UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

PROGRAMA DE MAESTRIA Y DOCTORADO EN INGENIERIA

FACULTAD DE INGENIERÍA

ESTUDIO E IMPLEMENTACION DE ALGORITMOS DE FILTRADO ADAPTABLE EN ARITMETICA DE PUNTO FIJO

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERIA

INGENIERIA ELÉCTRICA – PROCESAMIENTO DIGITAL DE SEÑALES

P R E S E N T A :

HECTOR GALINDO SALINAS

TUTOR:
DR. ROGELIO ALCÁNTARA SILVA

2008

JURADO ASIGNADO:

Presidente: DR. GARCÍA NOCETTI DEMETRIO FABIÁN

Secretario: DRA. MEDINA GÓMEZ LUCIA

Vocal: DR. ALCÁNTARA SILVA ROGELIO

1er. Suplente: DR. GARCÍA UGALDE FRANCISCO

2do. Suplente: DR. PÉREZ ALCÁZAR PABLO ROBERTO

Lugar o lugares donde se realizó la tesis:

CIUDAD UNIVERSITARIA MÉXICO D.F.

TUTOR DE TESIS:

ROGELIO ALCÁNTARA SILVA

_____

FIRMA

AGRADECIMIENTOS

A mis padres Adela y David por el amor y apoyo incondicionales, el constante aliento en todos y cada uno de mis desafíos y por el ejemplo de vida de ser y hacer siempre con integridad, dignidad, humanidad y pasión.

A mi hermano David por su cariño, solidaridad, amistad y apoyo incondicional.

A Guadalupe por su amor y comprensión y a Liz por su alegría y entusiasmo siempre contagiosos.

A mis tíos Mercedes y Abel por su constante preocupación e interés en mi desarrollo.

A mi tía Victoria ejemplo de tenacidad y superación.

A mis profesores, quienes han contribuido a mi formación y al enriquecimiento en la elaboración de este trabajo, particularmente al Dr. Francisco Javier Garcia Ugalde por su apoyo y atenciones.

Un agradecimiento especial al Dr. Rogelio Alcántara Silva por la amistad y confianza depositadas en mi, así como por sus generosas contribuciones en la dirección de esta tesis.

INDICE

RESUMEN

La gran mayoría de los algoritmos de procesamiento digital de señales son concebidos en primera instancia para su implementación en lenguajes de programación de alto nivel (C, MATLAB, Java, etc.) con tipos de datos de punto flotante, sin embargo, las características de cálculo intensivo, grandes anchos de banda y superiores capacidades de almacenamiento que demandan las aplicaciones modernas, hacen imprescindible la implementación de los algoritmos en arquitecturas de punto fijo; esto, con la finalidad de cumplir con los requerimientos de disminución de costos, bajo consumo de energía y tiempos de cálculo reducidos que exige el mercado. Dado que, la programación de los algoritmos en aritmética de punto fijo resulta ser una tarea altamente laboriosa, consumidora de tiempo y propensa a errores, cada vez más se hace necesario el uso de metodologías y herramientas de desarrollo que no solo simplifiquen sino que también permitan la sistematización de las diferentes tareas como la evaluación de la dinámica de las variables, las estrategias de cuantización y la traducción automática de punto flotante a punto fijo. En este trabajo presentamos los resultados de la implementación en aritmética de punto fijo con longitudes de palabra mínima de los algoritmos rápidos de los mínimos cuadrados bajo el contexto de las aplicaciones de predicción, la identificación y la igualación de canal. Para este fin, utilizamos una metodología de implementación y la herramienta especializada *Matlab Fixed Point Toolbox*. Una vez determinada la dinámica de las variables de los diferentes algoritmos, de manera heurística mediante simulaciones de punto flotante, se determinaron las longitudes de palabra mínimas con las cuales funcionaron correctamente. Nuestro estudio comparativo muestra que los algoritmos SFTF y FAEST pueden funcionar con longitudes de palabra de 16 bits bajo el contexto de la igualación de canal.

1. INTRODUCCIÓN

Muchos algoritmos de procesamiento de señales son expresados de manera natural en una representación de punto flotante, sin embargo los cálculos en punto flotante requieren una mayor superficie del procesador, o emulaciones de software más lentas. En muchas aplicaciones el costo resultante del sistema y/o el consumo de energía puede no ser aceptable. [Aam01]. Esta situación se resuelve normalmente con una versión *codificada a mano* del algoritmo original con un error aceptable debido a los efectos de la longitud de palabra finita. Sin embargo, el proceso de convertir manualmente aun el algoritmo más sencillo es una tarea consumidora de tiempo, tediosa y propensa a errores. Estos factores, además del hecho de que los lenguajes de programación de alto nivel como MATLAB o ANSI C, requieren de extensiones del lenguaje (bibliotecas, librerías, toolboxes o herramientas) para expresar efectivamente los algoritmos de punto fijo [LW90], han motivado el desarrollo de utilidades de conversión de punto flotante a punto fijo que automaticen al menos parcialmente el proceso [MVH+97], [KKS97], [AC99], [AC00], [Syn00].

En la presente era de la tecnología de la información, los productos de consumo requieren más poder de cálculo por parte de las arquitecturas de procesamiento, ejemplos de esto son el crecimiento en la demanda por acceso a internet inalámbrico de banda ancha y las aplicaciones de información relacionadas, haciendo a las arquitecturas de procesamiento dedicadas mucho más visibles tanto para los consumidores como para la academia.

El desarrollo de este tipo de arquitecturas pone especial énfasis en diferentes criterios de diseño en comparación con el computo de propósito general: Bajo consumo de energía y requerimientos de procesamiento de señales en tiempo real, combinados con la demanda constante del menor costo unitario para impactar todas las facetas del diseño.

Idealmente, se desea que una herramienta tome una representación de alto nivel en punto flotante de un algoritmo y genere automáticamente código maquina en punto fijo que provea una aproximación lo suficientemente buena en comportamiento de entrada/salida de la especificación original mientras se cumplen los requerimientos de procesamiento en tiempo real.

Una de las multiples aplicaciones en el procesamiento digital de señales son los ASIC de procesamiento de señales modernos, tales como cable modems integrados y terminales multimedia inalámbricos, los cuales son específicamente diseñados para algoritmos en precisión de punto flotante. Sin embargo, muy a menudo el estilo conceptual con el que estos algoritmos son implementados es con una precisión limitada, y se termina en una representación de punto fijo. Esto requiere de un diseñador que traduzca tipos de datos, operaciones y variables del algoritmo de punto flotante a tipos de datos, operaciones y variables del algoritmo en punto fijo, empleando para esto una estrategia de refinación.

Para cada objeto de punto flotante refinado, ciertas características de punto fijo (incluidas las longitudes de la parte fraccional y la parte entera, sobreflujo y comportamiento del redondeo) deben elegirse [CRS+99].

De manera general se puede decir que existen dos tipos de aproximaciones que llevan la responsabilidad del diseñador en la cuantización a un nivel más sustancial. [CRS+99]

- La **aproximación basada en la simulación** [SK95] la cual compara el desempeño del sistema con un modelo de referencia. Las longitudes de palabra se eligen heurísticamente mientras se cumple con algún criterio de error. Este proceso se repite hasta que las longitudes de palabra convergen. Una solución más elaborada la cual utiliza la técnica de sobreflujo de operadores y monitoreo de las características de la señal fue presentada en [SKW95].

  Este método produce resultados precisos pero [MVH+97] puede requerir procesos de simulaciones muy grandes en caso de una convergencia lenta, lo cual no es aceptable para sistemas complejos.

- Otro método sugiere una aproximación analítica [MVH+97]. Las longitudes de palabra se derivan utilizando estructuras de código fuente, anotaciones locales y un método de interpolación. Este método produce resultados rápidamente, pero es una aproximación conservadora que lleva a una sobreestimación de las longitudes de palabra de la señal [SK95].

En este trabajo nos vamos a concentrar únicamente en métodos basados en la simulación para realizar el estudio de las dinámicas y así determinar los rangos de las variables para emplear longitudes de palabra adecuadas para la implementación de los diferentes algoritmos empleando la herramienta de punto fijo de Matlab. Por lo tanto antes es necesario conocer antecedentes y conceptos tanto de la implementación de algoritmos en punto fijo como de los algoritmos de filtrado adaptable, temas que se desarrollan en los capítulos siguientes de esta tesis.

1.1. Implementación de Algoritmos en Aritmética de Punto Fijo

La implementación de algoritmos en aritmética de punto fijo implica definir la posición del punto decimal para cada dato empleado en la ejecución de la implementación respetando siempre las reglas de la aritmética de punto fijo para cada operación, con el objeto de mantener la funcionalidad del algoritmo y garantizar la ausencia de desbordamientos o sobreflujos, para satisfacer así los requerimientos de precisión [Men03].

La figura 1.1 muestra de manera general el proceso de llevar la implementación de un algoritmo de punto flotante a punto fijo, en el cual se puede apreciar que partiendo del código de MATLAB en punto flotante se realiza la evaluación de la dinámica de todas y cada una de las variables del algoritmo para conocer la cantidad de bits necesaria para poder representar los valores máximos y mínimos de las variables evitando sobreflujos. Una

vez conocida dicha dinámica se realiza la codificación de los datos en punto fijo, lo cual implica respetar las reglas aritméticas del punto fijo en las operaciones que intervienen en el algoritmo. Posteriormente se evalúa la precisión de la implementación, de manera que si cumple los requisitos de precisión de la implementación en punto flotante se considere una implementación en punto fijo valida.

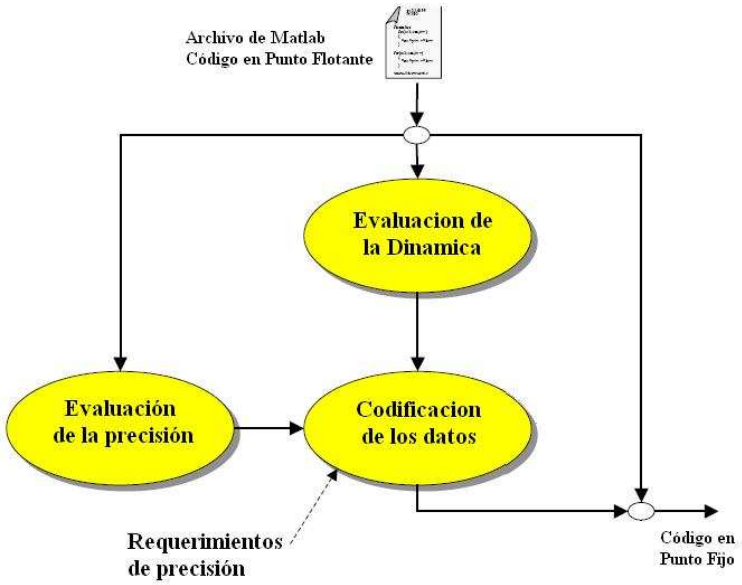


Fig. 1.1: Implementación de Punto Fijo a partir del algoritmo de Punto Flotante .

El realizar la implementación del algoritmo en punto fijo puede conllevar tanto ventajas desde el punto de vista material: minimizar la superficie del material empleado (pistas) así como el consumo de energía, como ventajas desde el punto de vista lógico: minimizar los tiempos de ejecución y el tamaño del algoritmo.

1.1.1. Operaciones Numéricas en Punto Fijo

El caso más evidente de la problemática que implica el realizar operaciones numéricas con datos representados en longitudes de palabra finita es el de la multiplicación de dos enteros binarios. El resultado de multiplicar dos valores enteros binarios de 16 bits es un valor entero binario de 32 bits. Sin embargo este resultado debe ser almacenado en una longitud de palabra finita de 16 bits. Los bits menos significativos no pueden ser truncados arbitrariamente del final del número, dado que ellos representan la magnitud del número y esta es una parte esencial de su representación [CH06].

Con la representación de punto fijo, una respuesta a este problema es el escalamiento de los valores numéricos en el sistema de manera tal que estos sean fracciones entre -1 y 1. Dado que el 1 positivo no se puede representar, el limite superior del rango solo puede llegar a $(1 - \varepsilon)$, en donde ε es el número más pequeño que puede ser representado por el número de bits en el sistema. Entonces, el máximo valor positivo posible de representar con 16 bits es el valor fraccional 0,999969482 y no 1. Para simplificar este rango es llamado

normalmente como −1 a 1.

El escalar valores en el rango −1 a 1 requiere representar todos los números incluyendo los datos de entrada y coeficientes del algoritmo como fracciones. Los números pueden ser normalizados como representaciones fraccionales moviendo la posición del punto hacia la izquierda de la palabra. Desplazar el punto hacia la izquierda una posición en la palabra binaria es el equivalente a dividir por 2, mientras que desplazarlo hacia la derecha una posición es el equivalente de multiplicarlo por 2. Siempre y cuando todos los valores sean escalados de la misma manera los resultados operacionales serán equivalentes.

Dado que dos fracciones multiplicadas siempre resultan en otra fracción, y la magnitud de un número fraccional no esta significativamente representada por sus bits menos significativos LSB, los LSBs pueden ser truncados permitiendo almacenar el resultado en la longitud finita requerida. Esto resuelve e problema del crecimiento de la magnitud de multiplicaciones con representación de punto fijo. Caso contrario que la multiplicación, sumar dos fracciones puede resultar en un valor mayor a la unidad. Cabe mencionar que para poder sumar dos números binarios estos deben tener el punto decimal en la misma posición de la palabra.

Si los resultados de alguna operación resultan ser mayores a la unidad ocurre lo que se conoce como una condición de sobreflujo y el o los resultados de los algoritmos son inválidos. Estas condiciones de sobreflujo tienden a causar estragos en el sistema. La solución a este problema es que los valores de entrada sean lo suficientemente pequeños para evitar condiciones de sobreflujo (o saturación si es que se esta trabajando con modo de saturación).

En conclusión, normalizar los valores en el rango de −1 a 1 le añade muchas cargas significativas a la implementación de algoritmos de punto fijo causando que se deban estudiar las dinámicas para evitar los sobreflujos asociados con sumas o multiplicaciones.

1.1.2. Efectos de la Longitud de Palabra Finita

Debido a que los datos en punto fijo deben ser representados en una longitud de palabra finita, es decir, con un número finito de bits, existen diferencias entre el desempeño de un sistema ideal (precisión infinita) y un sistema de punto fijo de una aplicación real.

Las fuentes de error del sistema pueden incluir [CH06]:

- Sobreflujo del registro, o modo de saturación del registro.
- Errores aritméticos.
- Errores en la representación de los coeficientes.

Mientras los efectos de cuantización añaden un ruido inicial (imprecisión o degradación de la señal) a la implementación de punto fijo, los efectos de la longitud de palabra finita añaden fuentes de ruido adicional a el sistema.

Algunas de estas fuentes incluyen [CH06]:

- Errores en la aritmética dentro de la implementación del algoritmo (truncamiento del punto fijo).
- Error de truncamiento cuando se almacenan los datos.
- Cuantización de los coeficientes que se deben almacenar en memoria.

Los efectos de la cuantización son no lineales, esto resulta en errores dependientes de la señal.

1.1.3. Reducción de Efectos de la Longitud de Palabra Finita

Al utilizar procesadores o arquitecturas de punto fijo se debe comparar la magnitud máxima posible de los datos resultante de una cadena de operaciones con la magnitud de la máxima representación numérica y ajustarse para reducir o prevenir las fuentes de error.

Cuando se identifican las areas problemáticas (via análisis, simulación o por prueba) existen algunas medidas correctivas que se pueden tomar [CH06]:

- Escalar los valores de entrada o los coeficientes.
- Seleccionar una arquitectura equivalente o alternativa.
- Sacar ventaja de las características de la arquitectura de los procesadores DSP.
- Implementar los bloques de cálculos sospechosos con modo de saturación.

El método de corrección más simple es el escalamiento. El escalar los valores se puede aplicar a ciertos tipos de algoritmos incluyendo los filtros.

1.2. Generalidades de Filtrado Adaptable

Para abordar los temas contemplados en los capítulos siguientes de esta tesis, es necesario introducir algunos conceptos de filtrado ya que la familia de algoritmos que se van a emplear en las implementaciones de punto fijo pertenece a la categoría de filtrado adaptable. Por lo tanto comenzaré introduciendo el término filtrado el cual es usado comúnmente para describir algún dispositivo ya sea hardware o software que es aplicado a un conjunto de datos con ruido para extraer la información que sea de nuestro interés. El ruido puede provenir de diversas fuentes, por ejemplo, los datos pueden ser el resultado de la medición de sensores con ruido o puede representar una componente de la señal que ha sido corrompida por la transmisión a través de un canal de comunicación. En cualquier caso tendríamos que usar un filtro para llevar a cabo las tres operaciones básicas del procesamiento de la información.

- 1. Filtrado. Implica la extracción de información de interés a un instante de tiempo t.
- 2. Suavizado. Difiere del filtrado en que la información de interés no necesita estar disponible en un instante de tiempo t y los datos medidos después del tiempo t pueden ser usados obteniendo esta información.
- 3. Predicción. es el pronóstico del proceso de la información, el objetivo aquí es derivar la información acerca de como sera la cantidad de interés en cierto tiempo $t + \tau$, para $\tau > 0$, mediante el uso de los datos medidos incluyendo el tiempo t.

Podemos decir que el filtro es lineal si la cantidad de interés filtrada, suavizada o predicha a la salida del dispositivo es una función lineal de las observaciones aplicadas a la entrada del filtro.

Ahora bien por filtro adaptable entendemos a aquel dispositivo que es autodiseñable, para la que se basa en un algoritmo recursivo que le permite al filtro desempeñarse satisfactoriamente en un ambiente en el cual el completo conocimiento de las características relevantes de la señal no se encuentran disponibles. El algoritmo parte de algunas condiciones iniciales predeterminadas, que representan completa ignorancia del ambiente. Aun en ambientes estacionarios encontramos que después de iteraciones sucesivas del algoritmo, este converge a la solución óptima de Wiener (ver apéndice) en algún sentido estadístico. En un ambiente no estacionario, el algoritmo ofrece *capacidad de Seguimiento*, de manera que puede rastrear variaciones en el tiempo en las estadísticas de los datos de entrada, proveyendo que las variaciones sean suficientemente lentas.

Como consecuencia directa de la aplicación de un algoritmo recursivo, los parámetros de un filtro adaptable son actualizados de una iteración a la siguiente, de manera que los parámetros se convierten en *dependientes de los datos*

El diseño de un filtro de Wiener (ver apéndice A) requiere información *a priori* acerca de las estadísticas de los datos que se van a procesar. El filtro es óptimo solo cuando las características estadísticas de los datos de entrada coinciden con la información a priori utilizada para el diseño del filtro cuando esta información no se conoce completamente.

Una gran variedad de algoritmos recursivos han sido desarrollados para la operación de filtros adaptables. Se puede concluir que la elección de un algoritmo depende de uno o más de los siguientes factores [Hay96][Esc97]:

- 1. Rango de convergencia. Este se define como el número de iteraciones requeridos por el algoritmo, para converger a la solución de Wiener óptima en el sentido de los mínimos cuadrados. Un rango de convergencia rápido le permite al algoritmo adaptarse rápidamente a ambientes estacionarios de estadísticas desconocidas.

- 2. Desajuste. Este parámetro provee una medida cuantitativa de cuanto se desvía el valor final del error medio cuadrático de un arreglo de filtros adaptables con respecto al mínimo error medio cuadrático que produce el filtro de Wiener.

- 3. Seguimiento. Un algoritmo de filtrado adaptable trabajando en un medio no estacionario requiere rastrear variaciones estadísticas en el ambiente. El desempeño de seguimiento del algoritmo, es influenciado por dos características: 1) el rango de convergencia y 2) las fluctuaciones en estado estacionario debidas al ruido del algoritmo.

- 4. Robustez. Para que un filtro adaptable se considere robusto, pequeñas perturbaciones solo pueden resultar en pequeños errores de estimación. Las perturbaciones pueden deberse a una variedad de factores internos o externos del filtro.

