{p+1}(k) = \gamma_p(k-1)\alpha_p^{-1}(k)\, \alpha_p^{-1}(k)$	
$\begin{bmatrix} A_{p+1}(k) \\ \cdots \\ \gamma_{p+1}(k) \end{bmatrix} = \begin{bmatrix} 0 \\ \cdots \\ W_p(k-1) \end{bmatrix} - \begin{bmatrix} I \\ \cdots \\ A_p(k-1) \end{bmatrix} \frac{\lambda^{-1} e_p^b(k)}{\alpha_p^b(k-1)} + \begin{bmatrix} D_p(k) \\ \cdots \\ \delta(k) \end{bmatrix}$	3.172
$A_p(k) = A_p(k-1) + W_p(k-1)\, \varepsilon_p^{rT}(k)$	3.173
$e_p^b(k) = -\lambda\alpha_p^b(k-1)\, \delta(k)$	3.174
$\gamma_p(k) = [1 + \gamma_{p-1}(k)\, e_p^{bT}(k)\delta(k)]^{-1}\, \gamma_{p-1}(k)$	3.175
$\varepsilon_p^b(k) = e_p^b(k)\, \gamma_p(k)$	3.176
$\alpha_p^b(k) = \lambda\alpha_p^b(k-1) + \varepsilon_p^b(k)\, e_p^{bT}(k)$	3.177
$W_p(k) = D_p(k) - B_p(k-1)\, \delta(k)$	3.178
$B_p(k) = B_p(k-1) + W_p(k)\, \varepsilon_p^{bT}(k)$	3.179
Filtrado	
$e_p(k) = d(k) + H_p^T(k-1)\, Y_p(k-1)$	3.180
$\varepsilon p^i(k) = e_p(k)\, \gamma_p(k)$	3.181
$H_p(k) = H_p(k-1) + W_p(k)\, \varepsilon_p^T(k)$	3.182

Tabla 3.7 Algoritmo rapido de Kalman a posteriori (FAEST). [CAR84].

III.7 ALGORITMOS ADAPTABLES CON RECURSIVIDAD EN EL ORDEN

El algoritmo rapido RLS se desarrolla con base al algoritmo de minimos cuadrados (LS) utilizando la matriz de covarianza $R_p[k]$ al orden p+1, $R_{p+1}[k]$, y el vector de la ganancia de Kalman $K_p[k]$ al orden p+1, $K_{p+1}[k]$, para este desarrollo es necesario hacer simultáneamente la prediccion hacia adelante (forward) y la prediccion hacia atras (backward).

Resumiendo las ecuaciones del algoritmo RLS:

$$R_{p+1}[k-1]A_{p+1}[k] = -r_{p+1}^{f}[k] \qquad 3.183$$

$$R_{p+1}[k]B_{p+1}[k] = -r_{p+1}^{b}[k] \qquad 3.184$$

$$R_{p+1}[k]H_{p+1}[k] = -r_{p+1}^{h}[k] \qquad 3.185$$

recordando las expresiones para r_p^f y r_p^b

$$r_p^f[k] = \sum_{j=0}^{k} \lambda^{k-j} Y_p[j-1]\, y(j) \qquad 3.186$$

$$r_p^b[k] = \sum_{j=0}^{k} \lambda^{k-j} Y_p[j]\, y(j-p) \qquad 3.187$$

los vectores r al orden p+1 y la matriz de R al orden p+1

$$r_{p+1}^f[k] = \sum_{j=0}^{k} \lambda^{k-j} Y_{p+1}[j-1]\, y(j) \qquad 3.188$$

$$r_{p+1}^b[k] = \sum_{j=0}^{k} \lambda^{k-j} Y_{p+1}[j]\, y(j-p-1) \qquad 3.189$$

$$R_{p+1}[k] = \begin{bmatrix} r_p[0] & r_p^{bT}[k] \\ r_p^f[k] & R_p[k-1] \end{bmatrix} = \begin{bmatrix} R_p[k] & r_p^b[k] \\ r_p^{bT}[k] & r_p[p] \end{bmatrix} \qquad 3.190$$

La ecuación 3.190 es similar a 3.114, es decir que sus vectores ya fueron definidos.

estos vectores tambien se pueden escribir:

$$\sum$$

$$\sum$$

de estas ecuaciones se observa que las componente de orden $p+1$ de los vectores, son iguales, es decir:

$$r_{p+1}^{f\,p+1}[k] = r_{p+1}^{b\,p+1}[k]$$

esto se puede obtener al invertir el orden de los productos dentro de las matrices.

Sustituyendo la inversa de la matriz $R_{p+1}[k]$ en forma particionada, y los vectores $r_i[k]$ para las soluciones forward y backward, entonces se obtienen los vectores solución de $A_{p+1}[k]$ y $B_{p+1}[k]$.

Se hace notar que en esta sustitución, la matriz inversa de $R_{p+1}[k]$ tiene dos formas, sin embargo, para que se obtenga coherencia en las operaciones matriciales en la solución forward, se sustituyen la matriz inversa de $R_{p+1}[k]$ donde aparecen los vectores backward, es decir, $r^b[k]$. Para la solución forward se sustituye la $R_{p+1}[k]$ que contiene los vectores $r^f[k]$. También hay que tomar en cuenta que en la *solución forward*, la matriz $R_{p+1}[k]$ está al tiempo k-1, efectuando las operaciones matriciales y sustituyendo igualdades anteriores se puede llegar a un algoritmo de recursividad en el orden que se resume a continuación (para su desarrollo ver anexo A.6).

A partir de la $K_{p+1}^f(t-1)$ y $K_{p+1}^b(t-1)$ se calculan al mismo orden $p+1$

$$\begin{bmatrix} A_p^{f\,p+1}[k] \\ A_p^{f\,p+1}[k] \end{bmatrix} = \begin{bmatrix} A_p[k] \\ 0 \end{bmatrix} + \begin{bmatrix} B_p[k-1] \\ 1 \end{bmatrix} K_{p+1}^f[k] \qquad \therefore \quad K_{p+1}^f[k] \triangleq -\frac{a_{p+1}^f[k]}{\alpha_p^f[k]}$$

3.193

$$\begin{bmatrix} B_p^{b\,p+1}[k] \\ B_p^{b\,p+1}[k] \end{bmatrix} = \begin{bmatrix} 0 \\ B_p[k-1] \end{bmatrix} + \begin{bmatrix} 1 \\ A_p[k] \end{bmatrix} K_{p+1}^b[k] \qquad \therefore \quad K_{p+1}^b[k] \triangleq -\frac{a_{p+1}^b[k]}{\alpha_p^b[k]}$$

3.194

despues de llegar a p_{max}-2 se incrementa una muestra en el tiempo, es decir, la recursividad en el tiempo.

Estas dos últimas ecuaciones permiten ligar los coeficientes del algoritmo recursivo en orden con los coeficientes de un algoritmo transversal, en sí estas ecuaciones son una recursión del tipo Levinson-Durbin. [HAY91]

El organigrama del algoritmo de recursividad en orden con predicción a priori se muestra en la tabla 3.8

Inicialización en tiempo $p = 0,1, \ldots p_{max} - 1$ $\alpha_0^f(N-1) = \alpha_0^b(N-1) = \alpha_0^f(N-2) = \delta I \; ; \; \delta > 1$ $g_0^f(N-2) = g_0(N-2) = g_{0max}(N-2) = 0$ $e_0^f(N-1) = e_0(N-1) = e_{0max}(N-1) = 0 \; , \; \gamma_0(N-2) = 1$ Inicialización en orden $e_0^f(k) = e_0^b(k) = y(k); \; e_0(k) = d(k) \; ; \; \gamma_0(k-1) = 1$ $\alpha_0^f(k) = \alpha_0^b(k) = \lambda\alpha_0^f(k-1) + y(k)y^T(k)$ recursión en el orden : $p = 0,1, \ldots, p_{max} - 2$	
$g_{p+1}^f[k-1] = \lambda g_{p+1}^f[k-2] + \gamma_p(k-2)e_p^b(k-2)e_p^{bT}(k-1)$	3.195
$K_{p+1}(k-1) = - g_{p+1}^{fT}[k-1] \; \alpha_p^f(k-1)$	3.196
$K_{p+1}^b(k-1) = - g_{p+1}^{fT}[k-1] / \alpha_p^b(k-1)$	3.197
$e_{p+1}^f(k) = e_p^f(k) + K_{p+1}^{bT} e_p^b(k-1)$	3.198
$e_{p+1}^b(k) = e_p^b(k-1) + K_{p+1}^{fT} e_p^f(k-1)$	3.199
$\alpha_{p+1}^f(k-1) = \alpha_p^f(k-1) + g_{p+1}^{aT}[k-1] K_{p+1}^b(k-1)$	3.200
$\alpha_{p+1}^b(k-1) = \alpha_p^b(k-2) + g_{p+1}^{cT}[k-1] K_{p+1}^f(k-1)$	3.201
$\gamma_{p+1}(k-1) = \gamma_p(k-1) - \gamma_p(k-1)^2 e_p^{aT}(k-1) \alpha_p^{bt}(k-1) e_p^b(k-1)$	3.202
Filtrado	
$g_{p+1}[k-1] = \lambda g_{p+1}[k-2] + \gamma_p(k-2)e_p^b(k-2)e_p^T(k-1)$	3.203
$K_{p+1}(k-1) = - \alpha_p^b(k-1)g_{p+1}(k-1)$	3.204
$e_{p+1}(k) = e_p(k) + k_{p+1}^T(k-1)e_p^b(k)$	3.205

Tabla 3.8 Organigrama del algoritmo recursivo en orden, predicción a priori, [ALC86].

III.8 CONVERGENCIA DEL ERROR CUADRATICO MEDIO EN DIFERENTES ALGORITMOS

En este punto se hace una comparación de los algoritmos tratados en este trabajo, en la figura 3.8 se muestran una comparación de la convergencia del error cuadrático. Es evidente que los algoritmos de tipo gradiente, tal como el LMS, su velocidad de convergencia es muy lenta, lo que hace concluir que para aplicaciones donde se requieren respuesta muy rapidas este algoritmo no es conveniente, cuando se hace crecer la constante μ el algoritmo tiende ha converger mas rápido sin embargo presenta problemas de estabilidad, en muchas aplicaciones lo que se hace es proponer constantes μ grandes y luego disminuirlas cuando el algoritmo ha convergido, [TOR97]. De la misma figura se observa que los algoritmos rápidos desarrollados a partir del algoritmo RLS tienen una razón de convergencia muy rápida, de ahí que algunos toman su nombre de "rápidos". Otra ventaja de los algoritmos RLS es que pueden operar adecuadamente en ambientes no estacionario. Ademas los algoritmos tipo LS y RLS son insensibles a las propiedades de correlación de los datos de entrada en contraste con los algoritmos tipo gradiente, el algoritmo LMS es muy sensible a la dispersión de los valores característicos de la matriz R. [CAR84],[WID85],[HAY91].

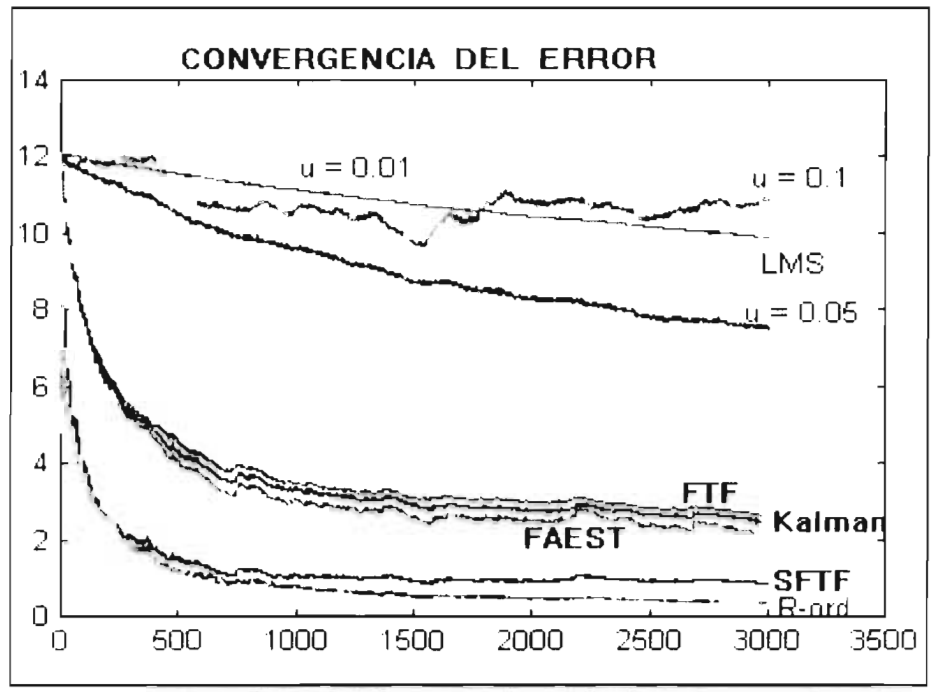


Figura 3.8 Convergencia de algoritmos

Para los algoritmos tipo RLS de la gráfica 3.8 se utilizó una $\lambda = 0.985$, $\gamma(0) = \alpha^f(0) = \alpha^b(0) = 1$.

De la figura 3.8 es obvia la razon de la busqueda de algoritmos cuya convergencia del error cuadratico sea mas rapida, sin embargo, esta no es la unica caracteristica deseable en una algoritmo adaptable. En la tabla 3.9 se hace resumen de los algoritmos desarrollados y se comparan la cantidad de operaciones teoricas por iteracion que efectúa cada algoritmo. De la tabla se observa que el algoritmo LMS es el que realiza menos operaciones por iteracion, pero ya se observo el inconveniente anterior de su velocidad de convergencia. Los algoritmos rapidos, RLS, son los que presentan menos cantidad de operaciones por iteracion.

Algoritmos	Prediccion Lineal		Filtrado	
	*	÷	*	÷
LMS	2p a 3p	1 (en NLMS)	3p	1 (NMLS)
RLS	$2p^2 + 6p$			
Rápido de Kalman (FRLS)	8p	2	10p	2
Rápido a posteriori (FAEST)	5p+8	2	7p+9	2
FTF	5p	2	7p	

Tabla 3.9 Tabla de comparación del numero de operaciones teoricas.

Hasta este punto se han expuesto, desarrollado y comparado algunos de los algoritmos de filtrado adaptable más usuales, en ellos son evidentes las ventajas de unos sobre otros, es decir, la velocidad de convergencia y el número de operaciones necesarias por iteración para su operación. Dependiendo de la aplicación, y si la velocidad de convergencia es adecuada o el número de operaciones por iteración no es muy excesiva para efectuar el procesamiento de entre muestras entonces se puede elegir uno u otro.

III.9 APLICACIONES DEL FILTRADO ADAPTABLE

En este punto se exponen las aplicaciones típicas del filtrado adaptable tales como: la predicción, el eco en las señales telefónicas, la igualación de canal y la eliminación de ruido en señales; se presenta un breve análisis de estos fenómenos y la eliminación de estos efectos.

En el capítulo II se mencionaron en general las aplicaciones del filtrado adaptable, en este capítulo se presentan resultados obtenidos de los algoritmos ya desarrollados, y se hace un análisis de cada una de las aplicaciones en particular. La presentación de estos resultados se hace con el objeto de verificar que efectivamente el filtrado adaptable es capaz de solucionar estos problemas, además se comprueba que los algoritmos desarrollados en secciones anteriores son funcionales.

Aunque sin duda alguna, la solución de estos problemas lleva consigo el procesamiento de gran cantidad de información y lo ideal es efectuar el procesamiento en tiempo real, por lo que con los avances tecnológicos de los Procesadores de Señales Digitales (DSP) se hace atractiva la implementación en arquitecturas modulares. La implementación en arquitecturas tipo DSP se presentan en el capítulo V.

III.9.1 Predicción

Tal como se mencionó en el capítulo II, la predicción es una estimación de la muestra actual de una señal deseada con base a una cantidad "p" de muestras pasadas de la señal, en este caso la señal deseada es la muestra al instante actual de la señal. Con base a la muestra actual deseada y la estimada se tiene un error de predicción que sometido a un criterio de minimización permite obtener un algoritmo adaptable para actualizar sus coeficientes.

A manera de ejemplo se tiene una señal senoidal de frecuencia variable, como se observa en la figura 3.9, tal como se discutió anteriormente, en la medida que el error converge, el algoritmo tiende a la solución óptima y la señal estimada es similar a la señal deseada casi a partir de la muestra 30. Los resultados de la figura 3.9 comprueban que si es posible efectuar el la predicción utilizando algoritmos de filtrado adaptable.

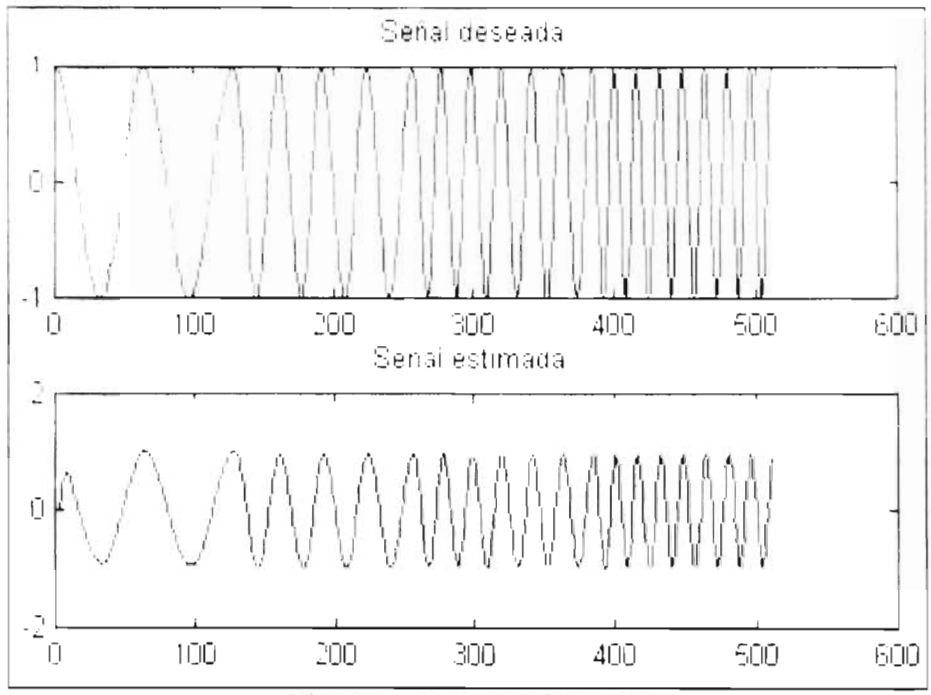


Figura 3.9 Predicción

III.9.2 El Efecto de Eco

El eco en una señal de voz, puede definirse como la repetición de un sonido reflejado por un cuerpo duro, para una onda electromagnética el efecto de eco se presenta cuando una onda electromagnética emitida por un radar y que vuelve a él después de haber sido reflejada por un obstáculo. El eco es simplemente un copia idéntica de una señal original de audio, pero retrasada una cantidad fija de tiempo; para una señal digitalizada es fácil producir eco utilizando un elemento simple de retraso. La señal retrasada es mezclada con la señal original para producir el eco. [CRO94].

El eco acústico se puede presentar por ejemplo en dos salas distantes conectadas por un sistema de altavoces y micrófonos. La señal proveniente del altavoz, filtrada por el canal acústico de la sala es re-inyectada en el micrófono, lo que trae como consecuencia que el locutor reciba un eco distante de su voz con un retardo perceptible (figura 3.10). Dado que la respuesta impulsional de la sala (el camino o ruta que sigue el eco) varía con el tiempo, se impone el uso de un filtrado adaptable. Un algoritmo adaptable hace una estimación de la respuesta impulsional asociada a la ruta del eco, minimizando la energía de una señal de error que es calculada como la diferencia entre el eco y el eco estimado. [SAN94].

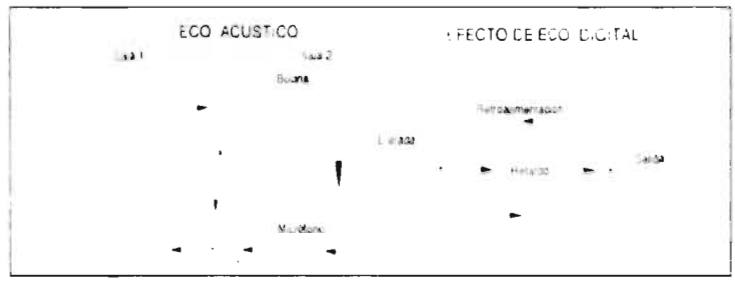


Figura 3.10, Efectos de eco

Un cancelador de eco que utiliza tecnicas de filtrado adaptable es ampliamente encontrado en aplicaciones practicas para resolver una variedad de problemas de comunicaciones.

III.9.2.1 Eco en Diferentes Sistemas

Eco en redes telefónicas

Una fuente de eco en una red telefónica puede ser una simple conexión entre dos usuarios (abonados o conversadores). Una conexión típica de una simple red telefónica consiste (figura 3.11) de dos alambres del lado de cada abonado que se conectan a un dispositivo híbrido, en medio de cada dispositivo híbrido existen cuatro cables, donde cada par de cables efectúa la transferencia en un sentido. [MHC89].

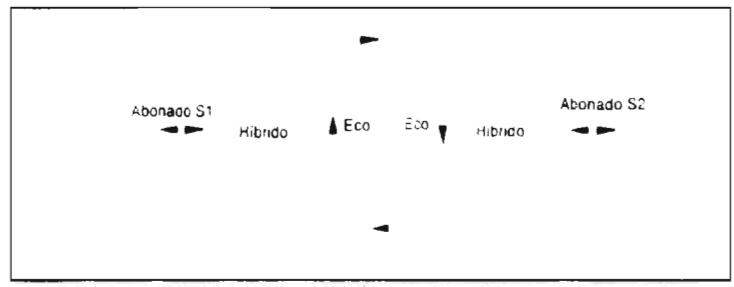


Figura 3.11 Conexión telefónica simple

Los dispositivos híbridos se encargan de convertir la conexión de dos a cuatro alambres y distribuir el sentido de la transmisión, además debe evitar el retorno de una señal a la fuente de emisión, de lo contrario esta sería una fuente de eco. El circuito de cuatro alambres puede usar cables de microondas, fibra optica y enlaces de microondas.

La función del circuito híbrido es separar las señales de transmisión y recepción para evitar un traslape entre ellas, en sí el circuito híbrido es un acoplador de impedancias de una línea bidireccional de transmisión, a lo largo de la línea de transmisión de cuatro alambres no se genera ningún eco, ya que las señales viajan en una sola dirección a través de un par de

alambres, sin embargo, en el lado del escucha tambien existe un circuito hibrido que refleja la señal al suscriptor, a esto se le llama eco lejano. El eco lejano es muy debil ya que se atenua el doble en el viaje redondo en la linea de cuatro hilos, mientras que el eco cercano es atenuado solo una vez, siendo mas facil de cancelar el eco lejano que el cercano por su debilidad comparado con la senal recibida.

El efecto del eco depende del retardo alrededor del lazo de la linea. Para retardos cortos, el eco resulta ser un deterioro insignificante, si la atenuacion es de 6 db o mas, el eco del que habla es indistinguible en el tono normal del telefono. Para conexiones de satelite, el retardo en cada uno de los cuatro alambres es al rededor de 600 ms como consecuencia de la gran altitud de los satelites. Esto significa que en el enlace el retardo total es aproximadamente de 1200 ms, lo cual es muy distorsionante para el que habla, lo que hace muy difícil establecer la conversación.

Esquema simple de eliminación de eco

Un anulador simple de eco es un interruptor que intenta abrir la ruta desde el escucha al que habla, cuando el escucha está en silencio, sin embargo, este anulador de eco efectua una supresion muy pobre ya que no puede efectuar el bloqueo en los periodos que se presenta la conversación simultánea. [HAY91],[K&T93].

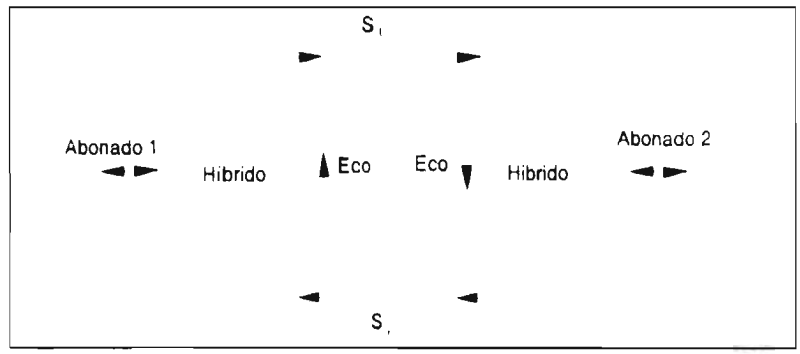


Figura 3.12 Esquema simple de eliminación de eco

Esta forma de eliminación de eco es muy pobre debido a que cuando los conversadores intenten hablar simultáneamente tendrá que haber cortes en la conversación.

Cancelación de Eco en Modems

El principal obstáculo para la transmisión de datos en full-duplex sobre una línea de transmisión de dos alambres de un teléfono es el eco. En la transmisión por modem los datos son transmitidos en forma analógica.

Clásicamente la multiplexión por división de tiempo (TDM) y la multiplexión por división de

frecuencia (FDM) son dos aproximaciones para separar la señal recibida del eco de la señal transmitida en un modem. En un sistema TDM cuando un modem está transmitiendo, el segundo modem solo está recibiendo, por lo que no existe un eco en el modem transmisor. En un sistema FDM, el ancho de banda de una línea telefónica es divida en dos para los dos modems, por ejemplo, si el primer modem utiliza la parte alta de la banda para transmitir al modem receptor, por otro lado, el receptor utiliza la parte baja de la banda para transmitir, como el eco recibido en un mismo modem está en diferente banda de frecuencia este puede separase por medio de filtrado convencional.

Sin embargo, un problema común en los sistemas TDM y FDM es que los modems en los extremos de la línea no pueden utilizar todo el tiempo disponible y el ancho de banda simultáneamente, es decir, que éstos no utilizan completamente los recursos del canal telefónico. Para lograr una eficiente transmisión full-duplex en una línea telefónica de dos alambres se puede utilizar la cancelación de eco para separar el eco y la señal recibida en el modem. [K&T93],[PAH95], figura 3.13.

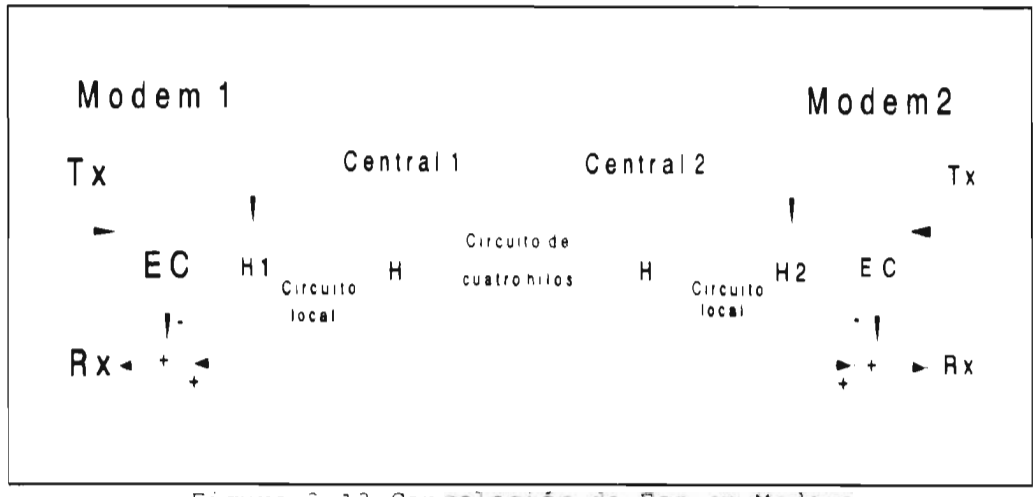


Figura 3.13 Cancelación de Eco en Modems

Eco en redes de servicios integrados ISDN

La cancelación de eco en la transmisión de datos de alta velocidad sobre una red local es llamado cancelador de eco en una red suscriptor. En una red ISDN los datos son analógicos sólo entre el suscriptor y la central, entre centrales los datos son transmitidos en forma digital, logrando así altas velocidades de transmisión, de hasta 144 Kbps (kilo bits por segundo).

La cancelación de eco para un suscriptor en una red ISDN es similar que para señales de modem, sin embargo, difiere en algunos aspectos debido a las diferencias en los requerimientos de cancelación, razón de símbolos (baudaje), constelación de señales y métodos de sincronización.

En ISDN es utilizada la cancelación no lineal de eco debido a dos razones. Primero, las no linealidades del cancelador de eco, este es más complejo que el cancelador lineal y la complejidad se incrementa exponencialmente con el posible tamaño de símbolos de datos del alfabeto, estos son más factibles para la transmisión de datos ISDN el cual tiene una simple señal de constelación. Segundo, ISDN tiene una razón más alta de datos y necesita un mejor desempeño en la cancelación (70 db) que un modem (55db). Como consecuencia es necesario el empleo de cancelador de eco no lineal para transmisión de datos ISDN, [PAH95].

Debido a que la transmisión de datos analógica es sólo entre el usuario y la central, se debe implementar un cancelador de eco en ambos extremos. En una red ISDN sólo se debe cancelar el eco cercano ya que no existe el eco lejano, requiriéndose una reducción del eco en 70 db.

III.9.2.2 Esquema para Resolver Problemas de eco

Para resolver el problema del eco en una línea telefónica se deben tomar en cuenta las siguientes consideraciones en la implementación de los algoritmos en tiempo real para la anulación de eco:

1) La respuesta impulsional del canal de eco es muy larga, del orden de 250 coeficientes para telefonía y hasta 8000 coeficientes para el caso de teleconferencia, [SAN94],[TI90].

2) Existen variaciones rápidas y difíciles de predecir en la respuesta impulsional de la ruta del eco, por lo que el algoritmo debe reaccionar rápidamente a estas variaciones, [CRO94].

3) Las señales de entrada (voz) no son estacionarias en la escala de la respuesta impulsional de la ruta del eco y presentan una importante dinámica espectral, [SAN94].

4) El retraso causado por el algoritmo de anulación debe ser mínimo para no perturbar la comunicación, [SAN94].

Cancelador de eco en una dirección

El principio de cancelador de eco en una dirección se ilustra en la figura 3.14

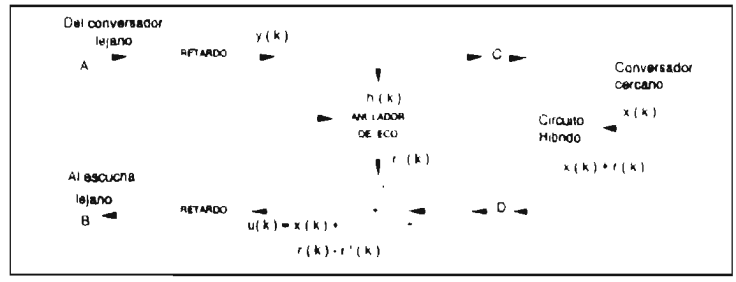


Figura 3.14 Cancelador de eco

La voz es transmitida en una dirección de los puntos A al C y en dirección opuesta de los puntos D a B. Todas la señales son muestreadas, es decir, que deben ocurrir digitalmente.

En la conversación doble, la señal de regreso al escucha contiene la voz del conversador lejano más el eco de la señal transmitida, una vez sintetizado el eco de la señal cercana y restada de las dos señales anteriores se obtiene la voz del conversador lejano.

El eco lejano o cercano puede ser visto como la salida del transmisor de señalas que pasa a través de un sistema lineal, si se puede estimar el sistema lineal, entonces la síntesis del eco puede ser precisa utilizando la señal transmitida. A la señal transmitida se le llama señal de referencia y es la que va a entrar a un filtro transversal para la estimación del eco. En el sentido que el modelo de cancelación de eco sea preciso, debe entrenarse antes para que la cancelación sea efectiva y cualquiera de los algoritmos adaptables puedan ser utilizados.

III.9.2.3 Planteamiento para Anulación de Eco

En el diseño de un cancelador de eco se involucran consideraciones tales como: la velocidad de adaptación, el efecto de las señales lejanas y cercanas.

Existen dos importantes medidas en el desempeño de la cancelación de eco:

- la velocidad de adaptación
- la precisión de la cancelación después de la adaptación.

La motivación de la cancelación de eco está dada por el desconocimiento de la función de transferencia de la ruta del eco, donde esta función de transferencia puede variar con el tiempo. En la mayoría de los casos la precisión final de la cancelación de eco es un factor crítico de diseño.

De la figura 3.14 se tienen las señales

y(k) señal que proviene del conversador lejano (señal de referencia).
$y(k)^T = [\, y(k),\ y(k-1),\ y(k-2),\ \ldots\ ,\ y(k-n+1)]$ 3.206

r(k) señal indeseable de eco.

x(k) señal que proviene del conversador cercano (deseada en el conversador lejano).

x(k) + r(k) señal del conversador cercano más el eco.

r'(k) es una réplica del eco generada por el cancelador de eco.

u(k) = x(k) + r(k) - r'(k) 3.207
señal que se transmite de regreso al conversador lejano

e(k) = r(k) - r'(k) 3.208
es el error en la cancelación de eco, idealmente debe tender a cero.

La respuesta al impulso del canal de eco h_k :
$\mathbf{h}^T = [\, h(0)\ ,\ h(1),\ \ldots\ h(n-1)]$ 3.209

definiendo los coeficientes de filtro transversal cancelador de eco:

$\mathbf{a}^T = [\, a(0)\ ,\ a(1),\ \ldots\ a(n-1)]$ 3.210

$$u(k) = x(k) + \sum_{m=0}^{\infty} h(m)\, y(k-m) - \sum_{m=0}^{n-1} a(m)\, y(k-m) \qquad 3.211$$

$$u(k) = (\mathbf{h} - \mathbf{a})\, y(k) + v(k) \qquad 3.212$$

donde:

$$v(k) = x(k) + \sum_{m=n}^{\infty} h(k)\, y(k-m) \qquad 3.213$$

que es el eco residual no cancelable, correspondiendo a los retardos que exceden el número de coeficientes en el filtro transversal, más el ruido y la señal de conversador cercano.

Debe notarse que los dos términos de e(k) son correlacionados cuando el muestreo de la señal

de referencia es correlacionada.

Esperanza del error cuadrático

Considerando el caso escalar:

$$e(k)^2 = \{ (h - a)\, y(k) + v(k) \} \{ (h - a)\, y(k) + v(k) \}$$

$$= (hy(k))^2 + (ay(k))^2 + (v(k))^2 - 2ahy(k)^2 + 2hy(k)v(k) - 2ay(k)v(k) \qquad 3.214$$

derivando parcialmente respecto de los parámetros **a** e igualando a cero:

$$\partial e(k)^2 / \partial \mathbf{a} = 0$$

$$\mathbf{a}y(k)^2 - \mathbf{h}y(k)^2 - y(k)v(k) = 0 \qquad 3.215$$

Obteniendo la esperanza matemática para el caso escalar:

$$\mathbf{a}E\{ y(k)^2 \} - \mathbf{h}E\{ y(k)^2 \} - E\{ y(k)v(k) \} = 0 \qquad 3.216$$

$$\mathbf{a} = \mathbf{h} + E\{ y(k)^2 \}^{-1} \; E\{ y(k)v(k) \} \qquad 3.217$$

Para propósitos de análisis es necesario asumir que la señal de referencia y(k) y la señal v(k) son conjuntamente estacionarias, definiendo para el caso vectorial:

$$\mathbf{p}(i) = E[v(i)y(i)] \qquad 3.218$$

$$\mathbf{\Phi} = \mathbf{R}(j) = E[y(i)y(i+j)] \qquad 3.219$$

donde:

$$\phi(i) = E[y(i)y^T(i+j)] = \begin{bmatrix} R_0 & R_1 & \cdots & R_{n-1} \\ R_1 & R_0 & \cdots & \cdots \\ \cdot & \cdot & \cdot & \cdot \\ R_{n-1} & \cdot & \cdot & R_0 \end{bmatrix} \qquad 3.220$$

Minimizando el error cuadrático medio $E[e(k)^2]$ respecto de los parámetros a(k) se puede obtener:

$$\mathbf{a}_{\text{óptimo}} = \mathbf{h} + \mathbf{\Phi}^{-1}\mathbf{p} \qquad 3.221$$

Asumiendo que la señal de referencia es mutuamente no correlacionada y x(k) no está

correlacionada con la muestra de referencia:

$$\Rightarrow \mathbf{p} = 0 \qquad y \qquad \Phi = R_0 \mathbf{I} \tag{3.222}$$

donde **I** es la matriz identidad.

Entonces los coeficientes del filtro óptimo son:

$$\mathbf{a}_{optimo} = \mathbf{h} \tag{3.223}$$

El bloque cancelador de eco genera un réplica del eco al aplicarse una señal de referencia a un filtro transversal.

Si la función de transferencia del filtro transversal es idéntica a la ruta del eco, entonces la réplica será idéntica al eco, con lo que se logra la cancelación. Dado que la función de transferencia de la ruta del eco entre los puertos C y D normalmente no es conocida previamente, el cancelador de eco adapta los coeficientes del filtro transversal.

Para reducir el error, el algoritmo de adaptación infiere la cancelación de error e(k) (cuando ninguna señal del conversador cercano está presente) la apropiada corrección de los coeficientes del filtro transversal.

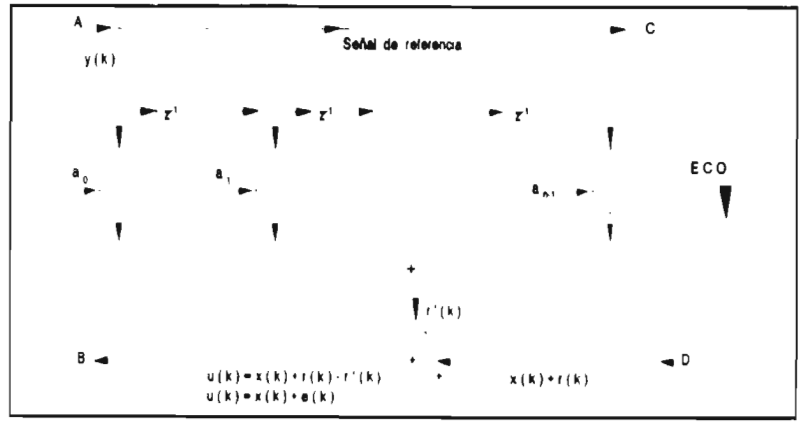


Figura 3.15 Estimador de eco utilizando un filtro transversal

El número de etapas del filtro está determinado por la duración de la respuesta al impulso de la ruta del eco entre C y D. Típicamente el tiempo en el cual la respuesta al impulso es significante es de 2 a 4 ms. Esto corresponde de 16 a 32 taps con 8 Khz de frecuencia de muestreo. Sin embargo, por la porción del circuito de cuatro alambres entre la localización del cancelador de eco y el circuito híbrido, la respuesta no tiende a cero, pero es retardada. El número de taps N,

debe ser lo suficiente grande para ajustar el retardo. Con N = 128 se pueden acomodar retardos de hasta 16 ms (o alrededor de 1,200 millas de línea), [TI90],[MES82].

Cancelador de eco en dos direcciones

En la realidad es necesario la cancelación de eco en ambas terminales, para este propósito se utilizan dos filtros adaptables, uno en cada extremo de la línea:

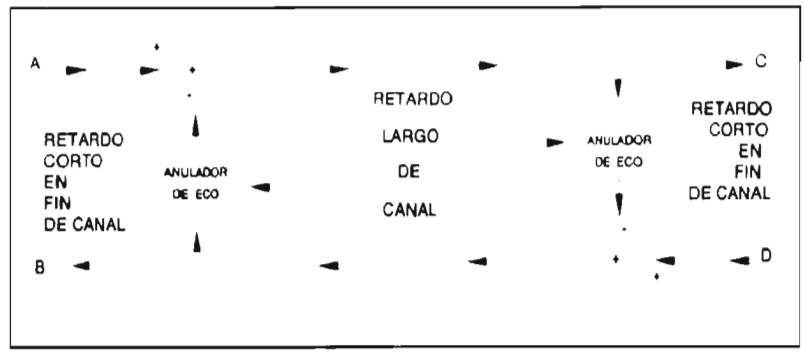


Figura 3.16 Cancelador de eco en dos direcciones

El cancelador digital de eco puede ser aplicado en una variedad de configuraciones de equipo de transmisión. En un teléfono, una parte de la señal generada en el micrófono se regresa a la bocina. Este tipo de eco puede ser deseable, ya que algunos experimentos han mostrado que si el suscriptor no escucha su propia voz asume que del otro lado de la línea no es escuchado, [LON90].

El eco lejano puede ser trasladado en frecuencia cuando este pasa a través de un convertidor de frecuencia (modulador o demodulador) sobre el circuito de cuatro alambres, esto ocasiona un corrimiento de la frecuencia del eco lejano causando un deterioro no lineal.