- 5. Requerimientos Computacionales. Aquí los aspectos de interés incluyen a) el número de operaciones requeridas para completar una iteración del algoritmo, b) el tamaño de las localidades de memoria requeridos para almacenar los datos y el programa y c) la inversión requerida para programar el algoritmo en una computadora.

- 6. Estructura. Este aspecto se refiere a la estructura del flujo de datos en el algoritmo, determinando la manera en que es implementado en hardware.

- 7. Propiedades Numéricas. Cuando un algoritmo es implementado numéricamente, se producen imprecisiones debidas a *errores de cuantificación*. Los errores de cuantificación se deben a la conversión analógica-digital de los datos de entrada y

a la representación digital en los cálculos internos. Normalmente, son los errores de cuantificación los que representan un serio problema de diseño. En particular, hay dos temas básicos de interés: estabilidad numérica y precisión numérica. La *estabilidad numérica* es una característica inherente de un algoritmo de filtrado adaptable. Por otro lado, la *precisión numérica* es determinada por el número de bits usados en la representación numérica de las muestras de datos y los coeficientes del filtro. Se dice que un algoritmo de filtrado adaptable es robusto numéricamente cuando es insensible a variaciones en la longitud de palabra utilizada en su implementación digital.

Como ya sabemos, un filtro para estimar una señal deseada puede diseñarse utilizando fórmulas estocásticas o determinísticas. En el caso determinístico, el diseño del filtro requiere del cálculo del promedio de ciertas cantidades utilizando el conjunto de datos dados que el filtro procesará. Por otro lado, el diseño de un filtro de Wiener requiere conocimiento *a priori* de las estadísticas de la señal. En sentido estricto, un gran número de realizaciones de la señal son necesarias para estimar confiablemente las estadísticas. Este procedimiento no es factible en la practica por que normalmente tenemos una sola realización de cada una de las secuencias de la señal. Para resolver este problema se asume que las secuencias de la señal son ergódicas, es decir que son estacionarias y que sus propiedades estadísticas y promedios temporales son idénticos. Por lo tanto, al utilizar promedios temporales, los filtros de Wiener pueden ser diseñados, aún cuando exista únicamente una realización de cada secuencia de la señal.

Aún cuando es posible medir directamente los promedios de la señal para obtener la información necesaria para el diseño de los filtros de Wiener óptimos (ver apéndice A), en la mayoría de las aplicaciones los promedios de la señal se utilizan de manera indirecta. Algunos algoritmos toman la salida del error del filtro, lo correlacionan de alguna manera con las muestras de entrada al filtro y utilizan ese resultado en una ecuación recursiva para ajustar los coeficientes del filtro de manera iterativa. Algunas de las razones para resolver el problema del filtrado adaptable de manera iterativa son[FB98]:

- El cálculo directo de los promedios necesarios y su empleo para el cálculo de los coeficientes del filtro requiere de la acumulación de una gran cantidad de muestras de la señal. Las soluciones iterativas, en cambio, no requieren acumulación de muestras de la señal, *resultando en un ahorro significativo de la cantidad de memoria.*

- La acumulación de muestras de la señal y su post procesamiento para generar la salida del filtro, como se requiere en las soluciones no iterativas, introduce un gran retardo en la salida del filtro. Esto es inaceptable en muchas aplicaciones. Por el contrario, *las soluciones iterativas no introducen un retardo significativo en la salida del filtro.*

- El uso de iteraciones resulta en soluciones adaptables con algunas capacidades de rastreo (*seguimiento*). Esto es, si las estadísticas de la señal cambian con el tiempo, entonces la solución proporcionada por un ajuste iterativo de los coeficientes del

filtro será capaz de adaptarse a las nuevas estadísticas.

- En general, las soluciones iterativas, son mucho más *simples* de codificar en software o implementar en hardware que aquellas que sus contrapartes no iterativas.

1.2.1. Estructuras de Filtros Adaptables

1.2.2. Filtros Transversales

La estructura más utilizada para la implementación de filtros adaptables es la estructura transversal, mostrada en la figura 1.2. Aquí el filtro tiene una salida única $y(k)$, y una entrada $x(k)$. La secuencia $d(k)$ es la señal deseada. La salida $y(k)$ es generada como una combinación lineal de muestras con retardo de la secuencia de entrada $x(k)$, de acuerdo a la ecuación

$$y(k) = \sum_{i=0}^{N-1} w_i(k)x(k-i) \tag{1.1}$$

en donde $w_i(k)$ son los coeficientes del filtro y N es la longitud del filtro. Las muestras de entrada, $x(k-i)$ para $i = 0, 1, \ldots, N-1$ son las entradas del filtro. Los coeficientes, $w_i(k)$; que pueden variar con el tiempo; son controlados por el algoritmo adaptable.

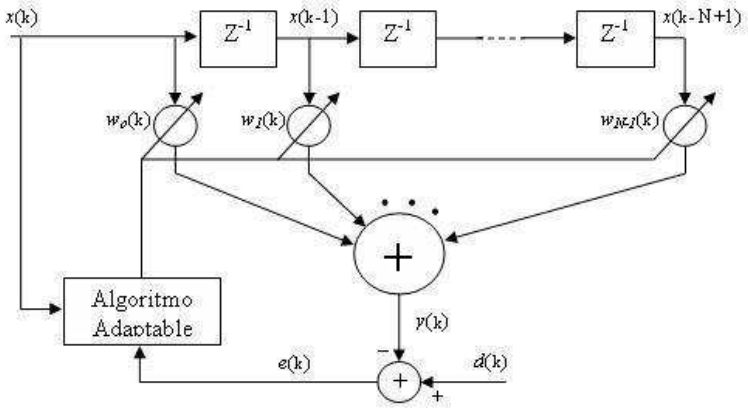


Fig. 1.2: Filtro Transversal Adaptable.

En algunas aplicaciones, como generación de espectros, las entradas del filtro no son las muestras retardadas de una sola entrada. En esos casos la estructura del filtro adaptable asume la forma mostrada en la figura 1.3. Esta es llamada combinador lineal, dado que su salida es una combinación lineal de las diferentes señales recibidas en su entrada:

$$y(k) = \sum_{i=0}^{N-1} w_i(k)x_i(k) \tag{1.2}$$

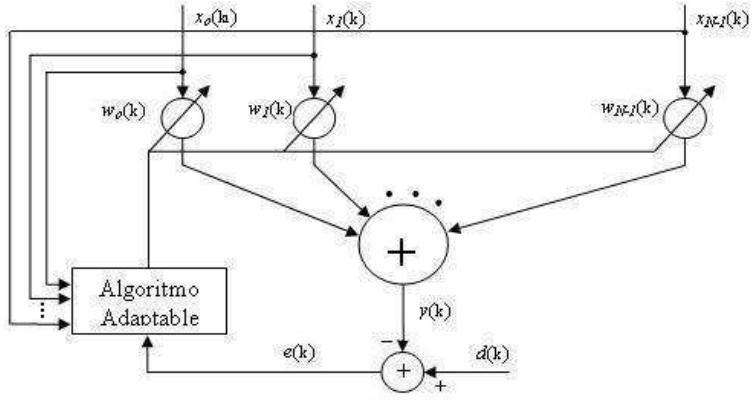


Fig. 1.3: Combinador Lineal Adaptable.

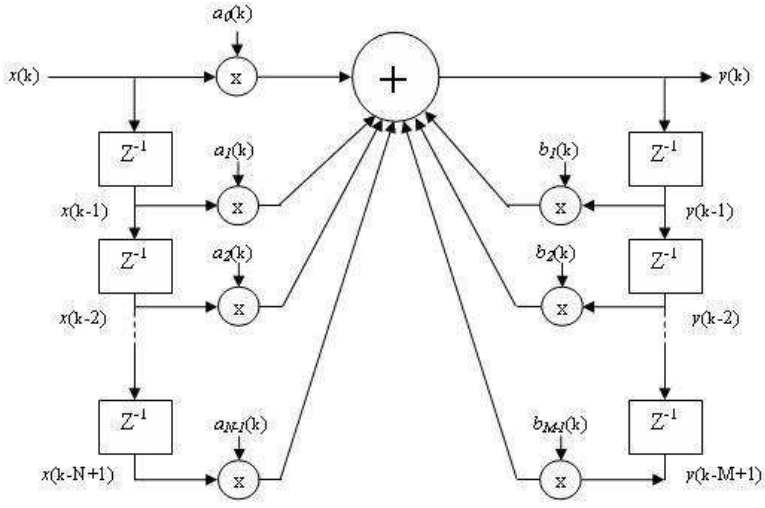


Fig. 1.4: Estructura de un Filtro IIR.

Nótese que la estructura de combinador lineal es más general que la transversal. La estructura en escalera (ladder) puede ser obtenida eligiendo $x_i(k) = x(k-i)$.

Las estructuras de las figuras 1.2 y 1.3 son de filtros no recursivos, es decir, el cálculo de la salida no cuenta con ningún mecanismo de retroalimentación. La figura 1.2 también es conocida como filtro de respuesta impulsional finita (FIR), dado que su respuesta al impulso es de duración finita en el tiempo. Un filtro con respuesta impulsional infinita (IIR) es gobernado por ecuaciones recursivas como (ver figura 1.4)

$$y(k) = \sum_{i=0}^{N-1} a_i(k)x(k-i) + \sum_{i=1}^{M-1} b_i(k)y(k-i) \tag{1.3}$$

en donde $a_i(k)$ y $b_i(k)$ son los coeficientes forward y feedback, respectivamente [CG86].

Los filtros IIR han sido utilizados en numerosas aplicaciones. Sin embargo, dadas las dificultades que implica la adaptación de filtros IIR; ya que sus polos pueden desplazarse fuera del circulo unitario por el proceso de adaptación y volverse inestables, y mayormente dado que no es el objetivo principal de este trabajo el estudio de dicha adaptación, nos enfocaremos únicamente en los filtros a respuesta impulsional finita

1.2.3. Filtros Lattice

La estructura de filtros lattice es una alternativa para realizar una Función de Transferencia de un filtro digital. A pesar de que la estructura de filtros lattice no tiene un número mínimo de multiplicadores y sumadores para la realización de una función de transferencia, tiene muchas propiedades ventajosas. Estas incluyen cascadeo de secciones idénticas, coeficientes con magnitudes menores a la unidad, prueba de estabilidad por inspección y buenas características de redondeo numérico. Además, la estructura lattice es particularmente útil para el filtrado adaptable dado que la solución recursiva de la estimación de los mínimos cuadrados produce naturalmente un filtro de estructura lattice. También, la estructura lattice ortogonaliza la señal de entrada en una base de etapa a etapa permitiendo capacidades de rápida convergencia y seguimiento. A pesar de que muchas técnicas alternativas han sido desarrolladas para estimar los coeficientes de reflexión que parametrizan la estructura lattice, el método de los mínimos cuadrados recursivos actualiza la estimación de los mínimos cuadrados a través de la observación de cada muestra. Este procedimiento nos lleva a una estimación óptima y requiere únicamente un costo computacional ligeramente mayor al de otras técnicas.

Las técnicas de estimación adaptables modifican los parámetros estimados del filtro de acuerdo a los datos observados actualizados. Para cada nueva muestra, la estimación recursiva utilizando el filtro lattice, como en el caso de la estructura transversal LS, genera nuevos coeficientes de reflexión y errores de predicción para cada orden del filtro. El cambiar cada coeficiente del filtro para cada nueva muestra es importante para aplicaciones en las cuales la rápida convergencia o el seguimiento de señales altamente variables son necesarios. Sin embargo, para aplicaciones en donde la dinámica es lenta, únicamente los resultados de observar la señal por cierto tiempo son importantes. [CG86]

Estructura General del Filtro Digital Lattice

La realización de filtros digitales lattice consiste en el cascadeo de etapas con dos entradas y dos salidas, como en las estructuras analógicas. Las configuraciones posibles para la realización de una función de transferencia digital incluyen un multiplicador asimétrico (figura 1.5) y dos multiplicadores simétricos (figura 1.6). Para el multiplicador lattice asimétrico, la estructura dentro de cada etapa equivale a una función de transferencia de un solo polo y un cero. El algoritmo para determinar la estructura asimétrica de la figura 1.5 se puede consultar en [MKH77] [CG86].

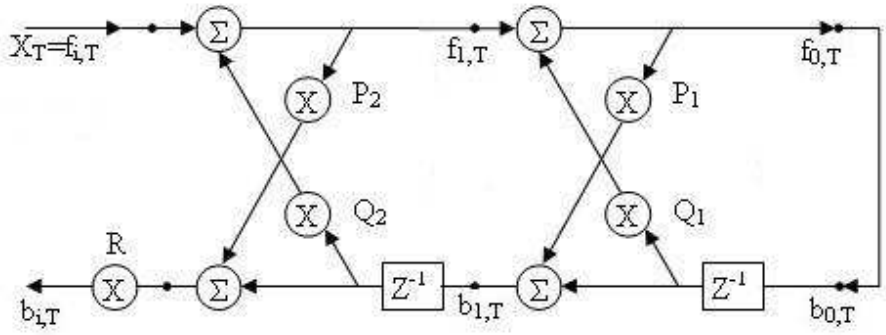


Fig. 1.5: Filtro Lattice Asimétrico.

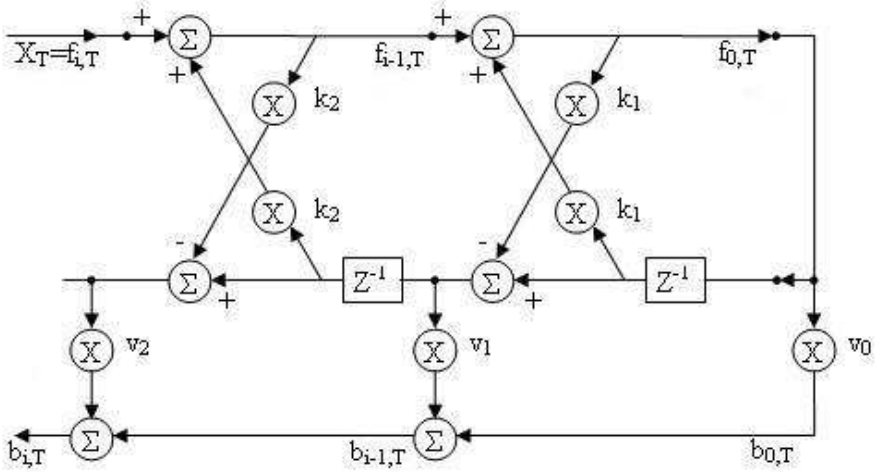


Fig. 1.6: Filtro Lattice Simétrico con dos multiplicadores.

En el caso de utilizar dos multiplicadores simétricos, se puede hacer una modificación para utilizar un solo multiplicador para así tener el menor número de multiplicadores posibles, sin embargo esto requiere de sumadores adicionales (ver 1.7).

Un cascadeo de secciones lattice, para formar un filtro lattice, puede implementar una función de transferencia de modo que tenga ventajas sobre las formas directa estándar, paralela o serie tradicional. La estructura de cascadeo en el filtro lattice propaga una señal forward $f_j(k)$ y una señal backward $b_j(k)$ en el instante k y en la sección j. La ecuación

fundamental que describe esta estructura de filtro lattice es (ver figura 1.8) [CG86].

$$f_{j+1}(k) = f_j(k) - k_{j+1}b_j(k-1)b_{j+1}(k) = b_j(k-1) - k_{j+1}f_j(k) \tag{1.4}$$

Los coeficientes k, son conocidos como coeficientes de reflexión o coeficientes de correlación parcial PARCOR por su origen en inglés (partial correlation).

La implementación de funciones de transferencia de filtros digitales en forma lattice ha sido examinada por Markel y Gray en [MG73], [MG75] y una representación canónica fue establecida por Morf en [Mor74], [MLNV77], y por Lee en [Lee80]. El algoritmo descrito por las formulas en 1.4 determina los coeficientes de reflexión k_i y los coeficientes v_i de la figura 1.6 que es equivalente a una forma directa de una función de transferencia con coeficientes b_i^P en el numerador y coeficientes a_i^P en el denominador.

Para una función de transferencia todo polo ($b_0^P = 1, b_j^P = 1, j > 1$), el filtro lattice se denomina filtro *feedback* y su estructura se muestra en la figura 1.8. El inverso del lattice *feedback*

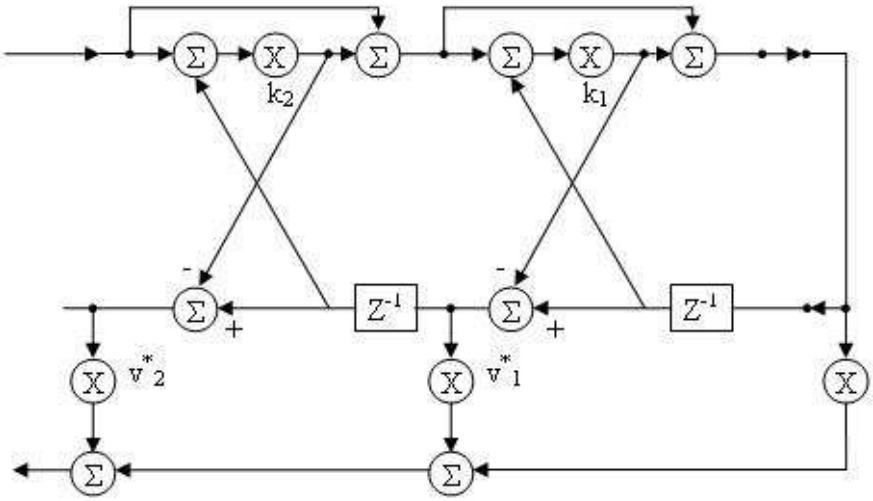


Fig. 1.7: Filtro Lattice Simétrico con un multiplicador.

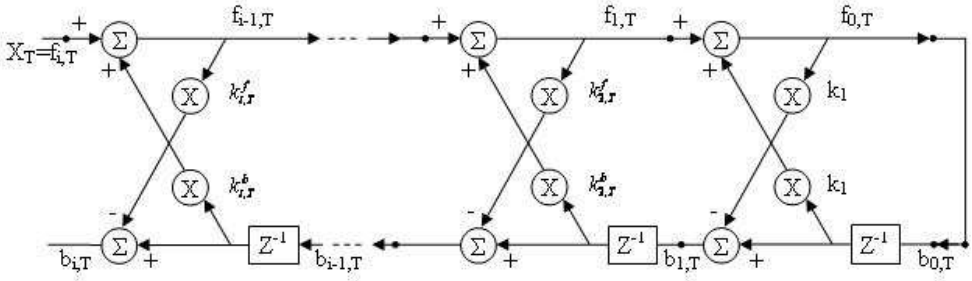


Fig. 1.8: Filtro Lattice Simétrico con un multiplicador.

1.3. Aplicaciones

Los filtros adaptables, por su naturaleza, son sistemas autodiseñables que se pueden ajustar a si mismos a diferentes ambientes. Como resultado, los filtros adaptables encuentran aplicación en diversos campos como control, comunicaciones, procesamiento de señales de radar o sonar, cancelación de interferencias, control activo de ruido, ingeniería biomédica, etc. La característica común de estas aplicaciones que las pone bajo el esquema de filtrado adaptable es que todas involucran un proceso de filtrado de algún tipo de señal para coincidir con una respuesta deseada. Los parámetros del filtro son actualizados haciendo un conjunto de medidas de las señales *subyacentes* y aplicar ese conjunto de medidas al algoritmo de filtrado adaptable tal que la diferencia entre la salida del filtro y la respuesta deseada sea minimizada ya sea en un sentido estadístico o determinístico. En este contexto, cuatro clases de aplicaciones son reconocidas: modelado, modelado inverso, predicción lineal y cancelación de interferencias.

1.3.1. Modelado

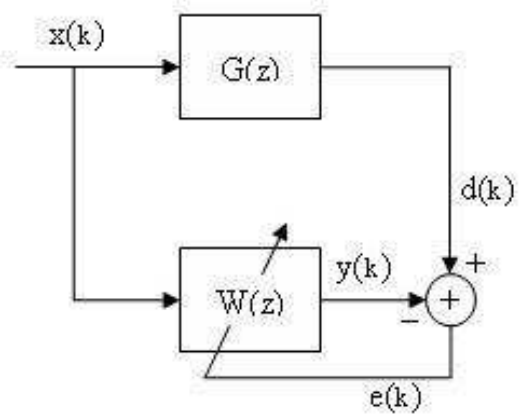


Fig. 1.9: Modelado de un Sistema Adaptable.

La figura 1.9 muestra el problema de modelado en el contexto de filtrado adaptable. El objetivo es estimar los parámetros del modelo $W(z)$, de la planta $G(z)$. En base a algún conocimiento a priori de la planta $G(z)$, una función de transferencia $W(z)$ con cierto número de parámetros ajustables es seleccionada. Los parámetros de $W(z)$ son escogidos a través de un algoritmo de filtrado adaptable tal que la diferencia entre la salida de la planta, $d(k)$, y la salida del filtro, $y(k)$, sea minimizada.

Una aplicación del modelado es la *identificación de sistemas*. En la mayoría de los sistemas de control modernos la planta bajo control es identificada en línea y el resultado es un regulador auto-ajustable (STR por sus siglas en inglés *self-tuning regulator*), como se muestra en la figura 1.10.

Otra es la *cancelación de eco*. En esta aplicación el filtro adaptable es utilizado para identificar la respuesta al impulso entre la fuente en la que se origina el eco y el punto en el cual el eco aparece. La salida de filtro adaptable, el cual es un estimado de la señal de eco, puede ser utilizado para cancelar el eco no deseado.

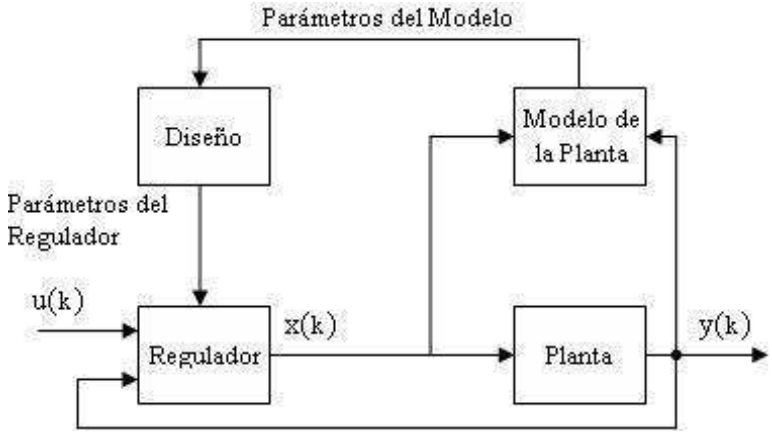


Fig. 1.10: Diagrama de bloques de un regulador Auto Ajustable.

Las características no ideales de los canales de comunicación resultan frecuentemente en alguna distorsión de la señal recibida. Para eliminar dicha distorsión se utilizan comúnmente los igualadores de canal. Esta técnica es equivalente a implementar la inversa de la respuesta del canal. El modelado directo del canal, sin embargo, tiene también una utilidad en algunas implementaciones de receptores de datos.

1.3.2. Modelado Inverso

El modelado inverso, también conocido como deconvolución, es otra aplicación de los filtros adaptables que se ha extendido en varias disciplinas de la ingeniería. La aplicación más ampliamente usada para el modelado inverso es en comunicaciones en donde un modelo inverso (también llamado igualador) es empleado para reducir la distorsión del canal. El concepto de modelado inverso también ha sido aplicado a sistemas de control adaptables en donde un controlador es diseñado y cascadeado con una planta de tal manera que la respuesta final de este cascadeo coincida con una respuesta deseada. El proceso de predicción puede ser visto como un esquema de modelado inverso.

1.3.3. Igualación de Canal

La figura 1.11 muestra el diagrama a bloques de un sistema de transmisión en banda base equipado con un igualador de canal. Aquí el canal representa la respuesta combinada del filtro transmisor, el canal verdadero y el filtro final. La secuencia de ruido aditivo, *v(k)*, se debe al ruido térmico de los circuitos electrónicos y a posibles interferencias de canales vecinos. Los símbolos de datos transmitidos, *s(k)* que aparecen como pulsos modulados en fase o amplitud son distorsionados por el canal.

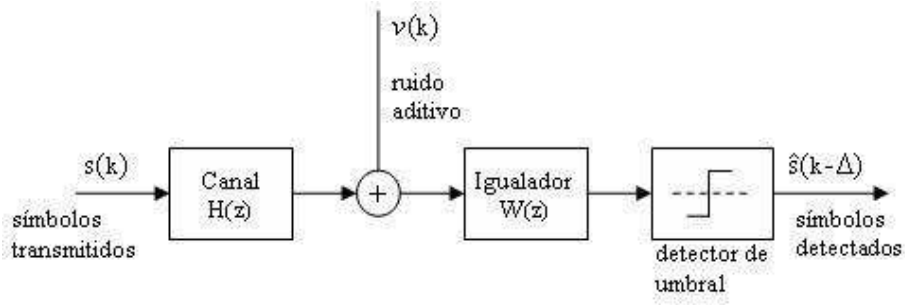


Fig. 1.11: Sistema de Transmisión en Banda Base con Igualador de Canal.

La distorsión con mayor importancia es el efecto *pulse-spreading*, el cual es causado cuando la respuesta al impulso del canal no es igual a la función pulso ideal, sino que la respuesta es diferente de cero para varios periodos de símbolo. Esta distorsión resulta de la interferencia de los símbolos con los símbolos adyacentes, haciendo la detección mediante un detector de umbral simple poco confiable. A este fenómeno se le conoce como *interferencia intersimbólica* (ISI). La presencia de muestras de ruido aditivo, *v(k)*, deteriora el desempeño de los receptores de datos. La función del igualador, como filtro, es corregir la distorsión introducida por el canal minimizando tanto como sea posible el efecto del ruido aditivo a la entrada del detector de umbral. Para un canal *H(z)*, un igualador con función de transferencia $1/H(z)$ haría su tarea perfectamente, resultando en una función de transferencia resultante del canal-igualador $H(z)W(z) = 1$, lo que implica que la secuencia de datos transmitidos $s(k)$, aparecerá en la entrada del detector sin distorsión alguna. Desafortunadamente esta es una situación ideal, la cual no puede ser utilizada en la mayoría de las aplicaciones prácticas.

Se puede notar que la función de transferencia inversa del canal, puede ser no causal si $H(z)$ tiene algún cero fuera del circulo unitario, haciéndolo así imposible de realizar en la práctica. Este problema se resuelve seleccionando el igualador tal que $H(z)W(z) \approx z^{\Delta}$ en donde Δ es un retraso entero adecuado. Esto quiere decir que una replica retrasada de los símbolos transmitidos aparecen en la salida del igualador.

También cabe hacer notar que al elegir $H(z) = 1/W(z)$ (o $W(z) \approx z^{\Delta}/H(z)$) nos puede llevar a realzar significativamente el ruido aditivo, $v(k)$, en aquellas frecuencias en las que la magnitud de $H(z)$ es pequeña. De ahí que al elegir un igualador, $W(z)$, debemos mantener un balance entre la interferencia intersimbólica residual y el realce del ruido a la salida del igualador.

La figura 1.12 muestra los detalles de un sistema de transmisión en banda base, equipado con un igualador adaptable. El igualador es implementado normalmente en forma de un filtro transversal. El entrenamiento inicial del igualador requiere de conocer los símbolos de datos transmitidos dado que ellos deben ser utilizados como las muestras de señal deseadas para adaptar los coeficientes del filtro. Esto debido al hecho de que la salida del igualador deben ser idealmente los mismos símbolos de datos transmitidos. Por lo tanto requeriremos un periodo de inicialización durante el cual el transmisor enviara una secuencia de símbolos de entrenamiento que son conocidos por el receptor. A este modo de operación se le llama *modo de entrenamiento*.

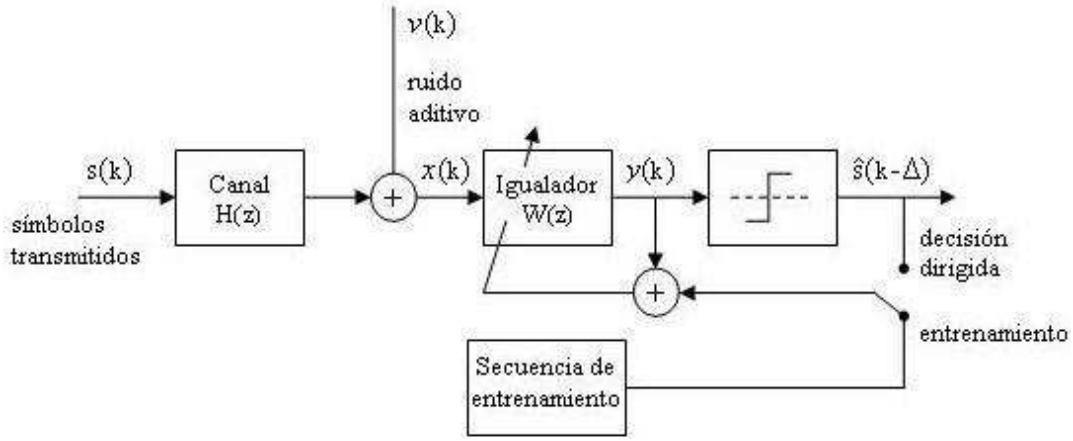


Fig. 1.12: Sistema de Transmisión en Banda Base con Igualador de Canal Adaptable.

Al finalizar el modo de entrenamiento, los coeficientes del igualador deberán haber converger a sus valores óptimos. Por consecuencia, los símbolos detectados deberán ser similares a los símbolos transmitidos con probabilidad cercana a uno. De ahí en adelante, los símbolos detectados pueden ser tratados como la señal deseada para adaptaciones futuras del igualador y por lo tanto posibles variaciones en el canal pueden ser rastreadas. A este modo de operación se le llama *modo de decisión dirigida*. Este modo trabaja satisfactoriamente mientras las variaciones del canal sean lo suficientemente lentas para que el algoritmo de adaptación sea capaz de seguir dichas variaciones satisfactoriamente.

1.4. Motivación

Actualmente la mayoría de los algoritmos de procesamiento digital de señales son concebidos en primera instancia para su implementación en lenguajes de programación de alto nivel (C, MATLAB, Java, etc.) con tipos de datos de punto flotante, sin embargo, las características de cálculo intensivo, grandes anchos de banda y superiores capacidades de almacenamiento que demandan las aplicaciones modernas, hacen imprescindible la implementación de los algoritmos en arquitecturas de punto fijo; esto, con la finalidad de cumplir con los requerimientos de disminución de costos, bajo consumo de energía y tiempos de cálculo reducidos que exige el mercado.

1.5. Problemática

Es un hecho que la programación de los algoritmos en aritmética de punto fijo resulta ser una tarea altamente laboriosa, consumidora de tiempo y propensa a errores, de tal manera que cada vez más se hace necesario el uso de metodologías y herramientas de desarrollo que no solo simplifiquen sino que también permitan la sistematización de las diferentes tareas como la evaluación de la dinámica de las variables, las estrategias de cuantización y la traducción automática de punto flotante a punto fijo.

1.6. Objetivo

En este trabajo, debido a que en la actualidad una de las herramientas tanto académica como comercialmente más empleadas para la implementación de algoritmos es MATLAB, se quiere validar la utilización de su Herramienta de Punto Fijo de (Matlab Fixed Point Toolbox) para llevar a cabo la implementación de algunos de los algoritmos rapidos de filtrado adaptable en aritmética de punto fijo con longitudes de palabra mínimas, dentro del contexto de una metodología de implementación de algoritmos en conjunto con una estrategia heurística, basada en la simulación, para la evaluación de la dinámica de las variables de los algoritmos y la validación de las implementaciones.

1.6.1. Hipótesis

Se puede realizar la implementación de una familia de algoritmos; en este caso los algoritmos rapidos de los mínimos cuadrados, en aritmética de punto fijo con longitudes de palabra mínima al emplear una metodología que involucre un estudio de las dinámicas de las variables del algoritmo y una herramienta de punto fijo apropiada.

1.7. Contribuciones

En este trabajo presentamos los resultados de la implementación en aritmética de punto fijo con longitudes de palabra mínima de los algoritmos rápidos de los mínimos cuadrados bajo el contexto de las aplicaciones de predicción, la identificación y la igualación de canal utilizando una metodología de implementación y la herramienta especializada *Matlab Fixed Point Toolbox*, así como el estudio de la dinámica de las variables de los diferentes algoritmos, hecho de manera heurística mediante simulaciones de punto flotante para la determinación de las longitudes de palabra mínimas con las cuales funcionaron correctamente. Nuestro estudio comparativo muestra que los algoritmos SFTF y FAEST pueden funcionar con longitudes de palabra de 16 bits bajo el contexto de la igualación de canal.

1.8. Organización de la Tesis

En el capítulo 2 se describen puntos a considerar de las metodologías y herramientas del diseño e implementación de algoritmos de procesamiento digital de señales, haciendo hincapié en la evaluación de la precisión de los sistemas en punto fijo, así como la evaluación de la dinámica de los datos mencionando algunos de los métodos existentes para la realización de estas dos tareas. De igual manera se menciona a grandes rasgos el desarrollo de la implementación de los algoritmos en aritmética de punto fijo. En el capítulo 3 se describen algunos fundamentos de filtrado adaptable, así como la familia de algoritmos rápidos de los mínimos cuadrados que se emplean en las implementaciones tanto en punto flotante como en punto fijo. En el capítulo 4 se presentan los conceptos necesarios a considerar al hacer uso de la herramienta de punto fijo para poder realizar las implementaciones de los algoritmos descritos en el capítulo 2 en aritmética de punto fijo. En el capítulo 5 se presentan los resultados obtenidos en las simulaciones de la implementación de los algoritmos tanto en aritmética de punto flotante como en aritmética

de punto fijo.

2. METODOLOGÍAS Y HERRAMIENTAS DE DISEÑO DE SISTEMAS PDS

Para llevar a cabo la implementación de algoritmos en aritmética de punto fijo cabe puntualizar que los procesadores de punto fijo representan y manipulan los números como enteros. Los procesadores de punto flotante primero representan los números en un formato de punto flotante, aun cuando también pueden soportar la representación y el cálculo entero. El formato de punto flotante implementa una representación numérica del valor como una combinación de mantisa (o parte fraccional) y un exponente.

Desarrollar y entender cuales aplicaciones son apropiadas para procesadores de punto fijo es altamente conveniente. El gran rango dinámico inherente disponible en diseños de punto flotante significa que las limitaciones del rango dinámico pueden ser prácticamente ignoradas en el diseño. Los procesadores de punto flotante pueden implementar tanto operaciones de punto flotante como operaciones enteras, haciéndolos más flexibles. Los procesadores de punto flotante tienden a ser más caros por que ellos implementan mayor funcionalidad (complejidad) en sílice y tienen buses más grandes (32 bits típicamente). La capacidad de punto flotante es apropiada para sistemas en los cuales los coeficientes de ganancia cambian con respecto al tiempo, o los coeficientes tienen rangos dinámicos grandes.

Los procesadores de punto flotante tienden a ser más amigables con lenguajes de alto nivel, por lo tanto puede ser más fácil desarrollar código para ellos. El proceso de desarrollo del código es también más independiente de la arquitectura. Por lo tanto, la facilidad relativa de desarrollo y las ventajas del lanzamiento han sido ponderadas contra el alto costo y la complejidad de hardware cuando se considera el diseño de implementaciones en punto flotante.

Comúnmente el menor costo y la alta velocidad de las implementaciones en DSP de punto fijo son ponderados contra el esfuerzo añadido en el diseño para el análisis de la implementación de algoritmos, y el escalamiento de los coeficientes de los datos para evitar sobreflujo del acumulador [CH06].

Entonces antes de comenzar el desarrollo de los algoritmos en punto fijo vale la pena puntualizar algunas de las ventajas y desventajas de la implementación de algoritmos en punto fijo, como pueden ser: [Zar06]

- Ventajas:
  - Mejor Implementación en Hardware
  - Circuitos más Simples
  - Tamaños de Chip más Pequeños
  - Menor Consumo de Energía
  - Disminución del costo unitario
- Desventajas
  - Mayor dificultad de programación
  - Tiempo de desarrollo más largo

Además de las ventajas y desventajas también cabe destacar que la implementación de algoritmos en punto fijo cubre una necesidad de los diseñadores de algoritmos y sistemas quienes buscan tomar en cuenta los efectos de la precisión finita cuando se lleva a cabo la implementación en hardware; ya sea en chips DSP de punto fijo o en hardware personalizado como FPGA's o ASIC.

Este tipo de implementaciones conlleva los siguientes retos: 1) Escalar apropiadamente las entradas, salidas y cantidades intermedias. 2) Limitar el grado de propagación de error en la secuencia de operaciones y 3) Asegurar el cumplimiento de especificaciones anteriores al prototipo de hardware.

2.1. Objetivos de la codificación de los datos

Al llevar a cabo la implementación en punto fijo debemos tener en consideración diversos aspectos para asegurar un correcto funcionamiento del algoritmo una vez implementado, a continuación se muestran algunos aspectos importantes a tomar en cuenta en la codificación de los datos en la aritmética de punto fijo [Men03]:

- Codificación en punto fijo
  - Definir para cada dato la posición del punto.
    Número de bits para las partes fraccionaria y entera.
  - Respetar las reglas de la aritmética de punto fijo.
- Objetivos y compromisos de la codificación en punto fijo.
  - Mantener la funcionalidad del algoritmo.
    Respetar las reglas de la aritmética en punto fijo.
    Garantizar la ausencia de desbordamientos.
  - Satisfacer el compromiso de precisión.

- Optimizar la implementación del algoritmo.

  Implantación física : minimizar la superficie y el consumo de energía

  Implantación lógica : minimizar los tiempos de ejecución y el tamaño del código.

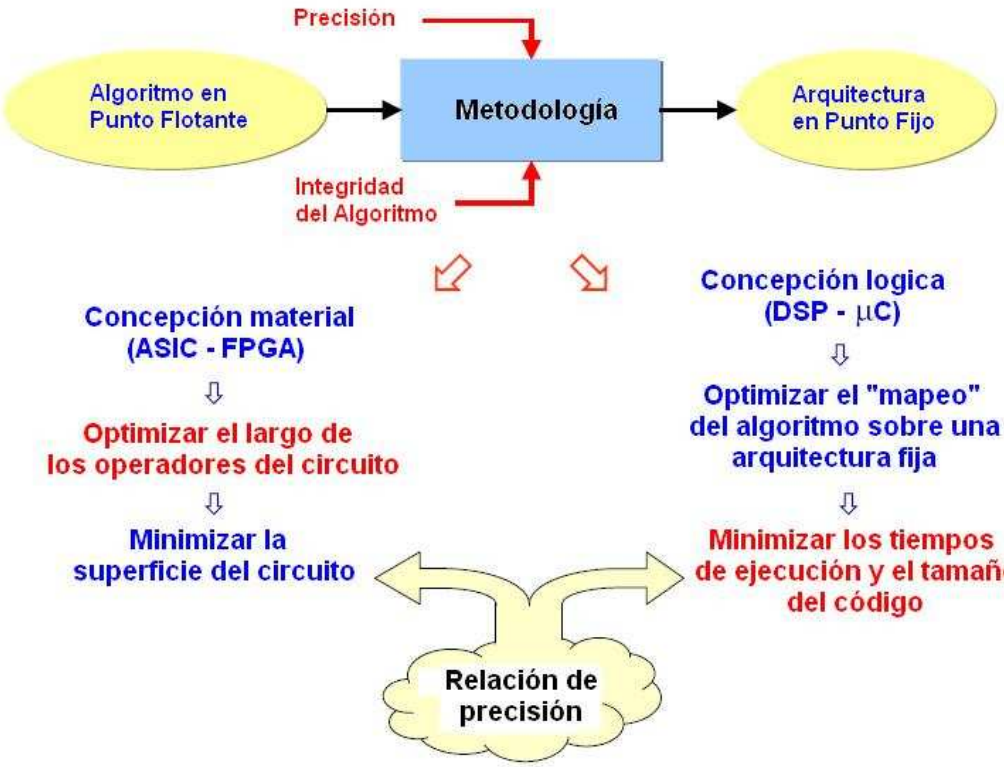


Fig. 2.1: Objetivos de la codificación de los datos. [Men03]

2.2. Evaluación de la precisión en sistemas de punto fijo

2.2.1. Métrica para la evaluación de la precisión

Para la medición de la precisión de los datos se pueden considerar los siguientes puntos [Men03]:

- Error de cuantificación asociado a un dato x:

  Diferencia entre el dato en precisión finita (punto fijo) y el dato en precisión infinita (valor exacto).

$$b_x = x_{precisionfinita} - x_{precisioninfinita}$$

- Propiedad del error de cuantificación

  Variable aleatoria (ergódica y estacionaria)

  Caracterizada por su potencia (momento de orden 2) P_{b_x}

- Métrica de evaluación de la precisión

  Relación Señal a Ruido de Cuantificación

$$RSBQ = \frac{P_y}{P_{b_y}}$$

  P_y : potencia de la señal.