Para el eco lejano y cercano sin corrimiento en frecuencia pueden ser vistos como una señal transmitida que pasa a través de un sistema lineal. Si se puede estimar el sistema lineal, entonces se puede sintetizar este eco utilizando la señal transmitida y estimando un sistema de parámetros. En el sentido de remover el eco de la señal recibida se le extrae el eco sintetizado.

Algoritmos de cancelación de eco.

Existe una gran gama de algoritmos adaptables que permite resolver este problema, sin embargo, el que se utiliza con más frecuencia es el LMS. El problema se plantea, para una señal y(k) de entrada y una salida deseada x(k) de un sistema se busca identificar la respuesta impulsional finita h_p por medio del cálculo de un error de modelado

$$e_p(k) = y_p^T(k)h_p(k) - y_p^T(k)a_p(k) \qquad 3.224$$

del cual se minimiza la esperanza matemática del error cuadrático $E(e_p^2)$, donde y_p es el vector que contiene las últimas p entradas al sistema.

$$h_{p+1}(k) = h_{p,N-1}(k) + \mu x_p(k)e_p(k) \qquad 3.225$$

esto demanda una carga de cálculo de aproximadamente 2p sumas y multiplicaciones, si el paso de adaptación μ es elegido convenientemente ($\mu \leq 2 / E(y_p^T(k)y_p(k))$) el algoritmo converge hacia una solución óptima con un error residual proporcional a μ.

Filtro transversal adaptable

La señal de eco reflejada r(k) al tiempo k puede ser escrita como la convolución de las señal lejana de referencia y(k) y la representación discreta de h(k) de la respuesta al impulso de la ruta del eco entre los puntos C y D (figura 3.14 y 3.15)

$$r(k) = \sum_{i=0}^{p-1} h(i)y(k-i) \qquad 3.226$$

Asumiendo linealidad y duración finita "p" de la ruta del eco, un cancelador de eco con "p" etapas, adapta "p" coeficientes a(k) del filtro transversal para producir una réplica del eco, r'(k), definido como:

$$r'(i) = \sum_{k=0}^{p-1} A_k y(i-k) \qquad 3.227$$

claramente si a(k) = h(k) para k= 0,1,.., p-1, entonces r(k) = r'(k)
que es buscado para la cancelación del eco.

Sin embargo, en general la respuesta al impulso de la ruta del eco h(k) es desconocida y puede variar lentamente con el tiempo, entonces, se requiere de un algoritmo de lazo cerrado para la adaptación de los coeficientes, que minimice el promedio o el error cuadrático medio (MSE) entre el eco y su réplica.

De la figura 3.14 se observa que la señal u(k) proveniente del conversador cercano está comprendida de x(k) y el error r(k)-r'(k), la cual no está correlacionada con y(k), entonces la esperanza de:

$$u^2(k) = x^2(k) + 2x(k)e(k) + e^2(k) \qquad 3.228$$

esta dada por:

$$E\{u^2(k)\} = E\{x^2(k)\} + E\{e^2(k)\} \qquad 3.229$$

donde el término $E\{e^2(k)\}$ debe de minimizarse para que la ecuación anterior se minimice.

Si no existe voz cercana (x(k)=0) entonces el mínimo es logrado por el ajuste de los coeficientes a(k) a lo largo de la dirección negativa del gradiente de $E\{e^2(k)\}$ en cada paso que se actualice la ecuación:

$$A_p(k+1) = A_p(k) - \mu \frac{\partial E(e^2(k))}{\partial a_p(k)} \qquad 3.230$$

sustituyendo 3.226 y 3.227 en 3.228, y de 3.230 la ecuación actualizada sería:

$$A_p(k+1) = A_p(k) + 2\mu E\{e(k)y(k-i)\} \qquad 3.231$$

donde μ es paso de adaptación del algoritmo.

En los capítulos 2 y 3 se estudió fundamentalmente los algoritmos de filtrado adaptable que funcionan bajo estructuras de filtros transversales, es decir, que cualquiera de los algoritmos, planteados y analizado su funcionamiento, se adecúa a este tipo de estructura, por lo tanto en las aplicaciones planteadas no es necesario entrar en detalles para cada algoritmo, sino sólo tener presente qué señal está asociada con la entrada y cuál con las señal deseada.

III.9.2.4 Pruebas de Cancelación de Eco

Bajo el esquema del anulador de eco en la figura 3.17, se muestran las pruebas realizadas para una señal y(k), su eco que se agrega a una señal senoidal x(k). Se tiene la eliminación de eco utilizando el algoritmo LMS con 100 coeficientes y del tipo FRLS con p=10.
En la figura 3.17 se observan las diferentes señales en cada punto.

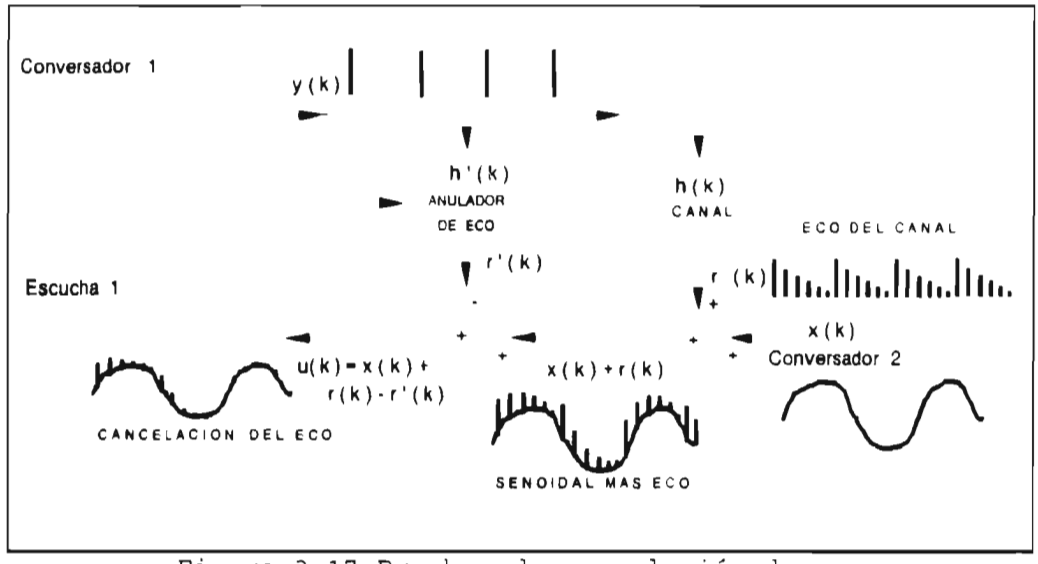


Figura 3.17 Pruebas de cancelación de eco

En la figura 3.17, el conversador 1 emita una señal de pulsos, el conversador 2 emite una señal senoidal x(k) que se va a contaminar con el eco de la señal y(k), el error será la resta de ésta señal contaminada menos el eco estimado por el sistema adaptable.

En la figura 3.18 se muestra una secuencia de señales utilizadas para la cancelación de eco, en la última señal se observa como se ha logrado eliminar el eco ocasionado por un tren de pulsos.

III.9.2.5 Resultados de Cancelación de Eco

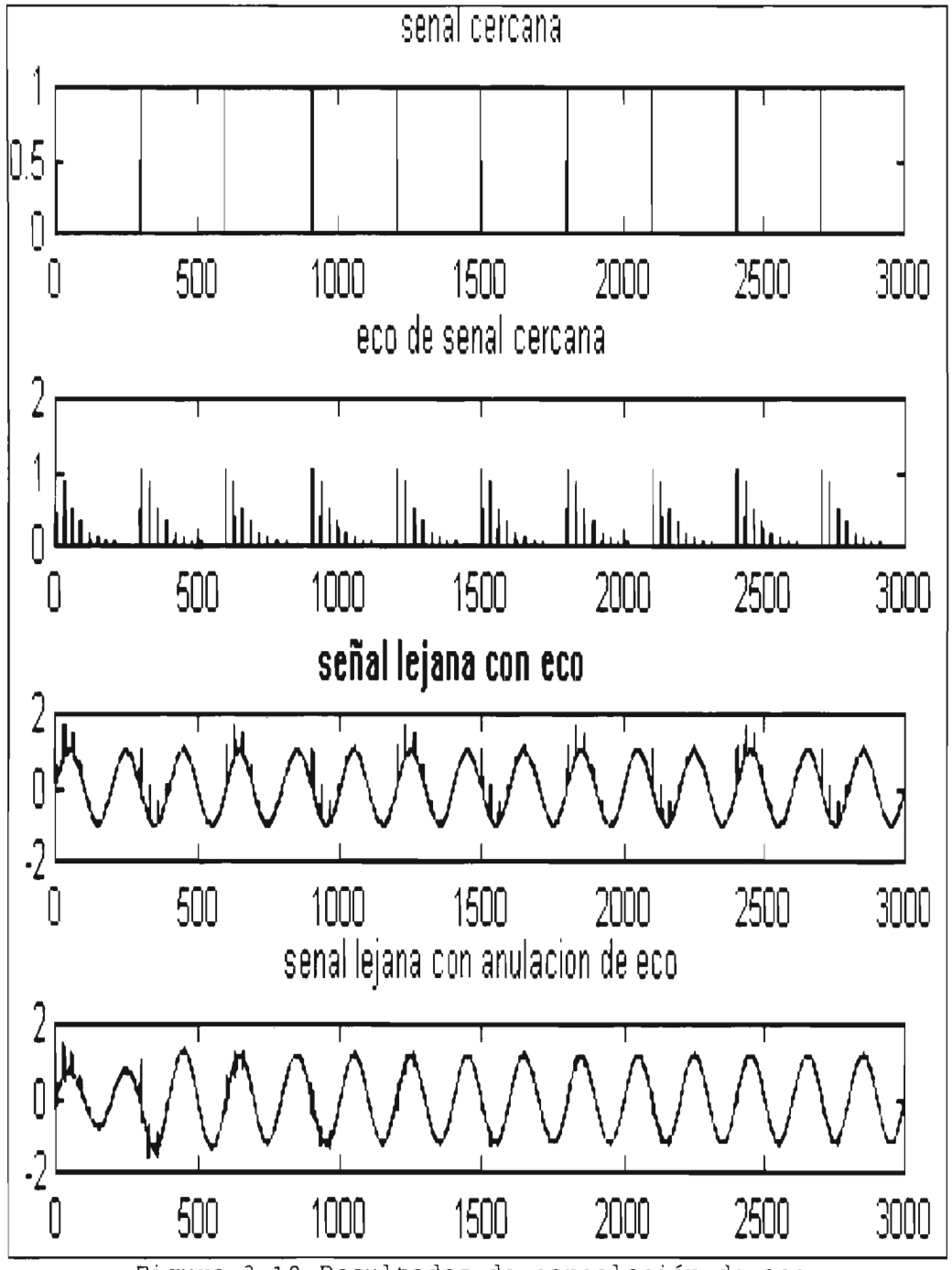


Figura 3.18 Resultados de cancelación de eco.

III.9.3 Igualación de Canal

Los métodos de los mínimos cuadrados (LS) son ampliamente utilizados en problemas donde dada una señal de salida de un sistema cuyas características son desconocidas se pretende determinar la señal de entrada al sistema. En un canal telefónico la señal de salida presenta una distorsión respecto de la señal original a causa de la interferencia intersímbolos (II) de los datos. Esta interferencia causa errores cuando se intenta recuperar los datos.

Para solucionar este problema es necesario diseñar un sistema correctivo en cascada con el sistema original (canal), para producir una réplica de la señal originalmente transmitida. En comunicaciones, este sistema correctivo es llamado *igualador de canal*.

En un sentido general para sistemas lineales, al sistema correctivo se le puede llamar *sistema inverso*, porque debe de tener una respuesta en frecuencia básicamente recíproca a la respuesta en frecuencia del sistema que causó la distorsión. Es decir, si el sistema original produce una salida y1(k) que es la convolución de y(k) con la respuesta al impulso del sistema h(k), entonces, la operación de tomar y1(k) y obtener y^(k) se le llama *deconvolución*, [PRN92],[K&T93].

La deconvolución es una operación lineal que remueve el efecto de una convolución previa sobre una secuencia de datos, [HAY91]. El modelo inverso de un sistema que tiene una función de transferencia desconocida, es un sistema cuya función de transferencia es el recíproco más aproximado de la función de transferencia desconocida. Algunas respuesta de modelos inversos contienen retardos que son incorporados deliberadamente para mejorar su comportamiento.

Un canal dispersivo, es cuando las señales de diferentes frecuencias viajan a diferentes velocidades o diferentes grupos de retardos. En un canal dispersivo, se puede utilizar un filtro adaptable en la recepción para igualar el canal, es decir, proveer una respuesta en frecuencia y fase para las señales en la banda de paso de la respuesta inversa del canal y así compensar la dispersión. En el receptor se obtiene la señal transmitida convolucionada con la respuesta al impulso del canal, el filtro igualador deconvoluciona las características del canal y reestablece la señal original.

La inclusión de *retardos* Δ generalmente permite valores mucho menores del error cuadrático medio mínimo y causa la convergencia de la respuesta adaptable al impulso. Una elección empírica es fijar el retardo Δ como la mitad de la longitud del filtro adaptable, [WID85],[WID96].

III.9.3.1 Igualación adaptable de canales telefónicos

En canales telefónicos la interferencia causada entre muestras sucesivas (interferencia entre símbolos) complica la transmisión y recepción entre símbolos.

El ruido en una línea telefónica es generalmente bajo y típicamente, el principal problema es la interferencia intersimbolos. Para afrontar este problema se requiere idealmente en el receptor igualar la función de transferencia que es esencialmente la inversa de la función de transferencia del canal.

Los datos binarios son generalmente codificados utilizando pulsos positivos para los UNOS y pulsos negativos para los CEROS. El canal produce en el receptor una señal analógica la cual es la convolución de los datos (pulsos) con la respuesta al impulso del canal.

Para un canal con respuesta en frecuencia ideal ($\pm$ f_c) y que los datos originales son transmitidos exactamente a la frecuencia de Nyquist, cuando una muestra es recibida a esta razón y fase, el receptor muestrea con un reloj sincronizado al reloj retardado del transmisor, entonces se obtiene una muestra de pulso único, esto se debe a que no existe interferencia de los pulsos vecinos, debido a que la respuesta al impulso de un canal ideal sus cruces por cero están uniformemente espaciados a intervalos de $1/(2f_c)$. Eligiendo la correcta frecuencia y fase de muestreo en el receptor permite que una de las muestras del canal salga en el pico de cada pulso individual de la función sinc (respuesta impulsional del canal ideal) mientras que los pulsos sinc vecinos pasan por cero [WID85]. Sin embargo, si se tiene la necesidad de transmitir señales a través del canal a tasa mayores que las permitidas por el ancho de banda, se impone la necesidad de igualar el canal con el fin de restaurar las señales tal como fueron transmitidas con el fin de efectuar la detección en el receptor, [TOR97].

En canales reales, las características difieren de uno a otro y además cambian en el tiempo. La respuesta al impulso del canal real se parece a la respuesta al impulso de un canal ideal (es decir un función sinc), pero sus cruces por cero no son uniformes en el tiempo. Esto ocasiona errores de interferencia intersimbolos. Debido a que la respuesta al impulso de un canal real tiende a caer en el tiempo en ambas direcciones, se puede reducir el efecto de intersímbolos por una señalización lenta, mucho más lenta que la razón de Nyquist, conduciendo a un adecuado espaciamiento entre respuestas al impulso. Este tipo de aproximación fue utilizado antes del advenimiento de la igualación de canales adaptable, [HAY91].

Las grandes velocidades y la confianza en las comunicaciones digitales han sido posibles gracias a la utilización de la igualación adaptable. Un igualador adaptable forma una función de transferencia inversa para el canal dentro de la banda de paso, fuera de esta banda la ganancia es pequeña o cero, dentro de la banda de paso la igualación produce una ganancia en magnitud plana y cero fuera de la banda de paso, la igualación también debe producir la distorsión en fase del canal.

En el proceso adaptable, un filtro transversal adaptable se utiliza en la configuración del modelo inverso.

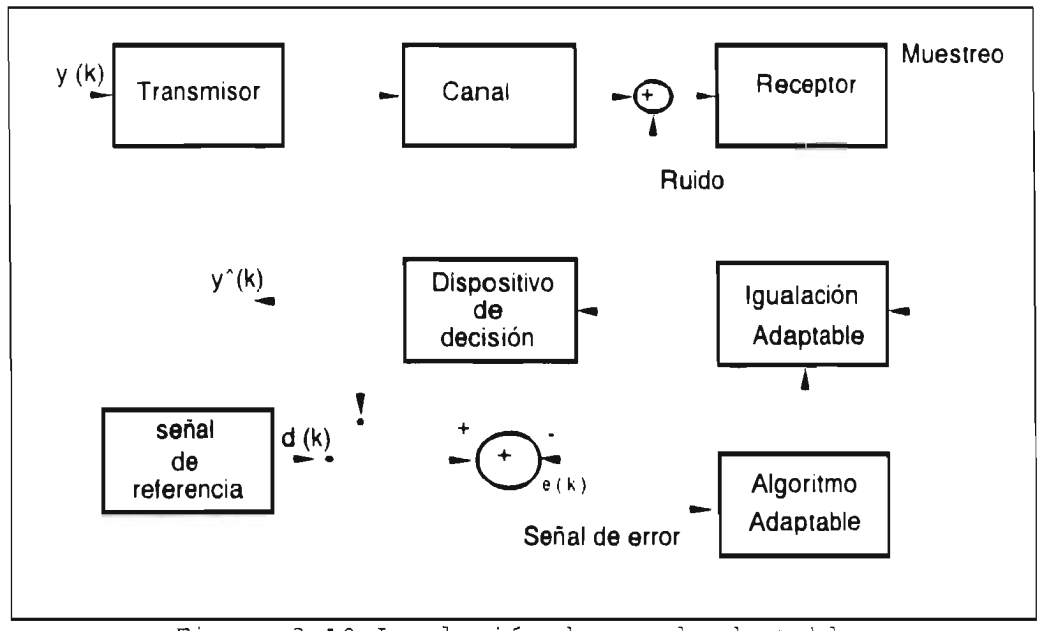


Figura 3.19 Igualación de canal adaptable.

De la figura 3.19, si la función de transferencia del canal es H(z), entonces claramente la función de transferencia del igualador está dada por:

$$G(z) = H^{-1}(z) \qquad 3.232$$

Esta función inversa debe ser inestable en los lugares donde H(z) posee ceros fuera del círculo unitario en el plano z (canales de fase no mínima). Un camino para obtener esta aproximación es redondear el problema aproximando la función inversa por un filtro finito, todo cero, es decir, desarrollar la división fraccionaria y truncar la serie. El problema de formar un igualador de fase no mínima es resuelto introduciendo un retardo finito en el igualador. [N&K93].

Por ejemplo, asumiendo la función de transferencia del canal:

$$H(z) = 0.5 + z^{-1} \qquad 3.233$$

la inversa del canal:

$$G(z) = z / (1 + 0.5z) \qquad 3.234$$

Esta función tiene un polo en z = -2, por lo que no se puede implementar el filtro inverso, sin embargo, efectuando la división se obtiene una serie:

$$G(z) = z - 0.5z^{-2} + 0.25z^{-3} - \ldots \qquad 3.235$$

La secuencia resultante es convergente (en el tiempo positivo) y puede truncarse después de un número finito de muestras, las cuales pueden ser determinadas por la interferencia intersimbólica (ISI) permitida por el receptor. Tomando tres muestras (truncando después del término z^{-3}) nos lleva a un filtro no causal que puede ser realizable introduciendo un retardo de tres muestras, es decir, multiplicando a G(z) por z^{-3}, entonces se tiene el igualador:

$$G_1(z) = Z^{-3}G(Z) = 0.25 - 0.5z^{-1} + z^{-2} \qquad 3.236$$

Esto da una solución simple cuando la función de transferencia del canal es conocida. Sin embargo, en la práctica H(z) no es conocida y puede variar en el tiempo.

III.9.3.2 Implementación de un igualador de canal

Si la función de transferencia del canal es H(z), entonces la función de transferencia del igualador está dada por:

$$G(z) = H^{-1}(z)$$

entonces,

$$H(z)\,G(Z) = 1 \qquad 3.237$$

en el tiempo discreto

$$y_1(k) = y(k) * h(k) \quad (* \text{ operación convolución}) \qquad 3.238$$

$$\hat{y}(k) = y_1(k) * g(k) = y(k) * h(k) * g(k) \qquad 3.239$$

si se quiere que $\hat{y}(k) = y(k)$

entonces, $h(k) * g(k) = \delta(k) \qquad 3.240$

con lo que se logra:

$$\hat{y}(k) = y(k)*\delta(k) = y(k) \qquad 3.241$$

es decir la señal transmitida y deseada en el receptor.

Con el objeto de probar los algoritmos en la igualación de canal, para la función de transferencia de un canal dada por:

$$H(z) = 0.251 + 0.935z^{-1} + 0.251z^{-2} \qquad 3.242$$

$$h(k) = \{ 0.251,\ 0.935,\ 0.251 \} \qquad 3.243$$

y bajo el diagrama de bloque presentado en la figura 3.20, se efectúa la simulación con el algoritmos ARK para encontrar g(k).

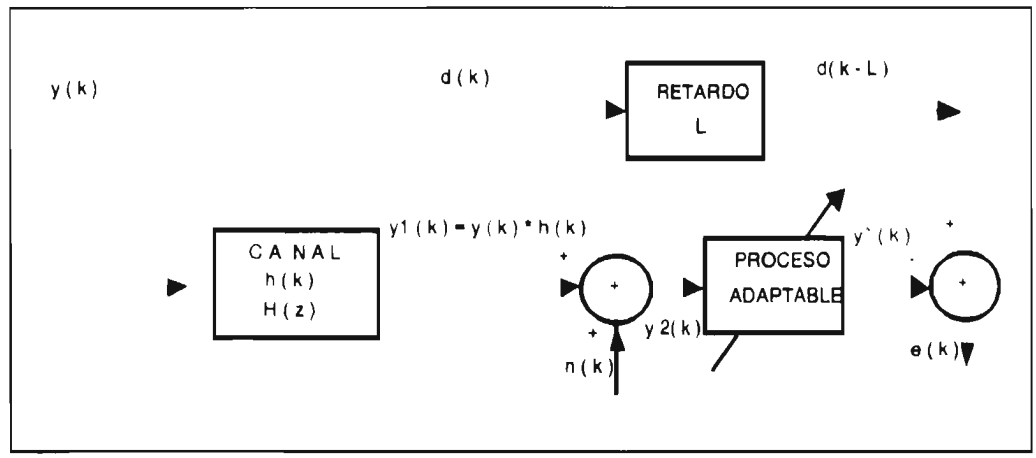


Figura 3.20 Esquema de igualación de canal.

En la figura 3.21 se muestra la respuesta al impulso h(k) del canal y la respuesta al impulso g(k) de un filtro inverso en el proceso adaptable de la figura 3.20. En la figura 3.22 se muestra el resultado de la convolución h(k)*g(k) (ecuación 3.240), que idealmente debe de ser un impulso centrado en el número de retardo utilizado.

Finalmente en la figura 3.23 se presentan los resultados de graficar H(z) y G(z), como se observa son casi inversas (ver simetría respecto al eje horizontal) a excepción del la parte inicial, que es cuando el algoritmo inicia a converger, es decir que $G(Z) = H^{-1}(z)$.

III.9.3.3 Resultados de la igualación de canal

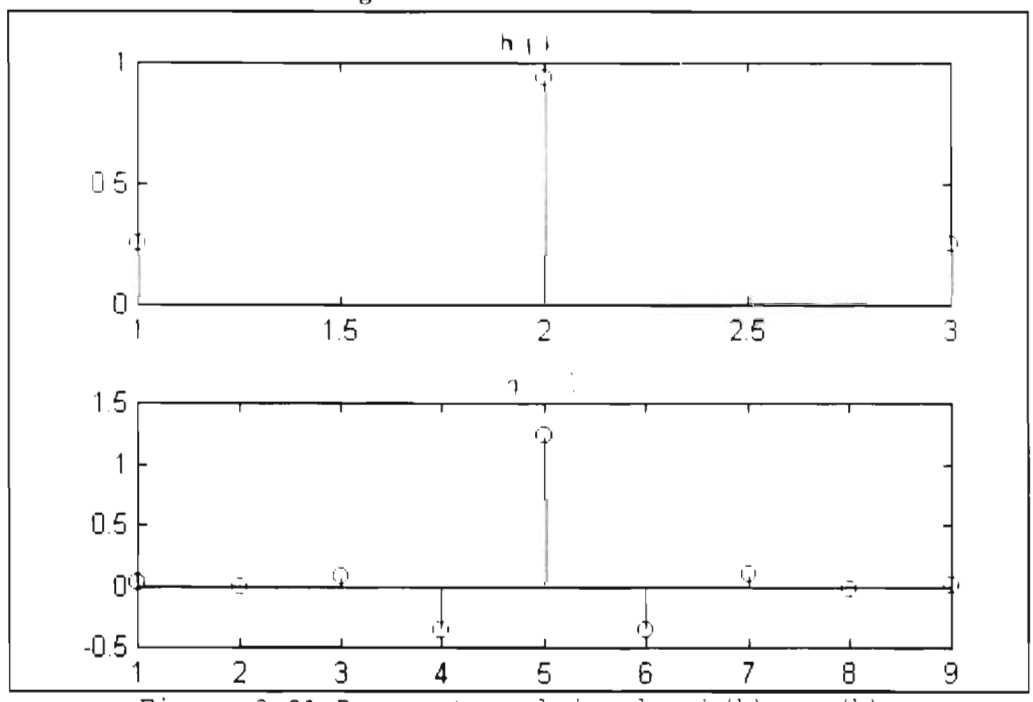


Figura 3.21 Respuestas al impulso h(k) y g(k).

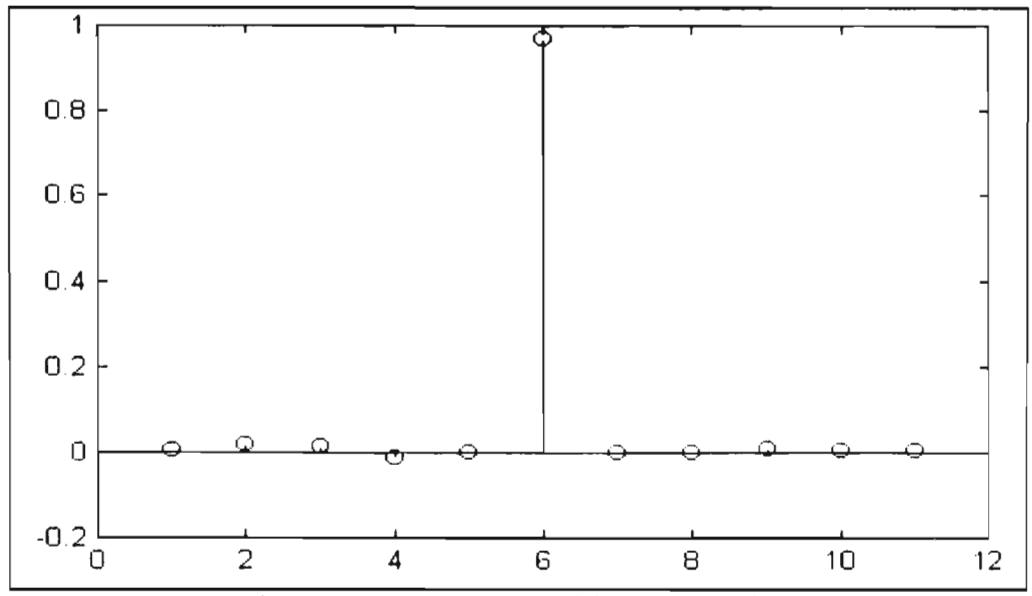


Figura 3.22 Convolución h(k)*g(k).

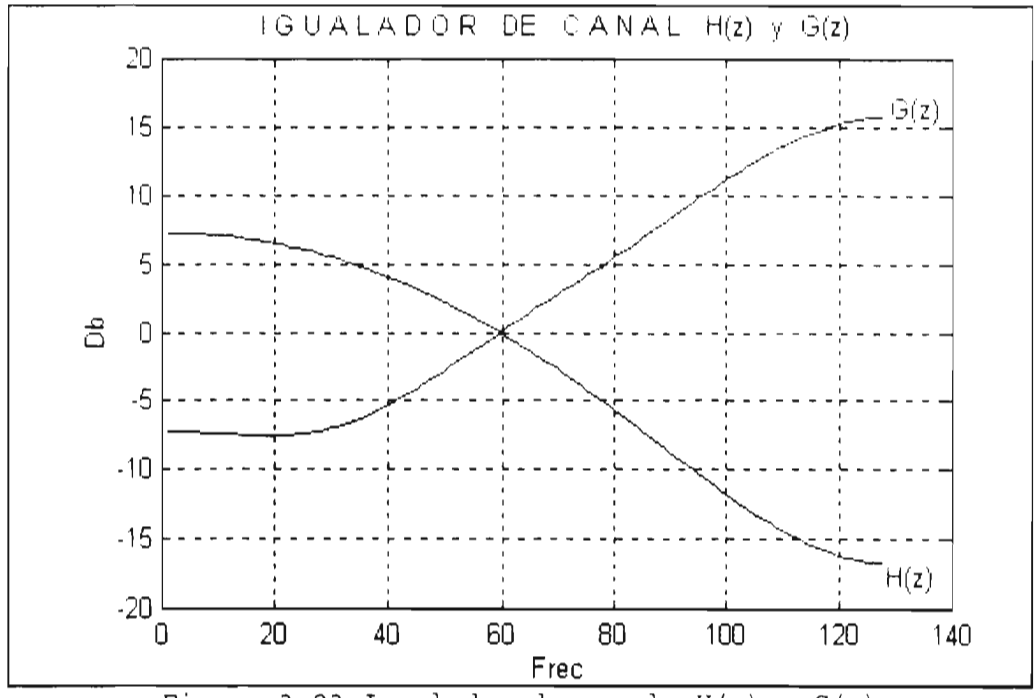


Figura 3.23 Igualador de canal, H(z) y G(z)

De los resultados obtenidos en la figura 3.22 y 3.23, se observa la convolución de h(k)*g(k) se aproxima a un pulso unitario, es decir que G(Z) se aproxima al inverso de H(z). Es decir, que estos resultados comprueban que la utilización de filtrado adaptable hace posible la igualación de canal llegando a resultados satisfactorios tal como se explicó en la introducción a este tema. Además hay que hacer notar que la respuesta al impulso h(k) de un canal real puede estar experimentando cambios, con lo que el filtrado adaptable es una solución adecuada para seguir estos cambios.

Un estudio más profundo para el problema de la igualación de canal utilizando filtrado adaptable se puede encontrar en [T0R97], donde además de abordar otros algoritmos de filtrado adaptable estudia diferentes canales característicos.

III.9.4 Eliminación de Ruido

En el proceso de eliminación de ruido en una señal se pueden utilizar filtros fijos los cuales deben estar basados en un conocimiento previo de la señal de interés y el ruido. En el diseño de filtros adaptables, éstos tienen la capacidad de ajustar sus parámetros automáticamente y no requieren un conocimiento previo de la señal ni de las características del ruido.

Una técnica que se utiliza frecuentemente para la eliminación de ruido utilizando filtrado adaptable consiste en tener dos micrófonos, uno donde se encuentra la señal deseada más el ruido y otro que capta el ruido del medio, o bien se utiliza un generador de ruido el cual debe tener las propiedades estadísticas del ruido real y no correlacionado con x(k). La señal de ruido n1(k) en conjunto con el bloque de adaptación hacen la estimación del ruido a restar a la señal. [WID75],[WID85],[ORF88],[WID96].

Para la figura 3.24 se tienen las señales:

x(k) + n(k) Señal deseada más ruido.

n1(k) Ruido no correlacionado con x(k) pero correlacionado con n(k), esta es la señal de referencia a cancelar.

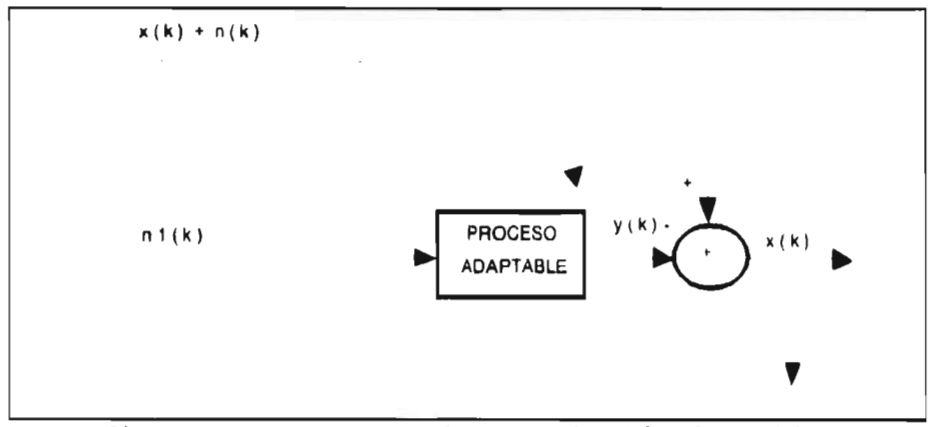


Figura 3.24 Esquema de cancelación de ruido.

En la figura 3.24 la señal de referencia de entrada es procesada por un filtro adaptable que ajusta los coeficientes del filtro en el sentido de que la señal de error de salida sea la señal x(k) deseada. Entonces el sistema puede operar bajo condiciones que cambian y ajustarse asimismo para minimizar el error.

En las siguientes gráficas se tiene la eliminación de ruido en una señal senoidal, una triangular y una señal de electrocardiograma (ECG) contaminada con una senoidal de 60 hz.

III.9.4.1 Resultados de la eliminación de ruido.

Con base al diagrama de bloques planteado en la figura 3.24, en la figura 3.35 se presentan los resultados de la eliminación de ruido en una señal senoidal utilizando algoritmos de filtrado adaptable.

Como se puede observar en la figura 3.25 se tiene una señal senoidal original, que es la señal deseada, la señal senoidal más ruido, y al final se tiene la señal filtrada utilizando algoritmos adaptables para lograr la eliminación de ruido. Como se ha comentado en otros resultados, y se observa en las señal senoidal ya filtrada, al inicio de la adaptación siempre se presenta los efectos de la adaptación hasta que el algoritmo adaptable converge, una vez que se logra la convergencia se tiene la señal deseada.

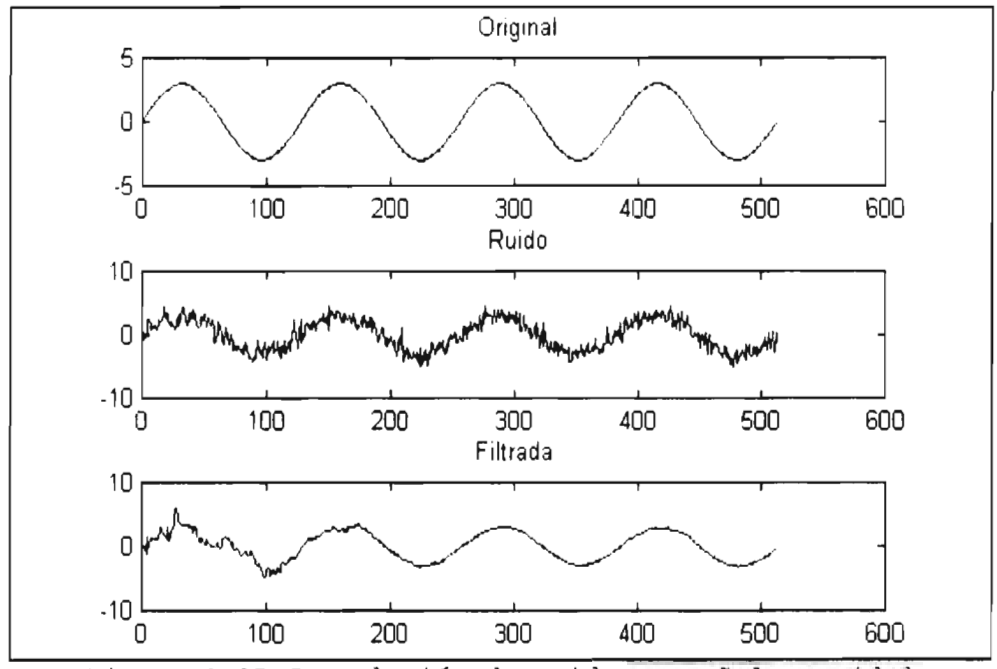


Figura 3.25 Cancelación de ruido en señal senoidal

En la figura 3.26 se tienen otros resultados de la aplicación de filtrado adaptable en la eliminación de ruido en una señal triangular. Como se observa en la figura 3.26, la señal con ruido no presenta mucha similitud a una señal triangular como en el caso para la señal senoidal con ruido (figura 3.25), sin embargo, una vez efectuado el proceso de filtrado se logra rescatar la señal triangular del ambiente ruidoso, observándose de nuevo que al inicio es notable el proceso de adaptación.

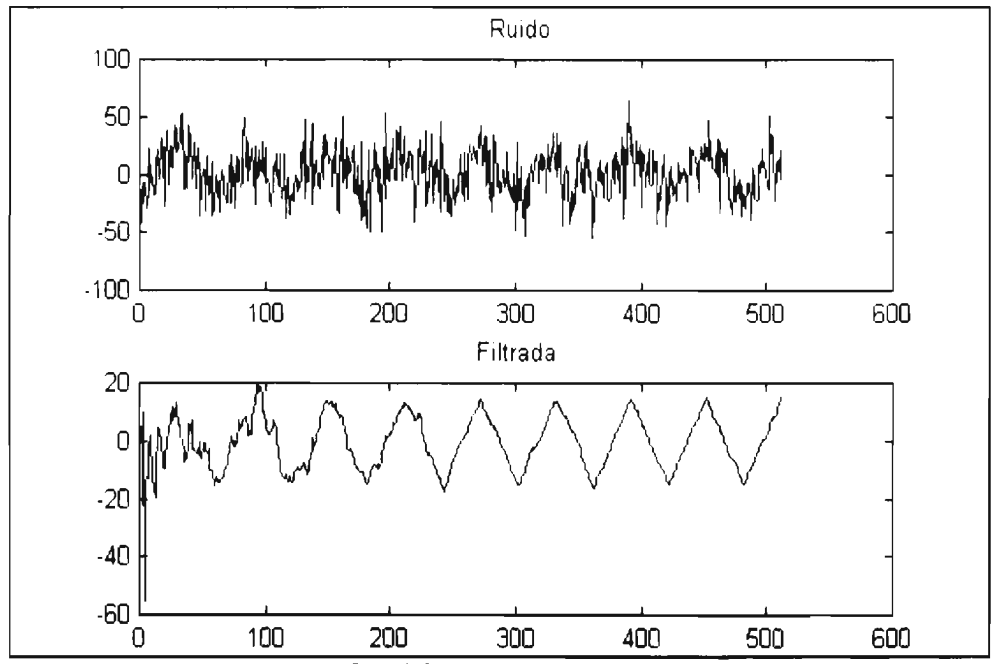


Figura 3.26 Cancelación de ruido en señal triangular

En la figura 3.27, se aplicó el mismo esquema de eliminación de ruido, a una señal de un electrocardiograma (ECG) afectada por la interferencia de una señal senoidal de 60 hz, como se observa en la señal ECG contaminada, existen zonas donde la interferencia es tan grande que la señal real es imperceptible. En la señal filtrada se observa como el proceso de filtrado adaptable logra eliminar el efecto de interferencia.