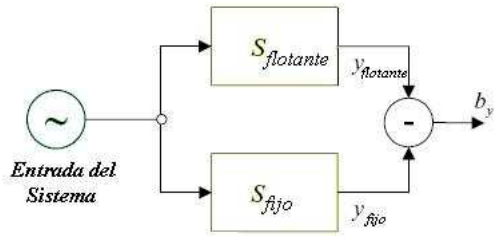
  P_{b_y} : potencia del ruido de cuantificación.

2.2.2. *Métodos basados en la simulación*

- **Principio**

Determinación de la potencia del ruido de cuantificación a partir de la simulación del sistema en punto fijo (y_{pfijo}) y en punto flotante ($y_{flotante}$).
El resultado en punto flotante es considerado como referencia.

La hipótesis se considera valida si el error asociado a la aritmética de punto flotante es despreciable en relación al resultado asociado a la aritmética de punto fijo.



$$P_{b_y} = \frac{1}{N_{pts}} \sum_{n=0}^{N_{pts}} \left(y_{flotante} - y_{fijo} \right)^2$$

Fig. 2.2: Objetivos de la codificación de los datos.

- Puesta en marcha: utilización de librerías para la emulación de la aritmética en punto fijo.

  Utilización de clase C++: systemC, gFix [KKW98]
  Tiempos de simulación elevados.

  Utilización de las características de la maquina huésped para acelerar la simulación en punto fijo.

  - Utilización de tipos optimizados: pFix [KKW98]
  - Generación de un código optimizado: FRIDGE [WBM97] [KCLM01]

    Reducción de tiempos de simulación.

    Aumento de tiempos necesarios para generar el código utilizado para la simulación.

- Adaptación del método CESTAC a punto fijo.

  Determinación del número de bits significativos al nivel de salida de la aplicación.

- Método CESTAC
  - Estimación del error de redondeado ligado a la aritmética de punto flotante a partir de algunas realizaciones de la aplicación [Tou99].
  - Hipótesis: El error en salida sigue una ley de probabilidad Gaussiana.

    Utilización del Test de Student para determinar el intervalo de confianza de la estimación de la media del error

    La deducción del número de bits significativos se hace a partir del intervalo de confianza.

2.2.3. *Métodos Analíticos*

- Método propuesto por Toureille [Tou99]
  - Determinación de la expresión analítica de RSBQ

    Determinación de la expresión de la potencia del ruido.

    Propagación de los momentos del ruido en el seno de GFD.
  - Definición de un modelo de propagación de los momentos del ruido para cada tipo de operador.

    Hipótesis simplificadoras.

    Procesamiento únicamente de las estructuras no recursivas.
- Desarrollo de una nueva metodología analítica.

  Estimación precisa de la potencia del ruido de cuantificación.

  Procesamiento de estructuras lineales recursivas.

2.3. *Evaluación de la dinámica de los datos*

La evaluación de la dinámica de los datos tiene como objetivos [Men03]

- Estimación del dominio de definición $[x_{min}, x_{max}]$ de cada dato x con vistas a deducir la posición del punto.

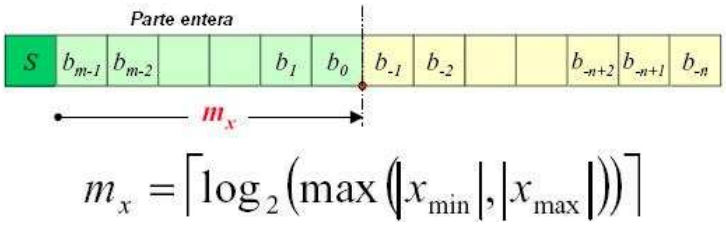


Fig. 2.3: Estimación del dominio de la dinámica.

- Criterios de calidad para la estimación.

  Precisión: minimizar el error de estimación.

  Evitar la presencia de bits no utilizados al nivel de los bits más significativos del dato

  Calidad: Garantizar la ausencia de desbordamientos o sobreflujos.

La evaluación de las dinámicas de un algoritmo se pueden llevar a cabo de diferentes maneras, ya sea empleando métodos basados en la simulación o emplear métodos analíticos.

2.3.1. Métodos basados en la simulación

Los métodos basados en la simulación pueden realizarse de una de las tres maneras siguientes:

- Determinación de la dinámica de un dato a partir de sus parámetros estadísticos.

  Simulación de los algoritmos y recolección de las muestras

  Determinación de los parámetros estadísticos y/o de los valores mínimos y máximos

- Determinación de la dinámica a partir de los valores mínimos X_{min} y máximos X_{max} obtenidos. [Aam01]

- Método propuesto por Kim [KKW98]

2.3.2. Métodos Analíticos

En lo que se refiere a los métodos analíticos para el estudio de las dinámicas de un algoritmo podemos encontrar:

- Propagación de la dinámica de las entradas en el seno de la aplicación.

  Utilización de los resultados de la aritmética de un intervalo [Kea96]

Operaciones	*min(z)*	*max(z)*
$z = x + y$	*min(x)+min(y)*	*max(x)+min(y)*
$z = x + y$	*min(x)-min(y)*	*max(x)-min(y)*
$z = x \times y$	*min(E)*	*max(E)*

E=(min(x)min(y),min(x)max(y),min(y)max(x),max(x)max(y))

  Procesamiento de estructuras no recursivas.

  Estimación pesimista.

  Sin considerar la correlación de los datos.

- Sistemas lineales: utilización de normas.

  Norma L1.

$$y_{\text{máx}\,1} = \max_{k}(|x(k)|)\sum_{m=-\infty}^{\infty}|h(m)|$$

  Sistemas lineales no recursivos: resultados idénticos a los obtenidos con la aritmética de intervalo.

  Norma Chebychev.

$$y_{\text{máx}\,2} = \max_{n}(|x(k)|)\max_{w}(|H(w)|)$$

  Señal de entrada del tipo

$$x(k) = \cos(w.k.T)$$

Después de contar con la implementación del algoritmo en punto flotante, se realiza el estudio de las dinámicas de las variables del algoritmo, a continuación se enumeran las etapas involucradas en el análisis de rango de las dinámicas para convertir un diseño a punto fijo.

1. Calcular o verificar el registro del rango de los mínimos/máximos
2. Calcular la parte entera necesaria que permita evitar un sobreflujo.
3. Calcular la parte fraccional.
4. Construir el objeto numérico de punto fijo.

2.4. Metodología de la Implementación en Punto Fijo

Para establecer una manera más descriptiva del procedimiento general para la implementación de algoritmos en aritmética de punto fijo, podemos enumerar las siguientes actividades a realizar durante el proceso.

1. **Establecer el flujo de la simulación (MATLAB®, Simulink®).** En esta etapa se debe analizar el flujo de los datos a través de toda la simulación del algoritmo y describirlo de manera gráfica.

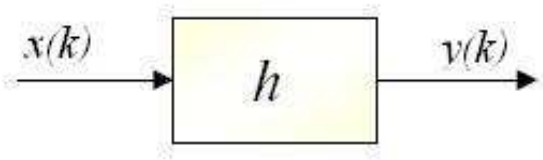


Fig. 2.4: Estimación del dominio de la dinámica.

2. **Desarrollar el algoritmo en punto flotante**. Enfocandose en la integridad algorítmica, prueba del concepto. Esto se refiere a la realización del programa en lenguaje de alto nivel en donde se implemente el algoritmo dando cumplimiento a los requerimientos de procesamiento descritos en el flujo de la simulación.

3. **Simulación** Simular iterativamente para observar el costo/beneficio de la implementación del algoritmo y validar de acuerdo a los requerimientos. Esto se lleva a cabo ejecutando la simulación con condiciones iniciales conocidas de manera que se puedan registrar en un banco de datos el comportamiento de las señales de entrada, salida, error, etc. que nos permitan evaluar el desempeño del algoritmo bajo algún criterio de desempeño.

4. **Conversión del diseño a punto fijo**. Enfocandose en la viabilidad del diseño basándose en las dinámicas (valores máximos y mínimos) de la implementación. En este paso se procede a llevar a cabo el estudio de dinámicas de las simulaciones en punto flotante de manera que se puedan garantizar la ausencia de sobreflujos en la implementación de punto fijo, así como disminuir los efectos de errores de precisión debidos a la adecuación del algoritmo en una longitud de palabra finita.

5. **Simulación**: Simular iterativamente para observar el costo/beneficio de la implementación del algoritmo y validar de acuerdo a los requerimientos originales. De igual manera que para las implementaciones en punto flotante, para las implementaciones en punto fijo se lleva a cabo una simulación iterativa de los algoritmos para verificar que las señales de entrada, salida, error, etc. se comporten de manera semejante a las simulaciones en punto flotante cumpliendo con los requerimientos de diseño originales.

Para llevar a cabo la conversión del diseño de punto flotante a punto fijo, nos podemos valer tanto de diversas metodologías como de variadas herramientas. En este trabajo nos enfocamos a utilizar la herramienta de punto fijo de Matlab, la cual ubica este proceso de conversión de aritmética de punto flotante a punto fijo como una etapa primaria para la implementación de algoritmos en punto fijo al emplear diversas herramientas de la familia de productos MathWorks como pueden ser Simulink, Real-time Workshop, DSP Blocksets y Code Composer, algunas de las cuales implican el empleo de productos específicos para la implementación como los blocksets de Texas Instruments.

2.5. Conclusiones

En esta sección del documento se describieron algunas ventajas y desventajas de la implementación de algoritmos en aritmética de punto fijo, de igual manera se presentaron algunos aspectos que se deben de considerar al seguir una metodología de implementación de algoritmos en punto fijo. Una de estas tareas es la codificación de los datos, con lo que se busca definir para cada dato la posición optima del punto decimal de manera que se respeten las reglas de la aritmética de punto fijo manteniendo la funcionalidad del algoritmo y garantizando la ausencia de desbordamientos en los acumuladores de la arquitectura de procesamiento en la que se implemente el algoritmo.

También se mencionan los puntos a considerar en la evaluación de la precisión de los sistemas de procesamiento en punto fijo, tales como el error de cuantificación y la relación señal a ruido de cuantificación. De igual manera se comentan algunos métodos para llevar a cabo la evaluación de la precisión y la evaluación de la dinámica de los datos, los cuales se pueden dividir en analíticos y basados en la simulación.

Al final de esta sección se describe a grandes rasgos todo el procedimiento para el desarrollo de la implementación de punto fijo, desde el establecimiento del flujo de la simulación del algoritmo, hasta la validación y verificación del diseño.

3. FUNDAMENTOS DE FILTRADO ADAPTABLE

Al abordar el estudio de los algoritmos de filtrado adaptable centraremos nuestra atención en el desarrollo de procedimientos o algoritmos para encontrar el vector de coeficientes óptimo $\hat{W}_p$. Es simple en principio; dado un registro de datos $x(k), y(k)$, uno calcula la matriz de covarianza R_p y el vector de correlación cruzada r_p y determina el vector $\hat{W}_p$. Aun cuando frecuentemente se elige calcular $\hat{W}_p$ de otra manera, algunas razones para realizar esto mediante este principio son [TJL87]:

- a) R_p puede no ser invertible.
- b) Aun cuando R_p es invertible en teoría, la precisión numérica requerida para invertirla apropiadamente esta más allá de las capacidades del hardware o de la computadora utilizada para la implementación del filtro.
- c) Pueden existir caminos más eficientes.

La meta es encontrar el vector de coeficientes $\hat{W}_p$ para el cual la función de desempeño J es minimizada. Un camino relativamente directo para encontrar $\hat{W}_p$ es buscar en la función J para encontrar su mínimo. Esto se puede realizar calculando J para todos los posibles valores de W_p y escogiendo el valor mínimo, también puede realizarse escogiendo aleatoriamente valores de W_p, calcular J y comparar si es menor que los valores ya calculados [WM76]. Sin embargo, una mejor aproximación es desarrollar un procedimiento de búsqueda ordenada que lo guié de forma metódica desde un valor inicial hasta el valor de W que minimize el valor del criterio J, el cual en este trabajo es el criterio de los mínimos cuadrados.

Este procedimiento se puede utilizar independientemente de la aplicación en la que se desee emplear el algoritmo, sea el modelado de señales o el modelado de sistemas, ya que para el modelado de señales se plantea el problema de predecir un valor estimado $\hat{x}(k)$ a partir de los valores pasados $x(k-1), x(k-2), \ldots, x(k-p)$, para lo cual se estiman los parámetros que permitan sintetizar el valor predicho de la señal, mientras que para el modelado de sistemas se plantea el problema de estimar los parámetros del modelo que correspondan según sea el caso a la respuesta en frecuencia de un sistema identificado o a la inversa de la respuesta en frecuencia de un sistema cancelador de interferencias.

Una vez resuelta la problemática de optimizar la estimación de los coeficientes bajo un esquema recursivo, los esfuerzos se enfocan en encontrar algún procedimiento más rápido para llevar acabo está tarea, resultando en el desarrollo de algoritmos recursivos más rápidos. En esta sección del documento se introducen algunos algoritmos rápidos de los mínimos cuadrados. Por lo tanto para adentrarse en el desarrollo de los algoritmos primero se muestra el desarrollo de la obtención de la solución del algoritmo de los mínimos cuadrados así como las recursiones de la solución, del vector de coeficientes del filtro, de la matriz de covarianza y del vector de correlación cruzada. Posteriormente se muestra el desarrollo de los algoritmos rapidos de los mínimos cuadrados.

3.1. Algoritmos de los Mínimos Cuadrados

El algoritmo RLS es bien conocido por sus propiedades de rápida convergencia debidas al alto costo computacional que implica, sin embargo, éste no puede ser utilizado para filtros adaptables de un orden alto en aplicaciones de tiempo real. En esos casos los algoritmos rápidos de los mínimos cuadrados recursivos FRLS (por su nombre en inglés, Fast Recursive Least Squares) han ido ganando importancia. Estos algoritmos requieren un gasto computacional menor y aun así proporcionan en cada iteración los mismos valores estimados de los parámetros y de la señal que el algoritmo RLS original descrito en la tabla 3.2.

3.1.1. Modelado de Señales

Para la problemática de predicción consideraremos que $x[k]$ es una señal discreta la cual puede ser caracterizada por la siguiente ecuación en diferencias:

$$x[k] = \sum_{i=1}^{p} a_i[k]x[k-i] + e[k]$$

Se plantea el problema de predecir a la señal $x[k]$ en términos de un conjunto de muestras pasadas de la misma, e.g. $x[k-1], x[k-2], \ldots, x[k-p]$, para lo cual podemos definir al predictor de $x[k]$ como una combinación lineal

$$\hat{x}[k] = -\sum_{i=1}^{p} a_i[k]x[k-1] \tag{3.1}$$

donde $a_i[k]$ son los parámetros que caracterizan a la señal en el instante k

La figura 3.1 ilustra los filtros de análisis y síntesis del problema de predicción.

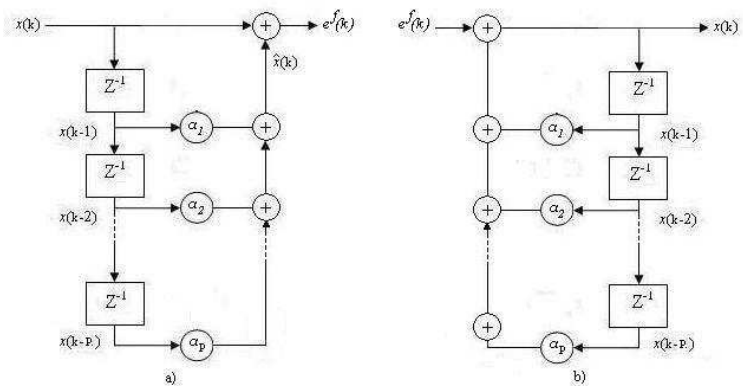


Fig. 3.1: a) Filtro de Análisis de Predicción. b) Filtro de Síntesis de Predicción.

Para evaluar la calidad del estimador, se introduce el siguiente error de predicción:

$$\varepsilon_p^f[k] = x[k] - \hat{x}[k] \tag{3.2}$$

o sustituyendo 3.1 en 3.2:

$$\varepsilon_p^f[k] = x[k] + \sum_{i=1}^{p} a_i[k]x[k] = x[k] + A_p^T[k]\mathbf{x}_p[k-1] \tag{3.3}$$

donde:

$$\mathbf{A}_p^T[k] \triangleq [a_1[k], a_2[k], \ldots, a_p[k]] \tag{3.4}$$

$$\mathbf{x_p}^T \triangleq [x[k-1], x[k-2], x[k-3], \ldots, x[k-p]] \tag{3.5}$$

Tomando el criterio de los mínimos cuadrados, definido como:

$$J_{ls} = \sum_{i=0}^{k} \varepsilon_p^f[i]\varepsilon_p^{f*}[i]$$

para minimizar el error de predicción establecido con respecto a los parámetros del modelo tenemos:

$$\begin{aligned}
&\tfrac{dJ_{LS}}{dA_p} = \tfrac{d}{dA_p}\sum_{i=0}^{k} \varepsilon_p^f[i]\varepsilon_p^{f*}[i] \\
&= \tfrac{d}{dA_p}\sum_{i=0}^{k} \Big[x[i] + A_p^T[k]\mathbf{x}_p[i-1]\Big]\Big[x[i] + A_p^T[k]\mathbf{x}_p[i-1]\Big]^* \\
&= \tfrac{d}{dA_p}\sum_{i=0}^{k} \Big[x[i]^2 + 2A_p^T[k]\mathbf{x}_p[i-1]x[i] + A_p^T[k]\mathbf{x}_p^T[i-1]\mathbf{x}_p[i-1]A_p[k]\Big] \\
&= \tfrac{d}{dA_p}\left\{\sum_{i=0}^{k} x[i] + 2A_p^T[k]\sum_{i=0}^{k}\mathbf{x}_p[i-1]x[i] + A_p^T[k]\left\{\sum_{i=0}^{k}\mathbf{x}_p[i-1]\mathbf{x}_p^T[i-1]\right\}A_p[k]\right\} \\
&= \tfrac{d}{dA_p}\Big\{r_0[k] + 2A_p^T[k]r_p^f[k] + A_p^T[k]R_p^f[k]A_p[k]\Big\} \\
&= 2r_p^f[k] + 2R_p^f[k]A_p[k]
\end{aligned}$$

lo cual da como resultado:

$$R_p^f[k]A_p[k] = -r_p^f[k] \tag{3.6}$$

con

$$R_p^f[k] = \sum_{i=0}^{k} \mathbf{x}_p[i-1]\mathbf{x}_p^T[i-1] \tag{3.7}$$

y

$$r_p^f[k] = \sum_{i=0}^{k} \mathbf{x}_p[i-1]x[i] \tag{3.8}$$

se puede verificar que $R_p^f[k]$ es igual a $R_p[k-1]$ donde

$$R_p^f[k-1] = \sum_{i=0}^{k-1} \mathbf{x}_p[i]\mathbf{x}_p^T[i] \tag{3.9}$$

3.1.2. Mínimos Cuadrados Recursivos

Debido a que en la mayoría de las aplicaciones existe una actualización constante en la respuesta de los sistemas, surge la necesidad de encontrar la solución del sistema de ecuaciones para cada instante de tiempo y como resultado se vuelve también necesario desarrollar el algoritmo de los mínimos cuadrados de manera recursiva.

3.1.3. Evaluación del Cálculo Recursivo del Predictor

Podemos verificar que tanto la matriz de covarianza, 3.7, como el vector, 3.8, debido a su estructura, pueden ser calculados de manera recursiva.

Evaluando $R_p^f[k]$ para $k = k+1$ tenemos que:

$$\begin{aligned} R_p^f[k] &= \sum_{i=0}^{k-1} \mathbf{x}_p[i-1]\mathbf{x}_p^T[i-1] + \mathbf{x}_p[k-1]\mathbf{x}_p^T[k-1] \\ R_p^f[k] &= R_p^f[k-1] + \mathbf{x}_p[k-1]\mathbf{x}_p^T[k-1] \end{aligned} \tag{3.10}$$

Similarmente se desarrolla una recursión para el vector de la recursion cruzada de las muestras de entrada y el valor predicho de la señal. Partiendo de la ecuación 3.8 se puede demostrar que [Alc86],[Hay91]:

$$r_p^f[k] = r_p^f[k-1] + \mathbf{x}_p[k-1]x[k] \tag{3.11}$$

Ahora podemos substituir la relación 3.11 en la ecuación 3.6, al instante k,

$$R_p^f[k]A_p[k] = -r_p^f[k-1] - \mathbf{x}_p[k-1]x[k] \tag{3.12}$$

substituyendo la ecuación, 3.6 y agregando $\pm\mathbf{x}_p[k-1]\mathbf{x}_p^T[k-1]A_p[k-1]$

$$R_p^f[k]A_p[k] = R_p^f[k-1]A_p[k-1] - \mathbf{x}_p[k-1]x[k] \pm \mathbf{x}_p[k-1]\mathbf{x}_p^T[k-1]A_p[k-1] \quad (3.13)$$

y agrupando de la siguiente manera,

$$R_p^f[k]A_p[k] = \{R_p^f[k-1]+\mathbf{x}_p[k-1]\mathbf{x}_p^T[k-1]\}A_p[k-1]-\{\mathbf{x}_p[k-1]x[k]+\mathbf{x}_p[k-1]\mathbf{x}_p^T[k-1]A_p[k-1]\} \quad (3.14)$$

reconociendo $R_p^f[k]$ y factorizando $\mathbf{x}_p[k-1]$ en la expresión 3.14

$$R_p^f[k]A_p[k] = R_p^f[k]A_p[k-1] - \mathbf{x}_p[k-1]\{x[k] + \mathbf{x}_p^T[k-1]A_p[k-1]\} \quad (3.15)$$

Pre-multiplicando 3.15 por la inversa de $R_p^f[k]$ y definiendo el error de predicción a priori como:

$$e_p^f[k] = x[k] + A_p^T[k-1]\mathbf{x}_p[k-1] \quad (3.16)$$

$$A_p[k] = A_p[k-1] - R_p^f[k]^{-1}\mathbf{x}_p[k-1]\{e_p^f[k]\} \quad (3.17)$$

y considerando a $R_p^f[k] = R_p[k-1]$, para tener la misma estructura que la matriz de covarianza que en el modelado de sistemas, de manera que el desarrollo posterior sea análogo:

$$R_p[k-1] = \sum_{i=0}^{k-1} \mathbf{x}_p[i]\mathbf{x}_p^T[i] \quad (3.18)$$

podemos entonces definir al vector,

$$k_p[k-1] = -R_p^{-1}[k-1]\mathbf{x}_p[k-1] \quad (3.19)$$

conocido en la literatura como la ganancia de Kalman, [FL78] [CMK83]

Con lo que finalmente

$$A_p[k] = A_p[k-1] + \mathbf{k}_p[k-1]e_p^f[k] \quad (3.20)$$

Siguiendo un procedimiento equivalente se puede demostrar que el cálculo recursivo de los parámetros de predicción se puede realizar con la siguiente relación involucrando el error a posteriori:

$$A_p[k] = A_p[k-1] + k_p^*[k-1]\varepsilon_p[k] \quad (3.21)$$

donde al vector

$$k_p^*[k] = -R_p^{-1}[k-1]\mathbf{x}_p[k] \quad (3.22)$$

se le conoce como la ganancia dual de Kalman, [CMK83].

3.1.4. Evaluación Recursiva de la Inversa de la Matriz de la Covarianza

De las definiciones 3.9 y 3.7, se puede verificar que la estructura de la matriz de covarianza $R_p^f k-1$ al instante k es equivalente. Por lo tanto, asumiendo que la matriz de covarianza $R_p^f(k)$ es definida, positiva y por lo tanto no singular, podríamos aplicar el lema de inversión de matriz a la ecuación recursiva 3.10. Haciendo primero las siguientes definiciones:

$$\begin{aligned} A &= R_p^f(k-1) \\ B &= \mathbf{x}_p(k), B^T = \mathbf{x}_p^T(k) \\ C &= B^T \\ D &= I \end{aligned} \tag{3.23}$$

Entonces:

$$R_p^{f-1}(k) = \left\{ R_p^f\,(k-1) + \mathbf{x_p}(k)\mathbf{x_p}^T(k) \right\}^{-1} \tag{3.24}$$

Y sustituyendo dichas definiciones en el lema de inversión de matriz obtenemos la siguiente ecuación recursiva para la inversa de la matriz de correlación:

$$R_p^{f-1}(k) = R_p^{f-1}\,(k-1) - R_p^{f-1}\,(k-1)\,\mathbf{x_p}(k) \left\{ I + \mathbf{x_p}^T(k) R_p^{f-1}(k-1)\mathbf{x_p}(k) \right\}^{-1} \mathbf{x_p}^T(k) R_p^{f-1}(k-1) \tag{3.25}$$

Por conveniencia de cálculo, pre-multiplicando 3.25 por $\mathbf{x}_p(k)$ y haciendo

$$P(k-1) = R_p^{f-1}(k-1) \tag{3.26}$$

tenemos

$$k_p(k) = \frac{P\,(k-1)\,\mathbf{x_p}(k)}{1 + \mathbf{x_p}^T(k) P(k-1)\mathbf{x_p}(k)} \tag{3.27}$$

en donde

$$1 + \mathbf{x_p}^T(k) P(k-1)\mathbf{x_p}(k) = \alpha = \gamma^{-1} \tag{3.28}$$

y sustituyendo 3.26 en 3.22 y a su vez sustituyendo en 3.27 obtenemos la siguientes relaciones entre la ganancia de Kalman y la ganancia dual de Kalman

$$k_p(k) = k_p^*(k)/\alpha \tag{3.29}$$

$$k_p(k) = k_p^*(k)\gamma \tag{3.30}$$

Entonces sustituyendo 3.27 en 3.25 tenemos

$$\begin{aligned} P(k) &= P(k-1) - k_p(k)\mathbf{x}_p^T(k)P(k-1) \\ R_p^{-1}(k) &= R_p^{-1}(k-1) - k_p(k)\mathbf{x}_p^T(k)P(k-1) \end{aligned} \tag{3.31}$$

De manera que el algoritmo RLS para la predicción se puede escribir como se muestra en la tabla 3.1.

Algoritmo RLS		
Inicialización $\hat{\mathbf{w}}_\mathbf{p}(0) = 0, \mathbf{x}_\mathbf{p}(0) = 0, \mathbf{R}^{-1}(0) = \delta^{-1} I.$ δ = constante pequeña positiva, I = matriz identidad		
Para $k = 1$ hasta M		
$e_p^f(k) = x_p(k) - \mathbf{A_p}^T(k-1)\mathbf{x_p}(k-1)$	P	(T2.1)
$\mathbf{A_p}(k) = \mathbf{A_p}(k-1) + \mathbf{k_p}(k-1)e_p^f(k)$	P	(T2.2)
$\mathbf{k_p^*}(k) = \mathbf{R}^{-1}(k)\mathbf{x_p}(k)$	P^2	(T1.1)
$\alpha(k) = 1 + \mathbf{k_p}^{*T}(k)\mathbf{x_p}(k)$ (ó $\gamma(k) = \left[1 + \mathbf{k_p}^{*T}(k)\mathbf{x_p}(k)\right]^{-1}$)	P	(T1.2)
$\mathbf{k_p} = \mathbf{k_p^*}/\alpha(k)$ (ó $\mathbf{k_p}(k) = \mathbf{k_p^*}(k)\gamma(k)$)	P	(T1.3)
$\mathbf{R}^{-1}(k) = \mathbf{R}^{-1}(k-1) - \mathbf{k_p}(k)\mathbf{k_p}^{*T}(k)$	P^2	(T1.4)

Tab. 3.1: Algoritmo de los Mínimos Cuadrados Recursivo para predicción.

3.1.5. Modelado de Sistemas

La estructura general de identificación de sistema utilizada es ilustrada en la figura 3.2, en donde $\mathbf{x}(k)$ denota la entrada de la señal de excitación con $\mathbf{x}(k) = 0$ para $k \leq 0$, $y(k)$ es la señal sin distorsiones del sistema desconocido y $e_f(k) = y(k) - x^T(k)\hat{w}_p(i)$ es el error de adaptación en la k-ésima iteración con un estimado de los parámetros de la i-ésima iteración.

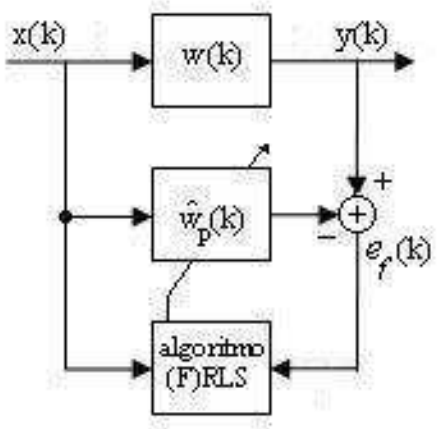


Fig. 3.2: Estructura de Identificación de Sistema.

La estructura general de la igualación de canal se muestra en la figura 3.3, en donde $\mathbf{y}(k)$ es la señal de los datos a transmitir a través de un canal con respuesta en frecuencia definida por los coeficientes $w(k)$, $x(k)$ es la señal recibida por el igualador y $\hat{y}(k)$ la señal estimada por el filtro igualador cuya respuesta en frecuencia esta definida por los coeficientes $\hat{w}_p(k)$ y que deben de representar el inverso de la respuesta en frecuencia del sistema o canal a igualar representado por los coeficientes $w(k)$.

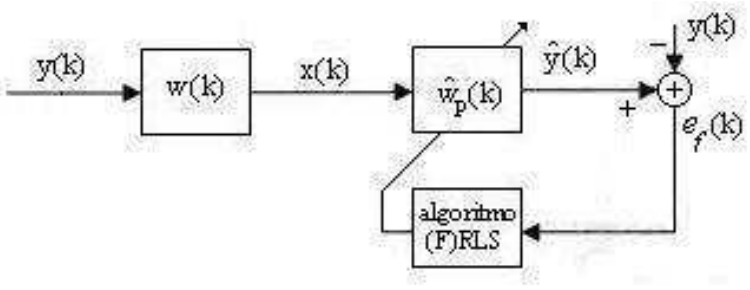


Fig. 3.3: Estructura de Igualación de Canal.

Ahora bien para la problemática de filtrado vamos a considerar a

$$y[k] = \sum_{i=0}^{P-1} W_i[k]x[k-i] \tag{3.32}$$

en donde podemos estimar la salida del filtro como

$$\hat{y}[k] = -\sum_{i=0}^{P-1} W_i[k]x[k] \tag{3.33}$$

para evaluar la estimación de los coeficientes de filtrado, establecemos nuevamente el error

$$e_f[k] = y[k] - \hat{y}[k] \tag{3.34}$$

de tal manera que el error puede quedar definido de la siguiente manera

$$e_f[k] = y[k] + \sum_{i=0}^{P-1} W_i[k]x[k] = y[k] + w_p^T[k]\mathbf{x}_p[k] \tag{3.35}$$

con los vectores:

$$w_p^T[k] = [w_0[k], w_1[k], \ldots, w_{p-1}[k]] \tag{3.36}$$

y

$$\mathbf{x}_p^T = [x[0], x[1], \ldots, x[p-1]] \tag{3.37}$$

Para minimizar el error tomamos el criterio de los mínimos cuadrados, el cual se define como:

$$J_{ls} = \sum_{i=0}^{k} e_f[i]e_f^*[i]$$

Se puede verificar que, repitiendo el procedimiento empleado en el modelado de sistemas para la minimización del error, obtenemos la solución del sistema de ecuaciones

$$R_p[k]W_p[k] = -r_p[k] \tag{3.38}$$

con

$$r_p[k] = \sum_{i=0}^{k} \mathbf{x}_p[i]y[i] \tag{3.39}$$

y

$$R_p[k] = \sum_{i=0}^{k} \mathbf{x}_p[i]\mathbf{x}_p^T[i] \tag{3.40}$$

3.1.6. Evaluación del Cálculo Recursivo del Filtro

Al igual que en el procedimiento anterior de la evaluación del cálculo recursivo del predictor, podemos verificar que tanto la matriz de covarianza, 3.40, como el vector, 3.39, debido a su estructura, pueden ser calculados de manera recursiva.

De manera tal que:

$$\begin{aligned} R_p[k] &= \sum_{i=0}^{k-1} \mathbf{x}_p[i]\mathbf{x}_p^T[i] + \mathbf{x}_p[k]\mathbf{x}_p^T[k] \\ R_p[k] &= R_p[k-1] + \mathbf{x}_p[k]\mathbf{x}_p^T[k] \end{aligned} \tag{3.41}$$

Y similarmente se desarrolla la recursión para el vector de recursion cruzada de las muestras de entrada y el valor deseado de la señal. Partiendo de la ecuación 3.39 se puede demostrar que [Alc86],[Hay91]:

$$r_p[k] = r_p[k-1] + \mathbf{x}_p[k]y[k] \tag{3.42}$$

Con las definiciones de la matriz de covarianza (3.41) y el vector (3.42) definiendo el error de estimación a priori como:

$$e_p^f[k] = x[k] + A_p^T[k-1]\mathbf{x}_p[k-1] \tag{3.43}$$

Obtenemos la recursión para el cálculo de los coeficientes del filtro como:

$$W_p[k] = W_p[k-1] + \mathbf{k}_p[k-1]e_p^f[k] \tag{3.44}$$

Nuevamente siguiendo un procedimiento equivalente se puede demostrar que el calculo recursivo de los coeficientes del filtro se puede realizar con la siguiente relación involucrando el error a posteriori, [Alc86]:

$$W_p[k] = W_p[k-1] + k_p^*[k-1]\varepsilon_p[k] \tag{3.45}$$

3.1.7. Recursión de la Matriz de Covarianza para Filtrado

Siguiendo el procedimiento como en el modelado de señales podemos hacer la analogía de la ecuación 3.8 con 3.39 y obtener la recursión para el vector de correlación:

$$r_p[k] = r_p[k-1] + \mathbf{x}_p[k]y[k] \tag{3.46}$$

Entonces de manera similar podemos obtener la recursión para el calculo de la inversa de la matriz de covarianza al aplicar el lema de inversión matricial a 3.40, tal que, obtenemos la ecuación siguiente:

$$R_p^{-1}(k) = P_p\,(k-1) - \frac{P_p\,(k-1)\,\mathbf{x_p}(k)\mathbf{x_p}^T(k)P_p(k-1)}{1 + \mathbf{x_p}^T(k)P_p(k-1)\mathbf{x_p}(k)} \tag{3.47}$$

Y podemos observar que la solución de las ecuaciones es la misma que para el caso de la predicción, sin embargo la diferencia radica en la definición de los elementos que componen tanto a la matriz de covarianza R_p como al vector de correlación cruzada r_p.

De acuerdo a las ecuaciones anteriores podemos estructurar el algoritmo de los mínimos cuadrados recursivo para el modelado de sistemas, el cual se muestra en la tabla 3.2.

Algoritmo RLS ($2N^2 + 4N$ MADPR)		
Inicialización $\hat{\mathbf{w}}_\mathbf{p}(0) = 0, \mathbf{x}_\mathbf{p}(0) = 0, \mathbf{R}^{-1}(0) = \delta^{-1} I.$ δ = constante pequeña positiva, I = matriz identidad		
Para $k = 1$ hasta M	MADPR	
$\mathbf{k}^*_\mathbf{p}(k) = \mathbf{R}^{-1}(k-1)\mathbf{x}_\mathbf{p}(k)$	P^2	(T1.1)
$\alpha(k) = 1 + \mathbf{k_p}^{*T}(k)\mathbf{x}_\mathbf{p}(k)$ (ó $\gamma(k) = \left[1 + \mathbf{k_p}^{*T}(k)\mathbf{x}_\mathbf{p}(k)\right]^{-1}$)	P	(T1.2)
$\mathbf{k}_\mathbf{p} = \mathbf{k}^*_\mathbf{p}/\alpha(k)$ (ó $\mathbf{k}_\mathbf{p}(k) = \mathbf{k}^*_\mathbf{p}(k)\gamma(k)$)	P	(T1.3)
$\mathbf{R}^{-1}(k) = \mathbf{R}^{-1}(k-1) - \mathbf{k}_\mathbf{p}(k)\mathbf{k_p}^{*T}(k)$	P^2	(T1.4)
Extensión para procesos conjuntos: formulación con el error a priori		
$e_p(k) = y(k) - \hat{\mathbf{w}}^T_\mathbf{p}(k-1)\mathbf{x}_\mathbf{p}(k)$	P	(T1.5a)
$\hat{\mathbf{w}}_\mathbf{p}(k) = \hat{\mathbf{w}}_\mathbf{p}(k-1) + \mathbf{k}_\mathbf{p}(k)e_p(k)$	P	(T1.6a)
formulación con el error a posteriori		
$\left\{e_p(k) = y(k) - \hat{\mathbf{w}}^T_\mathbf{p}(k-1)\mathbf{x}_\mathbf{p}(k)\right\}$	$\{P\}$	(T1.5b)
$\left\{\varepsilon^f_p(k) = \gamma(k)e_p(k)\right\}$	$\{1\}$	(T1.5c)
$\left\{\hat{\mathbf{w}}_\mathbf{p}(k) = \hat{\mathbf{w}}_\mathbf{p}(k-1) + \mathbf{k}^*_\mathbf{p}(k)\varepsilon^f_p(k)\right\}$	$\{P\}$	(T1.6b)

Tab. 3.2: Algoritmo de los Mínimos Cuadrados Recursivo (MAPDR = multiplicaciones y divisiones por recursión).

3.2. Algoritmos Rápidos

A continuación se describen los algoritmos rápidos de los mínimos cuadrados que se emplean en el estudio de este trabajo, los cuales fueron considerados como una muestra de los algoritmos más conocidos [SR92] y más estudiados por diferentes autores.