III.9.4.2 Eliminación de interferencia de 60 hz en una señal de ECG

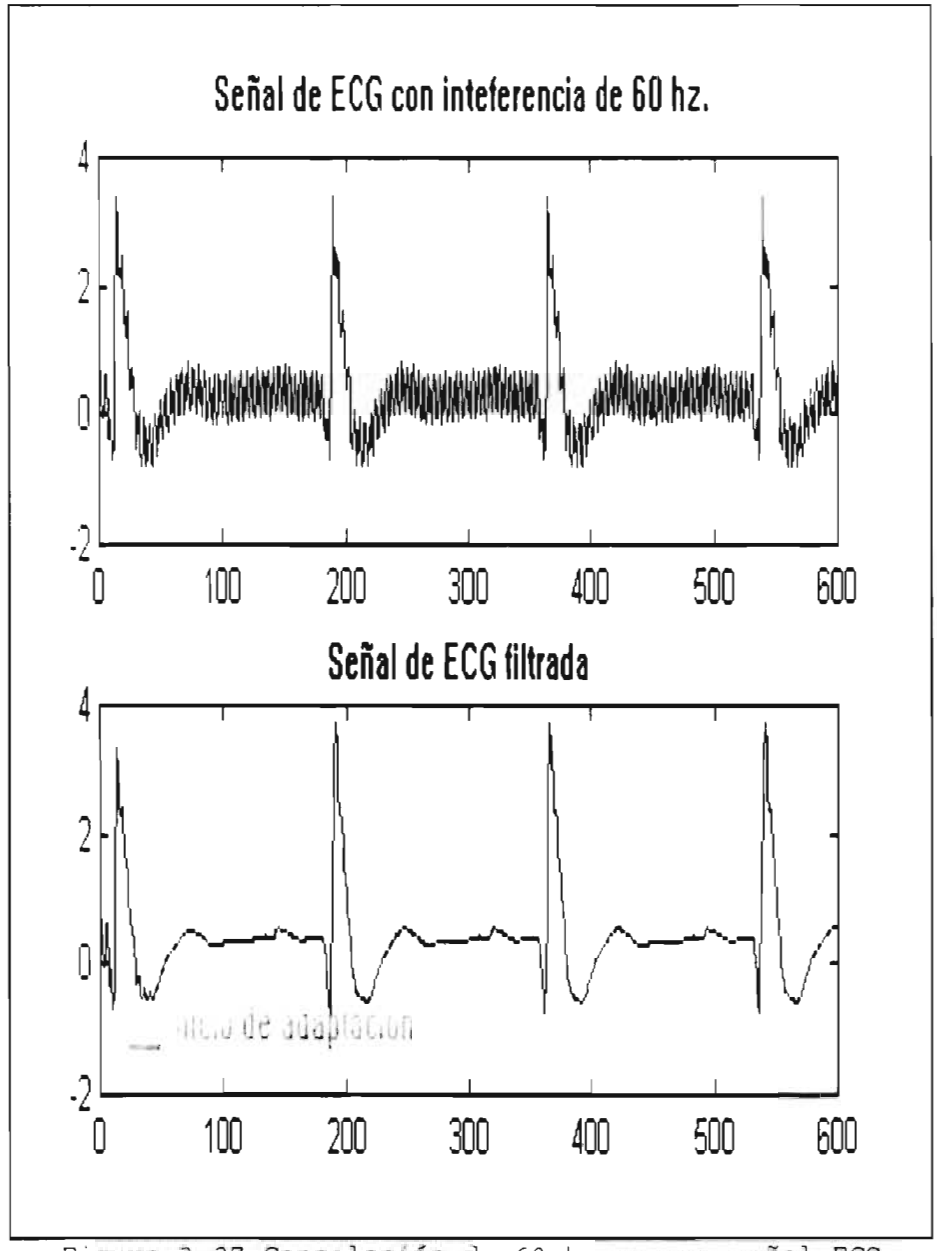


Figura 3.27 Cancelación de 60 hz en una señal ECG

En este capítulo se han desarrollado conceptos fundamentales del filtrado adaptable y las formas de optimizar algunos criterios para llegar a obtener un algoritmo en específico. Como consecuencia se han desarrollado algunos algoritmos con diferentes características, asimismo se ha hecho una comparación de las bondades de cada uno, de sus ventajas y de sus desventajas. Como parte final de capítulo se han presentado las aplicaciones más típicas en telecomunicaciones y se ha comprobando que efectivamente el filtrado adaptable cumple con el objetivo de eliminar efectos nocivos en comunicaciones.

En la actualidad existe una gran cantidad de algoritmos de filtrado adaptable, mucho más que los expuestos en este capítulo, sin embargo no es objeto de este trabajo abordarlos todos, pero con los que se han abordado se tiene un amplio panorama de los algoritmos adaptables, ya que de hecho existen pasos muy similares en el desarrollo de los mismos.

Uno de los algoritmos más utilizados es el algoritmo LMS por su sencillez de implementación y por contar con pocas operaciones por iteración (2p a 3p), sin embargo, es muy lento en converger. Por otro lado, los algoritmos tipo rápidos RLS presentan mayores velocidades de convergencia, con grados de implementación más complejos y en algunos casos con cantidad de operaciones cercanas al LMS, tal es el caso del FAEST y el FTF (de 5p operaciones). En la actualidad las propiedades de los algoritmos RLS los hace más convenientes para aplicaciones donde se requieren altas velocidades.

En el siguiente capítulo se continúa el estudio de los algoritmos adaptables mostrando algunas problemáticas que esto presentan, se presentan otros algoritmos que aminoran y en algunos casos resuelven estos problemas, ya que los algoritmos rápidos RLS presentan grandes ventajas de convergencia sobre los LMS, sin embargo, en la práctica estos pueden presentar inestabilidades.

Capítulo IV

ALGORITMOS ESTABILIZADOS

ALGORITMOS ESTABILIZADOS

En el capitulo anterior se han visto las características de los algoritmos de filtrado adaptable y se ha hecho una evaluación de éstos, sin embargo, los algoritmos antes deducidos se puede considerar que operan adecuadamente cuando se trabaja en aritmética de precisión infinita, bajo circunstancias de precisión finita o ciertas condiciones de las señales, la operación de los algoritmos puede cambiar. Al respecto existen varios estudios, donde se considera ciertos cambios en la operación los algoritmos, de ahi surge la idea de los algoritmos estabilizados. Básicamente, un algoritmo estabilizado parte de uno ya existente y trata de corregir su operación cuando ciertas variables hacen que los algoritmos diverjan rápidamente, o también se pueden cambiar secuencias de operaciones agregando nuevos cálculos, esto implica que puede afectar el número de operaciones por iteración.

En este capítulo se presentan algunos inconvenientes de los algoritmos adaptables rápidos, tal como el algoritmo rápido de Kalman (ARK). Este se ha utilizado en muchas aplicaciones, sin embargo, presenta problemas de estabilidad numérica debido a la acumulación de errores, sobre todo cuando se implementa en aritmética de precisión finita, [LIN84],[CIO87], para solucionar estos problemas se han hecho algunas propuestas que involucran cálculos extras que pueden hacer crecer el número de operaciones a costa de conservar la estabilidad de los algoritmos.

IV.1 ALGORITMO RAPIDO DE KALMAN ESTABILIZADO

El algoritmo ARK tiende a ser inestable cuando se utiliza el factor de olvido λ ($\lambda < 1$), [LJU85], sin embargo, λ es introducido para ponderar los errores pasados haciendo que el algoritmo sea adaptable.

Cuatro razones principales explican esta divergencia, [FAB85]:

a) Errores en la aproximación de las ecuaciones.
b) La matriz $R_{p,k}$ no tiene un comportamiento adecuado para valores pequeños de λ (factor de olvido) y/o la señal de entrada no es consistentemente excitada, el algoritmo es inestable independiente de la forma que sea implementado (RLS, ARK, lattice, etc).
c) Algunos de los algoritmos LS tienden a acumular errores de redondeo hasta diverger.
d) Efectos de inicialización.

La complicada estructura de un algoritmo hace que los errores se propaguen y sea difícil su análisis. Como se muestra en la figura 4.1 el algoritmo es más estable para valores de λ cercanos o iguales a uno, y para valores menores la convergencia es más rápida pero se hace inestable.

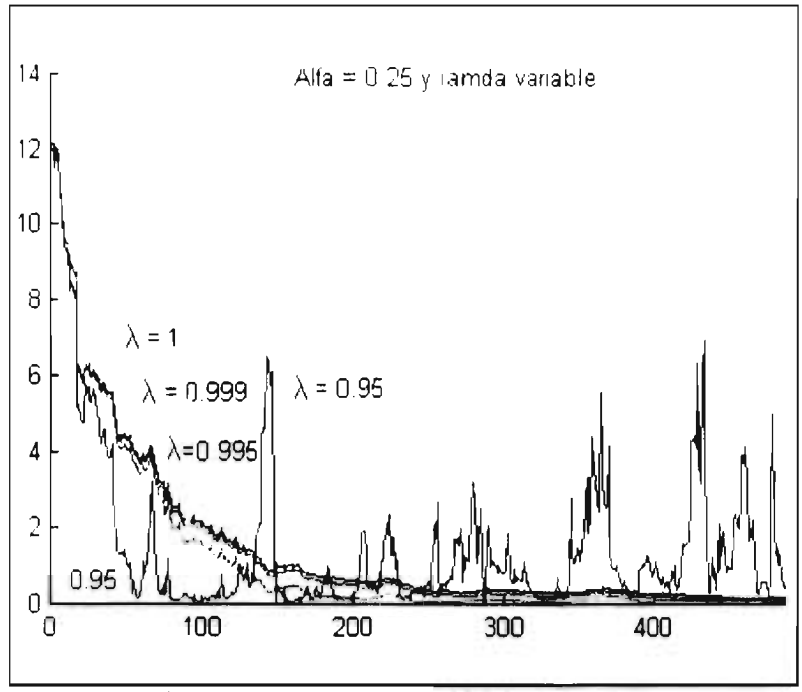


Figura 4.1 Inestabilidades para diferentes λ y $\alpha_0 = 0.25$

Una estrategia de estabilización de los algoritmos es la planteada por Alcántara, [ALC86], y también la propone Toplis [TOP88], donde el algoritmo se hace converger con un valor de λ pequeño y luego se cambia el valor de λ a un valor más alto cercano a uno. Esta estrategia resuelve el problema de divergencia, sin embargo, el algoritmo siempre tiene que estar verificando un umbral de convergencia. Algo similar se puede hacer en el algoritmo LMS, es decir, variar el valor de μ, tal como se explica en, [TOR97].

La inestabilidad en el algoritmo ARK se debe al predictor backward, [LIN84],[BEN88]. En el algoritmo ARK se presentan dos divisiones que deben ser positivas, en aritmética de precisión finita el cálculo de estos valores pueden ser negativos debido a efectos de redondeo. Para evitar los efectos de divergencia se deben de cumplir las siguientes desigualdades, [LIN84],[BEN88]:

$$1 + e_p^b(k)\, U_p(k) \leq 1 \qquad 4.1$$

$$0 \leq \varepsilon_p^f(k) / \alpha_p(k) \leq 1 \qquad 4.2$$

Sin embargo, la primera expresión es más sensible a burlar la estabilidad del algoritmo, [LIN84]. Benallal [BEN88] propone utilizar la variable de verosimilitud y el último elemento

de la ganancia de Kalman al orden p+1 para corregir el error de predicción backward, es decir, hacer un ajuste sobre este error calculado al tiempo k :

$$d^b(k) = e_p^b(k) + \alpha_p^t(k)\, U_p(k)/\lambda^p \qquad 4.3$$

El cálculo recursivo de los coeficientes de $B_p(k)$

$$B_p[k] = B_p[k-1] - (e_p^b(k) + \mu\, d^b(k))\, K_p[k] \qquad 4.4$$

El algoritmo es estable para, [BEN88]:

$$(4p+5) / (4p+7) < \lambda < 1 \quad y \quad \mu = 1 \text{ o } \mu = 1/\lambda \qquad 4.5$$

si p = 10 $0.957 < \lambda < 1$
si p = 100 $0.995 < \lambda < 1$

Estos resultados se muestran en la en la figura 4.1, donde se observa claramente que para $\lambda <$ 0.95 y p=10 , el algoritmo diverge, no así para la los valores probados de λ = 0.995, 0.999 y 1.

La estabilización propuesta por la corrección del error backward sólo agrega dos multiplicaciones, dos sumas y una división por iteración. El organigrama del algoritmo se presenta en la tabla 4.1

Inicializar $\alpha_p^f(0) = \delta > 0$ y $\lambda < 1$	
$e_p^f(k) = y(k) + A_p^T(k-1)Y_p(k-1)$	4.6
$A_p[k] = A_p[k-1] + K_p[k-1]e_p^f(k)$	4.7
$\epsilon_p^f(k) = y(k) + A_p^T[k]Y_p[k-1]$	4.8
$\alpha_p^f(k) = \lambda\alpha_p^f(k-1) + \epsilon_p^f e(k)_p^f(k)$	4.9
$K_{p+1}[k] = \begin{bmatrix} M_p[k] \\ U_p[k] \end{bmatrix} = \begin{bmatrix} 0 \\ K_p[k-1] \end{bmatrix} - \begin{bmatrix} 1 \\ A_p[k] \end{bmatrix} \alpha_p^{-1}(k)\epsilon_p^f$	4.10
$e_p^b(k) = y(k-p) + B_p^T[k-1]Y_p[k]$	4.11
$K_p[k] = (M_p[k] - B_p[k-1]Up(k)) / (1 + e_p^b(k)U_p(k))$	4.12
$d^b(k) = e_{pb}(k) + \alpha_p^f(k)\, U_p(k)/\lambda^N$	4.13
$B_p[k] = B_p[k-1] - (e_p^b(k) + \mu\, d^b(k))\, K_p[k]$	4.14
Filtrado	
$e_p^f(k) = d(k) + H_p^T(k-1)Y_p(k-1)$	4.15
$H_p[k] = H_p[k-1] + K_p[k]e_p^f(k)$	4.16

Tabla 4.1 Organigrama del algoritmo de Kalman estabilizado. [BEN88].

IV.2 ALGORITMO ESTABILIZADO FTF, (SFTF) [SLO88],[SLO91]

El algoritmo transversal rápido FTF parece computacionalmente atractivo para la solución de problemas de filtrado adaptable por la cantidad de 5p operaciones por iteración, sin embargo, presenta problemas de estabilidad numérica. [SLO88].

En esencia el algoritmo rápido FTF es un mecanismo que calcula rápidamente la ganancia de Kalman. La posibilidad de redundancia en el algoritmo FTF exhibe cambios en la dinámica de propagación de errores numéricos, la redundancia permite la retroalimentación de los errores numéricos calculando ciertas cantidades en dos formas diferentes.

El algoritmo SFTF introduce dos formas de calcular los errores backward utilizando constantes K_i, estos errores se retroalimentan para actualizar el vector de predicción hacia atrás B. Como la ganancia de Kalman se actualiza a través del vector B, entonces el cálculo de los errores backward influyen en éste cálculo. La complejidad del algoritmo es de orden 7p (para filtrado 9p).

Utilizando la notación del autor, el algoritmo FTF utiliza la misma actualización para $W_p(k)$ que el RLS, esta actualización es un reflejo de la estructura del vector $X^T_p(k) = [X_p(k)\ X^T_p(k-1)]$, por lo que se puede ir de $X_p(k-1)$ a $X_p(k)$ para hacer una actualización en orden.

$$[x(k)\ \ X_p(k-1)\ ] = X_{p+1}(k) = [X_p(k)\ x(k-p)] \qquad 4.17$$

La actualización de $r_p^{ps}(k)$ y $\gamma_p^{-s}(k)$ evita el uso de productos internos.

Diferente señales se pueden retroalimentar al algoritmo sin afectar la estructura RLS. La estructura propuesta retroalimenta señales en el cálculo de cantidades asociadas con ellas, es decir, que toma el valor final de aquellas cantidades combinadas en dos formas de cálculo.

$$r_p^{pf}(k) = B_p(k-1)\ X_{p+1}(k) \qquad \text{(producto interno entre vectores)} \qquad 4.18$$

$$r_p^{ps}(k) = -\lambda\beta_p(K-1)\ C_{p+1}^{psH}(k) \qquad 4.19$$

(producto escalar, denotado por s, H denonta Hermitiano)

$$r_p^{p}(k) = r_p^{ps}(k) + K(\ r_p^{pf}(k) - r_p^{ps}(k)\ ) \qquad 4.20$$

y similarmente para C_{p+1}^{p} y $\gamma_p^{-1}(k)$.

Para el caso de precisión infinita los valores $r_p^{pf}(k)$ y $r_p^{ps}(k)$ son iguales.

La cantidad $r_p^{p}(k)$ es utilizada en diferentes lugares del algoritmo, si se usan diferentes valores de K_i en los diferentes casos, entonces se tendrá más libertad en la dinámica de los errores del algoritmo. Es decir, que el algoritmo puede estabilizarse eligiendo los valores de K_i,[SLO88],[SLO91].

Las cantidades calculadas por el algoritmo serán independientes de K_i si se dispone de precisión infinita.

La influencia de los parámetros K_i puede resumirse :

K_4	Es sólo de importancia secundaria.
K_5	No es necesario para la estabilización (excepto en casos extremos), agrega libertad adicional en la forma de la dinámica del error.
$K_3 = K_6 = 0$	Lleva a un algoritmo de complejidad 8p.
$K_3 = 1$ y $K_6 = 1$	La componente de estado $\gamma_p^{-1}(k)$ se calculada en función de algunas otras componentes de estado.
$K_1 = K_2 = 0$	Llevan al algoritmo original 7p con divergencia exponencial.

Según estudios de Slock y Kailath, [SLO91], ha encontrado que los valores de $K_1 = 1.5$, $K_2 = 2.5$ y $K_3 = 1$ producen el mejor desempeño para que el algoritmo pueda ser estabilizado para un intervalo de variación de λ. El organigrama del algoritmo se muestra en la tabla 4.2.

Problema de predicción	
$e_p^p(k) = A_p(k-1)\, X_{p+1}(k)$	4.21
$C_{p+1}^{(0)H}(k) = -\lambda^{-1}\, \alpha_p^{-1}(k-1)\, e_p^p(k)$	4.22
$C_{p+1}(k) = [0 \;\; C_p(k-1)]^T + C_{p+1}^{(0)}(k)\, A_p(k-1) \;\; \sin C_{p+1}^p(k)$	4.23
$\gamma_{p+1}^{-1}(k) = \gamma_p^{-1}(k-1) - C_{p+1}^{(0)}(k)\, e_p^p(k)$	4.24
$r_p^{pf}(k) = B_p(k-1)\, X_{p+1}(k)$	4.25
$C_{p+1}^{pfH}(k) = -\lambda^{-1}\, \beta_p^{-1}(k-1)\, r_p^{pf}(k)$	4.26
$C_{p+1}^{ps}(k) = C_p^{p-1}(k-1) + C_{p+1}^{(0)}(k)\, A_p^p(k-1)$	4.27
$r_p^{ps}(k) = -\lambda\beta_p(k-1)\, C_{p+1}^{psH}(k)$	4.28
$r_p^{p(i)}(k) = K_i\, r_p^{pf}(k) + (1-K_i)\, r_p^{ps}(k) \;\; i = 1,2,5$	4.29
$C_{p+1}^p(k) = K_4\, C_{p+1}^{pf}(k) + (1-k_4)\, C_{p+1}^{ps}(k)$	4.30
$[C_p(k) \;\; 0]^T = C_{p+1}(k) - C_{p+1}^p(k)\, B_p(k-1)$	4.31
$\gamma_p^s(k) = \gamma_{p+1}^{-1}(k) - C_{p+1}^{ps}(k)\, r_p^{p(5)}(k)$	4.32
$\gamma_p^f(k) = 1 - C_p(k)\, X_p(k)$	4.33
$\gamma_p^i(k) = K_3\, \gamma_p^{-f}(t) + (1 - K_3)\gamma_p^{-s}(k)$	4.34
$e_p(k) = e_p^p(k)\gamma_p(k-1)$	4.35
$A_p(k) = A_p(k-1) + e_p(k)\, [0 \;\; C_p(k)]^T$	4.36
$\alpha_p^{-1}(k) = \lambda^{-1}\, \gamma_p^{-1}(k-1) - C_{p+1}^{(0)H}(k)\, \gamma_{p+1}(k)\, C_{p+1}^{(0)}(k)$	4.37
$r_p^{(i)}(k) = r_p^{p(i)}\, \gamma_p^s(k) \quad i=1,2$	4.38
$B_p(k) = B_p(k-1) + r_p^{(1)}(k)\, [C_p(k) \;\; 0]^T$	4.39
$\beta_p(k) = \lambda\beta_p(k-1) + r_p^{p(2)}(k) r_p^{p(2)H}(k)$	4.40
Filtrado	
$e_p^p(k) = d(k) + W_p(k-1)\, X_p(k)$	4.41
$e_p(k) = e_p^p(k)\gamma_p(k)$	4.42
$W_p(k) = W_p(k-1) + e_p(k)\, C_p(k)$	4.43

Tabla 4.2 Organigrama del algoritmo FTF estabilizado. [SLO88].

IV.3 ALGORITMO ESTABILIZADO DE BOTTO

El algoritmo estabilizado de Botto, [BOT89], introduce redundancia de ecuaciones, éste calcula cantidades especificas de diferentes formas, la diferencia de esta formas de cálculo se utilizan como una medida del error de redondeo acumulado, esta diferencia es utilizada para la corrección de variables en el algoritmo en cada iteración en el sentido de estabilizar al algoritmo.

El algoritmo rápido de Kalman (ARK) exhibe dos diferentes modos de divergencia, una hacia el infinito y la otra a cero, [BOT89]:

a) Hacia el infinito es inmediata aparente, dado que la mayoría de la variables del algoritmo crecen hacia el infinito.
b) Hacia cero es menos conocida, dado que es difícil de observar, esta consiste básicamente en que la ganancia tiende a cero para valores razonables del filtro.

Para el caso estacionario la divergencia a cero no es aparente, esto se convierte en un serio problema, porque aunque el error de predicción es grande, la ganancia es pequeña tal que el algoritmo no puede seguir eficientemente a la señal. La estabilización del algoritmo introduce 3p multiplicaciones más para un total de 7p a 10p multiplicaciones.

Botto, [BOT89], introduce una cantidad que tiene la propiedad de ser una medida de la acumulación de errores de redondeo. Esta cantidad es usada para corregir las variables del algoritmo en cada paso; esto incrementa la complejidad del algoritmo, pero resulta un algoritmo más estable. El método propuesto puede ser aplicado a diferentes versiones de ARK normalizados o no.

Es posible incrementar la estabilidad del ARK cuando se implementa utilizando el algoritmo FTF al introducir un factor que calcula el error de predicción backward $r_p^p(t)$ involucrando un producto interno (utilizando la notación y simbología del autor):

$$r_p^p(k) = x(k-p) - B_{p,k-1}^T X_{p,k} \qquad 4.44$$

en lugar de usar

$$r_p^p(k) = -\lambda \beta_p(k-1)\, C_{p+1,k}^{p+1} \qquad 4.45$$

Obviamente esto requiere otras p multiplicaciones llegando a un total de 8p.

Es decir, que existen dos formas de calcular $r_p^p(k)$ y parecen independientes, sin considerar los errores de redondeo, sus resultados parecen ser iguales, sin embargo, bajo precisión finita esto no es cierto. Se puede utilizar la diferencia de estos dos cálculos como una medida de la acumulación de los errores.

Denotando esta diferencia como $\xi_p(k)$

$$\xi_p(k) = r_p^p(k) + \lambda\beta_p(k-1)\, C_{p+1,k}^{p+1} \quad 4.46$$

$$\xi_p(k) = x(k-p) - B_{p,k-1}^T X_{p,k} + \lambda\beta_p(k-1)\, C_{p+1,k}^p + \beta_p(k-1) A_{p,k-1}^p \{ x(k) - A_{p,k-1}^T X_{p,k-1} \}/\alpha_p(k-1) \quad 4.47$$

$$\xi_p(k) = f(B_{p,k-1}, A_{p,k-1}) \quad 4.48$$

La variable $\xi_p(k)$ es una medida de la divergencia del algoritmo que incrementa su valor absoluto conforme se acumulan los errores [BOT89], se utiliza como una variable de corrección. El objetivo será reemplazar los valores de A y B por valores nuevos corregidos $\hat{A}$ y $\hat{B}$.

También se debe minimizar la ecuación 4.49 con los valores f(A, B).

$$\hat{\xi}_p(k) = f(\hat{B}_{p,k-1}, \hat{A}_{p,k-1}) \quad 4.49$$

expresando esta minimización resulta:

$$\hat{A}_{p,k-1} = A_{p,k-1} - k_p(k-1)\hat{\xi}_p(k) A_{p,k-1} C_{p,k-1} - k_p(k-1)\hat{\xi}_p(k) e_p^p(k) D_{p,k-2} \quad 4.50$$

$$\hat{B}_{p,k-1} = B_{p,k-1} - \hat{\xi}_p(k) D_{p,k} \quad 4.51$$

donde $D_{p,k} = (1/\lambda) R_{p,k}^{-1} [0 \ldots 1]^T$

$$\hat{r}_p^p(k) = r_p^p(k) - \{1/\gamma_p(k) - 1\}\hat{\xi}_p(k) \quad 4.52$$

$$\hat{e}_p^p(k) = \{1 - k_p(k-1) C_{p,k-1}^p \hat{\xi}_p(k)\} e_p^p(k) - k_p(k-1) A_{p,k-1}^p \{1/\gamma_p(k-1) - 1\}\hat{\xi}_p(k) \quad 4.53$$

$$\hat{\xi}_p(k) = \{1 + \{1/\gamma_p(k) - 1\} + k_p(k-1) A_{p,k-1}^{p2} \{1/\gamma_p(k-1) - 1\} + 2A_{p,k-1}^p k_p^2(k-1) C_{p,k-1}^p e_p^p(k) + k_p^2(k-1) D_{p,k-2}^p (e_p^p(k))^2 \} \quad 4.54$$

El desempeño del algoritmo no cambia si

$$\hat{A}_{p,k-1} \approx A_{p,k-1} \quad 4.55$$

El algoritmo necesita p multiplicaciones para calcular $r_p^p(k)$ y 2p para la corrección de $B_{p,k-1}$. El organigrama del algoritmo se presenta en la tabla 4.3.

Al tiempo k $C_{p,k-1}$, $A_{p,k-1}$ $B_{p\ k-1}$, $H_{p,k-1}$ $X_{p,k-1}$ $\gamma_p(k-1)$, $\alpha_p(k-1)$, $\beta_p(k-1)$, $y(k)$	
Cálculo de variables $\xi_p(k)$, $\hat{\xi}_p(k)$	
$e_p^p(k) = x(k) - A_{p,k-1}^T X_{p,k-1}$	4.56
$r_p^p(k) = x(k-p) - B_{p,k-1}^T X_{p,k}$	4.57
$\gamma_{p+1}(k) = \lambda\alpha_p(k-1)\gamma_p(k-1)/ (\lambda\alpha_p(k-1) + \gamma_p(k-1)e_p^{p2}(k))$	4.58
$\gamma_p(k) = \gamma_{p+1}(k)/(1+\gamma_{p+1}(k)r_p^p(k)(C_{p,k-1}^{p+1} + e_p^p(k)A_{p,k-1}^p / \lambda\alpha_p(k-1))$	4.59
$k_p(k-1) = \gamma_p(k-1)/ \lambda^p$	4.60
$\xi_p(k) = r_p^p(k) + A_{p,k-1}^p k_p(k-1) e_p^p(k) + \lambda\beta_p(k-1) C_{p,k-1}^p$	4.61
$\hat{\xi}_p(k) = \{1+(1/\gamma_p(k) - 1) + (k_p(k-1)A_{p,k-1}^p)^2\{1/\gamma_p(k-1)-1\}$ $+ 2A_{p,k-1}^p k_p^2(k-1)C_{p,k-1}^p e_p^p(k) \}^{-1} \xi_p(k)$	4.62
Corrección del filtro transversal $B_{p,k-1}$	
$\hat{e}_p^p(k) = \{1 - k_p(k-1)C_{p,k-1}^p\hat{\xi}_p(k)\}e_p^p(k) - \lambda^p A_{p,k-1}^p\{1-1/\gamma_p(k-1)\}\hat{\xi}_p(k)$	4.63
$C_{p+1,k} = [\ 0\ \ C_{p,k-1}\ ]^T - [\ 1\ - A_{p,k-1}^p]^T \hat{e}_p^p(k)/\lambda\alpha_p(k-1)$	4.64
$[\hat{B}_{p,k-1} -1]^T = \{[B_{p,k-1} -1]^T - \hat{\xi}_p(k)\ C_{p+1,k}\}/(1 + C_{p+1,k}^{p+1}\hat{\xi}_p(k))$	4.65
$\hat{r}_p^p(k) = r_p^p(k) - \{1/\gamma_p(k) - 1\}\hat{\xi}_p(k)$	4.66
Algoritmo clásico FTF	
$e_p(k) = \gamma_p(k-1)\ \hat{e}_p^p(k)$	4.67
$\alpha_p(k) = \lambda\alpha_p(k-1) + e_p(k)\hat{e}_p^p(k)$	4.68
$A_{p,k-1} = A_{p,k-1} - e_p(k)C_{p,k-1}$	4.69
$r_p(k) = \gamma_p(k)\ \hat{r}_p^p(k)$	4.70
$\beta_p(k) = \lambda\beta_p(k-1) + r_p(k)\hat{r}_p^p(k)$	4.71
$[C_{p,k}\ \ 0]^T = C_{p+1,k} - C_{p+1,k}^{p+1}[-B_{p,k-1}\ \ 1]^T$	4.72
$B_{p,k-1} = \hat{B}_{p,k-1} - r_p(k)C_{p,k}$	4.73
Filtrado de la señal y(k)	
$\varepsilon_p^p(k) = y(k) - H_{p,k-1}^T X_{p,k}$	4.74
$\varepsilon_p(k) = \gamma_p(k)\ \varepsilon_p^p(k)$	4.75
$H_{p,k} = H_{p,k-1} - \varepsilon_p(k)C_{p,k}$	4.76

Tabla 4.3 Algoritmo estabilizado FTF (Complejidad 10p), [BOT89].

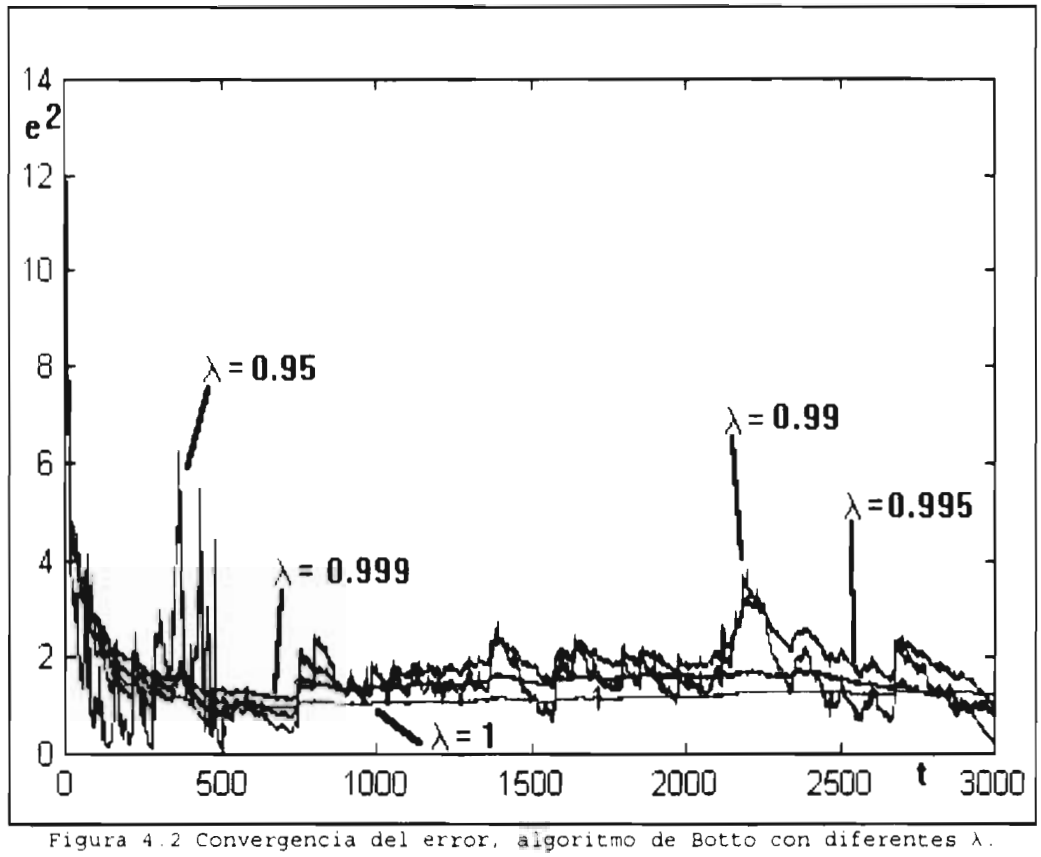


Figura 4.2 Convergencia del error, algoritmo de Botto con diferentes λ.

En la figura 4.2 se observa que el algoritmo de Botto tiende a ser inestable cuando los valores de λ son menores o iguales a 0.95. Según la referencia [SLO91] la estructura de solución presentada por Botto no es suficiente dado que esta cae en problemas de inestabilidades asociadas con $\gamma_p(k)$. Además, en la referencia [CIO87] se menciona que a pesar de los problemas de inestabilidad hallados en este algoritmo para algunos casos prácticos, la solución de Botto & Moustakides es práctica y efectiva. El mismo autor de este método de estabilización, [BOT89], menciona que el hecho que sus pruebas no diverjan, no significa que halla obtenido la solución al problema del algoritmo RLS, además señala que no realizó las pruebas suficientes de estabilidad.

En las figuras 4.3 se hace una comparación entre el algoritmo estabilizado de Botto y el algoritmo estabilizado SFTF, para un valor de $\lambda = 0.999$ y para un número de 3,000 muestras (Botto hace pruebas de su algoritmo para 10,000 muestras), como se observa el algoritmo de Botto tiene la ventaja de converge relativamente más rápido, sin embargo, cuando llega a converger de nuevo empieza a crecer y su comportamiento es similar al algoritmo SFTF. En seguida la convergencia del algoritmo de Botto crece y se mantiene estable pero con un valor de convergencia del error relativamente superior al SFTF, es decir, que no fue mucho lo que se ganó con una mejor convergencia inicial.

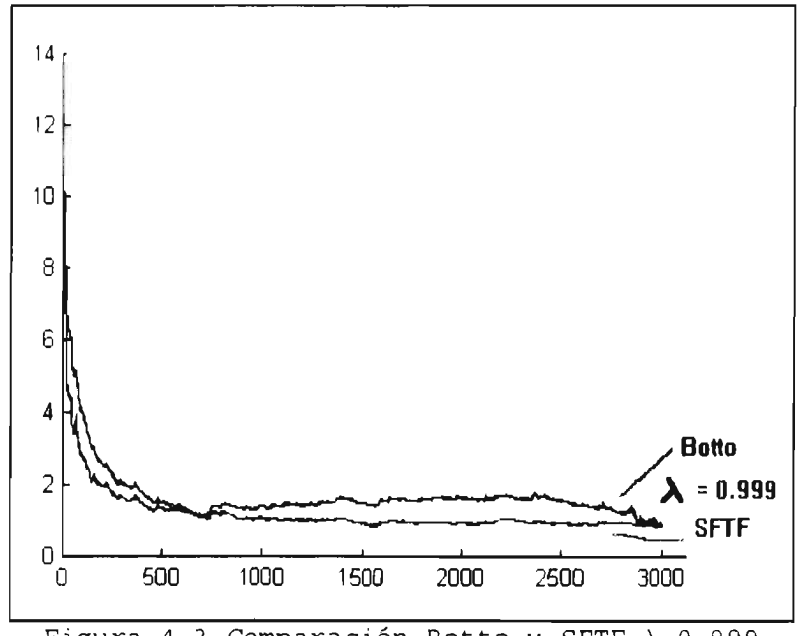


Figura 4.3 Comparación Botto y SFTF λ=0.999

En la figura 4.4 se muestras los resultados de la convervencia similares a los de la figura 4.3, sólo que para un $\lambda = 0.95$. Como se puede observar el algoritmo SFTF sigue comportandose sin mayores cambios respceto a los resultados anteriores donde $\lambda = 0.999$, mientras que el algoritmo de Botto tiene la misma tendencia inicial a la convergencia, sin embargo abruptamente empieza a mostrar inestabilidades, que eventualmente hacen que el error empiece a crecer produciendo inestabilidades. Esto muestra que para $\lambda < 0.95$ el algoritmo es inestable.

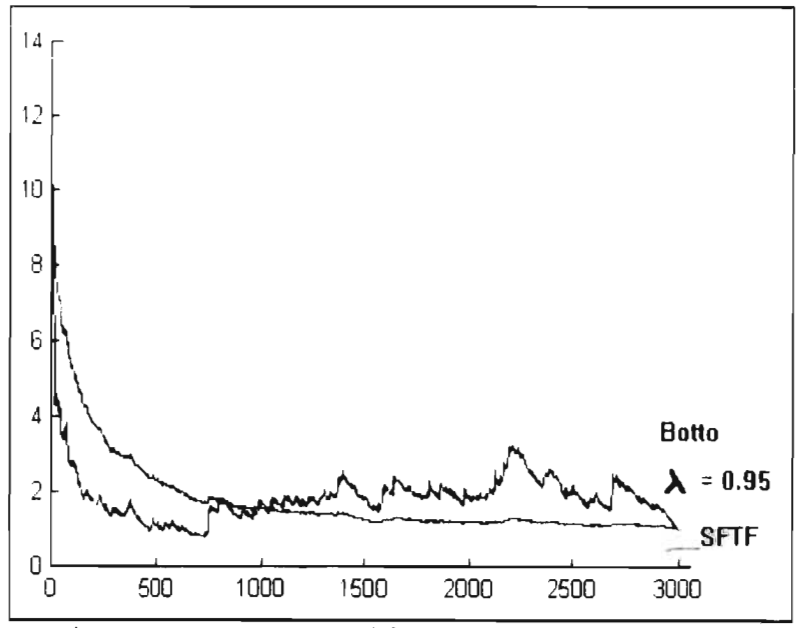


Figura 4.4 Comparación Botto y SFTF λ=0.95

Como se observó en las figuras 4.2, 4.3 y 4.4 y se comprobó para simulaciones para $\lambda < 0.95$, el criterio de estabilidad establecido en la ecuación 4.1 es válido para la interpretación de estas figuras, ya que para un orden p=10, λ debe ser mayor que 0.95 y el algoritmo de Botto es más estable cuando λ se acerca a uno (figura 4.2). Bajo las mismas condiciones probadas el algoritmo SFTF es más estable que el de Botto.

En la tabla 4.4 se hace un resumen global a nivel de operaciones que realizan por iteración los algoritmos desarrollados anteriormente. En las dos últimas filas se agregan los algoritmos estabilizados SFTF y el de Botto, donde se observa un crecimiento en el número de operaciones para efectuar un mejoramiento en comportamiento de los algoritmos. El algoritmo de Botto incrementa el número de operaciones y como mostró puede tener problemas de estabilidad.

Algoritmos	Predicción Lineal		Filtrado	
	*	–	*	÷
LMS	2p a 3p	1 (para NLMS)	3p	
RLS	$2p^2 + 6p$			
Rápido de Kalman	8p	2	10p	2
Rápido a posteriori (FAEST)	5p+8	2	7p+9	2
FTF	5p	2	7p	
SFTF	8p			
Botto	10p	8	12p	

Tabla 4.4 Comparación del número de operaciones

En este capítulo se han señalado algunas de las deficiencias que presentan los algoritmos adaptables rápidos y algunas formas de solucionar estos problemas, se ha mostrado gráficamente el comportamiento de la convergencias de éstos. Además se resume en una tabla el número de operaciones teóricas que ocupa cada uno de estos algoritmos.

Como conclusión de este capítulo podemos considerar que los algoritmos adaptables cuando se implementan en aritmética finita presentan problemas de estabilidad que hay necesidad de corregir para su buen desempeño. Sin embargo, los algoritmos estabilizados en su tarea de corregir estas deficiencias incrementan el número de operaciones por iteración. En la implementación de los algoritmos adaptables un desempeño adecuado se logra cuando el valor de λ se considera con valores muy cercanos a uno. Los algoritmos estabilizados pueden corregir los errores de divergencia pero no necesariamente garantizan esta corrección en todos los casos.

En el capítulo siguiente se abordará la implementación concreta de estas aplicaciones bajo el esquema de una arquitectura dedicada. Esto se hará tomando en consideración que ya se ha abordado toda la teoría de los algoritmos adaptables y se conocen sus características.

Capítulo V

EVALUACION DE LA ROBUSTEZ NUMERICA DE LOS ALGORITMOS DE FILTRADO ADAPTABLE

EVALUACION DE LA ROBUSTEZ NUMERICA DE LOS ALGORITMOS DE FILTRADO ADAPTABLE

En este capítulo se expone la metodología seguida para la implementación de los algoritmos, ya que es necesario seguir algunos pasos para tener una implementación adecuada. Además, se hace una presentación de las operaciones numéricas en formatos de precisión finita e infinita, un análisis de las fuentes de errores numéricos y su modelado, se evalúa la robustez numérica de los algoritmos, y se propone una arquitectura para la implementación de los Algoritmos de Filtrado Adaptable. La implementación se enfoca a Procesadores de Señales Digitales (DSP) por su alto desempeño en este tipo de algoritmos.

Se evalúan e implementan los algoritmos LMS y los algoritmos rápidos RLS en tarjetas de desarrollo que cuentan con un DSP, el TMS320C50 de la compañía Texas Instruments (TI). Estas tarjetas cuentan además con un convertidor analógico digital (A/D) y digital analógico (D/A) de 14 bits con puerto serie que permite conectarse directamente al puerto serie del TMS320C50.

V.1 METODOLOGIA DE IMPLEMENTACION

Una metodología permite sistematizar el diseño y desarrollo de actividades para realizarlas de una manera más óptima, proponer etapas y tareas cuyo desarrollo y evaluación permitan avanzar hacia el objetivo planteado, e ir fortaleciendo y corrigiendo deficiencias que se presenten. Durante el desarrollo del presente trabajo se ha seguido una metodología señalando las etapas ya desarrolladas en los capítulos correspondientes.

Para el diseño de sistemas de procesamiento digital de señales en la referencia [ALC92] se propone la siguiente metodología:

1) *Definición de problema real a resolver*

La problemática abordada por este trabajo en los capítulos dos y tres define los problemas factibles de resolver utilizando filtrado adaptable, tales como: predicción, identificación de sistemas, la deconvolución, cancelación de interferencias, igualación de canal y cancelación de eco. Además, en el capítulo tres se han abordado estos problemas con más profundidad.