Para comprender las ventajas que aprovechan los algoritmos rapidos con respecto a el algoritmo recursivo de los mínimos cuadrados matricial, debemos comenzar presentando como se obtiene la actualización en orden de la inversa de la matriz de covarianza de la señal, para lo cual comenzamos con la sustitución de la ecuación

$$\begin{aligned} &\mathbf{x}_{p+1}(k) = \begin{bmatrix} x(k) \\ \mathbf{x_p}(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{x_p}(k) \\ x(k-P) \end{bmatrix}, \\ &\mathbf{x}_{p+1} \in R_{P+1}, \mathbf{x_p} \in R_P. \end{aligned} \tag{3.48}$$

en

$$R_p[k] = \sum_{i=0}^{k} \mathbf{x}_p[i]\mathbf{x}_p^T[i]$$

al orden $p+1$ nos lleva a la escritura de la matriz de covarianza en la forma particionada

$$R_{p+1} = \begin{bmatrix} r_{p0}^f(k) & r_p^T(k) \\ r_p^f(k) & R_p(k-1) \end{bmatrix} = \begin{bmatrix} R_p(k-1) & r_p^b(k) \\ r_p^{bT}(k) & r_{p0}^b(k) \end{bmatrix} \tag{3.49}$$

La inversa de la matriz de covarianza se calcula con el lema de inversión matricial de una matriz particionada, [Alc86]:

$$R_{p+1}^{-1}(k) = \begin{bmatrix} 0 & 0 \\ 0 & R_p^{-1}(k-1) \end{bmatrix} + \begin{bmatrix} I \\ A_p \end{bmatrix} \alpha_p^{-f}(k) \begin{bmatrix} I & W_p^T(k) \end{bmatrix} \tag{3.50}$$

$$R_{p+1}^{-1}(k) = \begin{bmatrix} R_p^{-1}(k) & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} A_p \\ I \end{bmatrix} \alpha_p^{-b}(k) \begin{bmatrix} B_p^T(k) & I \end{bmatrix} \tag{3.51}$$

El cálculo de la ganancia de kalman directo al orden $p+1$, K_{p+1}, se obtiene post-multiplicando las ecuaciones 3.50 y 3.51 para $-x_{p+1}$ y de acuerdo a la expresión 3.48 con $I = 1$ tenemos que:

$$\begin{aligned} k_{p+1}(k) &= \left\{ \begin{bmatrix} 0 & 0 \\ 0 & R_{p+1}^{-1}(k-1) \end{bmatrix} + \begin{bmatrix} 1 \\ A_p(k) \end{bmatrix} \alpha_p^{-f}(k) \begin{bmatrix} 1 & A_p^T(k) \end{bmatrix} \right\} \begin{bmatrix} x(k) \\ x_p(k-1) \end{bmatrix} \\ &= \left\{ \begin{bmatrix} 0 & 0 \\ 0 & R_{p+1}^{-1}(k-1) \end{bmatrix} + \begin{bmatrix} \alpha_p^{-f}(k) & \alpha_p^{-f}(k)A_p^T(k) \\ A_p(k) & A_p(k)A_p^T(k)\alpha_p^{-f}(k) \end{bmatrix} \right\} \begin{bmatrix} x(k) \\ x_p(k-1) \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 \\ 0 & R_{p+1}^{-1}(k-1) \end{bmatrix} + \begin{bmatrix} \alpha_p^{-f}(k)x(k) & \alpha_p^{-f}(k)A_p^T(k)x_p(k-1) \\ A_p(k)\alpha_p^{-f}(k)x(k) & A_p(k)A_p^T(k)\alpha_p^{-f}(k)x_p(k-1) \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ k_p(k-1) \end{bmatrix} + \begin{bmatrix} x(k) + A_p^T(k)x_p(k-1) \\ A_p(k)x(k) + A_p(k)A_p^T(k)x_p(k-1) \end{bmatrix} \alpha_p^{-f}(k) \end{aligned}$$

$$
\begin{aligned}
&= \begin{bmatrix} 0 \\ k_p(k-1) \end{bmatrix} + \begin{bmatrix} \varepsilon^f(k) \\ Ap(k)\varepsilon^f(k) \end{bmatrix} \alpha_p^{-f}(k) \\
&= \begin{bmatrix} 0 \\ k_p(k-1) \end{bmatrix} + \begin{bmatrix} 1 \\ Ap(k) \end{bmatrix} \alpha_p^{-f}(k)\varepsilon^f(k)
\end{aligned} \tag{3.52}
$$

De manera que siguiendo el procedimiento anterior para la ganancia empleando los errores a posteriori encontramos las fórmulas siguientes, [Alc86]:

$$
k_{p+1}(k) = \begin{bmatrix} 0 \\ k_p(k-1) \end{bmatrix} - \begin{bmatrix} I \\ A_p(k) \end{bmatrix} \alpha_p^{-f}(k)\varepsilon^f(k) \tag{3.53}
$$

$$
k_{p+1}(k) = \begin{bmatrix} k_p(k) \\ 0 \end{bmatrix} - \begin{bmatrix} B_p(k) \\ I \end{bmatrix} \alpha_p^{-b}(k)\varepsilon_p^b(k) \tag{3.54}
$$

Además, si uno post multiplica las expresiones anteriores por $X_{p+1}(k)$ uno obtiene la ganancia de kalman dual al orden $p+1$, $k^*_{p+1}(k)$:

$$
k^*_{p+1}(k) = \begin{bmatrix} 0 \\ k^*_p(k-1) \end{bmatrix} - \begin{bmatrix} I \\ A_p(k-1) \end{bmatrix} \alpha_p^{-f}(k-1)e_p^f(k) \tag{3.55}
$$

$$
k^*_{p+1}(k) = \begin{bmatrix} k^*_p(k) \\ 0 \end{bmatrix} - \begin{bmatrix} B_p(k-1) \\ I \end{bmatrix} \alpha_p^{-b}(k-1)e_p^b(k) \tag{3.56}
$$

La solución de este sistema de ecuaciones nos permite calcular la recursion de la ganancia de Kalman y la ganancia dual (ver organigrama de los algoritmos).

3.2.1. Algoritmo FK (Ljung)

El primer algoritmo FRLS fue desarrollado por Ljung a partir del algoritmo RLS siguiendo una idea básica de Morf [Mor74] y presentada en 1978 [LMF78]. Este algoritmo es conocido generalmente como el algoritmo Rápido de Kalman FK (Fast Kalman algorithm) 3.3 y requiere $10P+3$ multiplicaciones y divisiones por recursión (MADPR). El principio básico en la derivación fue utilizar la llamada propiedad de invarianza al desplazamiento **shift invariant property** 3.48 del vector de la señal.

Es posible reemplazar el cálculo de las matrices inversas de covarianza $R(k)$ para cada recursión en la tabla 3.2 por algunas operaciones vectoriales. Este cambio significa una reducción considerable en el costo computacional, especialmente para valores de N muy grandes. Las ecuaciones T2.9 y T2.10 resultan de derivar el algoritmo FK [CMK83]. Estas ecuaciones son superfluas para el cálculo del algoritmo FK, pero son necesarias para derivar los algoritmos FRLS siguientes. Para el cálculo del filtro FIR con el vector de coeficientes $\hat{\mathbf{w}}(k)$ (extensión para procesos conjuntos), el algoritmo FK utiliza una formulación a partir del error a priori en la tabla 3.2.

Algoritmo FK ($10P + 3$ MADPR)		
Inicialización		
$f(0) = \mathbf{B_p}(0) = \mathbf{k_p}(0) = \hat{\mathbf{w}}_\mathbf{p}(0) = \mathbf{x_p}(0) = 0.$ $\alpha_p^f(0) = \delta$, constante pequeña positiva		
Para $k = 1$ hasta M	MADPR	
$e_p^f(k) = \mathbf{x_p}(k) - \mathbf{A_p}^T(k-1)\mathbf{x_p}(k-1)$	P	(T2.1)
$\mathbf{A_p}(k) = \mathbf{A_p}(k-1) + \mathbf{k_p}(k-1)e_p^f(k)$	P	(T2.2)
$\varepsilon^f(k) = \mathbf{x_p}(k) - \mathbf{A_p}^T(k)\mathbf{x_p}(k-1)$	P	(T2.3)
$\alpha_p^f(k) = \alpha_p^f(k-1) + e_p^f(k)\varepsilon^f(k)$	1	(T2.4)
$\begin{bmatrix} \mathbf{m}(k) \\ \mu(k) \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{k_p}(k-1) \end{bmatrix} - \frac{\varepsilon_p^f(k)}{\alpha_p^f(k)} \begin{bmatrix} -1 \\ \mathbf{A_p}(k) \end{bmatrix}$	$P+1$	(T2.5)
$e_p^b(k) = x(k-P) - \mathbf{B_p}^T(k-1)\mathbf{x_p}(k)$	P	(T2.6)
$\mathbf{k_p}(k) = \frac{\mathbf{m}(k) + \mu(k)\mathbf{B_p}(k-1)}{1-\mu(k)e_p^b(k)}$	$2P+1$	(T2.7)
$\mathbf{B_p}(k) = \mathbf{B_p}(k-1) + \mathbf{k_p}(k)e_p^f(k)$	P	(T2.8)
$\left\{\varepsilon_p^b(k) = x(k-P) - \mathbf{B_p}^T(k)\mathbf{x_p}(k)\right\}$	$\{P\}$	(T2.9)
$\left\{\alpha_p^b(k) = \alpha_p^b(k-1) + e_p^b(k)\varepsilon_p^b(k) = \frac{\alpha_p^b(k\|k)}{\mu(k)}\right\}$	$\{1\}$	(T2.10)
Extensión para procesos conjuntos: $e(k\|k-1) = y(k) - \hat{\mathbf{w}}_\mathbf{p}^\mathbf{T}(k-1)\mathbf{x_p}(k)$	P	(T2.11)
$\hat{\mathbf{w}}_\mathbf{p}(k) = \hat{\mathbf{w}}_\mathbf{p}(k-1) + \mathbf{k_p}(k)e(k\|k-1)$	P	(T2.12)

Tab. 3.3: Algoritmo Rápido de Kalman (Fast Kalman).

3.2.2. Algoritmo FTF (Cioffi y Kailath)

Después del algoritmo FK varios algoritmos FRLS fueron desarrollados con el objetivo de reducir aun más el costo computacional. En 1984 el algoritmo del Filtro Transversal Rápido FTF (Fast Transversal Filter por sus siglas en inglés) fue presentado por Cioffi y Kailath [CK84]. Posteriormente en [Ale86] se describe el algoritmo FTF sistemáticamente con proyecciones en el espacio vectorial. Esto permitió obtener una interpretación geométrica muy descriptiva del algoritmo, la cual es la principal ventaja del algoritmo FTF sobre otros algoritmos FRLS. Es fácil demostrar que el algoritmo FK puede convertirse con unos cuantos pasos computacionales en el algoritmo FTF, lo cual comprueba la equivalencia entre los dos algoritmos.

La diferencia básica con el algoritmo FK es que el parámetro del ángulo γ y conocido por 3.28 es introducida en el algoritmo FTF.

$$\gamma(k-1) = 1 - \mathbf{x^T}(k-1)k_p(k-1) \tag{3.57}$$
$$\gamma(k) = 1 - \mathbf{x^T}(k)k_p(k) \tag{3.58}$$
$$\gamma_+(k-1) = 1 - \mathbf{x_+^T}(k-1)k_{p+1}(k-1) \tag{3.59}$$

Si sustituimos en 3.57 las ecuaciones (T2.1),(T2.2) y (T2.3) obtenemos:

$$\gamma(k-1) = \frac{\varepsilon_p^f(k)}{e_p^f(k)} \tag{3.60}$$

De la misma manera (T2.6),(T2.8) y (T2.9) se pueden sustituir en 3.58, lo que resulta

$$\gamma(k) = \frac{\varepsilon_p^b(k)}{e_p^b(k)} \tag{3.61}$$

Si utilizamos en 3.59 la relación $\mathbf{x_+^T}(k) = \left[\mathbf{x^T}(k); x(k-N)\right]$ de 3.48 y por $\mathbf{k_{p+}}(k)$ la parte izquierda de (T2.5), es decir, $\mathbf{k_{p+}}(k) = \left[\mathbf{m}^T(k); \mu(k)\right]$, nos da como resultado con (T2.6),(T2.7) y 3.58

$$\gamma_+(k) = \left[1 - \mu(k)e_p^b(k)\right]\gamma(k) \tag{3.62}$$

Si por el contrario utilizamos en 3.59 la relación $\mathbf{x_+^T}(k) = \left[\mathbf{x_p}(k); \mathbf{x^T}(k-1)\right]$ de 3.48 y por $k_{p+}(k)$ la parte derecha de (T2.5), obtenemos con (T2.1),(T2.2),(T2.4) y 3.57

$$\gamma_+(k) = \frac{\alpha_p^f(k-1)}{\alpha_p^f(k)}\gamma(k-1) \tag{3.63}$$

El algoritmo se muestra en la tabla 3.4 en donde podemos apreciar que las ultimas cuatro formulas son las que difieren con el algoritmo FK (Tabla 3.3).

Algoritmo FTF con (T3.7a) (9P + 8 MADPR) con (T3.7b) 8P + 10 MADPR		
Inicialización $\mathbf{A_p}(0) = \mathbf{B_p}(0) = \mathbf{k_p}(0) = \hat{\mathbf{w}}_\mathbf{p}(0) = x(0) = 0 : \gamma(0) = 1$ $\alpha_p^f(0) = \delta$, constante positiva pequeña.		
Para $k = 1$ hasta M:	MADPR	
$e_p^f(k) = \mathbf{x_p}(k) - \mathbf{A_p^T}(k-1)\mathbf{x_p}(k-1)$	P	(T3.1)
$\mathbf{A_p}(k) = \mathbf{A_p}(k-1) + \mathbf{k_p}(k-1)e_p^f(k)$	P	(T3.2)
$\varepsilon_p^f(k) = \gamma(k-1)e_p^f(k)$	1	(T3.3)
$\alpha_p^f(k) = \alpha_p^f(k-1) + e_p^f(k)\varepsilon_p^f(k)$	1	(T3.4)
$\gamma_+(k) = \frac{\alpha_p^f(k-1)}{\alpha_p^f(k)}\gamma(k-1)$	2	(T3.5)
$\begin{bmatrix} m(k) \\ \mu(k) \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{k_p}(k-1) \end{bmatrix} - \frac{\varepsilon_p^f(k)}{\alpha_p^f(k)} \begin{bmatrix} -1 \\ \mathbf{A_p}(k) \end{bmatrix}$	$P+1$	(T3.6)
$e_p^b(k) = x(k-P) - \mathbf{B_p^T}(k-1)\mathbf{x_p}(k)$	P	(T3.7a)
$\left\{e_p^b(k) = \frac{\varepsilon^b(k-1)}{\gamma_+(k)}\mu(k)\right\}$	$\{2\}$	(T3.7b)
$\gamma(k) = \frac{\gamma_+(k)}{1-\mu(k)e_p^b(k)}$	2	(T3.8)
$\left\{\varepsilon_p^b(k) = \gamma(k)e_p^b(k)\right\}$	$\{1\}$	(T3.9)
$\left\{e^b(k) = e^b(k-1) + e_p^b(k)\varepsilon_p^b(k) = \frac{\varepsilon_p^b(k)}{\mu(k)}\right\}$	$\{1\}$	(T3.10)
$\mathbf{k_p}(k) = [m(k) + \mu(k)\mathbf{B_p}(k-1)]\,\frac{\gamma(k)}{\gamma_+(k)}$	$2P+1$	(T3.11)
$\mathbf{B_p}(k) = \mathbf{B_p}(k-1) + \mathbf{k_p}(k)e_p^b(k)$	P	(T3.12)
Extensión para procesos conjuntos		
$e(k\|k-1) = y(k) - \hat{\mathbf{w}}_\mathbf{p}^\mathbf{T}(k-1)\mathbf{x_p}(k)$	P	(T3.13)
$\hat{\mathbf{w}}_\mathbf{p}(k) = \hat{\mathbf{w}}_\mathbf{p}(k-1) + \mathbf{k_p}(k)e(k\|k-1)$	P	(T3.14)

Tab. 3.4: Algoritmo FTF (Fast Transversal Filter).

3.2.3. Algoritmo FAEST (Carayannis)

Este algoritmo se denomina así por su nombre en inglés (Fast A posteriori Error Sequential Technique) y se muestra en la tabla 3.5, ideado por Carayannis [CMK83] este algoritmo solo requiere $7P + 10$ MADPR. La principal diferencia entre los algoritmos FK y FTF con el algoritmo FAEST, es que los primeros dos utilizan el error a priori $e(k|k-1)$, mientras que el último emplea el error a posteriori $e(k|k)$. Esto implica únicamente un cambio en la formulación de la regla de iteración original. De cualquier manera, se obtienen para cada iteración los mismos valores que con el algoritmo RLS (Tabla 3.2).

Un elemento esencial para la conversión es el factor introducido en (T1.2):

$$\alpha(k) = [\gamma(k)]^{-1}$$

ya que este factor vincula las cantidades con * y sus correspondientes cantidades sin * en los algoritmos FK y FTF para el algoritmo FAEST como se muestra a continuación:

$$\mathbf{m}^*(k) = \mathbf{m}(k)\alpha(k-1)\frac{\alpha_p^f(k)}{\alpha_p^f(k-1)} \tag{3.64}$$

$$\mu^*(k) = \mu(k)\alpha(k-1)\frac{\alpha_p^f(k)}{\alpha_p^f(k-1)} \tag{3.65}$$

$$\mathbf{k_p}^*(k) = \mathbf{k_p}(k)\alpha(k) \tag{3.66}$$

Si reemplazamos todas las cantidades $\mathbf{m}(k), \mu(k)$ y $\mathbf{k_p}(k)$ en el algoritmo FTF de la tabla 3.4 de acuerdo a 3.66 por $\mathbf{m}^*(k), \mu^*(k)$ y $\mathbf{k_p}^*(k)$ y además reemplazamos $\gamma(k)$ por $\alpha(k)$ de acuerdo a 3.28, obtenemos directamente el algoritmo FAEST mostrado en la tabla 3.5.

Algoritmo FAEST (7P + 10 MADPR)		
Inicialización $\mathbf{A_p}(0) = \mathbf{B_p}(0) = \mathbf{k_p}^*(0) = \hat{\mathbf{w}}(0) = \mathbf{x_p}(0) = 0; \alpha(0) = 1$ $\alpha_p^f(0) = \alpha_p^b(0) = \delta$, constante positiva pequeña.		
Para $k = 1$ hasta M	MADPR	
$e_p^f(k) = \mathbf{x_p}(k) - \mathbf{A_p}^T(k-1)\mathbf{x_p}(k-1)$	P	(T4.1)
$\varepsilon_p^f(k) = e_p^f(k)/\alpha(k-1)$	1	(T4.2)
$\mathbf{A_p}(k) = \mathbf{A_p}(k-1) + \mathbf{k_p}^*(k-1)\varepsilon_p^f(k)$	P	(T4.3)
$\alpha_p^f(k) = \alpha_p^f(k-1) + e_p^f(k)\varepsilon_p^f(k)$	1	(T4.4)
$\begin{bmatrix} \mathbf{m}^*(k) \\ \mu^*(k) \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{k_p}^*(k-1) \end{bmatrix} - \frac{e_p^f(k)}{\alpha_p^f(k-1)} \begin{bmatrix} -1 \\ \mathbf{A_p}(k-1) \end{bmatrix}$	$P+1$	(T4.5)
$e_p^b(k) = \mu(k)\alpha_p^b(k-1)$	1	(T4.6)
$\mathbf{k_p}^*(k) = \mathbf{m}^*(k) + \mu^*(k)\mathbf{B_p}(k-1)$	P	(T4.7)
$\alpha(k) = \alpha(k-1) + \frac{\left[e_p^f(k)\right]^2}{\alpha_p^f(k-1)} - \mu^*(k)e_p^b(k)$	3	(T4.8)
$\varepsilon_p^b(k) = e_p^b(k)/\alpha(k)$	1	(T4.9)
$\alpha_p^b(k) = \alpha_p^b(k-1) + e_p^b(k)\varepsilon_p^b(k)$	1	(T4.10)
$\mathbf{B_p}(k) = \mathbf{B_p}(k-1) + \mathbf{k_p}^*(k)\varepsilon_p^b(k)$	P	(T4.11)
Extensión para procesos conjuntos		
$e(k\|k-1) = y(k) - \hat{\mathbf{w}}^{\mathbf{T}}(k-1)\mathbf{x_p}(k)$	P	(T4.12)
$e(k\|k) = e(k\|k-1)/\alpha(k)$	1	(T4.13)
$\hat{\mathbf{w}}(k) = \hat{\mathbf{w}}(k-1) + \mathbf{k_p}^*(k)e(k\|k)$	P	(T4.14)

Tab. 3.5: Algoritmo FAEST.