2) *Conceptualización del problema: solución matemática*

En este punto, con base en el análisis del problema se proponen soluciones. En el capitulo tres y cuatro se hizo un estudio y análisis de los algoritmos existentes en esta área, sus posibilidades de implementacion, características principales, estructuras y convergencia. En los capitulos dos y tres se han planteando las estructuras que nos llevan a la solución de estos problemas.

3) *Proposición de una solución basada en las técnicas de procesamiento digital de señales: solución algoritmica*

La implementación de los algoritmos matemáticos implica hacer una valoración de sus estructuras, para la consecución de los objetivos planteados, en el caso concreto de los algoritmos adaptables es importante tener presente la dinámica numérica de los algoritmos.

La solución algorítmica conlleva a examinar una gran gama de posibilidades que presenta el PDS (capítulos tres y cuatro), cuando se tienen dificultades de plantear esta solución se debe retornar a la conceptualización del problema.

4) *Simulación en lenguaje de alto nivel de la propuesta algorítmica: solución software*

Se hace una evaluación de los algoritmos utilizando lenguaje de alto nivel con señales generadas sintéticamente, se efectúan las operaciones en aritmética de punto flotante y de punto fijo. Se hace una comparación entre ambas implementaciones, se valoran las dinámicas, los errores y las rapidez de convergencia. En este punto, utilizando lenguajes de alto nivel, se efectúan simulaciones en aritmética entera para comparar con los resultados en aritmética flotante y evaluar el comportamiento de las variables y el desempeño de los algoritmos. Las simulaciones presentadas se han hecho en Lenguaje C con precisión numérica en punto flotante a 32 bits y en MATLAB con precisión numérica doble. Estas simulaciones se han hecho en los capítulos tres y cuatro, presentando resultados que validan la solución a la problemática planteada.

Cuando los resultados de estas simulaciones no son los esperados, entonces tenemos que hacer una revisión de la solución algorítmica en el sentido de obtener la mejor solución y volver a las simulaciones en lenguaje de alto nivel

La simulación en punto flotante facilita la estimación de los intervalos dinámicos de las variables del algoritmo y permite la validación de los resultados de la implementación en microprocesadores. Las condiciones de estabilidad son impuestas por la elección del predictor de energía forward, el factor de olvido, la longitud de palabra utilizada y las características de la señal a procesar, [ALC86].

5) *Simulación en el lenguaje de un dispositivo dedicado: solución micro-software.*

Con las evaluaciones del punto anterior, los algoritmos se trasladan a lenguaje ensamblador, para un DSP, concretamente el TMS320C50 de la familia TMS320 de Texas Instruments (TI), este

DSP opera en precision finita a 16 bits (Ver anexo B.1) . En este punto se utilizaran generadores de codigo. Concretamente en este capitulo se hacen las simulaciones utilizando herramientas de desarrollo ya existentes, tales como el simulador y un DSP starter kit (DSK) del TMS320C50. [TID94]. El DSK permite hacer simulaciones de señales de hasta cinco mil puntos en tiempo real

Las simulaciones que se hagan en lenguaje ensamblador tienen que coincidir con las simulaciones hechas en aritmetica entera y utilizando lenguajes de alto nivel, si se encuentran resultados muy diferentes hay que hacer una evaluación estricta del comportamiento numerico de los algoritmos.

6) ***Diseño y realización de una arquitectura de cálculo: solución software / hardware***

Una vez validada la solución algoritmica, las simulaciones en alto nivel y en ensamblador muestran la solución del problema, es decir, que se tiene la certeza que los programas operan adecuadamente para un ambiente y tipo de señales, entonces se propone una arquitectura capaz de soportar la carga computacional de estos algoritmos. Para este caso se hacen pruebas con tarjetas DSP como el DSK, el cual tiene capacidad de hacer adquisición de señales en tiempo real. Se hace notar que la tarjeta DSK como se adquirió originalmente tiene los componentes necesarios para correr los algoritmos.

Este punto implica hacer coincidir en la solución el hardware con el software, es decir, mostrar que los resultados obtenidos en la solución software y micro-software pueden llevarse a implementar en una arquitectura real.

7) ***Evaluación, pruebas y validación de un prototipo final: solución algorítmica-software-hardware***

Se hace una evaluación partiendo desde la solución matemática, la solución algoritmica, la simulación en lenguaje de alto nivel y la solución en las arquitecturas DSP. Este punto se deja como una perspectiva a futuro de este trabajo, ya que su realización implica la implementación de algoritmos en una arquitectura dedicada a una aplicación concreta que opere en tiempo real.

Un enfoque más amplio desde el punto de vista metodológico aplicado a los sistemas de procesamiento digital de señales se puede hallar en la referencia [ARR96].

V.2 IMPLEMENTACION EN ARITMETICA DE PRECISION FINITA Y FORMATOS NUMERICOS.

En el procesamiento digital de señales (PDS) es interesante implementar los algoritmos en arquitecturas dedicadas, ya que esto da una idea de como operan los algoritmos en tiempo real. En nuestro caso se presentan las implementaciones en aritmética de precisión finita y en

particular en el TMS320C50 de aritmetica entera a 16 bits. Para llevar a cabo tal implementacion se exponen los formatos numericos a utilizar y las operaciones a realizar entre estos

Cuantizacion numerica

En aritmetica de precisión finita o de punto fijo el programador decide la localización del punto binario que separa la parte entera de la parte fraccionaria de un número. esta determinación se hace con base a un evaluación previa de la dinámica de las señales, variables y constantes a utilizar.

Representación numérica en punto fijo

En la implementación de filtros digitales en aritmética de punto fijo, siempre existen sumas de productos, es decir, si se multiplican dos cantidades cuantizadas a L bits más un bit de signo, el resultado queda representado en 2L bits más dos bits de signo, si el resultado de la multiplicacion no se cuantiza y si el filtro es de tipo recursivo, entonces en un segundo producto el resultado queda en 3L bits más tres bits de signo, y si la recursión continúa hasta "n" la representación crecería a (n+1)L bits más (n+1) bits de signo. El almacenamiento de este resultado sería problemático, por lo que es necesario almacenarlo en localidades de memoria con longitud de palabra L+1, efectuando un recorte y corrimientos en el resultado para guardar sólo los bits más significativos y un bit de signo. Para esto existen dos procedimientos que son *truncar* o *redondear*. El resultado del producto de dos números en aritmética finita está limitado a la longitud de los registros donde se almacene.

La representación de un número en formato entero de longitud L bits se efectúa en complemento a dos, y se puede interpretar como:

Números enteros que varían en el intervalo

$$-2^{15} < num < 2^{15}-1 \quad (-32768 < num < 32767) \qquad 5.1$$

o números fraccionarios que varían de

$$-2^{-15} < num < 1 - 2^{-15} \quad (30.5 \times 10^{-6} < num < 0.9999695) \qquad 5.2$$

Dependiendo cómo se ubique el punto decimal en la palabra binaria, la representación del número en formato entero puede variar entre los máximos y mínimos establecidos anteriormente. Para la representación de un número x_L en punto fijo con una longitud de palabra de "n" bits, se tiene una distribución de los bits para la parte entera y parte fraccionaria:

$$Q_L = Q_e + Q_f + S \qquad 5.3$$

donde

Q_t = numero total de bits
Q_e = numero de bits para representar la parte entera.
Q_f = numero de bits para representar la parte fraccionaria.
S = un bit para el signo.

5.4

Dependiendo del número de bits "f" que utilice la parte fraccionaria a la representación se le llama Q_f. [TI89]

Por ejemplo, un número cualquiera en formato binario de 16 bits puede tener varias interpretaciones:

x = 0001 0000 1100 0101
en Q_0 x = 0001 0000 1100 0101 significa 4.293
en Q_8 x = 0001 0000.1100 0101 significa 16.76993
en Q_{12} x = 0001.0000 1100 0101 significa 1.04810

existiendo una relación implícita entre estos números:
1.04810 = 4.293 / 2^{12} (donde se tiene un corrimiento del punto de 12 bits)
16.76953 = 4.293 / 2^8 (donde se tiene un corrimiento del punto de 8 bits)

estas expresiones son ilustrativas de la forma en que se pueden manejar los valores numéricos cuando se utiliza aritmética entera, es decir, que la dinámica de cada variable a utilizar se puede escoger de acuerdo a un máximo estimado de la misma.

Truncamiento

Como se mencionó anteriormente, después de efectuar una multiplicación es necesario para almacenarlo en memoria, obtener un resultado de longitud de palabra L+1 y seguir operando. Sin embargo como es sabido que cuando se multiplican dos números de longitud L+1, se obtiene un resultado de longitud 2L+ 2, entonces para su almacenamiento efectuando un truncamiento se descartan los bits menos significativos del producto y se almacenan los más significativos conservando el signo. El error de truncamiento para un número positivo x satisface la desigualdad para número enteros en complemento a dos

$$0 \geq x_T - x \geq -2^{-L} \qquad 5.5$$

donde L es la longitud de bits después del punto binario
y x_T es el valor truncado de x, y $|x| < 1.0$

Para operaciones en punto flotante $x = 2^e m$, donde "m" es la mantisa y "e" el exponente. Si la mantisa es truncada a L bits y el error de truncamiento $(x - x_T)$ es escrito como una cantidad proporcional a x

$$x_T - x = \epsilon(-) x \qquad 5.6$$

Para la representacion de la mantisa en dos complemento, el error de truncamiento satisface la desigualdad

$$0 \geq \epsilon x \geq -2^{-L} 2^e \qquad 5.7$$

si "x" es positivo se satisface, [OPP72]:

$$2^{e-1} \leq x < 2^e \quad x > 0 \qquad 5.8$$

$$0 \geq \epsilon > -2^{-L} 2 \qquad \text{si } x > 0 \quad y \quad 0 \leq \epsilon < 2^{-L} 2 \quad \text{si } x < 0 \qquad 5.9$$

Redondeo

El redondeo de un número binario a L bits se lleva a cabo sumando un uno en la posición más significativa de la parte binaria donde se va a hacer el recorte, si se redondea 0.101010 (0.65625) a tres bits el resultado es 0.101 (0.625), si se redondea a cuatro bits el resultado es 0.1011 (0.6875).

En aritmética de punto fijo el error de redondeo para un número de L bits después del punto binario satisface la desigualdad

$$-2^{-(L+1)} \leq x_T - x \leq 2^{-(L+1)} \qquad 5.10$$

En aritmética de punto flotante el error de redondeo satisface la desigualdad

$$-2^e 2^{-(L+1)} \leq x_T - x \leq 2^e 2^{-(L+1)} \qquad 5.11$$

Si el error de redondeo es representado proporcional a "x", es decir:

$$x\epsilon = x_T - x \qquad 5.12$$

donde ϵ satisface desigualdad

$$0 \leq \epsilon \leq 2^{-L} \qquad 5.13$$

Aritmética de punto flotante

En aritmetica de punto flotante la representación de un numero consiste de una mantisa "m" multiplicada por un una base elevada a un exponente:

$$m\ b^{e} \qquad 5.14$$

donde la mantisa es la normalización del número, en valor absoluto esta entre 1 y 2, y la base es 2. Aunque la mantisa es representada en punto entero, en el verdadero valor del número el punto binario flota al multiplicarse por b^e. El exponente "e" es un entero cuyo valor determina la posicion del punto binario del número.

IEEE ha establecido un formato estándar en la representación de número en punto flotante, este formato en 32 bits está dado por:

31	30 23	22 0
s	exponente (e)	mantisa (m)

Los bits 0-22: Representan la magnitud de la mantisa, donde para ganar precisión existe un bit implícito entre el bit 22 y 23, es decir, que la mantisa está representada verdaderamente en 24 bits.

Los bits 23-30 Representan el exponente, el cual está expresado en un formato de offset por 127, es decir, que el verdadero valor del exponente de 2 es e-127.

El bit 31 Indica el signo de la mantisa.

Errores de redondeo en estructuras recursivas

Las operaciones básicas en filtrado digital son la multiplicación y la suma. Cuando se efectúa una multiplicación de cantidades menores que uno, representadas en punto entero no ocasionan sobreflujo, sin embargo, una suma puede dar lugar a sobreflujo, ejemplo una suma de 0.9999 + 0.1232 = 1.1231 sobrepasa la representación binaria en punto entero (donde todos L-1 bits son tomados como fraccionarios), [OPP72].

Una multiplicación de números en punto entero puede modelarse como un multiplicador de precisión infinita seguido por un sumador donde se agrega el ruido del redondeo (figura 5.1). Cada muestra redondeada de ruido es modelado como una variable aleatoria con una densidad de probabilidad uniforme, con media cero y varianza $2^{-2L}/12$ donde (L +1) es el número de bits incluyendo al signo y paso de cuantización $Q=2^{-L}$,[RAB75],[DEF88].

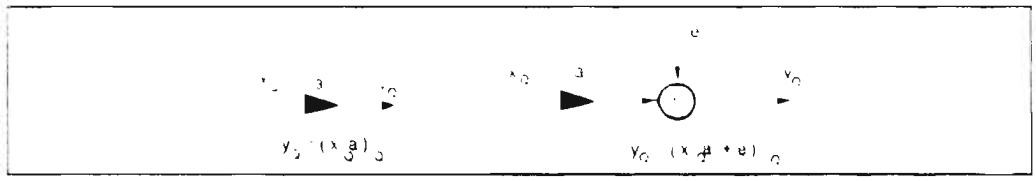


Figura 5.1 Modelo del error de cuantización

Truncamiento	$y_Q = (x_Q\, a)_Q$	5.15
Redondeo	$y_Q = (x_Q\, a + e)$	5.16

Para modelar los efectos de redondeo se hacen las suposiciones estadísticas

1) Cualquier muestra diferente de ruido de la misma fuente es no correlacionada.
2) Dos fuentes diferentes de ruido son no correlacionadas.
3) Cada fuente de ruido es no correlacionada con la secuencia de entrada.

Cada fuente de ruido es modelada como un proceso aleatorio de ruido blanco con densidad de potencia espectral $2^{-2L}/12$. Si la entrada es constante las suposiciones anteriores no son válidas.

En una operación de convolución tal como en un filtro digital, se tiene un punto de suma donde todas la fuentes de ruido son sumadas en ese punto, entonces existe un ruido equivalente:

$$e(k) = e_0(k) + e_1(k) + e_2(k) + \ldots e_{N-1}(k) \qquad 5.17$$

con media cero, utilizando la teoría de sistemas lineales y por la suposición dos, la varianza de salida debido a varias fuentes de ruido es igual a la suma de las varianzas individuales. [RAB75],[LIU69],[LIU71].

$$\sigma^2 = \sigma_0^2 + \sigma_1^2 + \sigma_2^2 + \ldots + \sigma_{N-1}^2 = N\, 2^{-2L}/12 \qquad 5.18$$

La estadística de redondeo del ruido depende donde se efectúe éste. Existe dos posibilidades, una es que las multiplicaciones se representen exactas y el redondeo sea efectuado al final de la suma, entonces solo se presenta una fuente de ruido, en el caso de redondeo el ruido es uniformemente distribuido entre -Q/2 y Q/2 con media cero y varianza $Q^2/12$. Si el producto de las multiplicaciones son redondeados antes de ser sumados, entonces el ruido de salida e(k) es la suma de N ruidos no correlacionados con una distribución uniforme entre -Q/2 y Q/2 con media cero y varianza $N\, Q^2/12$ (figuras 5.3a y 5.3b).[OPP72],[RAB75],[PRO92],[DEF88].

Modelo estadístico para redondeo aritmético

Un medio conveniente para analizar los efectos de cuantizacion es representar las estadísticas del error. La consideracion del modelo de densidad de probabilidad uniforme del error ha sido justificada experimentalmente para una amplia clase de entradas incluyendo señales aleatorias y voz. Este modelo no es válido para entradas constantes. [RAB75].

Los efectos de redondeo de un filtro digital recursivo pueden determinarse por el cálculo de la respuesta del filtro debido a una señal de ruido $e_i(k)$. si $h_i(k)$ es la respuesta al impulso del filtro. entonces la respuesta del filtro a la señal de ruido esta dada por

$$e_{on}(k) = \sum_{m=0}^{\infty} h_n(m)\, e_i(k-m) \qquad 5.19$$

la varianza del ruido debido a la fuente de ruido es:

$$\sigma^2_{o_n} = \sigma^2_{e_i} \sum_{k=0}^{\infty} h_n^2(k) \qquad 5.20$$

donde $\sigma^2_{e_i} = 2^{-2L}/12$. Como las fuentes son no correlacionadas. la varianza de la salida del ruido es la suma de las varianzas. [RAB75],[LIU69].

$$\sigma^2_{o} = \sum_{n} \sigma^2_{o_n} \qquad 5.21$$

Para un filtro de primer orden la respuesta al impulso de la fuente de ruido de entrada a la salida es $h(k)=a^k u_{-1}(k)$. entonces. [RAB75].

$$\sigma^2_{o} = \frac{1}{12} 2^{-2L} \frac{1}{1-a^2} \qquad 5.22$$

V.3 ERRORES NUMERICOS

Limitaciones del intervalo dinámico de operaciones en punto fijo

La operación de suma en operaciones de aritmética finita puede causar sobreflujo en un algoritmo digital. Una técnica para la prevención de sobreflujo es hacer escalamientos en ciertos

puntos de la señal, esta es una razon de la existencia de unidades de corrimiento a la entrada y salida de datos en un DSP.

Problemática de Implementación

Cuando se utiliza aritmética de precisión finita las propiedades numéricas de los algoritmos es diferente y las deficiencias numéricas eventualmente destruyen la operación normal. [ALC86]. Cuando se implementan algoritmos en aritmetica de punto fijo, los efectos de cuantización utilizando una longitud finita de palabra pueden modificar considerablemente los aspectos de precisión infinita.

Los principales problemas en la implementación de aritmética de punto fijo en algoritmos adaptables son:

1) La estimación de la dinámica de la variables
2) La determinación del número de bits necesarios para lograr un desempeño similar a aritmética de punto flotante.
3) La eleccion de las condiciones iniciales. [ALC87].

Efectos de longitud finita de palabra en el procesamiento digital de señales.

Cuando se implementan algoritmos de procesamiento digital de señales (PDS) se deben tomar en cuenta las siguientes fuentes que ocasionan errores durante el procesamiento. [RAB75]:

a) El ruido en la conversión análogo/digital (A/D).
b) El ruido de redondeo no correlacionado.
c) Las imprecisiones en la cuantización de los coeficientes (depende del ancho de palabra a utilizar).
d) El ruido correlacionado en el redondeo de las operaciones aritméticas.

En el estudio de los errores de una variable en un algoritmo de filtrado adaptable hay que tomar en cuenta. [CIO87]:

e) La generación de los errores, es decir, el error al tiempo actual.
f) La propagación de errores, es decir, cómo afectan y evolucionan en subsecuentes recursiones.
g) La acumulación de errores que influyen en el error actual, cómo los errores generados a diferentes instantes interactúan y se acumulan. [LJU85],[SLO91].

Dependiendo del tipo de aritmética utilizada, el tipo de cuantización para reducir los efectos de longitud de palabra y la estructura, se puede estimar cómo el desempeño del algoritmo es influenciado por cada uno de estos efectos.

Conversión análogo/digital (A/D)

En instrumentación el ruido ocasionado por la conversión A/D constituye una de las mayores fuentes de errores, sobre todo cuando la conversión se efectúa a menos de 10 bits. Un convertidor A/D es un dispositivo no lineal que mapea una entrada analógica continua en una secuencia de numeros naturales, en este proceso existe una diferencia de la entrada a la salida llamado ruido de cuantizacion o ruido de conversion, [RAB75],[DEF88],[PRO92]:

$$e(k) = s(k) - s_Q(k) \qquad 5.23$$

donde s(k) es la señal muestreada al tiempo "k" y $s_Q(k)$ la señal cuantizada.

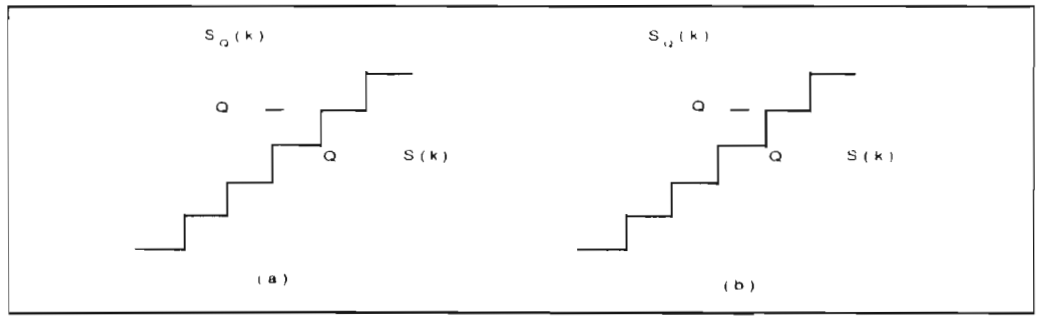


Figura 5.2 Conversión A/D

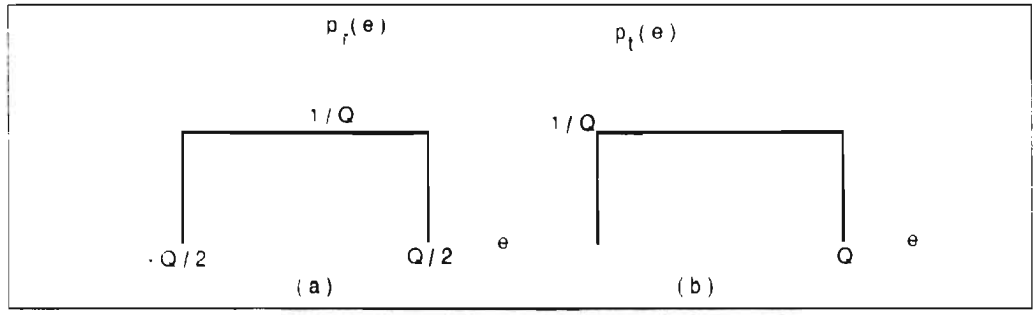
Dependiendo de la forma en que se cuantiza la señal s(k) se pueden obtener diferentes distribuciones del ruido cuantizado. Si el paso de cuantización utilizado es $Q = 2^{-L}$ (donde L+1 es la longitud de bits incluyendo al signo) entonces la relación entre $s_Q(k)$ y s(k) de redondeo se observa en la figura 5.2a.

Donde existe un número finito de cuantización para que todas las señales estén entre el valor E_{max} y un $-E_{max}$.

La señal de error satisface:

$$-Q/2 \leq e(k) \leq Q/2 \text{ para todo k.} \qquad 5.24$$

Bajo suposiciones no muy exigentes se puede comprobar que la distribución de la señal de error es uniforme entre -Q/2 y Q/2 para redondeo y entre 0 y Q para truncamiento. Para el redondeo la media es cero y la varianza $Q^2/12$. Widrow mostró que bajo la condición para una variable de entrada aleatoria con una función característica limitada en banda, la cuantización del ruido es uniformemente distribuido, esto es llamado *Teorema de cuantización*, [SRI77].



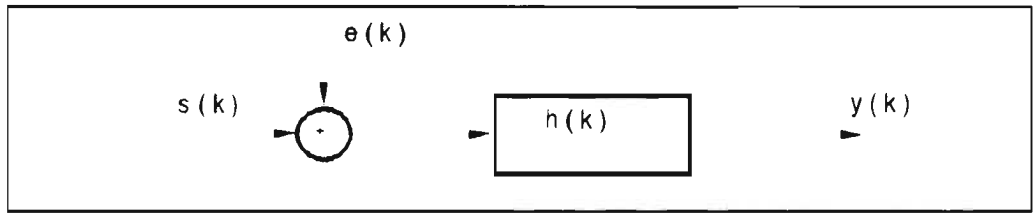
5.3 Distribución uniforme del ruido.

Para la cuantización por truncamiento la señal es representada por el nivel de cuantización superior pero no mayor que la señal como se observa en la figura 5.2b. El redondeo es equivalente al redondeo menor que una mitad que el paso de cuantización, en este caso la media es Q/2 y varianza $Q^2/12$, [RAB75],[DEF88],[PRO92].

Modelo lineal de cuantización

El ruido del conversión A/D puede representarse como una señal de entrada s(k) de precisión infinita más el error de cuantización e(k), si el sistema es lineal, entonces la salida y(k) en precisión infinita se obtiene (ver figura 5.4):

$$y(k) = s(k)\, h(k) + e(k)\, h(k) \qquad 5.25$$



5.4 Modelo lineal de cuantización.

Esta representación es básica en los efectos de longitud finita de palabra. De esta manera se puede obtener la relación señal a ruido de la secuencia procesada.

Una representación de la función de densidad de probabilidad y la viarianza del error con base al número de bits se da en la referencia [HEJ92], si Q =1.

$$f_e(e) = 1 + 2\sum_{n=1}^{\infty} J_0(2^{-1}\pi n)\cos(2\pi ne) \qquad 5.26$$

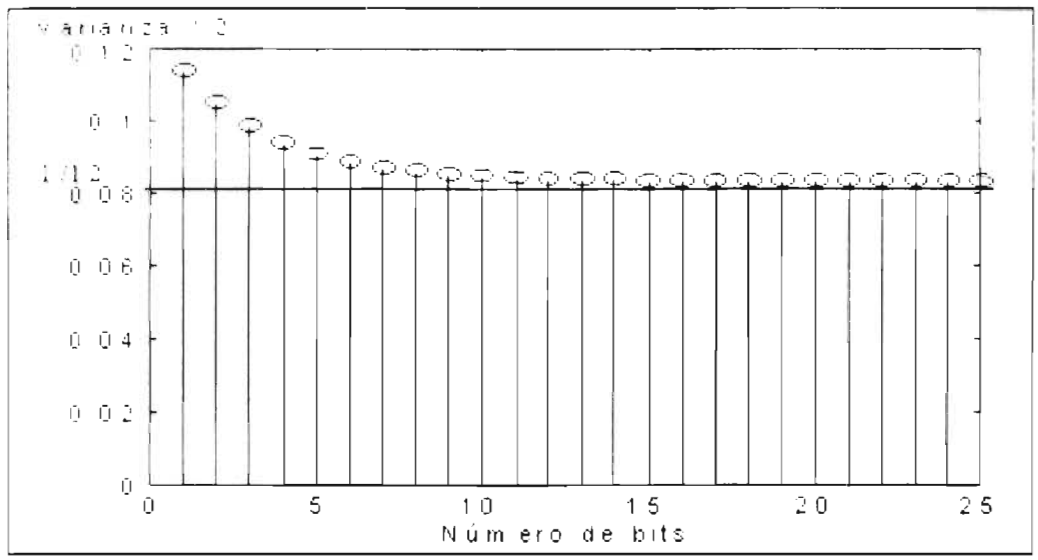


Figura 5.5 Varianza vs. número de bits

De la figura 5.5 se observa que para longitud de palabra $L > 10$ bits, la varianza converge al valor practico de 1/12. En la referencia [HEJ92] se muestra que para $L = 16$ bits la función de densidad de probabilidad del error de cuantización es muy cercano a una distribución uniforme entre 0 y 1. Si $L \geq 12$ el modelo de cuantización uniforme puede ser aceptado dentro de un porcentaje de precisión así como las estadísticas de primer orden. En referencia [FAW96] se concluye que el modelo con varianza $Q^2/12$ es más conveniente para señales senoidales donde la amplitud es suficientemente grande.

Para simulaciones en aritmética finita todas las variables y constantes han sido representadas al mismo número de bits, la figura 5.6 muestra que al correr los algoritmos adaptables como el ARK en aritmética finita en un número de bits menores de 12 bits para una señal senoidal el error diverge. Este resultado es coincidente con la figura 5.5, de la ecuación 5.27, que muestra la convergencia de la varianza para un número de bits mayor de 10. Es decir, que con estas pruebas se ligan las demostraciones teóricas de la cantidad de bits a utilizar y las simulaciones de los algoritmos. Se hace notar además que en las simulaciones de la figura 5.6, las dinámica de las variables se fijó igual para todas, sin embargo, una mejor simulación sería estar variando el Q_i de cada variable hasta encontrar el Q_i óptimo para cada algoritmo.

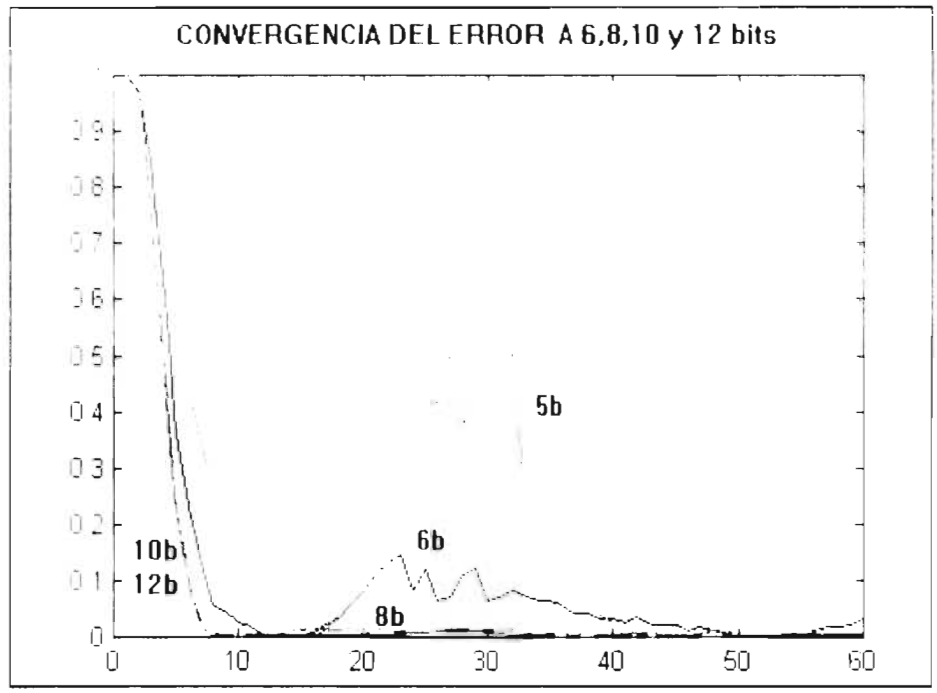


Figura 5.6 Convergencia del error a 6,8,10 y 12 bits

La mayoría de las señales analógicas contienen algún tipo de ruido, al aumentar el paso de cuantización, Q decrece a un valor relativamente pequeño respecto al pico de la señal, los bits menos significativos solo proveen una representación del ruido inherente a la señal analógica, es decir, que el incremento en el número de bits en un convertidor A/D, más allá de cierto punto sólo incrementa la precisión con que el ruido de la señal analógica es representada.

Definiendo la razón señal a ruido cuantizado, (SNR_Q)

$$SNR_Q = 10 \log_{10}(\sigma_x^2/\sigma_e^2) \qquad 5.28$$

donde σ_x^2 es la potencia de la señal x y σ_e^2 es la potencia del ruido cuantizado, para una potencia normalizada de x a $K\sigma_x$ y una distribución uniforme del error:

$$SNR_Q = 10 \log_{10}(1/K^2\sigma_e^2) = 10 \log_{10}(12/Q^2)$$
$$= 10 \log_{10}(12/2^{-2L}) = 6L + 10.8\text{-}20\log_{10}K \qquad 5.29$$

Si K=4 $\qquad SNR_Q = 6L - 1.24$ dB. $\qquad 5.30$

Para un convertidor de 8 bits $SNR_Q = 46.76$ dB, se observa que el ruido agregado por el

convertidor A/D es despreciable. A la relación SNR_Q también se le llama intervalo dinámico (ID) del cuantizador, y mide la capacidad del cuantizador al pasar la amplitud de la información sin estar contaminada por ruido, [DEF88],[PRO92].

$$\text{Si } K=1 \quad SNR_Q = 6L + 10.8 \qquad 5.31$$

se puede determinar el número de bits que se deben usar para representar un dato cuando el ID es especificado, [DEF88].

Intervalo dinámico

El intervalo dinámico (ID) de un filtro digital es definido como el número de bits disponibles no afectados por el ruido de los cálculos. Si se conoce la varianza de salida del ruido entonces el ID se puede expresar en db por, [RAB75],

$$ID = (bit_N - 1)20\log_{10}2 \qquad 5.32$$

donde bit_N es la posición del bit menos significativo del ruido y está dado por:

$$bit_N = INT(-\log 10\sigma_{x0} / \log_{10}2) \qquad 5.33$$

INT denota la parte entera.

Del análisis anterior, la ecuación 5.27, figuras 5.5 y 5.6, se puede concluir que una longitud de palabra de 12 bits en aritmética entera es una buena aproximación para la representación de las variables cuando se pretende implementar algoritmos en aritmética finita.

V.4 ARQUITECTURA PARA LA IMPLEMENTACION DE ALGORITMOS DE FILTRADO ADAPTABLE

Cuando se trabaja en aritmética de punto flotante el escalamiento de las variables lo hace automáticamente la computadora. Sin embargo, en punto fijo el diseñador debe controlar el escalamiento de las operaciones para prevenir sobreflujo, de lo contrario se tendría una explosiva divergencia.

En filtrado, los errores de cuantización se pueden combatir utilizando más bits para la representación de los coeficientes que para los datos de entrada, [CAR84]. Sin embargo, los autores Ljung S. & Ljung L., [LJU85], expresan que en términos numéricos no existe ningún ardid que mejore la organización de los cálculos que remuevan la propagación de errores, esto sólo se puede hacer cambiando el algoritmo. Un incremento en la longitud de palabra sólo hace que los errores tarden en hacerse evidentes. La estabilización de los algoritmos realmente

requiere un verdadero cambio en el algoritmo para que la dinamica de propagación de errores sea modificada

De acuerdo al analisis de la varianza del error hecha en el punto V.3, se llegó a la determinacion que una longitud de palabra adecuado para la representación de las señales y variables a manejar en un algoritmo puede ser mayor de 12 bits, es decir, que para la implementacion de los algoritmos es conveniente contar con un procesador con longitud de palabra de al menos 12 bits. Ademas, se requiere de un procesador muy rápido, con capacidad de efectuar suma de multiplicaciones eficientemente y capacidad de almacenamiento. Con estos fines se eligió un procesador de señales digitales, (DSP) el TMS320C50 de la compañía Texas Instruments (TI). [TI93].

El TMS320C50 es un DSP de 16 bits que imponen algunas limitantes:

1) La precisión (16 bits) provoca la alteración de algoritmos al codificar todas la variables con esta aritmética. Esto requiere un conocimiento previo y un aprendizaje de la evolución de las variables durante la ejecución. El programador debe tener un control de la evolución de las variables durante el proceso del algoritmo.

2) En la programación de arquitecturas DSP de 16 bits con pipeline se debe cuidar la minimización del número de operaciones y tomar ventaja de las unidades funcionales del procesador. En este punto, el TMS320C50 provee la capacidad de recuperación del pipeline en saltos condicionales, saltos a subrutinas y retornos condicionales, [TI93].

Algunas características de los DSP

Los DSP disponen de una arquitectura y un conjunto de instrucciones orientadas al Procesamiento Digital de Señales. La arquitectura de un DSP normalmente contiene buses separados para el direccionamiento simultáneo de dos operandos, un multiplicador por hardware, unidades de corrimientos, una unidad de direccionamiento, puertos seriales, temporizadores y un conjunto de instrucciones orientadas al PDS.

Entre las instrucciones de grandes posibilidades están el movimiento de bloques, multiplicaciones en un ciclo de instrucción y multiplicación acumulación para la realización de operaciones de convolución.

EL DSP TMS320C50 (Ver anexo B.1)

Es un procesador de punto fijo de la familia TMS320, su unidad de procesamiento central (CPU) está basada en la CPU del TMS320C25 con una arquitectura adicional que mejora el desempeño [TI93]. La familia C5x está diseñada para ejecutar 28 millones de instrucciones por segundo (MIPS).

Arquitectura

El TMS320C50 consiste de tres bloques principales:

- Unidad Central de Proceso (CPU)
- 10K x 16 bits de memoria interna con un K de memoria de doble acceso en un ciclo de instrucción.
- Interfaz a circuitos periféricos.

Su arquitectura utiliza dos buses separados, datos y programa, con instrucciones que aceptan transferencia de datos entre ambos espacios. El TMS320C50 efectúa aritmética de complemento a dos, utilizando la unidad aritmético lógica (ALU) y el acumulador de 32 bits.

Un *multiplicador* que efectúa multiplicaciones de 16 x 16 bits en complemento a 2 con resultado de 32 bits en un ciclo de instrucción.

El bloque de *corrimientos*, a la entrada de los datos de 16 bits y a otro la salida de datos de la ALU.

El TMS320C50 posee alto grado de paralelismo, ya que mientras se están operando datos en la Unidad Central Aritmético Lógica (CALU), se pueden estar ejecutando operaciones aritméticas en la Unidad Aritmética de Registros Auxiliares (ARAU), estas operaciones simultáneas se efectúan en un ciclo de instrucción.

Implementación de la convolución o producto interno entre vectores

Una de las operaciones fundamentales en el procesamiento digital de señales y aplicada en algoritmos adaptables es la convolución. Para la suma convolución

$$y(k) = \sum_{i=0}^{N-1} w_i \, x(k-i) \qquad 5.34$$

es decir, el producto interno entre los vectores **x** y **w**:

$$y(k) = \mathbf{x}^T\mathbf{w} = \mathbf{w}^T\mathbf{x} \qquad 5.35$$

En un DSP esta operación se puede efectuar eficientemente a través de las instrucciones de repetición y multiplicación acumulación, es decir, que la suma anterior se puede implementar:

RPTZ	N-1	; N número de repeticiones de la siguiente instrucción e inicializa el acumulador y registro producto en cero. Este ciclo es no interrumpible.
MACD	#loc_W_i,*-	; multiplica y acumula los coeficiente w_i por la localidad

	apuntada por un registro índice (al dato x(k)), y mueve el dato actual a la siguiente localidad (efectúa un retardo).
APAC	, suma último producto.

Cada multiplicación acumulación se efectua en un ciclo de instrucción.

Ademas de las instrucciones anteriores, para la implementación del algoritmo se utilizan las instrucciones de multiplicacion acumulación para efectuar convoluciones (MAC), repetición de bloques (RPTB) para efectuar ciclos en la actualizacion de los coeficientes, multiplicacion (MPY), el direccionamiento indirecto y corrimientos de datos o del acumulador en un mismo ciclo para el ajuste del punto fijo. También se realiza una subrutina para efectuar la división utilizando la instruccion SUBC (resta condicional).
(Ver anexo B.2 donde estan los programas en lenguaje ensamblador).

Aprovechando estas ventajas y caracteristicas que ofrecen los DSP's se implementan algunos algoritmos en lenguaje ensamblador del TMS320C50.

V.5 IMPLEMENTACION DE ALGORITMOS

La implementacion de los algoritmos se hace utilizando una tarjeta de evaluación DSK, [TID94], que se conecta a una computadora personal PC a través de un puerto serie. La tarjeta cuenta con un convertidor A/D y D/A de 14 bits (TLC32040 de TI) conectado al TMS320C50 por medio de su puerto serial y un programa monitor para establecer un protocolo de comunicación con la PC

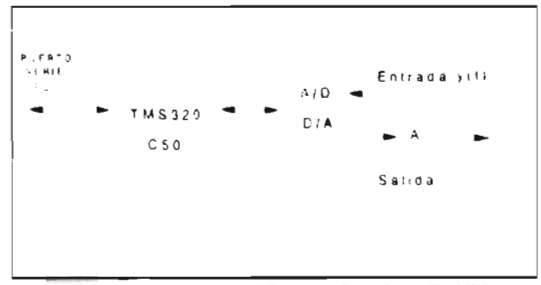


Figura 5.7 Arquitectura de la DSK

El muestreo se efectúa a razón de 8K muestras/segundo (cada 125 μs), el TMS320C50 recibe las muestras en un formato de 15 bits para la parte fraccionaria (Q15) y un bit para el signo, por software estas se pasa a formato Q11 por medio de tres corrimientos de bits a la derecha.

Procedimiento general de implementación

En procesamiento digital de señales y en la implementación de los algoritmos de filtrado adaptable se deben efectuar las siguientes operaciones:

1) Producto escalar o producto punto entre vectores.

$$\hat{y}(k) = \mathbf{A(k)}\ {}^{t}\mathbf{y}(k) \qquad 5.36$$

2) La suma de una escalar más un producto escalar.

$$e(k) = d(k) - \hat{y}(k) \qquad 5.37$$

3) Escalar por escalar más escalar por escalar.

$$\lambda * \alpha(k-1) + e^{f}(k) * e^{b}(k) \qquad 5.38$$

4) Vector más escalar por vector.

$$\mathbf{A}(k) = \mathbf{A}(k-1) + e(k)\mathbf{K}(k) \qquad 5.39$$

5) (Vector mas escalar por vector) por escalar

$$(\mathbf{M}(k) - e\,\mathbf{B}(k)) \quad (1 - U(k)e) \qquad 5.40$$

6) División entre escalares.

$$\varepsilon(k) \quad \sigma(k) \qquad 5.41$$

7) Raíz cuadrada para algoritmos normalizados.

Evaluación de operaciones aritméticas en el TMS320C50

En la tabla 5.1 se hace un resumen de las principales operaciones mencionadas que se deben efectuar en el TMS320C50 para la ejecución de los algoritmos de filtrado adaptable, el número de ciclos y el tiempo que se requieren por operación con el objeto de evaluar el desempeño de los algoritmos.

Operación	No. de ciclos	Tiempo (μs) p = 10 TMS320C50	Tiempo (μs) p = 10 TMS320C10[5]
+ , - , * , (+*)[4]	1	0.035	0.2
División	79	2.76	15
Estimación del error (escalar + producto escalar)	6 + p	0.56	8.6
Actualización del vector de parámetros (vector + escalar*vector)	6 + 5p	1.96	37
Raíz cuadrada (Algoritmo de [ALC86])	205	7.175	50

Tabla 5.1 Evaluación de operaciones aritméticas en el TMS320C50 y TMS320C10

[4] Estas operaciones son efectuadas por hardware en un ciclo de instrucción. La operación (+*) significa multiplicación acumulación. El ciclo de instrucción del TMS320C50 es de 35 ns.

[5] Estos son resultados de la referencia [ALC86]. El TMS32010 es el primer DSP de la familia TMS320 de TI a 16 bits y ciclo de instrucción de 200 ns, además este no incluye la operación de multiplicación acumulación (MAC).

Para la raíz cuadrada se utilizó el algoritmo desarrollado en la referencia [ALC86], este mismo algoritmo programado en el TMS320C10 utiliza 27 instrucciones mientras que en el TMS320C50 se requieren 23 instrucciones. De la tabla 5.1 se observa que un algoritmo que contenga muchas

divisiones o raíces cuadradas incrementa significativamente el tiempo de cálculo, en la tabla 5.2 se aprecia como se incrementan los tiempos.

Algoritmo LMS

Este algoritmo consta de tres ecuaciones, que corresponden a las operaciones 1, 2 y 4 antes mencionadas (ecuaciones 5.36, 5.37 y 5.39), es decir:

$$y(k) = \mathbf{x}^t\mathbf{w} = \mathbf{w}^t\mathbf{x} \qquad \text{en Q11} \qquad 5.42$$

$$e(k) = y(k) - \hat{y}(k) \qquad \text{en Q13} \qquad 5.43$$

$$\mathbf{w_N(k+1)} = \mathbf{w_N(k)} + \mu\, \mathbf{x_N}(k)\, e(k) \quad \text{en Q12} \qquad 5.44$$

$x(k)$ en Q11

Para efectuar el algoritmo LMS en lenguaje ensamblador para el TMS320C50 se requieren 36 + 6p instrucciones, es decir 42 localidades de memoria para el código de programa y 4 + 2p localidades de memoria para almacenamiento de variables (ver anexo B.2).

Algoritmo Rápido de Kalman

Para el algoritmo ARK las variables están en formato de punto entero, el formato para cada variable se determinó efectuando simulaciones de punto entero en lenguaje de alto nivel:

$y(k)$	: señal de entrada a estimar en Q11
$e_p^f(k)$ y $\epsilon_p^f(k)$	: errores forward a priori y posteriori en Q11
$A_p[k]$ y $B_p[k]$	: vectores de parámetros a estimar en Q12
$K_p[k]$	: vector ganancia de Kalman en Q13
$e_p^b[k]$	: error backward a priori en Q11
$\alpha_p^f(k)$	: energía en Q10
$M_p[k]$ y $U_p(k)$	: componentes del vector $K_{p+1}[k]$ en Q13

El algoritmo ARK se efectúa en 268 + 29p = { 110 + 29p + 2*(47 + 2*16) } instrucciones, donde 158 instrucciones corresponden a dos divisiones y "p" es la longitud de los vectores de coeficientes $A_p[k]$ y $B_p[k]$. Como la mayoría de instrucciones en el TMS320C50 se ejecutan en un ciclo, entonces para un ciclo de instrucción de 35 ns se tiene un tiempo de ejecución del algoritmo para N=10, 558*35 ns = 19.53 μs, quedando todavía un tiempo de 105.5 μs dentro de la ventana de muestreo de 125 μs (para frecuencias de muestro de 8K muestras/s).

El ARK programado utiliza 188 localidades de memoria para el código de programa y 19 + 5p localidades de memoria para datos (ver anexo B.2). Se hace notar que de las 189 + 29p, están

incluidas dos llamados a la subrutina de división, y dentro de la subrutina de división existen dos ciclos que repiten una instruccion 16 veces, para el cálculo de las localidades de memoria no se toman las repeticiones sobre las instrucciones ni la inicializacion de variables y del mismo DSP, es decir, 110 + 29 + 47 + 2 = 188. En la referencia [ALC86], el mismo algoritmo ocupa 193 localidades de memoria y utilizando un DSP TMS320C10 (de 200 ns de ciclo de instruccion) para p=10 tarda un tarda 215 μs, esto implica que la generacion TMS320C50 para este algoritmo supera en once veces la al TMS32010, ver tabla 5.3.

Para una señal de voz muestreada a 8 K muestras/s se tiene un tiempo de muestreo de 125 μs, si el TMS320C50 ejecuta cada instruccion en un ciclo (35ns), entonces entre cada muestra adquirida se podrian ejecutar hasta 3,500 instrucciones, lo que es suficiente para los algoritmos adaptables que deben implementarse en tiempo real, el algoritmo de Kalman sólo utiliza el 15.62% del tiempo de muestreo, esto implica que el DSP puede efectuar otros programas mas complejos o largos.

En la tabla 5.2 se resumen los tiempos por iteración para el algoritmo LMS y el ARK para diferentes valores de p, como se observa en la fila de número de instrucciones existe una diferencia entre el número de iteraciones teoricos de las tablas 3.9 y 4.4 y la implementación en una arquitectura real.

	Algoritmo LMS	Algoritmo ARK
Número de instrucciones	36 + 6p	268 + 29p
si p = 10 (tiempo de proceso)	3.36 μs	19.53 μs
si p = 128 (tiempo de proceso)	28.14 μs	139.30 μs
si p = 256 (tiempo de proceso)	55.02 μs	269.20 μs

Tabla 5.2 Comparación de tiempos por iteración para el algoritmo LMS y ARK, utilizando el TMS320C50.

La tabla 5.2 da una idea de las instrucciones necesarias para implementar los algoritmos en el TMS320C50 y su desempeño de acuerdo al ciclo de instrucción del DSP. Si se considera que en una conversación telefónica se tiene un retardo típico de 16 ms, se requieren al menos 128 bloques de retardo en una estructura transversal para la eliminación de eco, entonces si se considera un tiempo de muestreo de 125 μs el algoritmo de Kalman no podría ser aplicable bajo estas circunstancias aunque su desempeño es bueno para "p" pequeño ($p < 110$) tomando en cuenta que hay que efectuar otras instrucciones para la adquisición y entrega de resultados. Sin embargo, el algoritmo LMS puede ser capaz de operar en una estructura transversal con 512 retardos (tiempo de proceso 108.78 μs), además, la respuesta al impulso del eco cambia lentamente lo que implica la factibilidad de utilizar el algoritmo LMS.

Como se aprecia en estos resultados la eficiencia y velocidad del DSP determinan el número máximo de "taps" en la implementación de los algoritmos. Los vectores de entrada y los coeficientes para efectuar productos puntos se almacenan en los bloque B0 y B1 de memoria

interna del TMS320C50, ya que estos bloques permiten el acceso dual en un ciclo de instruccion y cada uno de estos bloque contiene 512 palabras de 16 bits. La variables auxiliares se guardan en el bloque B2

Comparación de algoritmos implementados en el TMS320C50

Siguiendo la metodología de implementación utilizada, los algoritmos que se programaron y probaron en Lenguaje C, se programaron en lenguaje ensamblador del TMS320C50 utilizando redondeo para las operaciones. En la tabla 5.3 se muestran los resultados de estas implementaciones comparando el número de instrucciones, el número de localidades de memoria programa, memoria dato y tiempos de ejecución para un orden p = 10. De nuevo se hace el comentario de observar la diferencia del número de instrucciones por iteración, existiendo una diferencia entre el número de iteraciones teóricas de las tablas 3.9 y 4.4 y la implementación en una arquitectura real.

Algoritmo	Instrucciones por iteración	TMS320C50		Tiempo (μs) por iteración p = 10	
		Loc. de programa ††	Loc. de Datos	TMS320C50	TMS32010[‡]
LMS	36 + 6p	77	24	3.3	
ARK	268 + 29p	306	69	19.53	215.0
FAEST	267 + 27p	373	97	18.795	201.7
FTF	341 + 21p	333	77	19.285	223.0
SFTF	479 + 26p	430	95	25.865	
Botto	1097+28p	575	81	48.0	444.2

Tabla 5.3 Comparación de tiempos para varios algoritmos.

†† Las localidades de programa incluyen las operaciones de inicialización de variables y configuración del TMS320C50, las cuales no son instrucciones que se repiten por iteración, por lo que no entran en los calculos del tiempo, este número de instrucciones por iteración crece de acuerdo al número de divisiones y ciclos del algoritmo.

‡ Resultados reportados en la referencia [ALC86].

De la tabla 5.3 se observa que la razon de tiempos de los algoritmos programados en el TMS32050 y el TMS320C10 es en promedio de 10.63 a uno, a pesar de que la razon entre ciclos de instruccion es de 5.71 a uno, es decir, que las instrucciones del TMS320C5O en conjunto disminuyen aun mas los tiempos, sin embargo, el TMS320C10 es más barato a razón de 25 a uno. La razon de costo sería un elemento importante a considerar en una implementación real.

Generación de señales de prueba

Las simulaciones para predicción se hicieron a una señal de ruido blanco filtrada por medio de un modelo autoregresivo (AR) de orden p = 10 con señales aleatorias y señales senoidales con ruido agregado (fig. 5.9):

$$H(Z) = 1 A(z) \qquad 5.45$$

$$A(z) = 1 - 1.3728z^{-1} + 1.6260z^{-2} - 1.6110z^{-3} + 1.5013z^{-4} - 1.245z^{-5} + 0.8914z^{-6} - 0.5519z^{-7} + 0.2105z^{-8} + 0.0475z^{-9} + 0.0144z^{-10} \qquad 5.46$$

que proviene de un conjunto de polos:

$0.85\ e^{\pm 0.1\pi} = 0.8040 \pm 0.2627j$
$0.89\ e^{\pm 0.3\pi} = 0.5231 \pm 0.7200j$
$0.87\ e^{\pm 0.5\pi} = 0.0000 \pm 0.8700j$
$0.89\ e^{\pm 0.7\pi} = -0.5231 \pm 0.7200j$
$0.20\ e^{\pm 0.7\pi} = -0.1176 \pm 0.1690j$

La señal filtrada se carga en la memoria de la tarjeta DSK que realiza el proceso y entrega resultados corridos en tiempo real.

En la figura 5.8 se observan los resultados obtenidos de la simulación en aritmética de punto flotante (s_flot) y aritmética de punto fijo (s_ent) para el algoritmo ARK, donde para un factor de olvido $\lambda = 0.95$ se tienen resultados divergentes en la implementación en punto entero. Se observa claramente que cuando el algoritmo llega a la muestra 95 diverge abruptamente si se implementa en aritmética de punto entero, mientras que en punto flotante aún no diverge, también desde la muestra 60 se observa la tendencia a diverger en aritmética de punto entero. La simulación en punto en flotante para $\lambda = 0.995$ sirve para observar un mejor comportamiento del algoritmo cuando λ tiende a uno.

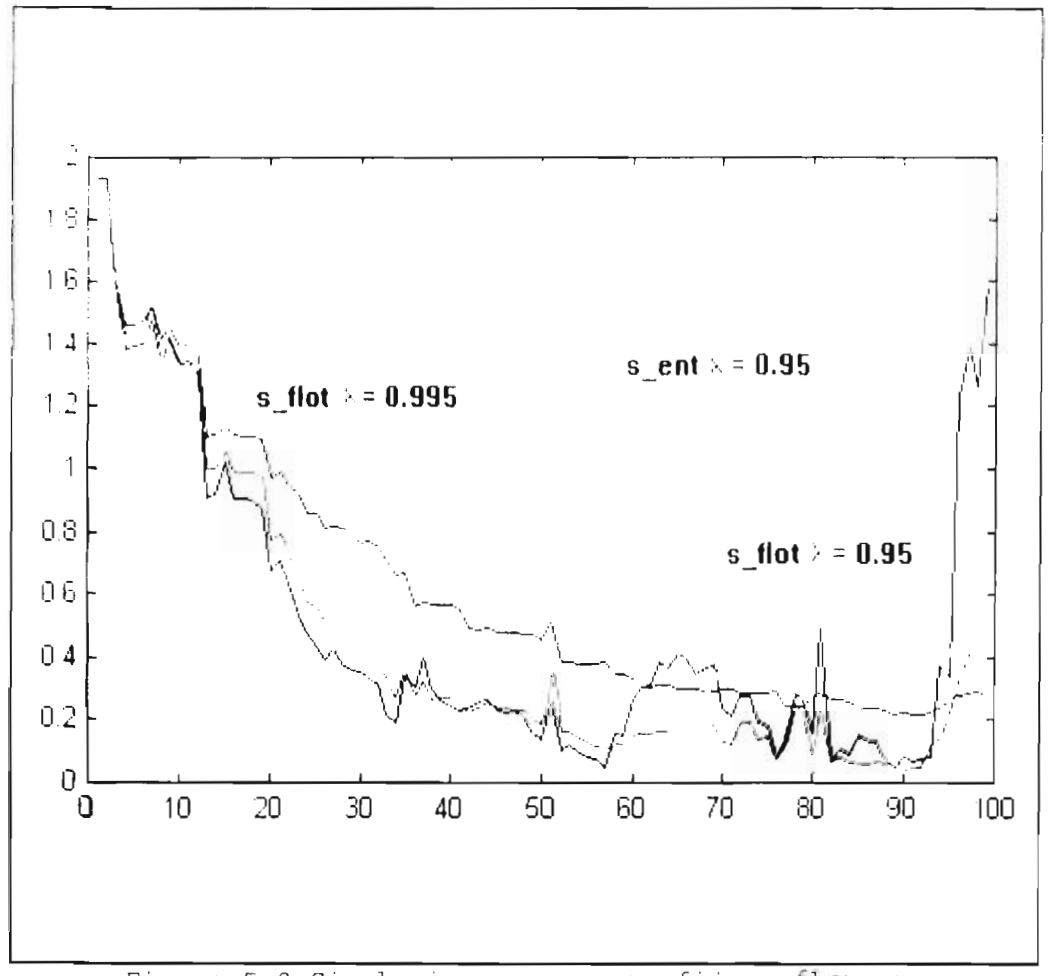


Figura 5.8 Simulaciones en punto fijo y flotante

V.6 RESULTADOS

En la figura 5.9a se muestran los resultados de salida de predicción de una señal senoidal sin ruido utilizando el ARK ejecutado en el TMS320C50 sobre la tarjeta DSK. Para apreciar el resultado de la predicción (señal estimada y.out) al inicio de la adaptación, en la figura 5. 10 se muestra esta parte inicial, donde se evidencia que una vez que el error converge, la señal estimada tiende a ser igual a la señal de entrada.

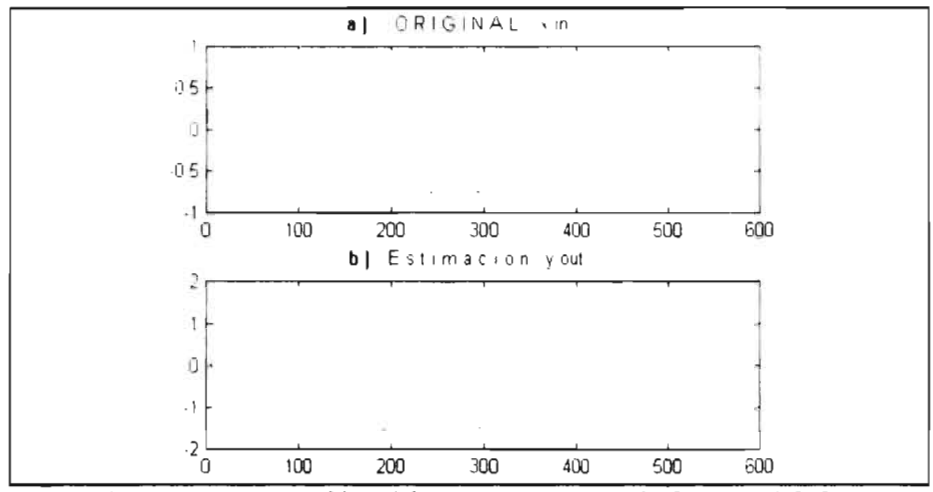


Figura 5.9 predicción para una señal senoidal.

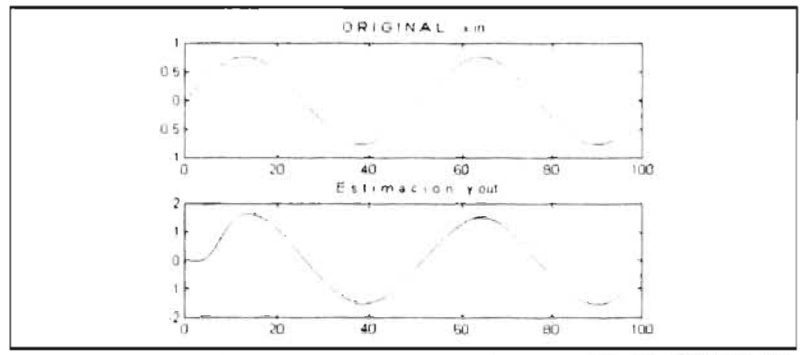


Figura 5.10 Implementaciones de predicción en el TMS320C50

Comparaciones del error cuadrático

En la siguiente grafica se muestra una comparación del error cuadratico normalizado de los algoritmos programados tanto en lenguaje de alto nivel como en ensamblador, utilizando la misma señal de entrada y las mismas condiciones iniciales para prediccion.

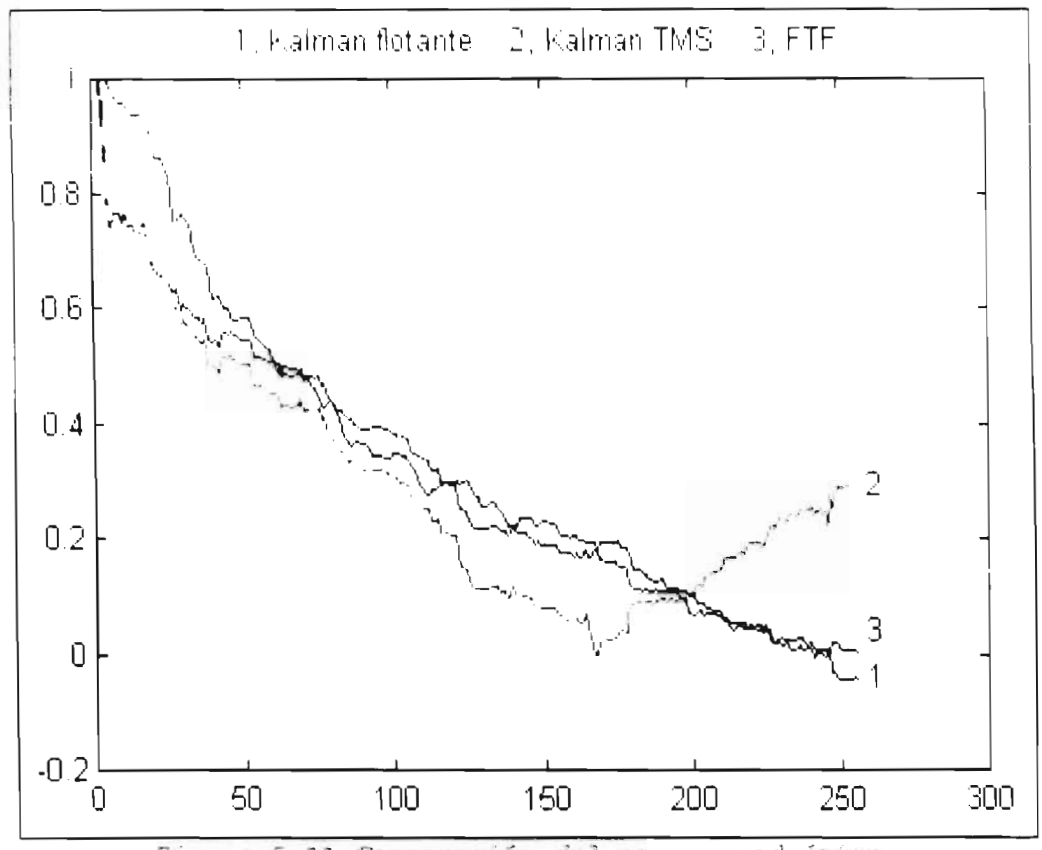


Figura 5.11 Comparación del error cuadrático.

La curva 1: Corresponde al error cuadrático para al algoritmo Rápido de Kalman programado en alto nivel, con $\lambda = 0.995$ y $\alpha^1(0) = 1$.

La curva 2: Corresponde al error cuadrático para el ARK en el TMS320C50 con $\lambda = 0.98$ y $\alpha^1(0) = 1.0$.

La curva 3: Corresponde al error cuadrático para el algoritmo transversal rápido FTF programado en el TMS32OC50 con $\lambda = 0.99$, $\alpha^1(0) = \alpha^2(0) = 1$.

Como se observa en la figura 5.11 el algoritmo FTF tarda más en converger, tal como se mostró en figura 3.8. La implementación del algoritmo ARK implementado en el TMS320C50, presenta

una convergencia mayor ya que se utilizó una $\lambda = 0.98$, a pesar de ello en la muestra 175 existe divergencia debida los errores de redondeo y que la lamda utilizada menor, sin embargo como se mostró en la figura 5.8 para simulaciones en aritmética de punto entero para una λ menor las divergencia se presenta mucho antes.

De los resultados anteriores se puede concluir:

- Que la eleccion de un algoritmo para operar en un DSP está determinado por los tiempos de ejecución y las necesidades de memoria.

- Un algoritmo que utilice muchas operaciones como sumas multiplicaciones, sumas y divisiones, implica recurrir a muchas lecturas y escrituras a memoria, haciendo que su ejecución tarde más tiempo que un algoritmo con operaciones vectorizables. En la referencia [ALC86] se concluye que el algoritmo rápido de Kalman respecto de otros rápidos, es el más conveniente de implementarlo en un DSP desde el punto de vista de los tiempos de ejecución (ver tabla 5.3).

- La utilización de DSP en la implementación de algoritmos adaptables ofrece ventajas como: la ejecución rápida y eficiente de productos internos entre vectores, un amplio conjunto de instrucciones que facilita la programación, cálculo rápido de operaciones aritméticas y gran cantidad de herramientas de desarrollo.

- La arquitectura de un DSP es diferente a la de un microprocesador o microcontrolador convencional, sobre todo por su capacidad de procesamiento en paralelo, operaciones en pipeline, ejecución rápida de productos y suma de productos.

- De las implementaciones llevada a cabo en el TMS320C50, el algoritmo que resultan tener mejor desempeño en cuanto a tiempos, numero de instrucciones y localidades de memoria (ver tabla 5.3) son el ARK y el Kalman a posteriori (FAEST), sin embargo, en lo personal considero que de estos dos el más fácil de programar es el ARK. El algoritmo de Botto presenta extremada cantidad de instrucciones y operaciones que dan lugar a muchos problemas de inestabilidad.

- Hasta este punto no se puede afirmar que la implementación de los algoritmos de filtrado adaptable está resuelto ya que estos deben de trabajar en tiempo real, sin embargo se presenta resultados que muestra su posibilidad real de implementación. Por otro lado en la realidad existen equipos donde los algoritmos operan en tiempo real pero el fabricante no da ninguna información al respecto.3

V.7 ARQUITECTURA PROPUESTA PARA OPERACION EN TIEMPO REAL

Con las pruebas realizadas hasta este momento y siguiendo con la metodología de implementación se hace una propuesta para operar en una aplicación concreta. Partiendo de la arquitectura básica de un DSP con las características mencionadas, sólo se agregan pocos dispositivos para la operación real, entre estos están: las interfaces necesarias para conectarse con el mundo exterior, la adquisición de señales, señales de control y nuevos bloques en el programa para la realización de la implementación completa.

En el caso de la cancelación de eco, el programa debe ejecutarse una vez por período de muestreo $T_s = 125\ \mu s$, el DSP es interrumpido cada período T_s, el programa debe enviar señales de control, leer la muestra del conversador lejano y(k) y la del conversador cercano x(k) y después del proceso escribir al conversador cercano.

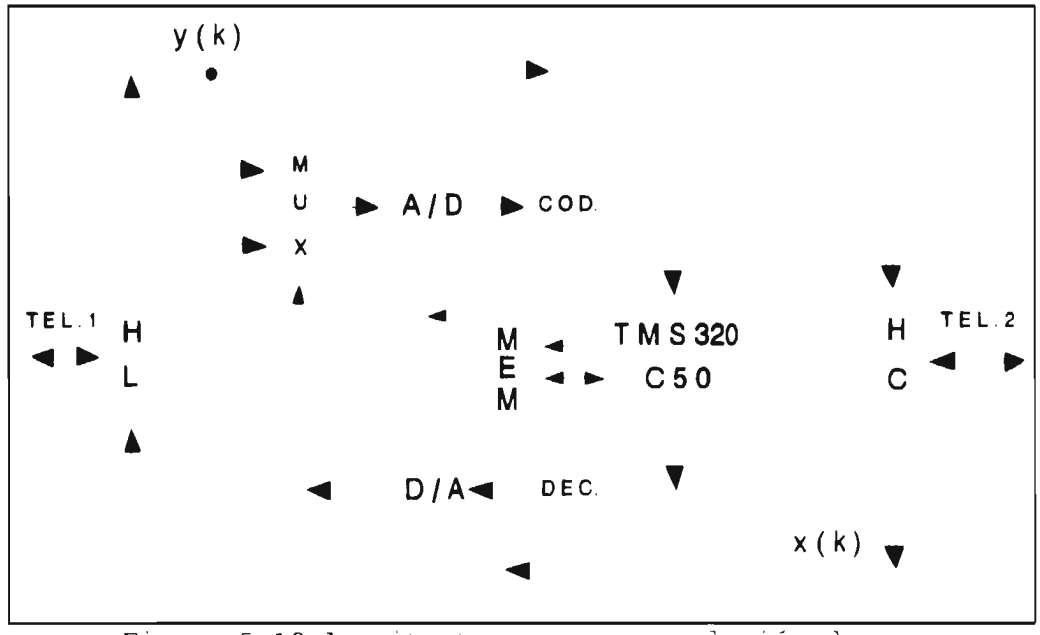


Figura 5.12 Arquitectura para cancelación de eco

En la arquitectura propuesta en la figura 5.12, se tiene el sistema general de una línea telefónica de dos cables conectados a través de dispositivo híbridos (HL y HC) a un sistema de cuatro cables, en esta parte es donde se realiza la eliminación del eco.

Las entradas pasan por un multiplexor analógico para reconocer de cual se trata, se tiene un convertidor A/D para obtener muestras digitales, un bloque codificador para obtener la muestras bajo un esquema de codificación que puede ser la ley μ, en el bloque TMS320C50 se ejecuta el programa que lleva todo el control y realiza el proceso, cuando el programa estima el eco de

la señal de entrada, este es restado de la señal cercana y luego el resultado es decodificado y enviado al convertidor D A.

En esta parte se supone que los programas y las pruebas realizadas son válidos para la solución del problema, es decir, que estos programas son los que efectúan el proceso principal, entonces solo se tienen que agregar los bloque de control, adquisición y entrega de resultados.

En este capitulo se ha abordado la evaluacion de los algoritmos en aritmética de punto entero utilizando un DSP, la problemática que esto implica, la evaluación de los errores, se han mostrado implementaciones para los formatos enteros dedicados y comparaciones de su implementación en lenguaje ensamblador. Los resultados importantes obtenidos son la implementación de las operaciones y los algoritmos de filtrado adaptable, así como una comparación con las mismas implementaciones con otro DSP de la misma familia.

Como conclusión del capitulo se tiene que con el TMS320C50 se logran tiempos de ejecución de los algoritmos en promedio 10 veces más rápido que el TMS320C10, sin embargo este último DSP es 25 veces más barato. Dependiendo si la aplicación implica operar a altas velocidades se pueden sacrificar costos. El algoritmo ARK respecto a los otros algoritmos rápidos RLS, es el más conveniente de implementar en el DSP TMS320C50 desde el punto de vista de tiempos de ejecución y programación.

Como culminación de la metodología utilizada se presentó en el la figura 5.12 una propuesta de arquitectura que se sustenta en todo el desarrollo del trabajo.

En el siguiente capítulo se dan las conclusiones y perspectivas generales de este trabajo con base a los resultados obtenidos en los diferentes capítulos y del estudio y análisis realizado.

Capítulo VI

CONCLUSIONES Y PERSPECTIVAS

CONCLUSIONES Y PERSPECTIVAS

Durante el desarrollo del presente trabajo se ha seguido una metodología que permite tener una visión general del problema de filtrado adaptable, desde su definición hasta llegar a la obtención de resultados ya sea a nivel algoritmico, soluciones software, micro-software o software/hardware. Esta metodología ha permitido definir y proponer soluciones a problemas de comunicaciones que se pueden afrontar utilizando filtrado adaptable.

El campo de aplicación de los Filtros Adaptables es muy extenso, sobre todo en la actualidad con el crecimiento de los sistemas digitales y las comunicaciones. Los algoritmos de filtrado adaptable son una herramienta muy poderosa, utilizada en varias áreas de telecomunicaciones para eliminación de problemas difíciles de resolver con filtros fijos. Esto se ha comprobado en los resultados obtenidos en el capítulo tres de este trabajo. Los Filtros Adaptables comparados con los filtros fijos presentan un mejor desempeño cuando se desconocen las características de la señal de entrada o sus estadísticas varían en el tiempo.

La utilización de cualquiera de los algoritmos de filtrado adaptables propuestos depende de la aplicación particular, es decir, si se dispone de mucho tiempo para operar o si los tiempos son cortos, se necesitan algoritmos que sean rápidos. El algoritmo LMS puede ser una solución sencilla cuando se trabaja con señales que cambian muy lentamente con el tiempo.

La desventaja en el uso del algoritmo LMS es su velocidad de convergencia y que es sensible al mal acondicionamiento de la matriz R. Por otro lado, los algoritmos rápidos RLS son rápidos en converger pero presentan problemas de inestabilidad y su implementación en ensamblador necesita muchas operaciones. La mayoría de los algoritmos rápidos que trabajan en estructuras transversales son derivados del algoritmo rápido de Kalman, éstos difieren esencialmente de la forma en que se establecen los cálculos de las variables intermedias.

Con la expansión de las comunicaciones de alta velocidad y las comunicaciones móviles a altas frecuencias (6 a 30 Ghz), la atenuación y dispersión de la transmisión es causada por los efectos atmosféricos: los algoritmos rápidos RLS son una alternativa de solución a los problemas tales como la cancelación de eco, la igualación de canal y la eliminación de ruido, ya que estos efectos son causados por fenómenos variables e impredecibles.

La convergencia de los algoritmos está ligada a las condiciones iniciales y la sensibilidad de los errores de cálculo a la longitud de palabra del sistema digital donde se ejecuten los algoritmos. Una fuente de inestabilidades de los algoritmos rápidos es el valor de factor de olvido λ pequeño, sin embargo, si $\lambda = 1$ el algoritmo no sigue las variaciones de la señal de entrada. Un forma de hacer converger los algoritmos es hacer λ pequeño antes de la convergencia y después hacerlo muy cercano a uno, [ALC86].

La convergencia del error cuadrático de los algoritmos se utiliza como un criterio de comparación del desempeño numérico de los mismos. Los errores de cuantización destruyen la estabilidad numérica de los algoritmos.

Los filtros adaptables son fáciles de implementar en arquitecturas para procesamiento digital de señales, sobre todo porque involucran operaciones de multiplicación acumulación (productos internos) y deben operar en tiempo real, ya que los DSP surgieron bajo la necesidad de efectuar eficientemente operaciones de multiplicación acumulación.

Una arquitectura de procesamiento digital de señales a 16 bits puede dar soluciones adecuadas para la implementación de los algoritmos adaptables. Las implementaciones en aritmética de punto entero en un DSP da lugar a realizar operaciones aritméticas mucho más rápidas, tal es el caso de efectuar el producto punto entre dos vectores, para el cual un DSP tiene un hardware e instrucciones especializadas.

Una medida de los requerimientos de cálculo en procesamiento digital de señales es el desempeño en términos de los ciclos por multiplicación de un procesador, un DSP normalmente utiliza un ciclo de instrucción por multiplicación y es lo que los hace adecuados para estas aplicaciones.

Con el DSP TMS320C50 se logran tiempos de ejecución de los algoritmos en promedio 10 veces más rápido que el TMS320C10, y la operación de raíz cuadrada se ha logrado reducir en 3 instrucciones, sin embargo, el DSP TMS320C10 es 25 veces más barato. Dependiendo si la aplicación implica operar a altas velocidades se pueden sacrificar costos.

La implementación de los algoritmos adaptables sería ideal efectuarlas sobre arquitecturas para DSP que operen en aritmética de punto flotante para evitar problemas de sobreflujo y otros problemas numéricos, con el inconveniente que una arquitectura en aritmética en punto flotante incrementa el costo de la implementación, sin embargo, con los avances actuales de la tecnología los DSP de punto flotante se están haciendo más accesibles y más rápidos, aunque siempre habrá aplicaciones que necesiten implementarse en aritmética de punto fijo.

Un algoritmo que utilice muchas operaciones como sumas multiplicaciones, sumas y divisiones, implica recurrir mucho a lecturas y escrituras a memoria, haciendo que su ejecución tarde más tiempo que un algoritmo con operaciones vectorizables. El algoritmo rápido de Kalman respecto de otros rápidos RLS, es el más conveniente de implementarlo en un DSP desde el punto de vista de los tiempos de ejecución.

El incremento del número de bits para la implementación de una algoritmo adaptable en aritmética de punto entero, no cambia la estabilidad numérica del algoritmo a partir de un número mínimo de bits necesarios, es decir, que la estabilidad del numérica depende exclusivamente del algoritmo, [LJU85],[CIO87].

La evaluacion del desempeño de los algoritmos es todavía un area de investigación abierta, donde se puede continuar investigando desde cinco puntos de vista: convergencia, inicialización, complejidad mínima de operaciones, estabilidad numérica y efectos de la longitud finita de palabra. Es decir, que una meta a futuro de los algoritmos adaptables es hacerlos más robustos y rapidos para implementarlos en aplicaciones en tiempo real.

La perspectiva futura de los algoritmos de filtrado adaptable es su implementacion en comunicaciones de alta velocidad, esto significa que este tema es inagotable y puede seguirse explotando desde diferentes enfoques: la búsqueda y mejoramiento de algoritmos, la simulación de soluciones en problemas reales y la implementación en arquitecturas dedicadas. Todavía, más allá de una arquitectura con un DSP, existe el área de procesamiento en paralelo donde se puede encontrar un campo adecuado para la implementación de los algoritmos adaptables, cuando las aplicaciones requieren de respuestas más rápidas. Además, en la actualidad con la rápida evolución de la tecnología, los nuevos DSPs son más poderosos, más rápidos y dedicados a áreas especificas de comunicaciones, tal es el caso de la nueva familia TMS320C6x que opera a 200 Mhz, 1600 MIPS y es capaz de ejecutar hasta ocho instrucciones en un ciclo de 5 ns, este es un DSP que podría mejorar el desempeño en las aplicaciones de algoritmos de filtrado adaptable para problemas de telecomunicaciones.

En este trabajo se ha presentado el desarrollo y análisis de los algoritmos de filtrado más utilizados lo que nos permite hacer una introducción y entendimiento del tema.

Se ha presentado la implementación de los algoritmos de filtrado adaptable mas conocidos en aritmética de punto flotante y punto entero en una arquitectura para procesamiento digital de señales el DSP TMS320C50, con resultados que validan las simulaciones en aritmética de punto flotante. Se hace notar que la implementación en un DSP de estos algoritmos se encuentra en muy escasas publicaciones o estudios.

Se han hecho comparaciones y evaluaciones entre algoritmos, mostrando la complejidad de implementación, sus características de convergencia, la cantidad de operaciones teóricas por iteración, la cantidad de operaciones reales al programarlos en una arquitectura dedicada, y la robustes de los algoritmos.

A partir del desarrollo de este trabajo se pueden emprender aplicaciones reales en la solución de los problemas más típicos en telecomunicaciones.

BIBLIOGRAFIA

BIBLIOGRAFIA

|ALC86| Alcántara S. Rogelio. Implantation d'algorithmes rapides sur des processeur de traitement du signal. Thèse de Docteur Ingènieur. ENST Paris. France. Septembre 1986.

|ALC87| Alcántara R., Prado J., Gueguen C. , Boudy J. & Favier G. Simulation sur ordinateur des effects de quantification dans les algorithmes de filtrage adaptifs. Gretst Nice. June 1987.

|ALC92| Alcántara S. R. Metodología de diseño de sistemas para el procesamiento digital de señales. DEPFI. UNAM. México D. F. 1992.

|ALE86| Alexander T. Adaptive Signal Processing. Springer-Verlag 1986.

|ALP86| Alcántara R., Prado J. & Gueguen C. Fixed-Point Implementation of the Fast Kalman Algorithm: Using a TMS32010 microprocessor. Signal Processing III: Theories and Applications I.T. Young et al. Editors. Elsevier Scinece Publishers B. V. (North-Holland) S.24 EURASIP. september 1986. Pg. 1335-1338.

|ARR96| Ricardo Arroyo Mendoza y Julio Enrique Orozco Torrentera. Metodología de diseño de sistemas de procesamiento digital de señales. Tesis de licenciatura. Facultad de Ingeniería. UNAM. México D.F. 1996.

|ATA92| Atay R., P. Baylon & M. Najim. Implementation of the stabilized fast transversal filter algorithm on fixed point DSP. ICASSP 1992 IEEE pp- IV 249-IV 252.

|BEN88| Benallal A. & Gilloire A. A new Method to stabilize RLS algorithms based on a fist-order model of the propagation of numerical errors. D1.4 IEEE 1988. Pg. 1373-1376.

|BOT89| Botto J. L. & Moustakides G. Stabilizing the Fast Kalman Algorithms. IEEE Trans. on acoustics, speech signal processing. Vol. 37 N. 9, sep 1989.

|CAN88| Campos M. & Antoniou A. A robust quasi-Newton adaptive filtering algorithm. Departament of Electrical and Computer Engineering, University of Victoria, Canada 1988.

|CAR84| Caraiscos Christos. A Roundoff Error Analysis of the LMS Adaptive Algorithm. IEEE Trans. Acoust., Speech Signal Processing, vol. ASSP-32, No. 1, pp 34-41, feb. 1984.

|CAR83| Carayannis G., Manolakis D. & Kaloupsidis N. A Fast Sequential Algorithm for Least-Squares Filtering and Prediction. IEEE Trans. on Acoustics, Speech and

Signal Processing. Vol. ASSP 31, No. 6 pg. 1394-1402, dec. 1983.

[CHA97] Chansarkar M. and Desai U. A robust recursive least squares algorithm. IEEE Transactions on signal processing. Vol. 45, No. 7, july 1997.

[CIO84] Cioffi J. M. Fast Transversal Filters for communications applications. Ph. D. Dissertation Thesis of Stanford University. March 1984.

[CIO87] Cioffi J.M. Limited-Precision Effects in adaptive Filtering. IEEE Trans. CAS-34, july 1987

[CIO90] Cioffi J.M. A fast echo canceller initialization method for the CCITT V.32 modem. IEEE Trans. on communications, Vol. 38, No. 5, May 1990.

[CRO94] Cronin Dennis. Examining Audio DSP Algorithms. Dr. Dobb's Journal, July 1994.

[DEF88] DeFatta D., Lucas J. & Hodgkiss W. Digital Signal Processing a system design approach. John Wiley & Sons . Singapore 1988.

[FAL78] Falconer D. D. and L. Lung. Application of fast Kalman estimation to adaptive equalization. IEEE Trans, Commun. Vol. COM-26, pp. 1439-1446.

[FAB85] Fabre P. & Gueguen C. Fast recursive least-squares algorithms: preventing divergene. IEEE, ICASSP 1985.

[FAB86] Fabre P. & Gueguen C. Improvement of the fast recursive least-squares via normalization a comparative study. IEEE Trans. on acoustics, speech and signal processing. Vol. ASSP-34 No. 2 april 1986.

[FAL78] Falconer D & L. Ljung. Application of fast Kalman estimation to adaptive equalization. IEEE Trans. Commun. Vol. COM-26, No. 10, October 1978.

[FAW96] Fawzy Wagdy M.ahmoud & Ng Wai-Man. Validity of Uniform Quantization Error Model for Sinusoidal Signals Without and With Dither. IEEE Trans. on Instrumentation and measurement. Vol. 38 No.3 june 1989.

[FLO] Flores de la Mota Idalía. Apuntes de matemáticas aplicadas. DEPFI, Facultad de Ingeniería, UNAM. México D. F.