3.3. Algoritmos Normalizados

3.3.1. Algoritmo GFTF (Cioffi y Kailath)

Una variante más de los algoritmos FRLS que requiere un menor número de pasos computacionales es el algoritmo del Filtro Transversal Rápido de Ganancia Normalizada GFTF (Gain-Normalized Fast Transversal Filter) mostrado en la tabla 3.6, el cual fue presentado en 1984 por Cioffi y Kailath [CK84]. El algoritmo GFTF requiere únicamente de $7N + 12$ MADPR. Se basa en el algoritmo FTF y la idea fundamental es normalizar los vectores de ganancia por los parámetros de ángulo correspondientes. Las ecuaciones 3.67 describen completamente la relación entre los algoritmos FTF y GFTF:

$$m(k)^* = \mathbf{m}(k)\,[\gamma_+(k)]^{-1} \tag{3.67}$$
$$\mu(k)^* = \mu(k)\,[\gamma_+(k)]^{-1} \tag{3.68}$$
$$\mathbf{k_p}^*(k) = \mathbf{k_p}(k)\,[\gamma(k)]^{-1} \tag{3.69}$$

Es aun más simple convertir el algoritmo FAEST en un algoritmo GFTF. Únicamente se tiene que reemplazar $\alpha(k)$ por $\gamma(k)$ de acuerdo a la siguiente relación:

$$\alpha(k) = [\gamma(k)]^{-1}$$

Algoritmo GFTF (7P + 12 MADPR)		
Inicialización $\mathbf{A_p}(0) = \mathbf{B_p}(0) = \mathbf{k_p}^*(0) = \hat{\mathbf{w}}(0) = \mathbf{x_p}(0) = 0 : \gamma(0) = 1$ $\alpha_p^f(0) = \alpha_p^b(0) = \delta$, constante positiva pequeña.		
Para $k = 1$ hasta M	MADPR	
$e_p^f(k) = \mathbf{x_p}(k) - \mathbf{A_p}^T(k-1)\mathbf{x_p}(k-1)$	P	(T5.1)
$\varepsilon_p^f(k) = \gamma(k-1)e_p^f(k)$	1	(T5.2)
$\mathbf{A_p}(k) = \mathbf{A_p}(k-1) + c(k-1)\varepsilon_p^f(k)$	P	(T5.3)
$\alpha_p^f(k) = \alpha_p^f(k-1) + e_p^f(k)\varepsilon_p^f(k)$	1	(T5.4)
$\gamma_+(k) = \gamma(k-1)\varepsilon^f(k-1)/\varepsilon^f(k)$	2	(T5.5)
$\begin{bmatrix} \mathbf{m}^*(k) \\ \mu^*(k) \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{k_p}^*(k-1) \end{bmatrix} - \frac{e_p^f(k)}{\alpha_p^f(k-1)} \begin{bmatrix} -1 \\ \mathbf{A_p}(k-1) \end{bmatrix}$	$P+1$	(T5.6)
$e_p^b(k) = \mu^*(k)\varepsilon^b(k-1)$	1	(T5.7)
$\gamma(k) = \left[1 - \gamma_+(k)\mu^*(k)e_p^b(k)\right]^{-1}\gamma_+(k)$	3	(T5.8)
$\mathbf{k_p}^*(k) = \mathbf{m}^*(k) + \mu^*(k)\mathbf{B_p}(k-1)$	P	(T5.9)
$\varepsilon_p^b(k) = \gamma(k)e_p^b(k)$	1	(T5.10)
$\alpha_p^b(k) = \alpha_p^b(k-1) + e_p^b(k)\varepsilon_p^b(k)$	1	(T5.11)
$\mathbf{B_p}(k)) = \mathbf{B_p}(k-1) + \mathbf{k_p}^*(k)\varepsilon_p^b(k)$	P	(T5.12)
Extensión para procesos conjuntos		
$e(k\|k-1) = y(k) - \hat{\mathbf{w}}^T(k-1)\mathbf{x_p}(k)$	P	(T5.13)
$e(k\|k) = \gamma(k)e(k\|k-1)$	1	(T5.14)
$\hat{\mathbf{w}}(k) = \hat{\mathbf{w}}(k-1) + \mathbf{k_p}^*(k)e(k\|k)$	P	(T5.15)

Tab. 3.6: Algoritmo GFTF.

3.3.2. Algoritmo NFTF (Fabre y Gueguen)

Debido a la acumulación de errores de redondeo, los algoritmos FRLS son propensos a tener inestabilidad numérica. Dado que esta inestabilidad se presenta en el caso de que aparezcan valores numéricos no favorables de algunas de las variables de un algoritmo FRLS se ha intentado crear rangos de valores más favorables.

El principio básico en este caso es utilizar las raíces de las energías de los errores de predicción $\bar{\alpha}_p^f(k) = \sqrt{\alpha_p^f(k)}, \bar{\alpha}_p^b(k) = \sqrt{\alpha_p^b(k)}$, así como de el parámetro del ángulo $\bar{\gamma}(k) = \sqrt{\gamma(k)} = \sqrt{[\alpha(k)]^{-1}}$. Si utilizamos estas expresiones para normalizar los errores de predicción, forward, backward y los parámetros de ganancia del filtro del algoritmo FAEST en la tabla 3.5 de la siguiente forma:

$$\begin{aligned}
\bar{e}_p^f(k) &= \bar{\gamma}(k-1)e_p^f(k)/\bar{\alpha}_p^f(k-1) && (3.70)\\
\bar{e}_p^b(k) &= \bar{\gamma}(k)e_p^b(k)/\bar{\alpha}_p^b(k-1) && (3.71)\\
\bar{\mathbf{A}}_{\mathbf{p}}(k) &= \mathbf{A}_{\mathbf{p}}(k)/\bar{\alpha}_p^f(k) && (3.72)\\
\bar{\mathbf{B}}_{\mathbf{p}}(k) &= \mathbf{B}_{\mathbf{p}}(k)/\bar{\alpha}_p^b(k) && (3.73)\\
\bar{\mathbf{k}}_{\mathbf{p}}(k) &= \mathbf{k}_{\mathbf{p}}(k)\bar{\gamma}(k) && (3.74)\\
\bar{\mathbf{m}}(k) &= \bar{\gamma}(k-1)\rho^f(k)\mathbf{m}(k) && (3.75)\\
\bar{\mu}(k) &= \bar{\gamma}(k-1)\rho^f(k)\mu(k) && (3.76)
\end{aligned}$$

con

$$\rho^f(k) = \frac{\bar{\alpha}_p^f(k-1)}{\bar{\alpha}_p^f(k)} \tag{3.77}$$

y

$$\rho^b(k) = \frac{\bar{\alpha}_p^b(k-1)}{\bar{\alpha}_p^b(k)} \tag{3.78}$$

llegamos exactamente al algoritmo del filtro transversal rápido normalizado NFTF de la tabla 3.7.

Una algoritmo FRLS normalizado fue propuesto en 1984 por Cioffi y Kailath [CK84], pero sin el parámetro $\bar{\gamma}(k)$ como se ha descrito en este documento. Posteriormente Fabre y Gueguen en [FG86] complementaron esta normalización introduciendo $\bar{\gamma}(k)$ y fue en 1986 que presentaron el algoritmo NFTF descrito en la tabla 3.7.

La normalización tiene como propósito reducir los rangos en la dinámica de las variables criticas $\alpha_p^f(k)$, $alpha_p^b(k)$ y $\gamma(k)$, y así mejorar el desempeño numérico de este algoritmo FRLS.

Algoritmo NFTF (11P + 18 MADPR)(+ 3 Raíces)		
Inicialización $\bar{\mathbf{A}}_\mathbf{p}(0) = \bar{\mathbf{B}}_\mathbf{p}(0) = \bar{\mathbf{k}}_\mathbf{p}(0) = \hat{\mathbf{w}}(0) = x(0) = 0 : \bar{\gamma}(0) = 1$ $\bar{\alpha}_p^f(0) = \bar{\alpha}_p^b(0) = \sqrt{\delta}$, constante positiva pequeña.		
Para $k = 1$ hasta M	MADPR	
$\bar{e}_p^f(k) = \bar{\gamma}(k-1)\left[\mathbf{x_p}(k)/\bar{\alpha}_p^f(k-1) - \bar{\mathbf{A}}_\mathbf{p}^T(k-1)\mathbf{x_p}(k-1)\right]$	$P+2$	(T6.1)
$\rho^f(k) = \left[1 + (\bar{e}_p^f(k))^2\right]^{-1/2}$	$2,1$	(T6.2)
$\zeta^f(k) = \rho^f(k)\bar{e}_p^f(k)$	1	(T6.3)
$\bar{\mathbf{A}}_\mathbf{p}(k) = \rho^f(k)\bar{\mathbf{A}}_\mathbf{p}(k-1) + \zeta^f(k)\bar{\mathbf{k}}_\mathbf{p}(k-1)$	$2P$	(T6.4)
$\bar{\alpha}_p^f(k) = \bar{\alpha}_p^f(k-1)/\rho^f(k)$	1	(T6.5)
$\begin{bmatrix} \bar{\mathbf{m}}(\mathbf{k}) \\ \bar{\mu}(k) \end{bmatrix} = \rho^f(k)\begin{bmatrix} 0 \\ \bar{\mathbf{k}}_\mathbf{p}(k-1) \end{bmatrix} - \zeta^f(k)\begin{bmatrix} -1/\bar{\alpha}_p^f(k)(k-1) \\ \bar{\mathbf{A}}_\mathbf{p}(k-1) \end{bmatrix}$	$2P+1$	(T6.6)
$e_p^b(k) = \bar{\mu}(k)\bar{\alpha}_p^f(k)$	1	(T6.7)
$\bar{\gamma}(k) = \left[\left(\bar{\alpha}_p^f(k)(k)/\bar{\alpha}_p^b(k)(k-1)\right)^2 - \left(e_p^b(k)\right)^2\right]^{-1/2}$	$4,1$	(T6.8)
$\bar{e}_p^b(k) = \bar{\gamma}(k)e_p^b(k)$	1	(T6.9)
$\rho^b(k) = \left[1 + \left(\bar{e}^b(k\|k-1)\right)^2\right]^{-1/2}$	$2,1$	(T6.10)
$\bar{\mathbf{k}}_\mathbf{p}(k) = \bar{\mathbf{m}}(\mathbf{k})/\rho^b(k) + \bar{e}_p^b(k)\bar{\mathbf{B}}_\mathbf{p}(k-1)$	$2P$	(T6.11)
$\bar{\alpha}_p^b(k) = \bar{\alpha}_p^b(k-1)/\rho^b(k)$	1	(T6.12)
$\bar{\mathbf{B}}_\mathbf{p}(k) = \rho^b(k)\left[\bar{\mathbf{B}}_\mathbf{p}(k-1) + \bar{\mathbf{k}}_\mathbf{p}(k)\bar{e}^b(k\|k-1)\right]$	$2P$	(T6.13)
Extensión para procesos conjuntos $e(k\|k-1) = y(k) - \hat{\mathbf{w}}^T(k-1)\mathbf{x_p}(k)$	P	(T5.14)
$[e(k\|k)/\bar{\gamma}(k)] = \bar{\gamma}(k)e(k\|k-1)$	1	(T6.15)
$\hat{\mathbf{w}}(k) = \hat{\mathbf{w}}(k-1) + \bar{\mathbf{k}}_\mathbf{p}(k)\,[e(k\|k)/\bar{\gamma}(k)]$	P	(T6.16)

Tab. 3.7: Algoritmo NFTF.

3.3.3. Algoritmo CFK (Lin)

Para contrarrestar la inestabilidad numérica de los algoritmos FRLS, tanto Lin [Lin84] como Cioffi y Kailath [CK84] propusieron reiniciar el algoritmo FRLS cuando se presente inestabilidad. Se comprobó que justo antes de comenzar la inestabilidad, la expresión $r(k) = 1 - \mu^*(k)e_p^b(k)$ se vuelve negativa aun cuando teóricamente siempre debe de estar en el intervalo $0 < r(k) \leq 1$. Por lo tanto sugirieron verificar esta variable y comenzar el algoritmo justo después de que $r(k)$ tome un valor ≤ 0.

Con un reinício del algoritmo al instante $k = k_i$ la ultima estimación del parámetro $\hat{w}(k_i - 1)$ puede utilizarse como un nuevo valor inicial. Los vectores pasados $A_p(k_i - 1)$ y $B_p(k_i - 1)$ pueden ser ignorados aun cuando estos se puedan volver inestables debido a errores de redondeo. El vector anterior $k_p^*(k_i - 1)$ también puede corromperse por errores de redondeo, por lo tanto debe ser reinicializado. Para afrontar este inconveniente Cioffi y Kailath propusieron en [CK84] el hacer $k_p^*(k_i) = 0$ en el reinício del algoritmo, lo cual implica el menor costo computacional. Desafortunadamente los últimos N valores medidos $x(k_i - N), \ldots, x(k_i - 1)$ no se toman en consideración. Esto significa que se asume implícitamente el caso pre-ventaneo el cual no se ve satisfecho en el caso de un reinício. Por otro lado Lin asume el caso sin ventaneo y derivó en [Lin84] el algoritmo FK análogamente a [LMF78] pero tomando en cuenta, $x(k_i - 1) \neq 0$. El resultado es un algoritmo FK extendido (Tabla 3.8) llamado algoritmo CFK (Covariance Fast Kalman) en el cual un vector $d(k)$ debe ser calculado adicionalmente. Utilizando los valores iniciales

$$k_p^*(k_i) = d(k_i) = \frac{\mathbf{x_p}(k_i)}{\delta(k_i) + \mathbf{x_p}^T(k_i)\mathbf{x_p}(k_i)} \tag{3.79}$$

obtenemos para $k \geq k_i$ antes del estimado de los parámetros RLS

$$\hat{\mathbf{w}}(k) = \hat{\mathbf{w}}(k-1) + k_p^*(k)e(k|k-1) \tag{3.80}$$

con el nuevo vector de ganancia $k_p^*(k) = R_i^{-1}(k)\mathbf{x_p}(k)$. Aquí

$$R_1(k) = \sum_{v=k_i}^{k} \mathbf{x_p}(v)\mathbf{x_p}^T(v) + \delta(k_i)\mathbf{I} \tag{3.81}$$

es la matriz de autocorrelación de la señal de excitación para k_i para el caso sin ventaneo y $h(k)$ minimiza la función de costo

$$\begin{aligned} J_i(k) = &\left[\sum_{v=k_i}^{k} y(v) - x^T(v)\hat{h}(k)\right]^2 + \delta(k_i) \\ &x\left[\hat{h}(k) - \hat{h}(k_i - 1)\right]^T\left[\hat{h}(k) - \hat{h}(k_i - 1)\right] \end{aligned} \tag{3.82}$$

Las constantes positivas $\delta(k_i)$ expresan en los reinicios la confianza en los valores estimados $\hat{\mathbf{w}}(k_i-1)$ utilizados de aquí en adelante. Aun cuando el costo computacional por cada recursión es mayor para el algoritmo CFK que para el algoritmo del tipo FK, el primero asegura una transición más suavizada en el reinício. Si el algoritmo CFK es empleado y el caso sin ventaneo realmente aplica, siempre existe $k_p^*(k) = k_{pi}^*(k)$ y $d(k) = 0$ porque $x(k_i-1) = 0$ y el algoritmo CFK (Tabla 3.8) es idéntico en cada iteración a el algoritmo FK (tabla 3.3) y requerirá $10N+3$ MADPR hasta el primer reinício.

Algoritmo CFK ($16P+3$ MADPR)		
Inicialización Para $k=1$ $\mathbf{A_p}(1) = \mathbf{B_p}(1) = 0; \mathbf{k_p}^*(1) = \mathbf{d}(1) = \frac{x(1)}{\delta + \mathbf{x_p}^T(1)\mathbf{x_p}(1)}$. $\hat{\mathbf{w}}(1) = \begin{cases} \mathbf{k_p}^*(1)e(1\|0) & primer\ inicio \\ \hat{\mathbf{w}}(0) & reinicio \end{cases}$ $\alpha_p^f(1) = \delta + x^2(1)$ $\delta = \begin{cases} \text{constante positiva pequeña al primer inicio} \\ \text{constante positiva pequeña conveniente al reinicio} \end{cases}$		
Para $k=2$ hasta M	MADPR	
$e_p^f(k) = \mathbf{x_p}(k) - \mathbf{A_p}^T(k-1)\mathbf{x_p}(k-1)$	P	(T7.1)
$\mathbf{A_p}(k) = \mathbf{A_p}(k-1) + \mathbf{k_p}^*(k-1)e_p^f(k)$	P	(T7.2)
$\varepsilon_p^f(k) = \mathbf{x_p}(k) - \mathbf{A_p}^T(k)x(k-1)$	P	(T7.3)
$\alpha_p^f(k) = \alpha_p^f(k-1) + e_p^f(k)\varepsilon_p^f(k)$	1	(T7.4)
$\begin{bmatrix} \mathbf{m}^*(k) \\ \mu^*(k) \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{k_p}^*(k-1) \end{bmatrix} - \frac{\varepsilon_p^f(k)}{\alpha_p^f(k)} \begin{bmatrix} -1 \\ \mathbf{A_p}(k) \end{bmatrix}$	$P+1$	(T7.5)
$e_p^b(k) = x(k-P) - \mathbf{B_p}^T(k-1)\mathbf{x_p}(k)$	P	(T7.6)
$r(k) = 1 - \mu^*(k)e_p^b(k)$	1	(T7.7)
Si $r(k) \leq 0$ hacer $k=1$ e ir a la inicialización		
$\mathbf{B_p}(k) = \frac{\mathbf{B_p}(k-1) + \mathbf{m}^*(k)e_p^b(k)}{r(k)}$	$2P$	(T7.8)
$\mathbf{k_{pi}}^*(k) = \mathbf{m}^*(k) + \mu^*(k)\mathbf{B_p}(k)$	P	(T7.9)
$\mathbf{d}(k) = \frac{\mathbf{d}(k-1) + \mathbf{k_{pi}}^*(k)\mathbf{x_p}^T(k)\mathbf{d}(k-1)}{1 - \mathbf{x_p}^T(1)\mathbf{k_{pi}}^*(k)\mathbf{x_p}^T(k)\mathbf{d}(k-1)}$	$4P$	(T7.10)
$\mathbf{k_p}^*(k) = \mathbf{k_{pi}}^*(k)x^T(1)\mathbf{k_{pi}}^*(k)$	$2P$	(T7.11)
Extensión para procesos conjuntos:		
$e(k\|k-1) = y(k) - \hat{\mathbf{w}}^T(k-1)\mathbf{x_p}(k)$	P	(T7.12)
$\hat{\mathbf{w}}(k) = \hat{\mathbf{w}}(k-1) + \mathbf{k_p}^*(k)e(k\|k-1)$	P	(T7.13)

Tab. 3.8: Algoritmo CFK.

3.4. Algoritmos Estabilizados

3.4.1. Algoritmo CFTF (Botto, Moustakides)

Dado que es imposible prevenir errores de redondeo en los algoritmos FRLS, se pensó en introducir una nueva cantidad $\xi(k)$, la cual es proporcional a los errores de redondeo presentes en el algoritmo FRLS, e interviene en el proceso de cálculo en dependencia con esta "medida de errores de redondeo". Al respecto Moustakides [Mou89] presentó en 1989 un algoritmo de filtro transversal rápido corregido (CFTF) (ver tabla 3.9) el cual desarrolló junto con Botto [BM86] en base a sus estudios recientes. El algoritmo CFTF resulta del algoritmo FAEST o del algoritmo GFTF con las siguientes extensiones fundamentales.

$$e^b_{(1)}(k|k-1) = \mu(k)\varepsilon(k-1)$$

y un segundo tiempo como en el caso del algoritmo FTF

$$e^b_{(2)}(k|k-1) = x(k-P) - b^T(k-1)\mathbf{x_p}(k)$$

entonces la diferencia de las dos ecuaciones

$$\xi(k) = e^b_{(2)}(k|k-1) - \mu(k)\varepsilon(k-1)$$

es formada. Sin los errores de redondeo el cálculo de ambos errores backward tendrían que coincidir completamente y $\xi(k) = 0$. Pero si $\xi(k) \neq 0$,$\xi(k)$ es una medida de la magnitud del error de redondeo.