[FRI82] Friedlander B. Lattice Filters for Adaptive Processing. IEEE, vol. 70, N. 8, pg. 829-867, Aug. 1982.

[GRO92] Grover Brown & Hwang Patrick Y.C. Introduction to random signals and applied

Kalman Filtering. Jhon Wiley . U.S.A. 1992.

[HAR78] Harris F. On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform. Proceedings of the IEEE Vol. 66 No. 1 Jan. 1978.

[HAY89] Haykin S. Modern Filters. Macmillan Publishing Company. USA 1989.

[HAY91] Haykin S. Adaptive Filter Theory. Prentice-Hall. U.S.A. 1991.

[HEJ92] Hejn Konrad W & Morling R. A Semifixed Frequency Method for Evaluating the Effective Resolution of A/D Conveters. IEEE Trans. on Instrumentation and Measurements Vol. 41 No. 2 april 1992.

[IEEE76] IEEE Trans. Antennas Propagation, Special issue on adaptive antennas. Vol. AP-24, No. 5, Sept. 1976.

[IEEE76] IEEE Trans. Antennas Propagation, Special issue on adaptive antennas. Vol. AP-12, mar. 1964.

[MAK75] Makhoul J. Linear Prediction: A Tutorial Review. Proc. IEEE, Vol. 63 No. 4, pp 561-580. Apr. 1975.

[KIM88] Kim D. & Alexander W. E. . Stability Analysis of the Fast RLS Adaptation Algorithm. D1.1 IEEE 1988.

[KOL39] Kolmogrorov A. N. Sur l'interpolation et extrapolation des suites stationaires. C. R. Acad. Sci., Paris. Vol. 28, pp. 2034-2045.

[KUB88] Kubin G. Stabilization of the RLS Algorithm in the absence of persistent excitation. IEEE 1988.

[K&T93] Kalouptsidis N. & Theodoridis S. Adaptive System Identification and Signal Processing Algorithms. Prentice Hall. Cambridge G. B. 1993.

[LEE81] Lee D. T., Morf M. and Friedlander B. Recursive least squares ladder estimation algorithms. IEEE Trans. Accoust., Speech and Signal processing. Vol ASSP-29, No. 3, June 1981.

[LIN84] Lin David W. On digital implementation of the Fast Kalman Algorithms. IEEE transactions on acoustics, speech and signal processing. Vol. ASSP-32 No. 5. October 1984.

[LIU69] Liu B. and Toyohisa Kaneko. Error analysis of digital filter realizad with floating-point arithmetic. Proceeding of the IEEE Vol. 57 No. 10, October 1969.

[LIU69] Liu B. Effect of finite word length on the accuracy of digital filter a review. IEEE Transactions on circuit theory. Vol. CT-18, No. 6, November 1971.

[LJU83] Ljung S. Fast algorithms for integral ecuations and least-square identification problems. Ph. D. dissertation. Linkoping Univ. Sweden 1983.

[LJU85] Ljung S. & Ljung L. Error propagation properties of recursive Least-squares Adaptation Algorithms. Automatica vol. 21, No. 2, Sweden 1985.

[LMF78] Ljung L., M. Morf & D. Falconer. Fast calculation of gain matrices for recursive estimation schemes. Int. J. Control, Vol. 27, No. 1, pp. 1-19, January 1978.

[LON90] Long G. and F. Ling. A new complex system identification method and its application to echo canceller fast initialization. In Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, Albuquerque, NM. Apr. 1990. pp 1671-1674.

[MAK77] Makhoul J. Stable and Efficient Lattice Methods for Linear predition. IEEE Transaction Acoustic, Speech, Signal Processing. Vol. ASSP-25 N.5 pg. 423-428, oct. 1977.

[MES82] Messerschmitt D. Echo cancellation in speech and dat transmission. IEEE selected areas in communications. Vol. SAC-2, No. 2. pp. 283-297. Mar. 1982.

[MHC89] Messerschmitt D., Hedberg D., Cole C., Haoui A. & Winship P. Digital Voice Echo Canceller with a TMS32020. Texas instruments 1989.

[OPP72] Oppenheim A. & Weinstein C. Effects of Infinite Register Length in Digital Filtering and Fast Fourier Transform. Proceedings of the IEEE, Vol. 60, No. 8, August 1972.

[ORF88] Orfanidis S. J. Optimun signal processing an introduction. 2nd. edition. McGraw-Hill. U.S.A. 1988.

[PAH95] Pahlavan Kaveh & Levesqe Allen H. Wirles Information networks. John Wiley and sons Inc. USA 1985.

[PRN92] Proakis J., Rader C. Lin F. & Nikias C. Advanced Digital signal Processing. Maxwell Macmillan. Ontario Canada 1992.

[PRO92] Proakis J. & Manolakis D. Digital Signal processing Principles, Algorithms and Aplications. Macmillan-Maxwell. Singapur 1992.

[PRO83] Proakis J. G. Digital Communications, MacGraw-Hill Inc. New York, 1983.

[QI96] Qi David. Acoustic echo canellation, algorithms and implementation on the TMS320C8x. Aplication report. SPRA063 Texas Instruments. USA 1996.

[RAB75] Rabiner L. & Gold B. Theory and application of Digital Signal Processing. Prentice-Hall . U.S.A. 1975

[SAN94] Sánchez V Jesús. Predicción y evaluación del desempeño de algoritmos adaptables implantados en máquinas paralelas: aplicación al caso de eliminación de eco en telefonía. 1er. Taller internacional de procesamiento paralelo. IIMAS UNAM agosto 1994.

[SIB87] Sibul L. Adaptive signal processing. IEEE press. USA 1987.

[SLO88] Slock D.T. & Kailath T. Numericcally stable fast recursive least square transversal filters. In Proc. ICASSP 88 Conf. (new York), Apr. 1988. pp 1365-1368.

[SLO91] Slock D.T. & Kailath T. Numerically Stable Fast Transversal Filter for Recursive Least Squares Adaptive Filtering. IEEE trans. on Signal processing. Vol. 39 No.1 Jan. 91.

[SRI77] Sripad A. & Snyder D. A necessary and sufficient condition for quantization errors to be uniform an white. IEEE Trans on Acoustics, speech and signal processing. Vol. ASSP-25 No. 5, octuber 1977.

[STR90] Strang G. Algebra lineal y sus aplicaciones. Ed. Addison-Wesley Iberoamericana. México D. F. 1990.

[TI89] Texas Instruments. TMS320C2x User's Guide. USA. 1989.

[TI90] Digital Signal Processing Aplications with the TMS320 Family. Theory, Algorithms and implementations Vol 1,2,3. Texas instruments, USA 1989,1990.

[TI91] Digital Signal Processing Aplications with the TMS320C30 Evaluation Module. Selected Application Notes. Texas instruments, USA 1991.

[TI93] Texas Instruments. TMS320C50 User's Guide. USA. 1993.

[TI94] Texas Instruments. TMS320C3x User's Guide. USA. 1994.

[TI94] Texas Instruments. Telecommunications Applications with the TMS320C5x, Aplication Book. Digital Signal Processing Products. Texas Instruments. U.S.A. 1994.

[TID94] Texas Instruments. TMS320C5x DSP Starter Kit. User's Guide. USA 1994.

[TI97] Texas Instruments. TMS320C50 User's Guide. USA. 1997.

[TOP88] Toplis B. and Pasupathy S. Tracking improvements in fast RLS algorithms using a variable forgetting Factor. IEEE Transactions on Acoustics and Signal Processing. Vol. 36 No. 2 Feb. 1988.

[TOR97] Torres F. Jose E. Estudio y comparación de algoritmos de filtrado adaptable para la igualación de canal. Tesis de maestría en ingeniería eléctrica. DEPFI. UNAM. México D. F. Junio de 1997.

[WAR82] Warren P. & Griffiths L. Finite Word Length Effects in the LMS adaptive Algortihm. Sixteenth conff. on circuits, Systems & Computer, Pacific Grove. Calif. Nov 1982.

[WID67] Widrow B., et al, Adaptive antenna systems. Proc. IEEE, Vol. 55, No. 12, pp 2143-2159, Dec. 1967.

[WID75] Widrow B, J. R. Glover, J. M. McCool, J. Kaunitz, C. S. Williams, R. Hearn, J. Zeidler, E. Dong, and R. Goodlin. Adaptive noise Cancelling: principles and applications. Proc. IEEE, Vol. 63, No.12 pp. 1692-1716, Dec. 1975.

[WID85] Widrow B. & Stearns S. Adaptive Signals Processing. Prentice-Hall Inc. New Jersey 1985.

[WID96] Widrow B. & E. Walach. Adaptive Inverse Control. Prentice Hall, New Yersey 1996.

ANEXOS

ANEXO A
HERRAMIENTAS Y DESARROLLOS

A.1 LEMA DE INVERSION MATRICIAL

Este lema es de gran importancia para el desarrollo de algoritmos recursivos, ya que permite encontrar la inversa de matrices que se pueden representar en forma particionada, ademas, las relaciones que se obtienen entre los elementos de la partición minimizan ciertas expresiones utilizadas en algoritmos recursivos.

El lema de inversión matricial permite obtener la matriz inversa de una matriz particionada en términos de los elementos de la partición.

Dada la matriz X y su inversa X^{-1}:

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \qquad X^{-1} = \begin{bmatrix} L & M \\ N & P \end{bmatrix} \qquad \text{A.1.1}$$

entonces el producto de X y su inversa da la matriz identidad:

$$X\,X^{-1} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} L & M \\ N & P \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} AL+BN & AM+BP \\ CL+DN & CM+DP \end{bmatrix} \qquad \text{A.1.2}$$

desarrollado el producto, se puede demostrar que:

$$X^{-1} = \begin{bmatrix} (A-BD^{-1}C)^{-1} & -A^{-1}B(D-CA^{-1}B)^{-1} \\ -D^{-1}C(A-BD^{-1}C)^{-1} & (D-CA^{-1}B)^{-1} \end{bmatrix} \qquad \text{A.1.3}$$

Los elementos de x^{-1} también pueden obtenerse por las relaciones:

$$(A-BD^{-1}C)^{-1} = A^{-1} + A^{-1}B(A-BD^{-1}C)^{-1} \qquad \text{A.1.4}$$

$$(D-CA^{-1}B)^{-1} = D^{-1} + D^{-1}C(A-BD^{-1}C)^{-1} \qquad \text{A.1.5}$$

$$A^{-1}B(D-CA^{-1}B)^{-1} = (A-BD^{-1}C)^{-1}BD^{-1} \qquad \text{A.1.6}$$

$$D^{-1}C(A-BD^{-1}C)^{-1} = (D-CA^{-1}B)^{-1}CA^{-1} \qquad \text{A.1.7}$$

Desarrollando los productos de A.1.2 .

$$CM + DP = I \qquad \text{A.1.8}$$

$$AM + BP = 0 \quad ==> \quad M = -A^{-1}BP \qquad \text{A.1.9}$$

$$-CA^{-1}BP + DP = I \qquad \text{A.1.10}$$

$$(D - CA^{-1}B)P = I \qquad \text{A.1.11}$$

$$P = (D - CA^{-1}B)^{-1} \qquad \text{A.1.12}$$

$$M = -A^{-1}B(D - CA^{-1}B)^{-1} \qquad \text{A.1.13}$$

$$AL + BN = I \qquad \text{A.1.14}$$

$$CL + DN = 0 \quad ==> \quad N = -D^{-1}CL \qquad \text{A.1.15}$$

$$AL + B(-D^{-1}CL) = I \qquad \text{A.1.16}$$

$$AL - BD^{-1}CL = I \qquad \text{A.1.17}$$

$$(A - BD^{-1}C)L = I \qquad \text{A.1.18}$$

$$L = (A - BD^{-1}C)^{-1} \qquad \text{A.1.19}$$

$$N = -D^{-1}C(A - BD^{-1}C)^{-1} \qquad \text{A.1.20}$$

con lo que se demuestra la matriz inversa de X.

Para las relaciones:

$$D^{-1}C(A-BD^{-1}C)^{-1} = (D-CA^{-1}B)^{-1}CA^{-1} \qquad \text{A.1.21}$$

de ecuación A.1.14

$$L = A^{-1} - A^{-1}BN \qquad \text{A.1.22}$$

sustituyendo en A.1.15

$$C(A^{-1} - A^{-1}BN) + DN = 0 \qquad \text{A.1.23}$$

factorizando a N y despejándola

$$N = -(D-CA^{-1}B)^{-1}CA^{-1} \qquad A.1.24$$

donde igualandola a A.1.17, entonces se cumple la relación A.1.21

$$(A-BD^{-1}C)^{-1} = A^{-1} + A^{-1}B(A-BD^{-1}C)^{-1} \qquad A.1.25$$

de ecuación A.1.14

$$L = A^{-1} - A^{-1}BN \qquad A.1.26$$

y sustituyendo N de ecuación A.1.24

$$L = A^{-1} + A^{-1}B(D-CA^{-1}B)^{-1}CA^{-1} \qquad A.1.27$$

igualando a L de ecuación A.1.19 se cumple la igualdad A.1.25

$$A^{-1}B(D-CA^{-1}B)^{-1} = (A-BD^{-1}C)^{-1}BD^{-1} \qquad A.1.28$$

despejando P de la ecuación A.1.8:

$$P = D^{-1} - D^{-1}CM \qquad A.1.29$$

sustituyendo A.1.29 en A.1.9:

$$AM - B(D^{-1} - D^{-1}CM) = 0 \qquad A.1.30$$

factorizando M y despejándola:

$$M = (A-BD^{-1}C)^{-1}BD^{-1} \qquad A.1.31$$

que es igual a la M obtenida en A.1.13

$$(D-CA^{-1}B)^{-1} = D^{-1} + D^{-1}C(A-BD^{-1}C)^{-1} \qquad A.1.32$$

como: $P = (D - CA^{-1}B)^{-1} = D^{-1} - D^{-1}CM$ A.1.33

sustituyendo la M obtenida en A.1.31

$$(D - CA^{-1}B)^{-1} = D^{-1} - D^{-1}C(A-BD^{-1}C) \qquad A.1.34$$

que es idéntica a A.1.32

Otra alternativa del lema de inversión matricial

Dado el sistema lineal expresado con una matriz particionada:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = - \begin{bmatrix} E \\ F \end{bmatrix} \qquad \text{A.1.35}$$

entonces el la solución del sistema puede expresarse:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} -A^{-1}E \\ 0 \end{bmatrix} + \begin{bmatrix} -A^{-1}B \\ 1 \end{bmatrix} Y \qquad \therefore \; Y = -(D-CA^{-1}B)^{-1}(F-CA^{-1}E) \qquad \text{A.1.36}$$

o alternativamente:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} 0 \\ -D^{-1}F \end{bmatrix} + \begin{bmatrix} 1 \\ -D^{-1}C \end{bmatrix} X \qquad \therefore \; X = -(A-BD^{-1}C)^{-1}(E-BD^{-1}F) \qquad \text{A.1.37}$$

A.2 RECURSIVIDAD SOBRE LA INVERSA DE LA MATRIZ R

En el capitulo tres (ecuación 3.68) se comprobó que la matriz de autocorrelación R de orden p se puede escribir en forma recursiva, aquí se comprobará la forma recursiva de la ecuación 3.83, partiendo.

$$R_p(k) = \lambda R_p(k-1) + Y_p(k)Y_p^T(k) \qquad A.2.1$$

premultiplicándola por la inversa de $R_p(k)$

$$I = \lambda R_p^{-1}(k)R_p(k-1) + R_p^{-1}(k)Y_p(k)Y_p^T(k) \qquad A.2.2$$

postmultiplicando por la inversa de $R_p(k-1)$

$$R_p^{-1}(k-1) = \lambda R_p^{-1}(k) + R_p^{-1}(k)Y_p(k)Y_p^T(k)R_p^{-1}(k-1) \qquad A.2.3$$

multiplicando por el vector $Y_p(k)$

$$R_p^T(k-1)Y_p(k) = \lambda R_p^{-1}(k)Y_p(k) + R_p^{-1}(k)Y_p(k)Y_p^T(k)R_p^{-1}(k-1)Y_p(k) \qquad A.2.4$$

factorizando $R_p^{-1}(k)Y_p(k)$:

$$R_p^T(k-1)Y_p(k) = R_p^{-1}(k)Y_p(k)\{ \lambda + Y_p^T(K)R_p^{-1}(k-1)Y_p(k) \} \qquad A.2.5$$

despejando

$$R_p^{-1}(k)Y_p(k) = R_p^T(k-1)Y_p(k) / \{ \lambda + Y_p^T(K)R_p^{-1}(k-1)Y_p(k) \} \qquad A.2.6$$

sustituyendo A.2.6 en A.2.3

$$R_p^T(k-1) = \lambda R_p^{-1}(k) + R_p^T(k-1)Y_p(k)Y_p^T(k)R_p^{-1}(k-1)/\{ \lambda + Y_p^T(K)R_p^{-1}(k-1)Y_p(k) \} \qquad A.2.7$$

despejando a $R_p^{-1}(k)$ se obtiene finalmente:

$$R_p^{-1}(k) = \frac{1}{\lambda}\{R_p^{-1}(k-1) - \frac{R_p^{-1}(k-1)Y_p(k)Y_p^T(k)R_p^{-1}(k-1)}{\lambda + Y_p^T(k)R_p^{-1}(k-1)Y_p(k)}\} \qquad A.2.8$$

Esta última ecuación es equivalente a la ecuación 3.83 donde $R_p^{-1}(k) = P(k)$.

A.3 MINIMOS CUADRADOS Y MATRIZ DE PROYECCION

Considerando al vector **p** como la proyección de un vector **b** sobre un subespacio vectorial **S** = { $\mathbf{a}_1$, $\mathbf{a}_2$, ... , $\mathbf{a}_k$ }, donde los **a**$_i$ son vectores independientes de R^n, es decir una base de S. Por simplicidad se hace k=2, entonces $\{\mathbf{a}_1,\mathbf{a}_2\}$ es un plano en R^n que contiene al origen y cualquier vector en **S** es una combinación lineal de $\mathbf{a}_1$ y $\mathbf{a}_2$.

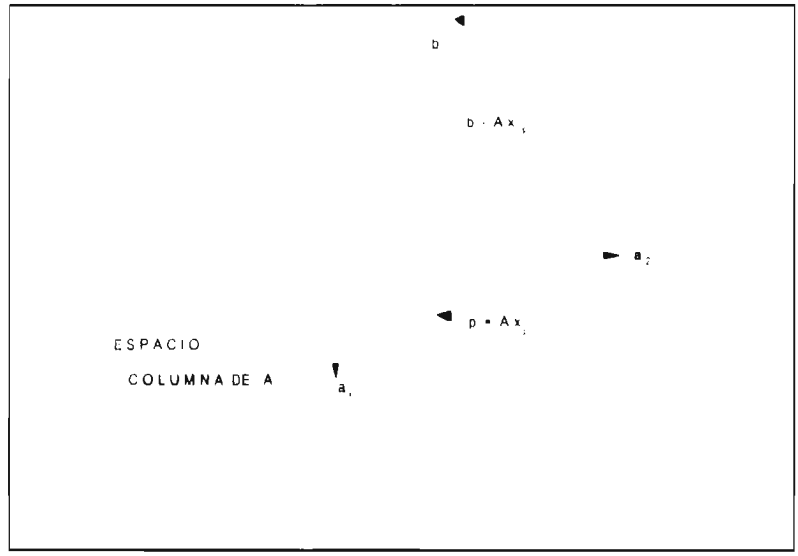


Figura A.3.1 Interpretación geométrica de las mínimos cuadrados

El vector **p** debe satisfacer:

a) Estar en el subespacio **S** = $\{\mathbf{a}_1,\mathbf{a}_2\}$
b) **b**-**p** debe ser perpendicular a cada vector de **S**.

Los vectores del subespacio S se pueden escribir como vectores columnas de una matriz **A**, es decir el espacio columna de la matriz **A** de 2x2. Entonces la identidad **Ax** = **b** tienen solución si y sólo sí **b** está en el espacio columna de **A**, todos los vectores de S tienen la forma **Ax** donde $\mathbf{x} = [x_1\ x_2]^T$. Como **p** está en el espacio de S entonces $\mathbf{p} = \mathbf{A}\,\mathbf{x}_p$ donde $\mathbf{x}_p = [r_1\ r_2\ ]^T$.
Como $\mathbf{b}\text{-}\mathbf{p} = \mathbf{b} - \mathbf{A}\mathbf{x}_p$ debe ser perpendicular a cada vector de S entonces el producto interno

$$< \mathbf{b} - \mathbf{A}\mathbf{x}_p\ ,\ \mathbf{A}\mathbf{x} > = 0 \qquad \text{A.3.1}$$

o

$$(\mathbf{A}\mathbf{x})^T (\mathbf{b} - \mathbf{A}\mathbf{x}_p) = \mathbf{x}^T (\mathbf{A}^T\mathbf{b} - \mathbf{A}^T\mathbf{A}\mathbf{x}_p) = \mathbf{0}$$

$$\mathbf{A}^T\mathbf{b} - \mathbf{A}^T\mathbf{A}\mathbf{x}_p = \mathbf{0} \qquad \text{A.3.2}$$

La matriz $\mathbf{A}^T\mathbf{A}$ es invertible ya que tiene el mismo rango que $\mathbf{A}$, despejando $\mathbf{x}_p$

$$\mathbf{x}_p = (\mathbf{A}^T\mathbf{A})^{-1}\,\mathbf{A}^T\,\mathbf{b} \qquad \text{A.3.3}$$

como

$$\mathbf{p} = \mathbf{A}\,\mathbf{x}_p = \mathbf{A}\,(\mathbf{A}^T\mathbf{A})^{-1}\,\mathbf{A}^T\,\mathbf{b} = \mathbf{P}\,\mathbf{b} \qquad \text{A.3.4}$$

donde $\mathbf{P} = \mathbf{A}\,(\mathbf{A}^T\mathbf{A})^{-1}\,\mathbf{A}^T$ se le conoce como la matriz de proyección para el subespacio S.

La distancia $\| \mathbf{b} - \mathbf{A}\mathbf{x}_p \|$ es la distancia mínima de proyectar $\mathbf{b}$ al subespacio S y $\mathbf{p}$ es la estimación del vector $\mathbf{b}$ en espacio S. Entonces $\mathbf{p} = \mathbf{P}\,\mathbf{b}$ es la *solución optima*.

Si la matriz $\mathbf{P}$ es simétrica real de nxn. Entonces los valores propios de $\mathbf{P}$ son reales y para cualquier matriz simétrica la multiplicidad algebraica de cada valor propio es igual a la multiplicidad geométrica, de tal forma que toda matriz simétrica es diagonalizable.

Toda matriz simétrica real de nxn tiene n vectores característicos ortonormales. Entonces si $\mathbf{Q}$ es una matriz ortogonal, $\mathbf{P}$ es diagonalizable si:

$$\mathbf{Q}^T\,\mathbf{P}\,\mathbf{Q} = \Lambda \qquad \text{A.3.5}$$

donde $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots \lambda_n)$ y los λ_i son los valores característicos de la matriz $\mathbf{P}$, y $\mathbf{Q}$ está formada de los vectores característicos ortonormales de $\mathbf{P}$.

Una matriz ortonormal es simplemente una matriz cuadrada con columnas y filas ortonormales, con propiedades $\mathbf{Q}^T = \mathbf{Q}^{-1}$, entonces

$$\mathbf{Q}^T\mathbf{Q} = \mathbf{I} \qquad \text{A.3.7}$$

$$\mathbf{Q}^{-1}\,\mathbf{P}\,\mathbf{Q} = \Lambda \qquad \text{A.3.6}$$

Esto quiere decir que $\mathbf{P}$ es ortogonalizable si y sólo sí $\mathbf{P}$ es simétrica.

A.4 DEDUCCION DE LA GANANCIA DE KALMAN

La ganancia de Kalman fue definida en 3.79 y está relacionada con la forma recursiva de la matriz $R_p^{-1}(k)$.

Definiendo ganancia de Kalman de la ecuación A.2.8 como:

$$K_p(k) = - R_p^{-1}(k-1)\, Y_p(k) / (\lambda + Y_p^T(k) R_p^{-1}(k-1) Y_p(k)) \qquad A.4.1$$

entonces la recursión sobre la inversa de R queda:

$$\lambda R_p^{-1}(k) = R_p^{-1}(k-1) + K_p(k)\, Y_p^T(k) R_p^{-1}(k-1) \qquad A.4.2$$

postmultiplicando por $Y_p(k)$

$$\lambda R_p^{-1}(k) Y_p(k) = R_p^{-1}(k-1) Y_p(k) + K_p(k)\, Y_p^T(k) R_p^{-1}(k-1) Y_p(k) \qquad A.4.3$$

sustituyendo el valor de $K_p(k)$ y factorizando

$$\lambda R_p^{-1}(k) Y_p(k) = \{ I_p(k) + R_p^{-1}(k-1)\, Y_p(k) / (\lambda + Y_p^T(k) R_p^{-1}(k-1) Y_p(k)) Y_p(k) \}\, R_p^{-1}(k-1) Y_p(k)$$

$$= \{ 1 + Y_p^T(k) R_p^{-1}(k-1) Y_p(k) - Y_p(k) R_p^{-1}(k-1) Y_p(k) / (\lambda + Y_p^T(k) R_p^{-1}(k-1) Y_p(k)) \}\, R_p^{-1}(k-1) Y_p(k) \qquad A.4.4$$

$$R_p^{-1}(k) Y_p(k) = R_p^{-1}(k-1) Y_p(k) / (\lambda + Y_p^T(k) R_p^{-1}(k-1) Y_p(k)) = - K_p(k) \qquad A.4.5$$

entonces

$$K_p(k) = - R_p^{-1}(k) Y_p(k) \qquad A.4.6$$

que es la ecuación 3.79 .

También la ganancia de Kalman con un retardo en el tiempo:

$$K_p(k-1) = - R_p^{-1}(k-1) Y_p(k-1) \qquad A.4.7$$

A.5 DEDUCCION DEL ALGORITMO DE KALMAN

En el capitulo tres se hizo el planteamiento del algoritmo rapido de Kalman, sin embargo, como la deduccion es muy larga e involucra muchas fórmulas, se presenta en este anexo el desarrollo completo, es decir, que se repiten algunas fórmulas del capitulo tres.

Partiendo de la ecuación de los mínimos cuadrados:

$$R_p(k-1)A_p(k) = -r_p^T(k) \qquad A.5.1$$

con el vector de parametros forward $A_p(k)$ de orden p

$$A_p^T(k) = [a_1^p, a_2^p, \ldots, a_p^p] \qquad A.5.2$$

la solución al orden p+1 sería:

$$R_{p+1}(k-1)A_{p+1}(k) = -r_{p+1}^T(k) \qquad A.5.3$$

definiendo :

$$A_{p+1}^T(k) = [a_1^{p+1}, a_2^{p+1}, \ldots, a_p^{p+1}] \qquad A.5.4$$

$$r_{p+1}^f[k] = \sum_{j=0}^{k} \lambda^{k-j} Y_{p+1}[j-1]\, y(j) \qquad A.5.5$$

$$R_{p+1}[k-1] = \sum_{j=0}^{k} \lambda^{k-j} Y_{p+1}[j-1]\, Y_{p+1}^T[j-1] \qquad A.1.6$$

$\hat{y}(k) = y(k)$ predictor hacia adelante
$\hat{y}(k-p) = y(k-p)$ predictor hacia atrás

$$Y_p^T[k-1] = [y(k-1), y(k-2), y(k-3) \ldots y(k-p)] \qquad A.1.7$$

$$Y_p^T[k] = [y(k), y(k-1), y(k-2) \ldots y(k-p+1)] \qquad A.1.8$$

entonces el vector Y_{p+1} puede escribir en forma abreviada como un vector de orden "p" y otro elemento para completar el orden "p+1":

$$Y_{p+1}^T[k] = [y(k), Y_p^T[k-1] = [Y_p^T[k], y(k-p)] \qquad A.1.9$$

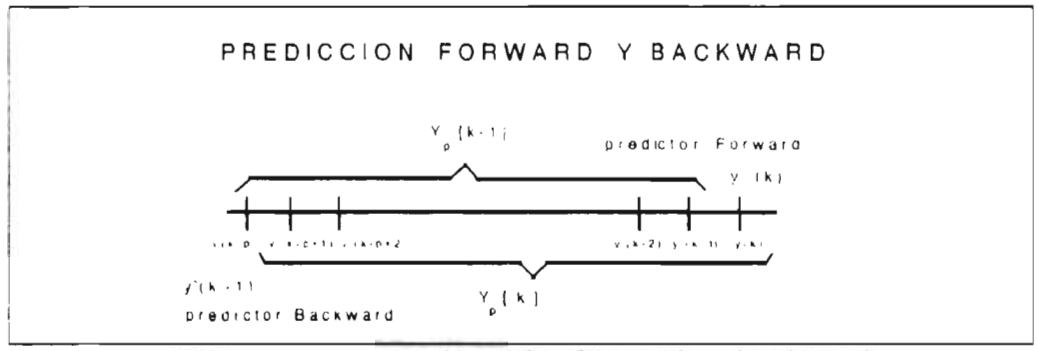


Figura A.5.1 Predicción forward y backward

escribiendo la matriz $R_{p-1}[k-1]$ al instante k y si l = j-1

$$R_{p-1}[k] = \sum_{j=0}^{k-1} \lambda^{k-j-1} Y_{p-1}[j-1]\, Y_{p-1}^{T}[j-1] = \sum_{l=-1}^{k} \lambda^{k-l} Y_{p-1}[l]\, Y_{p-1}^{T}[l]$$

A.5.10

donde por condiciones iniciales l = -1, pero puede empezar de cero, entonces l = 0.

La solución general al tiempo k+1 es:

$$R_{p-1}[k]\, A_{p-1}[k+1] = -r_{p-1}[k+1] \qquad \text{A.5.11}$$

sustituyendo en $R_{p-1}[k]$ la forma de $Y_{p-1}[k]$

$$R_{p-1}[k] = \sum_{j=0}^{k} \lambda^{k-j} \begin{bmatrix} y(j) \\ Y_p[j-1] \end{bmatrix} \begin{bmatrix} y(j) & Y_p^T[j-1] \end{bmatrix} \qquad \text{A.5.12}$$

en forma más compacta:

$$R_{p-1}[k] = \sum_{j=0}^{k} \lambda^{k-j} \begin{bmatrix} Y_p[j] \\ y(j-1) \end{bmatrix} \begin{bmatrix} Y_p^T[j] & y(j-1) \end{bmatrix} \qquad \text{A.5.13}$$

$$R_{p+1}[k] = \begin{bmatrix} r_y[0] & r_p^{fT}[k] \\ r_p^f[k] & R_p[k-1] \end{bmatrix} = \begin{bmatrix} R_p[k] & r_y^b[k] \\ r_p^{bT}[k] & r_y[p] \end{bmatrix} \qquad \text{A.5.14}$$

Se definen a $r_p^b[k]$ y $r_y[p]$ como:

$$r_p^b[k] = \sum_{j=0}^{k} \lambda^{k-j} Y_p[j]\, y(j-p) \qquad \text{A.5.15}$$

$$r_y[p] = \sum_{j=0}^{k} \lambda^{k-j} y(j-p)\, Y(j-p) \qquad \text{A.5.16}$$

$$R_p[k] = \sum_{j=0}^{k} \lambda^{k-j} Y_p[j]\, Y_p^T[j] \qquad \text{A.5.17}$$

Como en la solución de $A_{p+1}[k]$ aparece la ganancia de Kalman al orden p+1:

$$\mathbf{K_{p+1}[k] = - R_{p+1}^{-1}[k]\ Y_{p+1}[k]}$$

$$K_{p+1}[k] = - R_{p+1}^{-1}[k] \begin{bmatrix} y(k) \\ Y_p[k-1] \end{bmatrix} = - R_{p+1}^{-1}[k] \begin{bmatrix} Y_p[k] \\ y(k-p) \end{bmatrix} \qquad \text{A.5.18}$$

sustituyendo la ecuación de $\mathbf{R_{p+1}^{-1}[k]}$ en las dos versiones de $\mathbf{K_{p+1}[k]}$

$$K_{p+1}[k] = - \begin{bmatrix} r_y[0] & r_p^{fT}[k] \\ r_p^f[k] & R_p[k-1] \end{bmatrix}^{-1} \begin{bmatrix} y(k) \\ Y_p[k-1] \end{bmatrix} = - \begin{bmatrix} R_p[k] & r_y^b[k] \\ r_p^{bT}[k] & r_y[p] \end{bmatrix}^{-1} \begin{bmatrix} Y_p[k] \\ y(k-p) \end{bmatrix}$$

A.5.19

Se debe encontrar la solución para $\mathbf{K_{p+1}[k]}$ en función de $\mathbf{A_p[k]}$, $\mathbf{K_p[k-1]}$, $\mathbf{B_p[k]}$ y $\mathbf{K_p[k]}$.

Para el predictor hacia atrás, también se tiene la ecuación del error hacia atrás (backward):

$$\mathbf{e_p^b = y(k-p) - \hat{y}(k-p) = y(k-p) + B_p^T[k] Y_p[k]} \qquad \text{A.5.20}$$

donde $B_p[k]$ es el vector solucion del predictor hacia atras, y esta dado por:

$$B_p^t[k] = [b_{p1}, \ldots, b_1] \qquad A.5.21$$

que corresponde a la solución del sistema backward:

$$B_p[k] = - R_p^{-1}[k] r_p^b[k] \qquad A.5.22$$

También se demuestra que $B_p[k]$ se puede calcular recursivamente utilizando la ganancia de Kalman:

$$B_p[k] = B_p[k-1] + K_p[k] e_p^b(k) \qquad A.5.23$$

$$B_p[k] = B_p[k-1] + W_p[k-1]\, \gamma_p[k-1] e_p^b(k) \qquad A.5.24$$

de estas ecuaciones, los errores backward a priori $e_p^b(k)$ y a posteriori $\epsilon_p^b(k)$

se relacionan : $\epsilon_p^b(k) = \gamma_p(k) e_p^b(k)$ A.5.25

donde

$$A_p[k] = A_p[k-1] + K_p[k-1] e_p^f(k) \qquad A.5.26$$

$$A_p[k] = A_p[k-1] + W_p[k-1]\, \gamma_p[k-1]\, e_p^f(k) \qquad A.5.27$$

los errores forward a priori $e_p^f(k)$ y a posteriori $\epsilon_p^f(k)$

se relacionan : $\epsilon_p^f(k) = \gamma_p(k) e_p^f(k)$ A.5.28

Utilizando el lema de inversión

Para el cálculo del vector de la ganancia de Kalman al orden p+1, se debe calcular la inversa de la matriz R_{p+1}, como está particionada, entonces se puede utilizar el lema de inversión para una matriz X particionada (ver anexo A.1)

$$X^{-1} = \begin{bmatrix} (A - BD^{-1}C)^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ -D^{-1}C(A - BD^{-1}C)^{-1} & (D - CA^{-1}B)^{-1} \end{bmatrix} \qquad A.5.29$$

donde se cumplen las relaciones:

$$(A - BD^{-1}C)^{-1} = A^{-1} + A^{-1}B(A - BD^{-1}C)^{-1} \qquad A.5.30$$

$$(D\text{-}CA^{-1}B)^{-1} = D^{-1} + D^{-1}C(A\text{-}BD^{-1}C)^{-1} \qquad A.5.31$$

$$A^{-1}B(D\text{-}CA^{-1}B)^{-1} = (A\text{-}BD^{-1}C)^{-1}BD^{-1} \qquad A.5.32$$

$$D^{-1}C(A\text{-}BD^{-1}C)^{-1} = (D\text{-}CA^{-1}B)^{-1}CA^{-1} \quad [\text{Ver anexo A.1}] \qquad A.5.33$$

igualando matrices entre el lema y R_{p+1}, para el caso backward se tiene:

$$A = R_p[k] \qquad A.5.34$$
$$B = r_p^b[k] \qquad A.5.35$$
$$C = r_p^{bT}[k] \qquad A.5.36$$
$$D = r_{po}^b(k) \qquad A.5.37$$

para el caso forward:

$$A = r_{po}^f(k) \qquad A.5.38$$
$$B = r_p^{fT}[k] \qquad A.5.39$$
$$C = r_p^f[k] \qquad A.5.40$$
$$D = R_p[k\text{-}1] \qquad A.5.41$$

como:

$$B_p[k] = -R_p^{-1}[k]r_p^b[k] \qquad A.5.42$$

definiendo las energías de error backward y forward:

$$\alpha_p^f(k) = r_{po}^f(k) + r_p^{fT}[k]A_p[k] \qquad A.5.43$$

$$\alpha_p^b(k) = r_{po}^b(k) + r_p^{bT}[k]B_p[k] \qquad A.5.44$$

$$r_{po}^f[k] = \sum_{j=0}^{k} \lambda^{k-j} y^2(j) \qquad r_{po}^b = \sum_{j=0}^{k} \lambda^{k-j} y^2(j-p) \qquad A.5.45$$

descomponiendo la inversa de $R_{p+1}[k]$ en dos partes, para el **caso backward** y utilizando el lema de inversión matricial :

$$R_{p+1}[k]^{-1} = \begin{bmatrix} R_p^{-1}[k] & 0 \\ 0 & 0 \end{bmatrix} +$$

$$\begin{bmatrix} R_p^{-1}[k]\, r_p^b[k]\,(r_{pp}^b[k] - r_p^{bT} R_p^{-1}[k]\, r_p^b[k])^{-1} r_p^{bT}[k]\, R_p^{-1}[k] & B_p[k]\,\alpha_p^{-b}[k] \\ \alpha_p^{-b}[k]\, B_p^T[k] & \alpha_p^{-b}[k] \end{bmatrix}$$

A.5.46

sustituyendo las ecuaciones anteriores $R_{p+1}[k]$ queda reducida:

$$R_{p+1}[k]^{-1} = \begin{bmatrix} R_p^{-1}[k] & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} B_p[k] \\ 1 \end{bmatrix} \alpha_p^{-b}[k] \begin{bmatrix} B_p^T[k] & 1 \end{bmatrix}$$

A.5.47

de manera similar para el caso forward:

$$R_{p+1}[k]^{-1} = \begin{bmatrix} \alpha_p^{-f}[k] & \alpha_p^{-f}[k]\, A_p^T[k] \\ A_p[k]\,\alpha_p^{-f} & R_p^{-1}[k]\, r_p^f[k]\,(r_{pp}^f[k] - r_p^{fT} R_p^{-1}[k-1]\, r_p^f[k])^{-1} r_p^f[k]\, R_p^{-1}[k-1] \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 \\ 0 & R_p^{-1}[k-1] \end{bmatrix} \qquad \text{A.5.48}$$

sustituyendo las ecuaciones anteriores $R_{p+1}[k]$ se reduce a:

$$R_{p+1}[k]^{-1} = \begin{bmatrix} 1 \\ A_p[k] \end{bmatrix} \alpha_p^{-f}[k] \begin{bmatrix} 1 & A_p^T[k] \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & R_p^{-1}[k-1] \end{bmatrix}$$

A.5.49

donde las matrices $R_{p+1}[k]$ obtenidas para el caso forward como backward son iguales, entonces multiplicando por sus respectivos vectores $Y_p[k]$ forward y backward.

$$K_{p+1}[k] = -\left[\begin{bmatrix} R_p^{-1}[k] & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} B_p[k] \\ 1 \end{bmatrix} \alpha_p^{-b}[k] \begin{bmatrix} B_p^T[k] & 1 \end{bmatrix}\right] \begin{bmatrix} Y_p[k] \\ y(k-p-1) \end{bmatrix} =$$

$$= \begin{bmatrix} K_p[k] \\ 0 \end{bmatrix} - \begin{bmatrix} B_p[k] \\ 1 \end{bmatrix} \alpha_p^{-1}(k)\epsilon_p^b[k] \qquad \text{A.5.50}$$

$$R_{p+1}[k] = -\left[\begin{bmatrix} 1 \\ A_p[k] \end{bmatrix} \alpha_p^{-1}[k] \begin{bmatrix} 1 & A_p^T[k] \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & R_p^{-1}[k-1] \end{bmatrix}\right] \begin{bmatrix} y(k) \\ Y_{p-1}[k] \end{bmatrix} =$$

$$= \begin{bmatrix} 0 \\ K_p[k-1] \end{bmatrix} - \begin{bmatrix} 1 \\ A_p[k] \end{bmatrix} \alpha_p^{-1}(k)\epsilon_p^f \qquad \text{A.5.51}$$

de estas dos ecuaciones se puede calcular recursivamente el valor de la ganancia de Kalman, sin necesidad de la matriz $R_p[k]$ inversa.

Además, se puede demostrar que la energías hacia adelante y hacia atrás del error se calculan recursivamente como:

$$\alpha_p^f(k) = \lambda\alpha_p^f(k-1) + \epsilon_p^f e_p^f(k) \qquad \text{A.5.52}$$

$$\alpha_p^b(k) = \lambda\alpha_p^b(k-1) + \epsilon_p^b e_p^b(k) \qquad \text{A.5.53}$$

Comprobación para el caso forward. La energía de los errores se definen :

$$\alpha_p^f(k) = r_{po}^f(k) + r_p^{fT}(k)A_p(k) \qquad \text{A.5.54}$$

$$\alpha_p^b(k) = r_{po}^b(k) + r_p^{bT}(k)B_p(k) \qquad \text{A.5.55}$$

donde:

$$r_p^f(k) = \lambda\, r_p^f(k-1) + \mathbf{Y}_p(k-1)y(k) \qquad \text{A.5.56}$$

$$r_{po}^f(k) = \lambda\, r_{po}^f(k-1) + y^2(k) \qquad \text{A.5.57}$$

$$A_p[k] = A_p[k-1] + K_p[k-1]e_p^f(k) \qquad A.5.58$$

Recordando que

$$K_p(k-1) = - R_p^{-1}(k-1)Y_p(k-1) \qquad A.5.59$$

$$R_p(k-1)A_p(k) = - r_p^f(k) \qquad A.5.60$$

entonces:

$$\alpha_p^f(k) = \lambda\alpha_p(k-1) + \{ y(k) + A_p^T(k)Y_p(k-1) \} * e_p^f(k) \qquad A.5.61$$

ya que $\epsilon_p^T = y(k) + A_p^T(k)Y_p(k-1)$ A.5.62

queda demostrado que:

$$\alpha_p^f(k) = \lambda\alpha_p^f(k-1) + \epsilon_p^f e_p^f(k) \qquad A.5.63$$

Caso backward:

$$\alpha_p^b(k) = r_{po}^b(k) + r_p^{bT}(k)A_p(k) \qquad A.5.64$$

donde: $r_p^b(k) = \lambda\, r_p^b(k-1) + Y_p(k-1)y(k)$ A.5.65

$$r_{po}^b(k) = \lambda\, r_{po}^b(k-1) + y^2(k) \qquad A.5.66$$

$$B_p[k] = B_p[k-1] + K_p[k-1]e_p^b(k) \qquad A.5.67$$

Recordando que

$$K_p(k-1) = - R_p^{-1}(k-1)Y_p(k-1) \qquad A.5.68$$

$$R_p(k-1)B_p(k) = - r_p^b(k) \qquad A.5.69$$

entonces:

$$\alpha_p^b(k) = \lambda\alpha_p^b(k-1) + \{ y(k) + B_p^T(k)Y_p(k-1) \} e_p^b(k) \qquad A.5.70$$

ya que $\epsilon_p^T = y(k) + B_p^T(k)Y_p(k-1)$ A.5.71

entonces queda demostrado que:

$$\alpha_p^b(k) = \lambda\alpha_p^b(k-1) + \epsilon_p^b e_p^b(k) \qquad A.5.72$$

Resumen del ARK

	Inicializar $\alpha_p^f(0) = \delta > 0$ y $\lambda < 1$	
1	$e_p^f(k) = y(k) + A_p^T(k-1)Y_p(k-1)$	A.5.72
2	$A_p[k] = A_p[k-1] + K_p[k-1]e_p^f(k)$	A.5.73
3	$\epsilon_p^f(k) = y(k) + A_p^T[k]Y_p[k-1]$	A.5.74
4	$\alpha_p^f(k) = \lambda\alpha_p^f(k-1) + \epsilon_p^f e(k)_p^f(k)$	A.5.75
5	$K_{p+1}[k] = \begin{bmatrix} M_p[k] \\ U_p[k] \end{bmatrix} = \begin{bmatrix} 0 \\ K_p[k-1] \end{bmatrix} - \begin{bmatrix} 1 \\ A_p[k] \end{bmatrix} \alpha_p^{-1}(k)\epsilon_p^f$	A.5.76
6	$e_p^b(k) = y(k-p) + B_p^T[k-1]Y_p[k]$	A.5.77
7	$K_p[k] = (M_p[k] - B_p[k-1]Up(k)) / (1 + e_p^b(k)U_p(k))$	A.5.78
8	$B_p[k] = B_p[k-1] + K_p[k]e_p^b(k)$	A.5.79
	Otra posibilidad es cambiando el orden de 7 y 8 y sus formas:	
7	$B_p[k] = (M_p[k]e_p^b(k) - B_p[k-1]) / (1 + e_p^b[k]U_p[k])$	A.5.80
8	$K_p[k] = M_p[k-1] - B_p[k]U_p(k)$	A.5.81

Tabla A.1 Organigrama del algoritmo rápido de Kalman, [LMF78],[ALC86].

A.6 DEDUCCION DE ALGORITMOS RECURSIVOS EN EL ORDEN

En este anexo se desarrollan las fórmulas para las actualizaciones al orden superior "p+1" del orden "p" actual. Partiendo del algoritmo RLS que se desarrolla con base al algoritmo LS y utilizando la matriz de covarianza $R_p[k]$ al orden p+1, $R_{p+1}[k]$, y el vector de la ganancia de Kalman $K_p[k]$ al orden p+1, $K_{p+1}[k]$, para este desarrollo es necesario hacer simultáneamente la prediccion hacia adelante (forward) y la predicción hacia atrás (backward).

Resumiendo las ecuaciones del algoritmo RLS:

$$R_{p+1}[k-1]A_{p+1}[k] = -r_{p+1}^{f}[k] \qquad \text{A.6.1}$$

$$R_{p+1}[k]B_{p+1}[k] = -r_{p+1}^{b}[k] \qquad \text{A.6.2}$$

$$R_{p+1}[k]H_{p+1}[k] = -r_{p+1}^{b}[k] \qquad \text{A.6.3}$$

recordando las expresiones para r_p^f y r_p^b

$$r_p^f[k] = \sum_{j=0}^{k} \lambda^{k-j} Y_p[j-1]\, y(j) \qquad \text{A.6.4}$$

$$r_p^b[k] = \sum_{j=0}^{k} \lambda^{k-j} Y_p[j]\, y(j-p) \qquad \text{A.6.5}$$

los vectores r al orden p+1 y la matriz de R al orden p+1

$$r_{p+1}^f[k] = \sum_{j=0}^{k} \lambda^{k-j} Y_{p+1}[j-1]\, y(j) \qquad \text{A.5.6}$$

$$r_{p+1}^b[k] = \sum_{j=0}^{k} \lambda^{k-j} Y_{p+1}[j]\, y(j-p-1) \qquad \text{A.6.7}$$

$$R_{p+1}[k] = \begin{bmatrix} r_y[0] & r_p^{fT}[k] \\ r_p^f[k] & R_p[k-1] \end{bmatrix} = \begin{bmatrix} R_p[k] & r_y^b[k] \\ r_p^{bT}[k] & r_y[p] \end{bmatrix} \qquad \text{A.6.8}$$

estos vectores también se pueden escribir:

$$r_{p+1}^{f}[k] = \begin{bmatrix} r_p^{f}[k] \\ r_{p+1}^{b,p+1}[k] \end{bmatrix} \quad \cdots \quad r_{p+1}^{f}[k] = \sum_{j=0}^{k} \lambda^{k-j} y(j-p-1)\, y(j)$$

A.6.9

$$r_{p+1}^{b}[k] = \begin{bmatrix} r_{p+1}^{b,p+1}[k] \\ r_p^{b}[k] \end{bmatrix} \quad \cdots \quad r_{p+1}^{b,p+1}[k] = \sum_{j=0}^{k} \lambda^{k-j} y(j)\, y(j-p-1)$$

A.6.10

de estas ecuaciones se observa que las componente de orden "p+1" de los vectores son iguales, es decir:

$$r_{p+1}^{f,p+1}[k] = r_{p+1}^{b,p+1}[k] \qquad \text{A.6.11}$$

esto se puede obtener al invertir el orden de los productos dentro de las matrices.

Sustituyendo la inversa de la matriz $R_{p+1}[k]$ en forma particionada, y los vectores $r_p[k]$ para las soluciones forward y backward, entonces se obtienen los vectores solución de $A_{p+1}[k]$ y $B_{p+1}[k]$.

Se hace notar que en esta sustitución, la matriz inversa de $R_{p+1}[k]$ tiene dos formas. También hay que tomar en cuenta que en la *solución forward*, la matriz $R_{p+1}[k]$ está al tiempo k-1, efectuando las operaciones matriciales y sustituyendo igualdades ya conocidas:

$$\begin{bmatrix} A_p^{p+1}[k] \\ A_{p+1}^{p+1}[k] \end{bmatrix} = -\left(\begin{bmatrix} R_p^{-1}[k-1] & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} B_p[k-1] \\ 1 \end{bmatrix} \alpha_p^{-b}[k-1] \begin{bmatrix} B_p^{T}[k-1] & 1 \end{bmatrix} \right) \begin{bmatrix} r_p^{f}[k] \\ r_{p+1}^{f,p+1}[k] \end{bmatrix}$$

A.6.12

$$\begin{bmatrix} A_p^{p+1}[k] \\ A_{p+1}^{p+1}[k] \end{bmatrix} = \begin{bmatrix} A_p[k] \\ 0 \end{bmatrix} - \begin{bmatrix} B_p[k-1] \\ 1 \end{bmatrix} \left(B_p^{T}[k]\, r_p^{f}[k] + r_{p+1}^{f,p+1}[k] \right) \alpha_p^{b}[k-1]$$

A.6.13

Para el *caso backward*:

$$\begin{bmatrix} B_p^{f,p+1}[k] \\ B_p^{b,p+1}[k] \end{bmatrix} = -\left(\begin{bmatrix} 0 & 0 \\ 0 & R_p^{-1}[k-1] \end{bmatrix} + \begin{bmatrix} 1 \\ A_p[k] \end{bmatrix} \alpha_p^{b}[k] \begin{bmatrix} 1 & A_p^T[k] \end{bmatrix} \right) \begin{bmatrix} r_p^{b,p+1}[k] \\ r_p^{f}[k] \end{bmatrix}$$

A.514

efectuando operaciones matriciales y sustituyendo igualdades conocidas:

$$\begin{bmatrix} B_p^{f,p+1}[k] \\ B_p^{b,p+1}[k] \end{bmatrix} = \begin{bmatrix} 0 \\ B_p[k-1] \end{bmatrix} - \begin{bmatrix} 1 \\ A_p[k] \end{bmatrix} \left(A_p^T[k]\, r_p^{b}[k-1] + r_{p+1}^{b,p+1}[k] \right) \alpha_p^{f}[k]$$

A.6.15

Se puede comprobar que el primer término escalar entre paréntesis para el caso forward y backward son iguales:

$$B_p^T[k-1]r_p^f[k] = -r_p^{bT}[k-1]R_p^{-1}[k-1]r_p^f[k] = r_p^{bT}[k-1]A_p[k] \qquad \text{A.6.16}$$

$$B_p^T[k-1]r_p^f[k] = A_p^T[k]r_p^b[k-1] \qquad \text{A.6.17}$$

entonces se comprueba que los términos backward y forward son iguales:

$$B_p^T[k-1]r_p^f[k] + r_{p+1}^{f,p+1}[k] = A_p^T[k]r_p^b[k-1] + r_{p+1}^{b,p+1}[k] = g_{p+1}^c[k] \qquad \text{A.6.18}$$

Entonces los vectores solución se pueden expresar:

$$\begin{bmatrix} A_p^{p+1}[k] \\ A_p^{p+1}[k] \end{bmatrix} = \begin{bmatrix} A_p[k] \\ 0 \end{bmatrix} + \begin{bmatrix} B_p[k-1] \\ 1 \end{bmatrix} K_{f,p+1}^{b}[k] \qquad \therefore \qquad K_{p+1}^{b}[k] \triangleq -\frac{g_{p+1}^{c}[k]}{\alpha_p^{b}[k]}$$

A.6.19

$$\begin{bmatrix} B_{p+1}^{p+1}[k] \\ B_p^{p+1}[k] \end{bmatrix} = \begin{bmatrix} 0 \\ B_p[k-1] \end{bmatrix} + \begin{bmatrix} 1 \\ A_p[k] \end{bmatrix} K_{p+1}^{f}[k] \qquad \therefore \qquad K_{p+1}^{f}[k] \triangleq -\frac{g_{p+1}^{c}[k]}{\alpha_p^{f}[k]}$$

A.6.20

Energías residuales

De las energías residuales al orden p, se tienen las energías al orden p+1:

$$\alpha_{p+1}^{f}(k) = r_{p0}^{f}(k) + r_{p+1}^{fT}(k)A_{p+1}[k] \qquad \text{A.6.21}$$

$$\alpha_{p+1}^{b}(k) = r_{po}^{b}[k] + r_{p+1}^{br}(k)B_{p+1}[k] \qquad \text{A.6.22}$$

que se pueden desarrollar sustituyendo los valores de $A_{p+1}[k]$ y $B_{p+1}[k]$ encontradas anteriormente, es decir:

caso forward

$$\alpha_{p+1}^{f}[k] = r_{yy}[0] + \left[r_{p}^{bT}[k] \quad r_{p+1}^{f,p+1}[k]\right]\left(\begin{bmatrix}A_p[k]\\0\end{bmatrix} + \begin{bmatrix}B_p[k-1]\\1\end{bmatrix} K_{p+1}^{f}[k]\right)$$

A.6.23

desarrollando y sustituyendo los términos equivalentes:

$$\alpha_{p+1}^{f} = \alpha_{p}^{f}[k] + K_{p+1}^{b}[k]g_{p+1}^{c}[k] \qquad \text{A.6.24}$$

caso backward

$$\alpha_{p+1}^{b}[k] = r_{yy}[p] + \begin{bmatrix}r_{p+1}^{b,p+1}[k]\\r_{p}^{bT}[k]\end{bmatrix}\left(\begin{bmatrix}0\\B_p[k-1]\end{bmatrix} + \begin{bmatrix}1\\A_p[k]\end{bmatrix} K_{p+1}^{f}[k]\right)$$

A.6.25

desarrollando y sustituyendo los términos equivalentes:

$$\alpha_{p+1}^{b} = \alpha_{p}^{b}[k] + K_{p+1}^{f}[k]g_{p+1}^{c}[k] \qquad \text{A.6.26}$$

La actualización del algoritmo todavía no es posible, ya que no se dispone de los términos:

$$B_p[k-1] \text{ ni } \alpha_p^b[k-1] \qquad \text{A.6.27}$$

si se utiliza la actualización temporal para $B_p[k]$

$$B_p[k] = B_p[k-1] + W_p[k]\epsilon_p^b(k) \qquad \text{A.6.28}$$

$$B_p[k] = B_p[k-1] + K_p[k]e_p^b(k) \qquad \text{A.6.29}$$

entonces se necesita $K_{p+1}[k]$ o $W_{p+1}[k]$

donde ya se demostró para el algoritmo rápido de Kalman que (A.5.50):

$$K_{p+1} = \begin{bmatrix} K_p[k] \\ 0 \end{bmatrix} - \begin{bmatrix} B_p[k] \\ 1 \end{bmatrix} \alpha_p^{-b}(k)\, \epsilon_p^b[k] \qquad \text{A.6.30}$$

como $\gamma_p(k) = \epsilon_p^b(k) / e_p^b(k)$ A.6.31

y $W_{p+1}[k] = K_{p+1}[k]\, \gamma_p^{-1}(k)$ A.6.32

$$W_{p+1} = \begin{bmatrix} W_p[k] \\ 0 \end{bmatrix} - \begin{bmatrix} B_p[k-1] \\ 1 \end{bmatrix} \lambda^{-1} \alpha_p^{-b}(k-1)\, e_p^b[k] \qquad \text{A.6.33}$$

Otras recursividades

$$\epsilon_p^f(k) = y(k) + A_p^T[k] Y_p[k-1] \qquad \text{A.6.34}$$

al orden p+1

$$\epsilon_{p+1}^f(k) = y(k) + A_{p+1}^T[k] Y_{p+1}[k-1] \qquad \text{A.6.35}$$

$$\epsilon_{p+1}^f(k) = y(k) + \left(\begin{bmatrix} A_p^T[k] & 0 \end{bmatrix} + \begin{bmatrix} B_p^T[k-1] & 1 \end{bmatrix} K_{p+1}^b \right) \begin{bmatrix} Y_p[k-1] \\ y(k-p-1) \end{bmatrix} \qquad \text{A.6.36}$$

sustituyendo cantidades ya conocidas:

$$\epsilon_{p+1}^f(k) = \epsilon_p^f(k) + K_{p+1}^b[k] \epsilon_p^b[k-1] \qquad \text{A.6.37}$$

similarmente para el caso backward:

$$\epsilon_p^b(k) = y(k-p) + B_p^T[k] Y_p[k] \qquad \text{A.6.38}$$

al orden p+1

$$\epsilon_{p+1}^b(k) = y(k-p) + B_{p+1}^T[k] Y_{p+1}[k] \qquad \text{A.6.39}$$

$$\epsilon^{f}_{p+1}(k) = y(k-p) + \begin{bmatrix} 0 & B_p^{T}[k-1] \end{bmatrix} + \begin{bmatrix} 1 & A_p^{T}[k] \end{bmatrix} \begin{bmatrix} y(k-p) \\ Y_p[k] \end{bmatrix}$$

A.6.40

sustituyendo cantidades ya conocidas:

$$\epsilon_{p+1}(k) = \epsilon_p^{f}(k-1) + K_{p+1}^{T}[k]\epsilon_p^{b}[k] \qquad \text{A.6.41}$$

se puede demostrar también que la g_{p+1}^{c} puede escribirse recursiva en el tiempo:

$$g_{p+1}^{c}[k+1] = \lambda g_{p+1}^{c}[k] + \epsilon_p^{b}[k]e_p^{f}[k+1] \qquad \text{A.6.42}$$

como

$$r_{p+1}^{b,p+1}[k] = \lambda r_{p+1}^{b,p+1}[k-1] + y(k)y(k-p-1) \qquad \text{A.6.43}$$

$$A_p^{T}[k] = A_p[k-1] + e_p^{f}[k]K_p[k-1] \qquad \text{A.6.44}$$

$$r_p^{b}[k] = \lambda r_p^{b}[k-1] + Y_p[k]y(k-p) \qquad \text{A.6.45}$$

sustituyendo estas ecuaciones en $g_{p+1}^{c}[k]$

$$g_{p+1}^{c}[k] = \{A_{pT}[k-1] + e_p^{f}[k]K_p^{T}[k-1]\}\{\lambda r_p^{b}[k-2] + Y_p[k-1]y(k-p-1)\}$$
$$+ \lambda r_{p+1}^{b,p+1}[k-1] + y(k)y(k-p-1) \qquad \text{A.6.46}$$

efectuando operaciones y agrupando

$$g_{p+1}^{c}[k] = \lambda\{A_{pT}[k-1]r_p^{b}[k-2] + r_{p+1}^{b,p+1}[k-1]\} + A_{pT}[k-1]\,Y_p[k-1]y(k-p-1)$$
$$e_p^{f}[k]K_p^{T}[k-1]r_p^{b}[k-2] + e_p^{f}[k]K_p^{T}[k-1]Y_p[k-1]y(k-p-1)\} + y(k)y(k-p-1) \qquad \text{A.6.47}$$

$$g_{p+1}^{c}[k] = \lambda g_p^{c}[k-1] + \{y(k) + A_p^{T}[k-1]Y_p[k-1]\}y(k-p-1)$$
$$+ e_p^{f}(k)K_p^{T}[k-1]\{\lambda r_p^{b}[k-2] + Y_p[k-1]y(k-p-1)\} \qquad \text{A.6.48}$$

$$g_{p+1}^{c}[k] = \lambda g_p^{c}[k-1] + e_p^{f}(k)\{y(k-p-1) + K_p^{T}[k-1]r_p^{b}[k-1]\} \qquad \text{A.6.49}$$

como la ganancia de Kalman está definida por

$$K_p[k-1] = -R_p^{-1}[k-1]Y_p[k-1] \quad \text{y} \quad r_p^{b}[k-1] = -R_p[k-1]B_p[k-1] \qquad \text{A.6.50}$$

y recordando que la matriz de R es simetrica

$$g_{p+1}^{c}[k] = \lambda g_{p}^{c}[k-1] + e_{p}^{f}(k)\{y(k-p-1) \qquad \text{A.6.51}$$

$$+ Y_{p}^{T}[k-1]R_{p}^{-1}[k-1]R_{p}[k-1]B_{p}[k-1]\} \qquad \text{A.6.52}$$

entonces:

$$g_{p+1}^{c}[k] = \lambda g_{p}^{c}[k-1] + e_{p}^{f}(k)\{\, y(k-p-1) + Y_{p}^{T}[k-1]B_{p}[k-1]\,\} \qquad \text{A.6.53}$$

donde el último término entre llaves es el error backward a posteriori: $\epsilon_{p}^{b}(k)$

$$g_{p+1}^{c}[k] = \lambda g_{p}^{c}[k-1] + e_{p}^{f}(k)\epsilon_{p}^{b}(k-1) \qquad \text{A.6.54}$$

similarmente para la definición backward de g

$$g_{p+1}^{c}[k] = B_{p}^{T}[k-1]r_{p}^{f}[k] + r_{p+1}^{f,p+1}[k] \qquad \text{A.6.55}$$

$$g_{p+1}^{c}[k] = \{B_{pT}[k-2] + e_{p}^{f}[k-1]K_{p}^{T}[k-1]\}\{\lambda r_{p}^{b}[k-1] + Y_{p}[k-1]y(k)\}$$
$$+ \lambda r_{p+1}^{b,p+1}[k-1] + y(k)y(k-1) \qquad \text{A.6.56}$$

efectuando operaciones y agrupando:

$$g_{p+1}^{c}[k] = \lambda\{B_{pT}[k-2]r_{p}^{f}[k-1] + r_{p+1}^{f,p+1}[k-1]\} + y_{p}[k-1]y(k-p)$$

$$e_{p}^{f}[k]K_{p}^{T}[k-1]\{\, r_{p}^{b}[k-2] + K_{p}^{T}[k-1]Y_{p}[k-1]y(k)\} + B_{p}^{T}[k-2]Y[k-1]y(k) \qquad \text{A.6.57}$$

$$g_{p+1}^{c}[k] = \lambda g_{p+1}^{c}[k-1] + y(k)\{y_{p}(k-p) + B_{p}^{T}[k-2]Y_{p}[k-p]\}$$

$$+ e_{p}^{f}(k)K_{p}^{T}[k-1]\{\, \lambda r_{p}^{b}[k-1] + Y_{p}[k-p]y(k)\} \qquad \text{A.6.58}$$

$$g_{p+1}^{c}[k] = \lambda g_{p+1}^{c}[k-1] + e_{p}^{b}(k)\, y(k) + e_{p}^{b}K_{p}^{T}[k-1]r_{p}^{b}[k] \qquad \text{A.6.59}$$

como la ganancia de Kalman está definida por

$$K_{p}[k-1] = -R_{p}^{-1}[k-1]Y_{p}[k-1] \quad \text{y} \quad r_{p}^{f}[k-1] = -R_{p}[k-1]A_{p}[k] \qquad \text{A.6.60}$$

como R es simétrica

$$g_{p+1}^{c}[k] = \lambda g_{p}^{c}[k-1] + e_{p}^{f}(k)\{\, y(k) + A_{p}^{T}[k]Y_{p}[k-1]\,\} \qquad \text{A.6.61}$$

$$g_{p+1}^{f}[k] = \lambda g_{p}^{f}[k-1] + e_{p}^{f}(k)\epsilon_{p}^{f}(k) \qquad \text{A.6.62}$$

ya que

$$\epsilon_{p}^{f}(k) = \gamma_{p}(k)\, e_{p}^{f}(k) \qquad \text{A.6.63}$$

$$\alpha_{p}^{f}(k) = \lambda\alpha_{p}^{f}(k-1) + \epsilon_{p}^{f}e_{p}^{f}(k) \qquad \text{A.6.64}$$

$$\alpha_{p}^{b}(k) = \lambda\alpha_{p}^{b}(k-1) + \epsilon_{p}^{b}e_{p}^{b}(k) \qquad \text{A.6.65}$$

A partir de la $K_{p,1}^{f}(t-1)$ y $K_{p,1}^{b}(t-1)$ se calculan al mismo orden p+1

$$\begin{bmatrix} A_{p}^{p+1}[k] \\ A_{p+1}^{p+1}[k] \end{bmatrix} = \begin{bmatrix} A_{p}[k] \\ 0 \end{bmatrix} + \begin{bmatrix} B_{p}[k-1] \\ 1 \end{bmatrix} K_{f,p+1}^{b}[k] \qquad \therefore \qquad K_{p+1}^{b}[k] \triangleq -\frac{\sigma_{p+1}^{f}[k]}{\alpha_{p}^{b}[k]}$$

A.6.66

$$\begin{bmatrix} B_{p}^{p+1}[k] \\ B_{p}^{p+1}[k] \end{bmatrix} = \begin{bmatrix} 0 \\ B_{p}[k-1] \end{bmatrix} + \begin{bmatrix} 1 \\ A_{p}[k] \end{bmatrix} K_{p+1}^{f}[k] \qquad \therefore \qquad K_{p+1}^{f}[k] \triangleq -\frac{\sigma_{p+1}^{b}[k]}{\alpha_{p}^{f}[k]}$$

A.6.67

después de llegar a p_{max}-2 se incrementa una muestra en el tiempo, es decir la recursividad en el tiempo.

Resumen del algoritmo recursivo en orden con predicción a priori

Inicialización en tiempo $p = 0,1,\ldots p_{max} - 1$

$\alpha_p^f(N-1) = \alpha_p^b(N-1) = \delta I \; ; \; \delta > 1$
$g_p^c(N-1) = g_p(N-1) = g_{pmax}(N-2) = 0$
$e_p^b(N-1) = e_p(N-1) = e_{pmax}(N-1) = 0 \; ; \; \gamma_p(N-2) = 1$

Inicialización en orden

$e_0^f(t) = e_0^b(t) = y(t) \; ; \; \gamma_0(t-1) = 1$
$\alpha_0^f(t) = \alpha_0^b(t) = \lambda\alpha_0^f(t-1) + y(t)y^T(t)$

recursión en el orden : $p = 0,1, \ldots, p_{max} - 2$

$g_{p+1}^c[t-1] = \lambda g_{p+1}^c[t-2] + \gamma_p(t-2)e_p^b(t-2)e_p^{fT}(t-1)$	A.6.68
$\mathbf{K}_{p+1}^f(t-1) = - g_{p+1}^{cT}[t-1] / \alpha_p^f(t-1)$	A.6.69
$\mathbf{K}_{p+1}^b(t-1) = - g_{p+1}^{cT}[t-1] / \alpha_p^b(t-1)$	A.6.70
$e_{p+1}^f(t) = e_p^f(t) + \mathbf{K}_{p+1}^{bT} e_p^b(t-1)$	A.6.71
$e_{p+1}^b(t) = e_p^b(t-1) + \mathbf{K}_{p+1}^{fT} e_p^f(t-1)$	A.6.72
$\alpha_{p+1}^f(t-1) = \alpha_p^f(t-1) + g_{p+1}^{cT}[t-1]\,\mathbf{K}_{p+1}^b(t-1)$	A.6.73
$\alpha_{p+1}^b(t-1) = \alpha_p^b(t-2) + g_{p+1}^{cT}[t-1]\,\mathbf{K}_{p+1}^f(t-1)$	A.6.74
$\gamma_{p+1}(t-1) = \gamma_p(t-1) - \gamma_p(t-1)^2 e_p^{bT}(t-1)\,\alpha_p^{-b}(t-1)\,e_p^b(t-1)$	A.6.75

Tabla A.2 Organigrama de algoritmo recursivo en orden, predicción a priori, [ALC86].

ANEXO B
ARQUITECTURAS DEL DSP Y PROGRAMAS EN ENSAMBLADOR

B.1 ARQUITECTURA DEL DSP TMS320C50

Los procesadores de señales digitales (DSP) son microprocesadores especializados para efectuar operaciones numéricas en tiempo real, en aplicaciones donde se requieren numerosas calculos aritméticos en un tiempo limitado. Por sus ampliaciones especificas los DSPs tienen arquitecturas que son significante diferentes a los microprocesadores tradicionales.

Los DSPs disponen de una arquitectura y un conjunto de instrucciones orientadas al Procesamiento Digital de Señales. Un DSP Normalmente contiene buses separados para el direccionamiento simultáneo de dos operandos, un multiplicador, corrimientos, una unidad de direccionamiento, puertos seriales y temporizadores. Entre las instrucciones de grandes posibilidades están el movimiento de bloques, multiplicaciones en un ciclo de instrucción, multiplicación acumulación para la realización de operaciones de convolución.

El TMS320C50 (TMS50 o C5x) es un procesador de punto fijo de la familia TMS320 de Texas Instruments (TI), está basado en el TMS320C25 con una arquitectura adicional que mejora el desempeño, entre estas características están:

- Ciclo de instrucción más rápido
- Respuesta rápida a interrupciones.
- Manipulación de bits vía la unidad paralela lógica.
- Bloques de repetición.
- Buffer circular.
- Gran cantidad de memoria interna.
- Corrimientos de 0 a 16 bits en el acumulador.
- Estados de espera por software.
- Acumulador Buffer.

Los dispositivos de la generación C5x son capaces de efectuar operaciones en dos veces la velocidad de la generación C2x y su código es compatible hacia arriba con las familias C1x y C2x. La familia C5x está diseñada para ejecutar 28 Millones de instrucciones por segundo (MIPS).

La operación pipeline

El TMS50 aumenta su potencialidad de ejecución de instrucciones con el mecanismo pipeline de cuatro niveles. La ejecución de instrucciones en pipeline consiste en una secuencia de operaciones externas de bus que ocurren durante la ejecución interna de la instrucción.

Las arquitecturas de los DSPs tienen algunos rasgos en comun: mientras una instrucción es buscada, otra decodificada y otra ejecutada, es decir, una forma de pipeline de tres estados.

En un programa tipico para DSP una instrucción buscará dos operandos de memoria, los multiplicara, sumará el producto anterior al acumulador, escribirá el resultado en memoria e incrementará el contenido de los registros de dirección; en forma secuencial llevará varios ciclos de tiempo, mientras que en pipeline se optimizará el tiempo de las operaciones mencionadas.

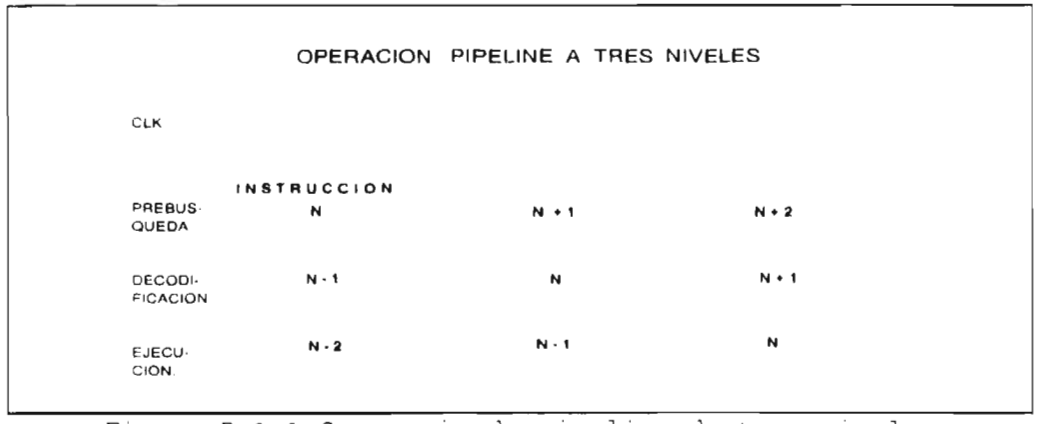


Figura B.1.1 Secuencia de pipeline de tres niveles

En esta arquitectura, las operaciones aritméticas son transferidas a estados sucesivos, donde unidades separadas de hardware proveen los cálculos a cada estado. Las computadoras que operan bajo este concepto son conocidas de procesamiento vectorial.

Una dificultad que se presenta con el pipeline son las ramificaciones, es decir, es difícil lograr eficientes ramificaciones debido a que el contador de programa debe continuar en otra dirección, perdiéndose la secuencia de búsqueda y por lo tanto la operación de pipeline. Sin embargo, el TMS50 ha incorporado a su conjunto de instrucciones una solución que consiste en producir saltos y llamados a subrutina con retardo, es decir, después del llamado a subrutina o el salto a otra localidad se agregan dos instrucciones que se ejecutan antes de hacer el salto.

Características principales del DSP TMS320C50

- 35/50 ns de ciclo de instrucción para instrucciones en punto entero (28.6/20 MIPS).
- 9K x 16 bits de memoria RAM interna, con acceso en un ciclo.
- 2K x 16 bits en memoria ROM acceso en un ciclo.
- 1056 x 16 bits en memoria RAM interna, puede accesarse dos veces por ciclo de máquina.

- 224k x 16 bits máximo espacio direccionable en memoria externa (64 K palabras[1] de programa, 64 k palabras de datos 64 k puertos I/O y 32 k palabras globales)
- Acumulador (ACC) de 32 bits y un acumulador buffer (ACCB) de 32 bits.
- Unidad Lógica Paralela (PLU) de 16 bits.
- Multiplicador paralelo de 16 x 16 con capacidad de productos de 32 bits.
- Instrucciones de multiplicación/acumulación en un ciclo simple.
- Ocho registros auxiliares con una unidad aritmética para direccionamiento indirecto.
- Ocho niveles de pila.
- 0 a 16 corrimientos a la izquierda.
- Dos buffer de direccionamiento indirecto para direccionamiento circular.
- Instrucciones de repetición de bloques.
- Instrucciones para el manejo de movimiento de bloques entre datos y programa.
- Puerto serial síncrono full-duplex para comunicación directa con otros 'C5x y otros dispositivos seriales.
- Puerto serial de múltiple acceso por división de tiempo (TDM).
- Un temporizador.
- 64k puertos I/O paralelos, con 16 puertos mapeados en memoria.
- 16 estados de espera programables por software, para programa, datos o espacio I/O.
- Operación de DMA.
- Cuatro niveles de Pipeline.
- Direccionamiento indexado.
- Direccionamiento de bit reverso para efectuar FFT's radix 2.
- Generador interno de reloj.
- Polarización de 5 volts, con modo power down.
- Empacado en 132 pines.
- 28 registros mapeados en memoria.
- Cuatro interrupciones externas y cinco internas una del timer y cuatro para puerto serie.
- 32 interrupciones por software.

Otros DSPs de la familia C5x

La familia TMS320C5x está disponible en diferentes versiones que se caracterizan por la distribución de la memoria interna:

	RAM	**ROM**
TMS320C50	10K	2K
TMS320C51	2K	8K
TMS320C52	1K	4K
TMS320C53	4K	16K
TMS320C56	7K	32K

Algunas de estas versiones están disponibles para polarización de 5 y 3.3 volts.

---

[1] una palabra equivale a 16 bits

ARQUITECTURA

El TMS50 consiste de tres bloques básicos:

- Unidad Central de Proceso (CPU)
- Memoria
- Interfaz a circuitos periféricos.

Su arquitectura utiliza dos buses separados, de datos y programa, con instrucciones que aceptan transferencia de datos entres ambos espacios. El TMS50 efectúa aritmética de complemento a dos, utilizando la ALU y el acumulador de 32 bits. Contiene una Unidad Lógica Paralela (PLU), que ejecuta operaciones lógicas sobre la memoria de datos sin afectar el contenido del acumulador.

Descripción general del TMS320C50

La destacada funcionalidad del TMS320C50 (TMS50) para el PDS, se debe a su tipo de arquitectura (tipo Hardware) que maximiza el procesamiento mediante el establecimiento de dos estructuras de buses de memoria separadas (memoria programa y memoria datos) para incrementar la velocidad de ejecución, contando con instrucciones para realizar transferencia de datos entre ambos espacios.

Externamente, las memorias de programa y datos son multiplexadas sobre el mismo bus externo para maximizar el intervalo de direccionamiento para dichos espacios mientras se minimizan los pines del dispositivo.

La flexibilidad en el diseño del sistema es incrementada al contar con tres bloques de datos internos en memoria RAM de doble acceso DRAM (un total de 1056-Palabras), donde uno de ellos puede ser configurado como memoria programa o memoria dato y 9 K-palabra de simple acceso (SRAM), además el TMS50 es capaz de direccionar externamente 64K-palabras en un espacio de memoria dato, para facilitar la implementación de algoritmos para DSP.

La memoria interna ROM de 2K-Palabras puede ser usada para reducir el costo de sistemas. Los programas en memoria ROM interna pueden ejecutarse a alta velocidad desde este espacio de memoria. Externamente el espacio de memoria de programa direccionable es de 64K-Palabras.

El TMS50 funciona con una aritmética en modo complemento a dos, empleando una ALU y un Acumulador de 32 bits. La ALU es una unidad aritmética de propósito general que opera con palabras de 16 bits provenientes de la RAM de datos o derivadas de instrucciones inmediatas o por el empleo del registro producto (PR) del multiplicador de 32 bits. La ALU puede efectuar operaciones booleanas. El Acumulador almacena los resultados de la ALU y a su vez es una segunda entrada a la ALU.

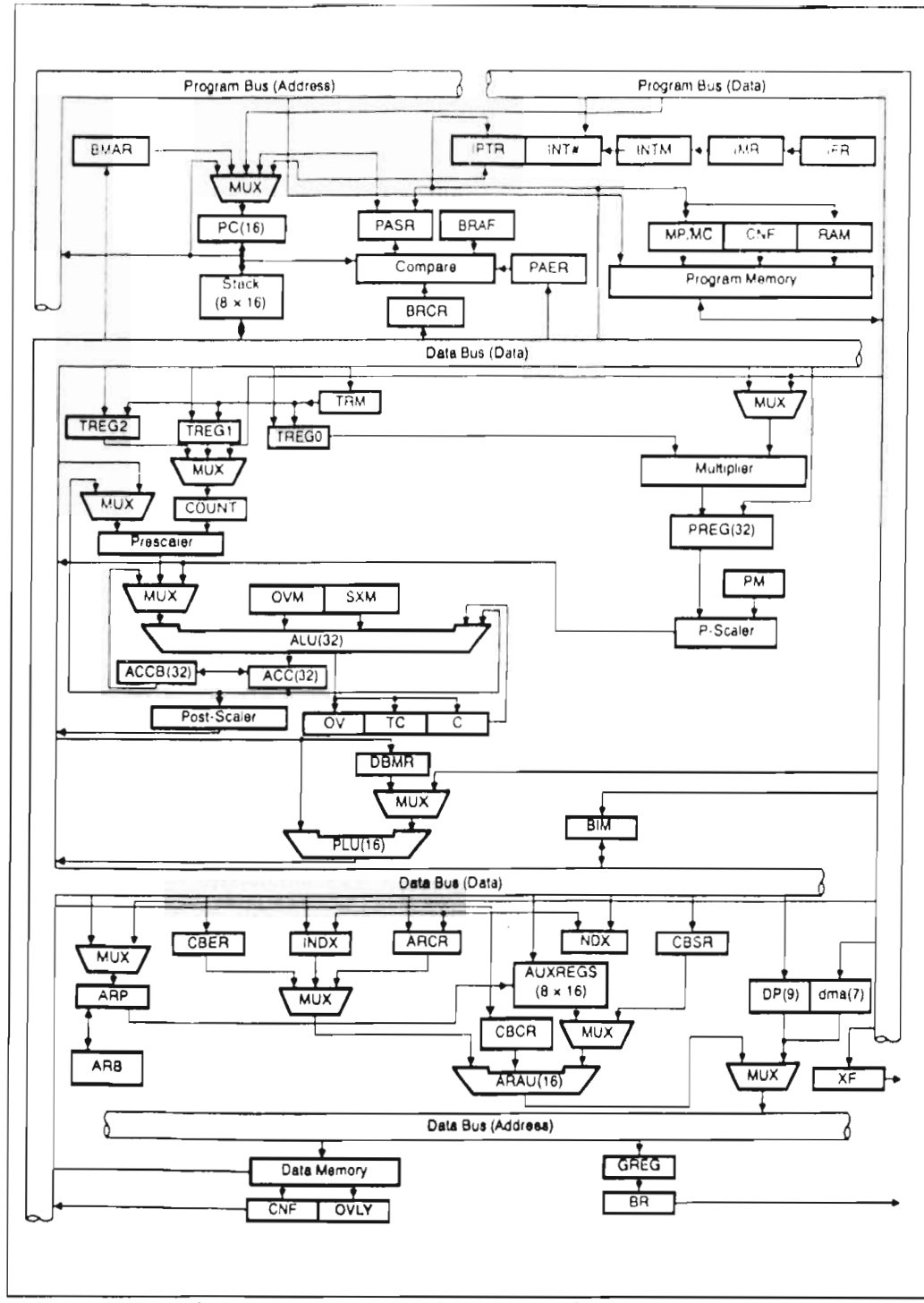


Figura B.1.2 Arquitectura del TMS320C50

La longitud total del Acumulador es de 32 bits, la cual es dividida en parte alta (bits 31 al 16 ACCH) y parte baja (bits 15 al 0 ACCL), y para el manejo de datos se tiene instrucciones de almacenamiento para cada una de las partes (alta y baja) del Acumulador.

El *multiplicador* realiza operaciones de 16x16 bits en modo dos complemento con un resultado de 32 bits en un sólo ciclo de instrucción. El multiplicador está compuesto de tres elementos: el registro TREG0 (de 16 bits) para almacenar temporalmente el multiplicado, el registro P (de 32 bits) para almacenar el producto y un arreglo multiplicador. Los valores del multiplicador provienen de la memoria dato, memoria programa (cuando se emplean las instrucciones MAC/MACD), o son derivados de una instrucción inmediata MPY (multiplicación inmediata). La rapidez del multiplicador integrado permite la ejecución de operaciones fundamentales en el DSP como la convolución, correlación y filtrado.

El registro de *corrimiento* del TMS50 tiene una entrada de 16 bits y está conectado al bus de datos, mientras que su salida de 32 bits está conectada a la ALU. Este registro proporciona corrimientos a la izquierda de 0 a 16 bits sobre el datos de entrada y son programados mediante instrucciones. Adicionalmente, se tiene la capacidad de que el procesador funcione con escalamiento numérico, extracción de bits, aritmética extendida y prevención de sobreflujo.

La *interfaz local* de la memoria del TMS consiste de un bus de datos paralelo de 16 bits (D15-D0), un bus de direcciones de 16 bits (A15-A0), tres pines para selección de memoria dato/programa o espacio de entrada/salida (/DS,/PS e /IS), y varias señales de control del sistema. La señal R/w controla el sentido de transferencia del dato, y la señal /STRB provee un tiempo válido para la transferencia de información, además contiene las señales de salida /RD y /WE que se pueden conectar directamente a memorias RAM con pines de entrada /OE y /WE. Cuando emplea las memorias ROM, RAM internas o memoria de programa externa de alta velocidad, el TMS50 ejecuta instrucciones a la máxima velocidad sin tiempos de espera. Para direccionar memorias externas lentas emplea de la señal READY es para generar tiempos de espera suficientes para la transferencia de información.

El *stack* por hardware de ocho niveles es empleado para almacenar el contenido de contador de programa durante interrupciones y llamadas de subrutinas. Las instrucciones PUSH y POP permiten salvar y recuperar información contenida en el stack. Las interrupciones empleadas en el dispositivo son mascarables.

Las operaciones de *control* son proporcionadas en el TMS50 por un timer interno de 16 bits mapeado en memoria, un contador de repetición, cuatro interrupciones externas mascarables, y una interrupción interna generada por el puerto serie o el timer.

Un *puerto serie* interno full-duplex provee comunicación directa con dispositivos seriales como codecs, convertidores seriales A/D, y otros dispositivos seriales. Los registros del puerto serial mapeados en la memoria (registros de transmisión/recepción de dato) pueden operar en modo byte (ocho bits) o modo word (16 bits). Cada registro tiene una entrada de reloj externa, una entrada de sincronía y registros de corrimiento. Contiene un puerto serial (TDM, time división

multiplexed) para aplicaciones de múltiples procesadores.

El TMS50 para aplicaciones de *múltiples procesadores* tiene la capacidad de distribuir el espacio global de memoria de dato y de comunicación con este espacio mediante las señales BR (requerimiento de bus) y READY. El registro de distribución de la memoria global de dato (GREG) de ocho bits especifica hasta 32K-Palabras de memoria dato como memoria global externa. El contenido del registro determina el tamaño del espacio global de memoria. Si la instrucción corriente direcciona un operando dentro de este espacio, /BR es insertado para requerir el control del bus. La extensión del ciclo de lectura/escritura de memoria es controlado con la línea READY.

El procesador TMS50 soporta *Acceso Directo a Memoria* (DMA) para su memoria externa dato/programa empleando las señales /HOLD y /HOLDA. Otro procesador puede tomar por completo el control de la memoria externa del TMS por medio de la señal /HOLD (activa baja), esta señal provoca que el TMS mantenga en estado de alta impedancia sus líneas de direccionamiento, datos y control. Sin embargo, de manera concurrente al modo DMA el TMS50 continuará ejecutando su programa si éste opera con la RAM y ROM internas.

La arquitectura del TMS50 está construida alrededor de dos buses: el de programa y el de datos. El bus de programa proporciona el código de instrucción y operandos inmediatos de la memoria de programa. El bus de datos interconecta varios elementos, como lo son la Unidad Aritmética Lógica Central (CALU) y los registros auxiliares a los datos RAM.
Tanto el bus de programa como el de datos, pueden transferir datos de memoria dato RAM interna y de la memoria programa interna o externa al multiplicador en un ciclo de instrucción para operaciones de multiplicación/acumulación.

El TMS50 tiene un alto grado de paralelismo, es decir, que mientras está operando sobre la CALU, puede implementar operaciones aritméticas en la Unidad Aritmética de Registros Auxiliares (ARAU). Tal paralelismo resulta en un poderoso conjunto de operaciones aritméticas, lógicas y manipulación de bits que se ejecutan en un solo ciclo de máquina.

REGISTROS

El TMS50 consta de 28 registros mapeados en la parte baja de la memoria dato, en las localidades 4 a 5F, estos se listan a continuación:

Registro	**Dirección (hex.)**	**Descripción**
-	0-3	Reservado
IMR	4	Registro de máscara de interrupción.
GREG	5	Reg. para localización global de memoria.
IFR	6	Reg. de banderas de Interrupción.
PMST	7	Reg. de modo estado del procesador.

RPTC	8	Reg. Contador de repetición.
BRCR	9	Contador de repetición de bloque.
PASR	A	Dirección inicial de programa para bloque de repetición.
PAER	B	Dirección final de programa para bloque de repetición.
TREG0	C	Reg. temporal para multiplicando.
TREG1	D	Reg. temporal para contador de corrimiento dinámico.
TREG2	E	Reg. temporal usado como apuntador de bit en prueba dinámica de bit.
DBMR	F	Manipulación dinámica de bit.
AR0-AR7	10-17	Registros auxiliares.
INDX	18	Indice.
ARCR	19	Reg. Auxiliar de comparación.
CBSR1	1A	Dirección inicial de buffer circular 1.
CBER1	1B	Dirección final de buffer circular 1.
CBSR2	1C	Dirección inicial de buffer circular 2.
CBER2	1D	Dirección final de buffer circular 2.
CBCR	1E	Registro de control de buffer circular.
BMAR	1F	Registro de dirección de movimiento de bloque.
	20-35h	Periféricos mapeados en memoria.
	36-4Fh	Reservado.
	50-5F	Puertos mapeados en memoria PA0-PA15.

Estos registros pueden cargarse al acumulador en direccionamiento indirecto o directo a través de la instrucción LAMM (carga el acumulador con registro mapeado en memoria) sin necesidad de hacer cambio de página. Para cambiar el contenido de estos registros se utiliza el acumulador salvándolo con la instrucción SAMM (salva el acumulador en registro mapeado), ejemplo:

```
LACL #1Ah       ; ACC = 1Ah
SAMM BRCR       ; Carga el ACC en el registro contador de bloque.
LAMM RPTC       ; ACC = RPTC
```

Organización de la memoria del TMS50

El TMS50 posee un total de 1056-Palabras de 16 bit de DRAM interna, de las cuales 544-Palabras son siempre memoria dato y las 512 restantes pueden ser configuradas como memoria programa o dato. También provee 2k-Palabras ROM.

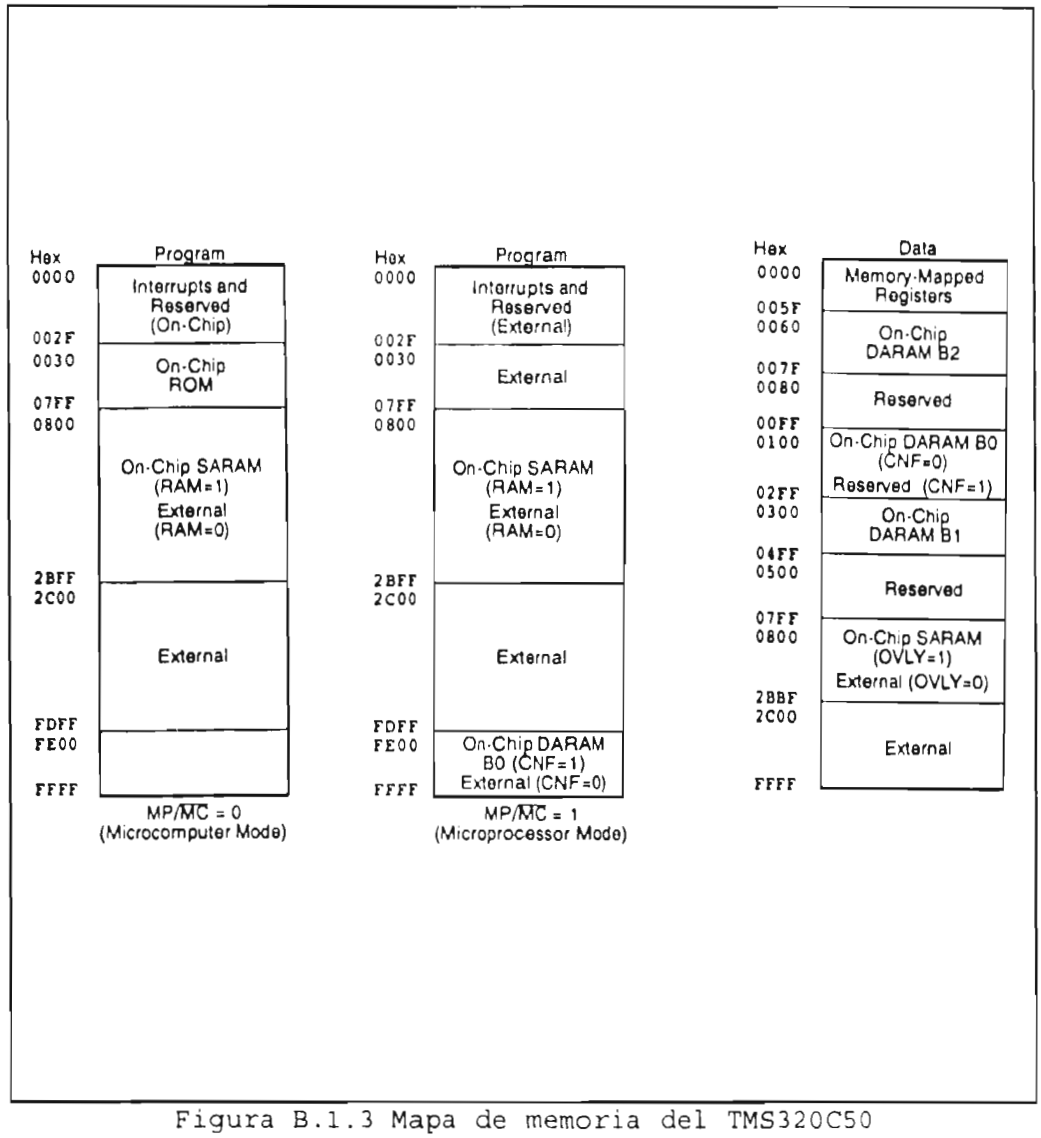


Figura B.1.3 Mapa de memoria del TMS320C50

Las 1056 Palabras en DRAM interna se dividen en tres bloques separados: B0, B1 y B2. El

bloque B0 de 512-Palabras es configurado como memoria programa (por instrucción SETC CNF) o dato (por medio de instrucciones CLRC CNF). Como memoria dato, B0 reside en las paginas 2 a 5 del mapa de memoria de datos, y como memoria de programa de $FDFF a $FFFF. Las 544 Palabras (bloques B1 y B2) siempre son configuradas como memoria dato. B1 esta localizado en las páginas 6 a 9 abarcando 512 localidades mientras que B2 está en las treinta y dos localidades superiores de la página 0 (cada página consta de 128 palabras cada una, es decir que el mapa completo tiene 512 paginas).

La memoria dato interna y las localidades reservadas para registros son mapeadas dentro de las primeras 1024-Palabras del espacio total de memoria dato (configuración con la instrucción CLRC CNF).

Las 4K-Palabras de ROM interna pueden ser un programa mascarable. Esta área de memoria permite la ejecución de programas a máxima velocidad sin necesidad de memorias externas veloces para almacenar el programa. El mapeo de estas 4K palabras es hecho por medio del pin de selección MP/mc (Microprocesador/microcomputadora). Manteniendo MP/mc en alto mapea este bloque de memoria como externo, y MP/mc en bajo lo mapea en ROM interna.

El TMS50 provee tres espacios de direccionamiento, para memoria programa, memoria dato e I/O. Estos espacios son distinguidos externamente por medio de las señales /PS (programa), /DS (dato) e /IS (puertos I/O). Las señales /PS, /DS, /IS Y /STRB son activadas sólo cuando un espacio externo de memoria está siendo direccionado. Durante un direccionamiento
interno estas señales permanecen en estado inactivo alto para prevenir conflictos de direccionamiento cuando el B0 es configurado como memoria programa. La señal R/w determina el sentido del flujo de los datos, es decir, lectura (alto) y escritura (bajo).

MODOS DE DIRECCIONAMIENTO

El TMS50 puede direccionar un total de 64K-Palabras de memoria de programa y 64K Palabras de memoria de datos. Los 16 bits del bus de direcciones (DAB) direccionan la memoria de datos de dos formas: modo de direccionamiento directo e indirecto.

Modo de direccionamiento directo

En este modo la memoria total de datos está dividida en páginas. Los 9 bits del apuntador de pagina DP pueden apuntar 512 páginas de 128 palabras cada una (64 K palabras). El dato de memoria direccionado es especificado por los 7 bits menos significativos de la instrucción para apuntar la palabra deseada dentro de la página (opera como un de offset sobre la página). Antes de usar este modo es necesario cargar el número de página en el registro DP, esto se hace con la instrucción LDP #pag.

Modo de direccionamiento indirecto

Este modo es muy eficiente para el direccionamiento de tablas y arreglos, en la instrucción no se expresa ninguna dirección de memoria. Los 16 bits de la dirección son seleccionados por el registro auxiliar en uso, direccionando el dato de memoria a través del bus de registro auxiliar buffer (AFB). El direccionamiento indirecto permite en la misma instrucción hacer modificaciones al contenido de los registros auxiliares, entre éstas están: postincremento (*+), postdecremento (*-), desplazamiento con registo INDX (*0+ o *0-) y direccionamiento de carry reverso para implementar la transformada rápida de Fourier (FFT) radix 2 (*BRO+).

Registros Auxiliares

El TMS50 posee 8 Registros Auxiliares: AR0-AR7, los cuales pueden ser utilizados para direccionamiento indirecto de memoria dato (como se observa en la figura), o almacenamiento temporal de datos. Estos registros son seleccionados por un apuntador de Registros Auxiliares de tres bits (ARP), cargándose con los valores de 0 a 7 para designar desde AR0 a AR7 respectivamente.

Los registros auxiliares están conectados a una Unidad Aritmética de Registros Auxiliares (ARAU), esta unidad efectúa operaciones en aritméticas no signada sobre los registros auxiliares en paralelo con la unidad CALU.

Direccionamiento circular

El direccionamiento circular es utilizado para implementar el deslizamiento de una ventana, la cual contiene los datos más recientes a ser procesados, los siguientes registros son utilizados para el direccionamiento circular:

CBSR1	1A	Dirección inicial de buffer circular 1.

CBER1	1B	Dirección final de buffer circular 1.
CBSR2	1C	Dirección inicial de buffer circular 2.
CBER2	1D	Dirección final de buffer circular 2.
CBCR	1E	Registro de control de buffer circular.

El registro CBCR es de 8 bits que se definen:

0-2	CAR1	Identifica que registro auxiliar es utilizado para el buffer circular 1.
3	CENB1	Habilita buffer circular 1 con un uno, con cero lo deshabilita.
4-6	CAR2	Identifica que registro auxiliar es utilizado para el buffer circular 2.
7	CENB1	Habilita buffer circular 2 con un uno, con cero lo deshabilita.

Antes de utilizar el direccionamiento circular hay que cargar los registros de inicio y final de buffer y escribir al registro de control CBCR.

El control del buffer circular lo efectúa la unidad ARAU haciendo la comparación:

IF(AR(ARP))=(CBER) entonces AR(ARP)=CBSR
lo contrario AR(ARP) = AR(ARP) + PASO

significa, que si el buffer circular ha terminado el registro índice en uso se recarga con la dirección inicial del direccionamiento circular.

El direccionamiento circular es útil en la implementación de convoluciones y filtros digitales.

Direccionamiento de registros mapeados

Este direccionamiento permite un acceso rápido a los registro mapeados en memoria. Opera similarmente al direccionamiento directo y se utiliza para acceder a los registros mapeados, en este caso se hertzio a los 9 bits más significativos de la dirección sean cero independiente del valor que tenga el registro de página, esto permite hacer un acceso directo a estos registros sin necesidad de hacer un cambio de página. La instrucciones utilizadas para este direccionamiento son LAMM y SAMM.

Modo inmediato corto y largo

Cuando un operando inmediato es empleado, el operando está contenido en la instrucción, el operando corto es una constante de 1 a 13 bits, en este caso el código de la instrucción es de 16 bits. Para caso de operandos inmediatos de 16 bits permite hace corrimientos a la izquierda sobre el operando de hasta 16 bits, el código de instrucción es de 32 bits, el corrimiento está en la parte baja de la primera palabra y el operando queda contenido en la segunda palabra de la instrucción, es decir que su ejecución es más lenta. La diferencia entre un operando corto o largo es el tamaño del operando y si existe corrimiento el operando es de tipo largo. Ejemplo:

Operando corto

```
ADD  #0C1h      ; suma al ACC una constante corta 0Ch
MPY  #03Ah      ; multiplica el registro TREG0 por la constante 3Ah
```

Operando largo

```
ADD  #111Fh,1   ; suma al ACC una constante larga con corrimiento de un bit a la
                  izquierda
MPY  #0124Ah    ; multiplica el registro TREG0 por la constante 124Ah
```

Modo de direccionamiento a registros

Este es un tipo de direccionamiento especial que opera con la unidad lógica paralela (PLU), la cual efectúa operaciones lógicas directamente sobre cualquier localidad de memoria dato sin afectar el contenido del ACC, permite direccionamiento directo e indirecto y operaciones con el registro de manipulación dinámica de bits (DBMR).

Movimiento de memoria a memoria

Son instrucciones que se utilizan para el movimiento de datos y programa, permiten utilizar eficazmente la configuración de la RAM interna. La instrucción BLDP mueve un bloque de datos de memoria dato a memoria programa y BLPD mueve un bloque de programa a memoria de datos. Para el empleo eficiente de estas instrucciones, se utilizan las instrucciones de repetición RPT y RPTZ.

La instrucción DMOV permite mover una palabra de la dirección actual en la memoria dato en RAM interna a la próxima localización alta, mientras que el dato de la dirección de localización está siendo operado en el mismo ciclo por CALU. Una operación de la ARAU también puede ejecutarse en el mismo ciclo cuando se usa el modo de direccionamiento indirecto. La función de DMOV es útil en la implementación de algoritmos donde se utilizan operadores de retardo Z^{-1}, tales como convolución y filtrado digital, donde el dato es pasado a través de una ventana de tiempo. La función de movimiento de dato puede ser utilizada dentro de los bloques B0, B1 y B2. Las instrucciones TBLR y TBLW (tabla de lectura y escritura) permiten transferir palabras entre el espacio de programa y de dato.

B.2 PROGRAMAS EN LENGUAJE ENSAMBLADOR

A manera de ejemplo se presenta la programación en lenguaje ensamblador del algoritmo LMS y el ARK por ser los más representativos.

```
*******************************************************
*       ALGORITMO DE FILTRADO ADAPTABLE LMS
*       Archivo :   lms_0.asm
*       Entrada x(k) en loc.   1000h
*       Salida  x'(k) en loc   1500h
*******************************************************
        .mmregs              ; incluye registros mapeados
        .ds     0F00h        ;
*
*       Vectores del algoritmo
*
YS      .word    0                    ;   Salida
ER      .word    0                    ;   error
ER1     .word    0                    ;   e*u
U       .q15     0.08
*
YK    .word   0
YK1   .word   0,0,0,0,0,0,0,0,0,0    ; Y(K-1) Aparta 10 loc. de memoria
YK2   .word   0,0,0,0,0,0,0,0,0,0
YK3   .word   0,0,0,0,0,0,0,0,0,0
YK0   .word   0,0,0,0,0,0,0,0,0
YKN   .word   0                      ; YN muestra penúltima
*
AK    .word     0,0,0,0,0,0,0,0,0,0  ; A(K-1)
AK2   .word     0,0,0,0,0,0,0,0,0,0
AK3   .word     0,0,0,0,0,0,0,0,0,0
AK0   .word     0,0,0,0,0,0,0,0,0
AN    .word     0                    ; Ultima A
*
N         .set      40               ; N
N1        .set      39               ; N-1
UNO       .set      1
TOT       .set      512              ; Total de Datos
*****************************************************************

          .ps     00A00h
          .entry
          SETC  INTM        ; des-habil. int. mask.
          SETC    SXM       ;
          SPM     0         ; sin corrimiento de Preg al cargarse a ACC

          LAR     AR5,#01500h  ; Apunta a Datos de Salida
          LAR     AR6,#TOT     ; Total de datos 256
          LAR     AR7,#01000h  ; Apunta a Datos Entrada

INICIO
          LDP     #YK
          MAR     *,AR7
          LACC    *+,13        ; Dato en Q15+12 = Q27 ,(12)
          SACH    YK           ; Dato entrada en Q11   (Q12)
*
          LAR     AR0,#YKN     ; primer retraso de senal
          MAR     *,AR0
          ZAP
          RPT     #N
CIC1      MACD    #AK,*-       ; obtiene y^(n) = Yki*Aki*2^23 (24)
```

```
            APAC
            ADD     #UNO,11
*
            MAR     *,AR5
            SACH    *+,4            ; YS salida en Q11
            SACH    YS,3            ; 4
*
            LACL    YK
            SUB     YS
            SACL    ER              ; ER en Q12
*
            LT      U               ; ER en Q15
            MPY     ER              ; U  en Q12
            PAC                     ; U*ER*2^27
            SACH    ER1,2           ; ER1 Q13
*
            LACL    #N1
            SAMM    BRCR
            LAR     AR0,#AK
            LAR     AR1,#YKN
            MAR     *,AR0
            LT      ER1

            RPTB    CIC2
            LACC    *,12,AR1        ; ACC  = AK*2^25            (25)
            MPY     *-,AR0          ; PREG = ER1*YK1*2^25       (25)
            APAC
*            ADD     #UNO,11
CIC2        SACH    *+,4            ;         Q13          w(k+1)
*
            MAR *,AR6

            BANZ INICIO,*-

        .end
```

```
*************************************************************************
* ALGORITMO RAPIDO DE KALMAN, para ejecutarse en DSP Starter Kit TMS50 *
*************************************************************************
        .mmregs                 ; incluye registros mapeados
        .ds     0F00h           ;
*
*       Vectores del algoritmo
*
YK      .word   0
YK1     .word   0,0,0,0,0,0,0,0,0    ; Y(K-1) Aparta 9 loc. de memoria
YN      .word   0                    ; YN muestra penúltima
YNM     .word   0                    ;    última muestra
YN1     .word   0                    ; reserva para DMOVE
AK      .word   0,0,0,0,0,0,0,0,0    ; A(K-1)
AN      .word   0       ;            ; Ultima A
BK      .word   0,0,0,0,0,0,0,0,0    ; B(K-1)
BN      .word   0                    ;
K       .word   0,0,0,0,0,0,0,0,0    ; M(n)
KN      .word   0                    ; último K
MK      .word   0                    ; MK[0] = epf/alfa
MK1     .word   0,0,0,0,0,0,0,0,0    ;
MP1     .word   0                    ; U(k)
*
YS      .word   0                    ;   "     de Salida
EFPK    .word   0                    ; Error forward a posteriori. En Q11
EFP     .word   0                    ;  "       "       priori
EBPK    .word   0                    ;  "    Backward a posteriori.
ALFA    .word   0                    ;
CTE_K   .word   0                    ; EFP/ALFA
UNO8    .word   0                    ; Almacena UNO en Q8
*         Para ls división
NUMERA    .word   00h                ; Numerador de entrada a Divide
NUMERA_F  .word   00h                ;   "                     Divide Fracción
DENOM     .word   00h                ; Denominador de entrada a Divide
COCIEN    .word   00h                ; Cociente de salida de Divide
COCIEN_F  .word   00h                ;    "          "    de Divide Fraccionario
RESUL     .word   00h                ; Une los dos cocientes
RESIDU    .word   00h
TEMSG     .word   00h                ; Signo
*
N         .set    10
N1        .set    9                  ; N-1
UNO       .set    1
TOT       .set    512                ; Total de Datos
*************************************************************
        .ps     00A00h
        .entry
        SETC  INTM      ; des-habil. int. mask.
        SETC    SXM     ;
        SPM     0       ; sin corrimiento de Preg al cargarse a ACC
*************************************************************************
**      Inicializa variables para KALMAN
**      Inicia ALFAn(k) = Delta  = 0400H  ( 1 en Q10 )
*
        LDP     #YK
        LAR     AR0,#YK
        MAR     *,AR0

        ZAP
        RPT     #45     ; limpia valores iniciales
        SACL    *+
```

```
        LACC    #UNO,10   ; acc =  uno*2^10
        SACL    ALFA      ; ALFA = UNO en Q10, 1024 ==> 1.0
        LACC    #UNO,8    ;
        SACL    UNO8      ; Uno en Q8
** ************************************************
*   Calcula Efpk(k+1) = y(k+1)+Ant(k)*YN(k)
*   YN, AN y YNM en Q11,Q12 y Q11 ==> Efpk  en Q11
*************************************************************
      LAR     AR5,#01500h  ; Apunta a Datos de Salida
      LAR     AR6,#TOT     ; Total de datos 256
      LAR     AR7,#01000h  ; Apunta a Datos Entrada

KALMAN
      LDP     #YK       ; página de variables
      MAR     *,AR7
      LACC    *+,12     ; ACCH = Señal entrada Q27
      SACH    YK        ; Señal de entrada en  Q11 = 27 -26
*
      LAR     AR0,#YK1  ; loc. de primer coef. A
      MAR     *,AR0
      ZAP               ; reg P = 0 , ACC = 0
      RPT      #N1
CIC1  MAC      #AK,*+    ; YKi*AKi*2^23

      APAC              ; Suma al Acc último producto
      ADD     #UNO,11   ; Redondea el Acc
      NEG               ; para obtener salida mismo signo
*
      MAR     *,AR5     ; PARA SALIDA
      SACH    *+,4      ; YS salida estimada en Q11 --->
*
         NEG               ; continua algoritmo signo original
         ADD     YK,12     ; YK + A . Y , Acc = XK*2^23h
         ADD     #UNO,11   ; Redondea el Acc
         SACH    EFPK,4    ; EPFK = ACCH*2^4   ==> EFPK en Q11, ARP=AR0

*        MAR     *,AR5     ; PARA SALIDA Del error
*        SACH    *+,4      ; YS salida estimada en Q11 --->
*****************************************************************
*   Calcula     An(k+1) = An(k) + Kp(k)*EFPK(k+1)
*   K y EFPK en Q13 y Q11 ===>  An en q12
*****************************************************************
        MAR      *,AR0
        LACC    #N1
        SAMM    BRCR
        LAR     AR0,#AK
        LAR     AR1,#K     ;  KN

        LT      EFPK       ; T = EPFK   en   Q11
*
      RPTB    CIC2
        LACC    *,12,AR1   ; ACC = AN*2^24 , AN en q12
        MPY     *+,AR0     ; P = ERPK*KNN*2^24
        APAC               ; ACC = ACC + P
        ADD     #UNO,11     ; Redondea el Acc
CIC2    SACH    *+,4        ; AP(i) = ACCH*2^4    , ARP = AR0
*************************************************************
*  Calculo de Efp(k+1) = y(k+1) + Ant(k+1)*YK1(k)
*     Y, AN y YK1 en Q11, Q12 y Q11 , ==> EfP en Q11
*****************************************************************
        LAR     AR0,#AK
        MAR     *,AR0
```

```
        ZAP                 ; ACC=PR=0
        LACC    YK,12       ; YK en Q11 * 2^12 --> Q23
        RPT      #N1
CIC3    MAC      #YK1,*+        ; P=(AN-i)*(YN-i)
        APAC
        ADD     #UNO,11     ; redondeo
        SACH    EFP,4       ; EFP= ACCH*2^4  en Q11    ..AR0
*****************************************************************
*   Calculo de ALFAn(k+1)=ALFAn(K)+EFPK(K+1)*EfP(k+1)
*   Efpk y Efp en Q11  ===> ALFAn EN Q10
*****************************************************************
        LACC     ALFA,12    ; ACC=ALFA*2^12 --> Q22
        LT      EFPK        ;
        MPY     EFP         ; P = EFPR*EFPO*2^22
        APAC
        ADD     #UNO,11
        SACH    ALFA,4      ; Alfa Q(22+4-16) = Q10

*     MAR     *,AR5       ; PARA SALIDA Del error
*     SACH    *+,4        ; YS salida estimada en Q11 --->

*****************************************************************
*   Calculo de Kn+1(0)=Efp(k+1)/ALFA(k+1) = CTE_K
*   EFP y ALPHAn en Q11 y Q10  ==> Kn+1(1) en Q13
*****************************************************************
        LACC    EFP,1         ; Acc = EFP*2^24
        SACL    NUMERA      ; Numera en Q8
        LACC    ALFA,2      ; Acc = Alfa*2^24
        SACL    DENOM       ; Denom  en Q8
           CALL     DIVIDE      ; ---->> Enf(k+1)/ALPHAn(k+1)
        LACL    RESUL       ; Resul viene en Q12  ,1
        SACL    CTE_K,1     ; CTE_K  en Q13
*
        NEG
        SACL    MK,1        ; Valor de MK en  Q13
*****************************************************************
*   Calculo de Kn+1(k+1)=[0 Kn(k)]T +Kn+1(1)*[1 An(k+1)]T
*   kn,Kn+w(1) y An en Q13,Q13 y Q12, ===> Kn+1(k+1) en Q13
*****************************************************************
        LACC    #N1
        SAMM    BRCR
*
        LAR      AR0,#K
        LAR      AR1,#AK
        LAR      AR2,#MK1

        MAR      *,AR0
        LT       CTE_K
        RPTB     CIC4
        LACC    *+,12,AR1     ; ACC=KNN*2^25
        MPY     *+,AR2         ; P =CTE_K*AN*2^25
        SPAC
        ADD      #UNO,11       ; Redondea
CIC4    SACH     *+,4,AR0      ; KNN=ACCH*2^4 =Q13 ....  AR0
*****************************************************************
*  Calculo de EBPK(k+1) = y(k+1) + BnT(k)*YK(k+1)
* YK,Bn y YN EN Q11,Q12 Y Q11 ===> EBPK en Q11
*****************************************************************
        MAR     *,AR0
        LAR     AR0,#YK
        ZAP                 ; ACC = PR = 0 ,
        LACC     YNM,12     ; ACC = YN*2^23
```

```
        RPT     #N1
CIC5    MAC     #BK,*+      ; P=(BN-1)*(YN-1)*2^23  MACD
        APAC
        ADD     #UNO,11
        SACH    EBPK,4      ; EPR=ACCH*2^4 ..............AR0
*****************************************************************
*   Kp(k)=[Mp(k) -Bp(k+1)*Up(k+1)]/[1+Un(k+1)*EBPK(k+1)]
* Mn,Un y EBPK en Q13,Q13 y Q11 ===> Bn en Q12
*****************************************************************
* Calcula  1 [1+Un(k+1)*EbPR(k+1 ]      +++++    ( - ) +
        ZALR    UNO8    ; ACCH = 0100
        LT      MP1     ; U(k)
        MPY     EBPK    ; P=(Un(k)*EBPK)*2^24
        APAC
        ADD     #UNO,15
        SACH    DENOM,2       ; DENOM en Q8
        LACC    UNO8,2
        SACL    NUMERA      ;
          CALL    DIVIDE  ; ---> ACCL=[1/[1+Un(k+1)*Eonb(k+1)] en Q8
        LACC    RESUL       ; en Q12
*
        SACL    CTE_K       ; CTE_K en Q12
*****************************************************************
*  Calcula Kp[K]=[Mp(k) -Bp(k+1)*U(n+1)]     Q13 - Q12*Q13            +
        LAR     AR0,#MK  ;
        LAR     AR1,#BK ;
        LAR     AR2,#K
        MAR      *,AR0    ;
        LACC    #N1
        SAMM    BRCR

        RPTB    CIC6
        LT      MP1         ; U(k) Q13
        LACC    *+,12,AR1   ; ACC=BN*2^25
        MPY     *+,AR2      ; P=(MKN*KMK)*2^25
        SPAC
        ADD     #UNO,11
        SACH    *,4         ; Temporal en Q13
        LT      *           ; T=(Bn-MN*Eonb)*2^25
        MPY     CTE_K       ; P=(t*(1/(1-Un*Eonb))*2^25
        PAC
        ADD     #UNO,11
CIC6    SACH    *+,4,AR0      ; K=ACCH*2^13  ........AR0   4
*****************************************************************
* Calcula Bn(k+1) = B(k)+kp(t)*EBPK      EBPK en Q11
* K y Bn en Q13,Q13 y Q12 ===> Bn+1 en Q12
*****************************************************************
        LAR     AR0,#BK
        LAR     AR1,#K
        MAR      *,AR0
        LT      EBPK
        LACC    #N1
        SAMM    BRCR
        RPTB    CIC7
        LACC    *,12,AR1 ; ACC=KMK*2^24
        MPY     *+,AR0   ; P=KNN*EBPR*2^24
        APAC
        ADD     #UNO,11
CIC7    SACH    *+,4       ; Bn=ACCH*2^12    ...AR0
*****************************************************************
* Movimiento de una muestra
        LAR     AR0,#YN    ; Yk
```

```
        MAR     *,AR0
        RPT     #N1         ; N
        DMOV    *-

        MAR     *,AR6
        BANZ    KALMAN,AR7


*****************************************************************
*           SUBRUTINAS
*****************************************************************
*     SUBRUTINA   DE  DIVISION
*     Entradas:  NUMERA        Q8
*                DENOM         Q8
*     Salida:    RESUL         18
*****************************************************************
DIVIDE                        ;
*
        LDP     #NUMERA
        CLRC    SXM           ; Solo para la división
        LT      NUMERA
        MPY     DENOM
        SPH     TEMSG         ; salva el signo temporal
*
        LACC    DENOM,16      ; para obtener verdadero absoluto
        ABS
        BSAR    15            ; *** no funciona corrimiento de 16
        SFR
        SACL    DENOM         ; Salva |DENOMINADOR|
*
        LACC    NUMERA,16     ; Alinea
        ABS
        BSAR    15
        SFR
        SACL    NUMERA        ; Salva | NUMERADOR |
****
        LACL    DENOM
        XC      #2,EQ         ; Si ACC = 0 ? ejecuta dos inst. Siguientes
        LACC    UNO,12        ;  ACC =  1  en Q12 evita división por cero
        SACL    DENOM       ;  Numerador = 1
***
        LACC    NUMERA
        SUB     DENOM         ; ACC  = NUMERA - DENOM
        BCND    DIV_E,GT      ; ---> Va hacer división  NUM > DEN
        LACC    #0
        SACL    COCIEN        ; En división fraccionaria el Coc. es cero
        LACC    NUMERA
        SACL    NUMERA_F
        B       DIV_F         ; ---> "    "       "       NUM < DEN
*****************************************************************************
*    DIVISION  ENTERA DONDE  | NUMERA | > | DENOM |
*    Inicia división si divisor y dividendo alineados
*
DIV_E
        LACC    NUMERA
        RPT     #15           ; 16 ciclos de división
        SUBC    DENOM

       SACL    COCIEN       ; Almacena cociente y residuo en el retorno
       SACH    NUMERA_F      ;
*
*****************************************************************************
```

```
*    DIVISION FRACCIONARIA     NUMERA  <  , DENOMINADOR  :
*    Inicia división si divisor y dividendo alineados
*
DIV_F
*
        LACC    NUMERA_F,16
        ABS
        RPT     #15          ; 16 ciclos de división
        SUBC    DENOM
        SACL    COCIEN_F      ; Almacena cociente y residuo en el retorno
        SACH    RESIDU      ;
*****************************
        LACC    COCIEN,16     ; Junta cociente entero y fraccionario
        OR      COCIEN_F      ;
        BSAR    4           ;  en Q8 ,8 x; en Q12 ,4 ; Q14 ,2  **4
        SACL    RESUL         ;

        BIT     TEMSG,0
        BCND    FIN_D,NTC
****
        LACL    #0           ; Si el signo es negativo
        SUB     RESUL
        SACL    RESUL
**
FIN_D
        SETC    SXM           ;
        RET

        .end
```

GLOSARIO

G.1 ABREVIATURAS Y SIGNIFICADO DE PALABRAS

ACC	Acumulador.
ACCL	Acumulador parte baja (bits 0-15).
ACCH	Acumulador parte alta (bits 16-31).
A/D	Conversión análogo digital.
ALU	Unidad aritmético lógica, del inglés "Arithmetic Logic Unit"
APAC	Suma al acumulador el contenido del registro producto.
ARK	Algoritmo rápido de Kalman.
AR	Modelo autoregresivo, del inglés "autoregresive".
ARMA	Modelo autoregresivo movimientro promedio, del inglés "autoregresive moving average".
ARAU	Unidad aritmética de registros auxiliares.
Backward	Hacia atrás.
BLP	Del inglés "Backward linear prediction".
CALU	Unidad central aritmético lógica.
D/A	Conversión digital analógica.
DMA	Acceso directo a memoria, del inglés "direct memory address".
DSK	DSP starter kit.
DSP	Procesador de señales digitales, del inglés "digital signal processing".
ECG	Electrocardiograma.
FAEST	Algoritmo rápido secuencial o Kalman a posteriori.
FFT	Transformada rápida de Fourier, del inglés "Fast Fourier Transform".
FIR	Respuesta finita al impulso, del inglés "finite impulse response".
FLP	del inglés "Forward linear prediction".
Forward	Hacia adelante.
FRLS	Algoritmo rápido RLS.
FTF	Filtro transversal rápido, del inglés "fast transversal filter".
ID	Intervalo dinámico
IEEE	Institute of electrical and electronics engineers.
IIR	Respuesta infinita al impulso, del inglés "infinite impulse response". .
INT	Parte entera de un número.
Lattice	Celosía, enrejado, entramado.
LMS	Algoritmo estocástico promedio aproximado (Algoritmo de Widrow), del inglés "least mean squares".
LS	Mínimos cuadrados, del inglés "least squares".
LSL	Mínimos cuadrados lattice, del inglés "least squares lattice".
MAC	Instrucción de multiplicación acumulación.
MACD	Instrucción de multiplicación acumulación y movimiento de datos.
MADS	Multiplicaciones y divisiones por recursión.
MC	Mínimos cuadrados.
MFLOPS	Millones de instrucciones en punto flotante por segundo.

MIPS	Millones de instrucciones por segundo.
ms	Milisegundos.
MSE	Error promedio cuadrático, del inglés "Mean square error".
MUX	Multiplexor
NLMS	Algoritmo normalizado LMS, del inglés "normalized least mean squares".
ns	Nanosegundos.
PARCOR	Parcial correlación.
PLU	Unidad lógica paralela.
PC	Computadora personal.
Q	Paso de cuantización.
Q15	Formato de punto entero, 15 bits para la parte fraccionaria de un número.
RAM	Memoria de acceso aleatorio, para lectura y escritura.
RLS	Algoritmo recursivo de los mínimos cuadrados.
ROM	Memoria de sólo lectura.
RPT	Instrucción que repite la siguiente instrucción N+1 vez.
RPTB	Instrucción que repite un bloque N+1 vez.
RPTZ	Similar a RPT pero inicializa el acumulador a cero.
SFTF	Algoritmo estabilizado FTF.
SUBC	Instrucción de resta condicional.
TDM	Multiplexión por división de tiempo.
TI	Texas Instruments.
TMS50	Procesador de señales digitales TMS320C50 de Texas Instruments.
μs	microsegundos
VLSI	Muy alta escala de integración.
W-H	Wiener & Hopf.

G.2 SIMBOLOS UTILIZADOS

k	Variable que representa el tiempo discreto.
$\varepsilon_p^b(k)$	Error backward a posteriori.
$\varepsilon_p^f(k)$	Error forward a posteriori.
α_p^b	Energía del error backward.
α_p^f	Energía del error forward.
$\gamma_p(k)$	Constante de verosimilitud.
μ	Constante de adaptación el algoritmo LMS
$A_p(k)$	Vector de coeficientes de predicción forward de p x 1.
$B_p(k)$	Vector de coeficientes de predicción backward de p x 1.
d(k)	Señal deseada.
e(k)	Señal de error en un algoritmo adaptable.
$e_p^b(k)$	Error backward a priori.
$e_p^f(k)$	Error forward a priori.
$k_p(k)$	Vector de coeficientes de reflexión lattice de p x 1.
$K_p(k)$	Vector de coeficientes de la ganancia de Kalman p x 1.
$M_p(k)$	Igual a la ganancia de Kalman $K_p(k)$.
N	Orden de un filtro FIR.
p	Orden de predicción.
$P_p(k)$	Inverso de $R_p(k)$.
$R_p(k)$	Matriz de autocorrelación de p x p.
$U_p(k)$	Ultimo elemento de la ganancia de $K_{p+1}(k)$.
x(k),u(k)	Señal de entrada a una algoritmo adaptable.
y^(k)	Señal estimada en un algoritmo adaptable.