La siguiente idea es cambiar $A_p(k)$ y $B_p(k)$ por medio de una corrección $A_p(k)+\triangle A_p(k)$ y $B_p(k)+\triangle B_p(k)$ de tal manera que los errores de redondeo en el algoritmo sean minimizados. Los valores de corrección convenientes $\triangle A_p$ y $\triangle B_p$ se pueden encontrar derivando un criterio de calidad basado en los mínimos cuadrados de $\xi(k)$ con respecto a $\triangle A_p(k)$ y $\triangle B_p(k)$. Este criterio de calidad incluye un factor de ponderación adecuado ρ permitiendo que la influencia de la corrección pueda ser determinado.

Para $\rho = 0$ no existe corrección y el algoritmo CFTF es idéntico a el algoritmo GFTF. Los vectores de corrección $\triangle A_p(k)$ y $\triangle B_p(k)$ no son utilizados únicamente para corregir $A_p(k)$ y $B_p(k)$ sino también para corregir todas las cantidades en las que $A_p(k)$ y $B_p(k)$ están incluidos. Como resultado, $\xi(k)$ cambia a $\xi_c(k)$, $e^f(k)$ a $e^f_c(k)$ y $e^b(k)$ a $e^b_c(k)$. El índice c en cada caso indica las cantidades corregidas. Para la derivación realizada hasta el momento se puede apreciar que la expresión $([\gamma(k)]^{-1} - 1)$ aparece en las cantidades corregidas. Para evitar la inestabilidad numérica para el caso $\gamma(k) \to 0$, se introduce la aproximación $[\gamma(k)]^{-1} - 1 \approx 1 - \gamma(k)$ aunque es valida únicamente para $\gamma(k) \approx 1$, pero en realidad esta puede ser $0 < \gamma(k) \leq 1$.

Mientras tengamos $\xi(k) = 0$, se mantiene $\triangle A_p(k) = 0$,$\triangle B_p(k) = 0$, $e^f_c(k) = e^f(k)$ y $e^b_c(k) = e^b(k)$. Para este caso el algoritmo CFTF es idéntico a el algoritmo GFTF y provee exactamente los mismos valores estimados que el algoritmo RLS original. Sin embargo, tan pronto como $\xi(k) \neq 0$, las intervenciones correctivas adicionales causaran desviaciones

del estimado RLS óptimo. Para aplicaciones prácticas esta diferencia debe ser únicamente de menor significancia.

Algoritmo CFTF ($8P + 35$ MADPR)

Inicialización

$\mathbf{A_p}(0) = \mathbf{B_p}(0) = \mathbf{k_p}(0) = \hat{\mathbf{w}}(0) = \mathbf{x_p}(0) = 0$
$\gamma(0) = 1$
$\alpha_p^f(0) = \alpha_p^b(0) = \delta$, constante positiva pequeña
ρ, constante positiva adecuada

Para $k = 1$ hasta M	MADPR	
$e_p^f(k) = \mathbf{x_p}(k) - \mathbf{A_p}^T(k-1)\mathbf{x_p}(k-1)$	P	(T8.1)
$e_p^b(k) = x(k-P) - \mathbf{B_p}^T(k-1)\mathbf{x_p}(k)$	P	(T8.2)
$\gamma_+(k) = \frac{\gamma(k-1)\alpha_p^f(k-1)}{\alpha_p^f(k-1)+\gamma(k-1)\left[e_p^f(k)\right]^2}$	4	(T8.3)
$\vartheta(k) = 1 - \gamma_+(k)e_p^b(k) \times \left[c^N(k-1) - \frac{e_p^f(k)\mathbf{A_p}^N(k-1)}{\alpha_p^f(k-1)}\right]$	4	(T8.4)
$\gamma(k) = \gamma_+(k)/\vartheta(k)$	1	(T8.5)
$\zeta(k-1) = \gamma(k-1)\mathbf{A_p}^N(k-1)$	1	(T8.6)
$\xi(k) = e_p^b(k) - \alpha_p^b(k-1)\mathbf{k_p}^N(k-1) + \zeta(k-1)e_p^f$	2	(T8.7)
$\xi_c(k) = \xi(k)\left[1 + \rho(1-\gamma(k)) + \rho\zeta^2(k-1) \times (1-\gamma(k-1))\right]^{-1}$	5	(T8.8)
$e_{pc}^f(k) = e_p^f(k) - \rho\xi_c(k)\zeta(k-1) \times [1-\gamma(k-1)]$	3	(T8.9)
$\alpha_p^f(k) = \alpha_p^f(k-1) + \gamma(k-1)\left[e_{pc}^f(k)\right]^2$	3	(T8.10)
$e_{pc}^b = e_p^b(k) - \rho\xi_c(k)\left[1-\gamma(k)\right]$	2	(T8.11)
$\alpha_p^b(k) = \alpha_p^b(k-1) + \gamma(k)\left[e_{pc}^b(k)\right]^2$	3	(T8.12)
$\begin{bmatrix} \mathbf{m}(k) \\ \mu(k) \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{k_p}(k-1) \end{bmatrix} - \frac{e_{pc}^f(k)}{\alpha_p^f(k-1)} \begin{bmatrix} -1 \\ \mathbf{A_p}(k-1) \end{bmatrix}$	$P+1$	(T8.13)
$\mathbf{A_p}(k) = \mathbf{A_p}(k-1) + \gamma(k-1) \times \left[e_p^f(k) + \rho\xi_\partial(k)\zeta(k-1)\right]\mathbf{k_p}(k-1)$	$P+3$	(T8.14)
$\mathbf{k_p}(k) = \mathbf{m}(k) + \mu(k)\mathbf{B_p}(k-1)$	P	(T8.15)
$\mathbf{B_p}(k) = \mathbf{B_p}(k-1) + \gamma(k) \times \left[e_p^b(k) + \rho\xi_c(k)\right]k_p(k)$	$P+2$	(T8.16)
Extensión para procesos conjuntos:		
$e(k\|k-1) = y(k) - \hat{\mathbf{w}}^T(k-1)\mathbf{x_p}(k)$	P	(T8.17)
$e(k\|k) = \gamma(k)e(k\|k-1)$	1	(T8.18)
$\hat{\mathbf{w}}(k) = \hat{\mathbf{w}}(k-1) + \mathbf{k_p}(k)e(k\|k-1)$	P	(T8.19)

Tab. 3.9: Algoritmo CFTF.

3.4.2. Algoritmo SFTF (Benallal y Gilloire)

En 1985, Ljung y Ljung [LL85] propusieron analizar la propagación de errores por redondeo en el estado espacial. En este contexto, un algoritmo FRLS puede ser descrito como un sistema dinámico no lineal en el tiempo discreto y puede ser examinado por su estabilidad. Desafortunadamente un análisis general de estabilidad es extremadamente difícil de realizar dada la alta complejidad de un sistema no lineal. El análisis de estabilidad es menos problemático para un sistema linealizado apropiadamente.

Para la derivación del algoritmo del filtro transversal rápido estabilizado (SFTF) Benallal y Gilloire [BG88] linealizaron en 1988 el algoritmo GFTF y encontraron que este sistema dinámico linealizado es inestable. La concepción básica del algoritmo SFTF es intervenir correctivamente como un lazo de control en el algoritmo GFTF original de tal manera que el sistema dinámico linealizado sea estable. Para este propósito, encontramos en primera instancia una cantidad de medida $\xi(k)$ la cual es proporcional al error de redondeo. Para obtener $\xi(k)$, se calcula el error backward dos veces, una como se hace en el algoritmo FTF (ver tabla 3.4) y una segunda vez como en el caso del algoritmo GFTF (ver tabla 3.6) y entonces formar la diferencia

$$\xi(k) = e_p^b(k) - \mu\alpha_p^b(k-1)$$

Esta es exactamente la misma cantidad de medida del error de redondeo que la utilizada en el algoritmo CFTF, pero esta se utiliza de diferente manera en el algoritmo SFTF. El producto de la cantidad de error $\xi(k)$ y una variable de control η proveen un valor adicional de corrección para el error backward $e_p^b(k)$. En el algoritmo SFTF, se introducen tres variables de control $\eta_\gamma, \eta_\varepsilon$ y η_β las cuales tienen un efecto aditivo en $\gamma(k), \alpha_p^b(k)$ y $B_p(k)$ via el error backward si $\xi(k) \neq 0$. El cálculo de los valores adecuados para $\eta_\gamma, \eta_\varepsilon$ y η_β es muy difícil. En [BG88] algunos valores adecuados fueron determinados empíricamente y dos juegos de variables de control $\eta_\gamma = \eta_\varepsilon = 0, \eta_\beta = 1$ y $\eta_\gamma = \eta_\varepsilon = \eta_\beta = 1$ fueron finalmente establecidos.

Para el caso en el que $\eta_\gamma = \eta_\varepsilon = 0, \eta_\beta = -1$ el algoritmo SFTF es idéntico al algoritmo GFTF.

Una extensión de este concepto de estabilización para un factor de olvido $\lambda \leq 1$ fue publicado por Slock y Kailath [SK91] en 1991. Su nuevo algoritmo inicialmente parte del mismo principio estabilizador como el algoritmo SFTF, pero utiliza seis variables de control K_1 a K_6 para corregir las cantidades $B_p(k), \alpha_p^b(k), \gamma(k) y \mu(k)$[SK88]. Desafortunadamente todavía no ha sido establecido claramente como elegir K_1 a K_6 para garantizar la estabilidad para cualquier señal de entrada.

Algoritmo SFTF ($8P + 17$ MADPR)

Inicialización

$\mathbf{A_p}(0) = \mathbf{B_p}(0) = \mathbf{k_p}(0) = \hat{\mathbf{w}}(0) = \mathbf{x_p}(0) = 0$
$\gamma(0) = 1$
$\alpha_p^f(0) = \alpha_p^b(0) = \delta$, constante positiva pequeña
ρ, constante positiva adecuada

Para $k = 1$ hasta M	MADPR	
$e_p^f(k) = \mathbf{x_p}(k) - \mathbf{A_p}^T(k-1)\mathbf{x_p}(k-1)$	P	(T9.1)
$\varepsilon^f(k) = \gamma(k-1)e_p^f(k)$	1	(T9.2)
$\mathbf{A_p}(k) = \mathbf{A_p}(k-1) + \mathbf{k_p}(k-1)\varepsilon^f(k)$	P	(T9.3)
$\alpha_p^f(k) = \alpha_p^f(k-1) + e_p^f(k)\varepsilon^f(k)$	1	(T9.4)
$\gamma_+(k) = \gamma(k-1)\alpha_p^f(k-1)/\alpha_p^f(k)$	2	(T9.5)
$\begin{bmatrix} \mathbf{m}(k) \\ \mu(k) \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{k_p}(k-1) \end{bmatrix} - \frac{e_p^f(k)}{\alpha_p^f(k-1)} \begin{bmatrix} -1 \\ \mathbf{A_p}(k-1) \end{bmatrix}$	$P+1$	(T9.6)
$e_p^b(k) = x(k-P) - \mathbf{B_p}^T(k-1)\mathbf{x_p}(k)$	P	(T9.7)
$\xi(k) = e_p^b(k) - \mu(k)\alpha_p^b(k-1)$	1	(T9.8)
$e_p^\gamma(k) = e_p^b(k) + \eta_\gamma\xi(k)$	1	(T9.9)
$e_p^\varepsilon(k) = e_p^b(k) + \eta_\varepsilon\xi(k)$	1	(T9.10)
$e_p^\beta(k) = e_p^b(k) + \eta_\beta\xi(k)$	1	(T9.11)
$\gamma(k) = [1 - \gamma_+(k)\mu(k)e^\gamma(k)]^{-1}\gamma_+(k)$	3	(T9.12)
$\mathbf{k_p}(k) = \mathbf{m}(k) + \mu(k)\mathbf{B_p}(k-1)$	P	(T9.13)
$\alpha_p^b(k) = \alpha_p^b(k-1) + \gamma(k)\left[e_p^\varepsilon(k)\right]^2$	3	(T9.14)
$\mathbf{B_p}(k) = \mathbf{B_p}(k-1) + \mathbf{k_p}(k)\gamma(k)e_p^\beta(k)$	$P+1$	(T9.15)
Extensión para procesos conjuntos:		
$e(k\|k-1) = y(k) - \hat{\mathbf{w}}^T(k-1)\mathbf{x_p}(k)$	P	(T9.16)
$\hat{\mathbf{w}}(k) = \hat{\mathbf{w}}(k-1) + \mathbf{k_p}(k)\gamma(k)e(k\|k-1)$	$P+1$	(T9.17)

Tab. 3.10: Algoritmo SFTF.

3.5. Conclusiones

En este capítulo se trata el problema del desarrollo de algoritmos para encontrar un vector de coeficientes óptimo, así como algunas de las razones para llevar a cabo dicho desarrollo. Además también se trata de establecer las motivaciones para el problema de las técnicas de solución recursivas, de manera que estas soluciones garanticen tanto una buena aproximación para el vector de coeficientes óptimo, como una rapidez de convergencia lo suficientemente aceptable para llevar a cabo su implementación.

También se menciona el desarrollo del algoritmo recursivo de los mínimos cuadrados así como su costo computacional para ejemplificar de una manera simple las implicaciones del costo del conjunto de operaciones por cada recursion y su repercusión global para la implementación del algoritmo. El desarrollo del algoritmo recursivo de los mínimos cuadrados (RLS) nos sirve de base para comprender de mejor manera los algoritmos rápidos que se emplean más adelante en el presente trabajo, ya que todos ellos encuentran en el algoritmo RLS una plataforma de partida a pesar de que se modifican para hacer las operaciones ya no de manera matricial si no operan de una manera vectorial aprovechando el principio de la invarianza en el tiempo para los vectores de la señal reemplazando así el cálculo de matrices inversas de covarianza.

De igual manera se trata la estimación de parámetros con el algoritmo recursivo de los mínimos cuadrados para lo cual es necesario introducir los conceptos de modelado autorregresivo, modelado que nos permite representar señales y/o sistemas lineales como ecuaciones en diferencias, de modo que la estimación de los coeficientes se puede realizar bajo el criterio de los mínimos cuadrados. Y para el caso del algoritmo de los mínimos cuadrados, nos permite contar con un juego de parámetros para cada instante de tiempo.

En la segunda parte de este capítulo se muestran y se explican brevemente algunos de los algoritmos rápidos de los mínimos cuadrados separándolos en tres principales categorías: la primera en la cual se consideran los algoritmos rápidos de Kalman (FK), el algoritmo de filtro transversal de Cioffi y Kailath (FTF) y el algoritmo de error a posteriori de Carayannis (FAEST); en la segunda categoría de algoritmos normalizados so consideran al algoritmo normalizado en ganancia de Cioffi y Kailath (GFTF), el algoritmo normalizado de Gueguen y Fabre (NFTF) y el algoritmo de Lin (CFK), y por último, la tercer categoría de los algoritmos estabilizados como lo son los algoritmos de Moustakides (CFTF) y de Benallal y Gilloire (SFTF), en los cuales se introduce una cantidad que se puede considerar como una medida de error de redondeo y que interviene en el cálculo de los errores backward y forward.

Se puede constatar también que para los algoritmos rápidos no estabilizados (FK, FTF, FAEST, GFTF y NFTF) se puede llegar a cada uno de dichos algoritmos empleando transformaciones matemáticas entre ellos. También se observa que en todos ellos se obtienen los valores estimados para el error de predicción forward $e(k|k-1)$ y para los coeficientes $\hat{w}(k)$ para cada iteración [SR92].

4. METODOLOGÍA DE IMPLEMENTACIÓN DE ALGORITMOS PDS EN ARITMÉTICA DE PUNTO FIJO

Para establecer de una manera más detallada el procedimiento general para la implementación de algoritmos en aritmética de punto fijo, podemos enumerar las siguientes actividades a realizar durante el proceso.

Establecimiento del Flujo de la Simulación (MATLAB®, Simulink®)

Al implementar cualquier algoritmo el primer paso en esta estrategia es establecer el flujo de la simulación, es decir, detallar paso a paso el algoritmo para poder llevar a cabo su implementación en punto flotante, empleando algún lenguaje de programación, que para nuestro caso es Matlab.

Este procedimiento se realizó para cada uno de los algoritmos mencionados en el capítulo 3, de manera que se garantice la funcionalidad de cada uno de ellos. Para este paso se utiliza comúnmente una descripción gráfica del algoritmo mediante un diagrama de flujo.

Desarrollo del algoritmo en punto flotante

Una vez establecido el flujo del algoritmo se procede a la implementación en Matlab, enfocandose en la integridad algorítmica, para obtener así un buen funcionamiento de la implementación en punto flotante. Este paso se llevo al cabo de manera que se generó una biblioteca de los algoritmos a evaluar en las simulaciones en punto flotante.

Validación de la implementación en Punto Flotante

Para validar la implementación en punto flotante se procede a simular iterativamente dicha implementación para observar la relación costo/beneficio de la implementación del algoritmo, estableciendo para esto algún criterio que al cumplirse permita validar la implementación de acuerdo a los requerimientos.

En este punto se ejecutaron las simulaciones en punto flotante para obtener los resultados de cada una de ellas y evaluar estos resultados con los requerimientos originales.

Conversión del Diseño a Punto Fijo.

Esta tarea la podemos dividir en dos etapas principales, la primera **Realizar un estudio de Dinámicas** y la segunda **Empleo de la Herramienta de Punto Fijo para Definir el Algoritmo**.

Estudio de Dinámicas

En esta parte del proceso de implementación, se realiza un estudio de las dinámicas de cada una de las variables; es decir, se registran los valores máximos y mínimos de entrada y salida de cada una de las variables para poder establecer, para cada una de ellas, las longitudes de las partes entera y fraccional necesarias para representar todos los valores posibles que tomen durante la ejecución del algoritmo con una longitud de palabra finita.

La longitud de la parte entera es importante puesto que es la parte de la palabra binaria en la que podemos evitar saturación de los acumuladores en la ejecución de la implementación, mientras que la longitud de palabra de la parte fraccional es la que nos va a proporcionar la mayor precisión posible.

Validación de la Simulación en Punto Fijo

Simular iterativamente para observar el costo/beneficio de la implementación del algoritmo y validar de acuerdo a los requerimientos originales.

4.1. Definición del Algoritmo con la Herramienta de Punto Fijo

En esta etapa del procedimiento, nos debemos de adecuar al funcionamiento de la herramienta para poder representar las partes entera y fraccional de cada uno de los valores del algoritmo.

La herramienta Fixed-Point Toolbox de Matlab provee tipos de datos de punto fijo y permite el desarrollo de algoritmos en aritmética de punto fijo. Esta herramienta nos permite crear los siguientes tipo de objetos[Mat06]:

- **fi** - Define un objeto numérico de punto fijo en el espacio de trabajo de Matlab. Cada objeto **fi** se compone de un valor de dato, un objeto **fimath** y un objeto **numerictype**.
- **fimath** - Controla la manera en que los operadores aritméticos rigen los objetos del tipo **fi**.
- **fipref** - Define las preferencias de despliegue, registro y override del tipo de datos de los objetos **fi**
- **numerictype** - Define los atributos de tipo de datos y escalamiento de los objetos **fi**

- **quantizer** - Cuantiza los conjuntos de datos

El Fixed-Point Toolbox nos provee diversas características al trabajar con él, como son:

- La habilidad de definir tipos de datos de punto fijo, así como los métodos de escalamiento, redondeo y sobreflujo en el espacio de trabajo de Matlab.
- Simulaciones reales y complejas.
- Aritmética de punto fijo básica:
  - Operadores aritméticos $+, -, *, .*$ para señales reales y de punto binario.
  - División al emplear la función `divide` para señales de punto binario.
- Expansión arbitraria de largo de palabra para `intmax('uint16')`.
- Registro de mínimos, máximos, overflows y underflows.
- Evitar el uso de tipos de datos singles, doubles o scaled doubles.
- Conversión entre binarios, hexadecimales, dobles y enteros.
- Operadores relacionales, lógicos y de bit a bit.
- Funciones con matrices como **ctranspose** y **horzcat**
- Funciones estadísticas como **max** y **min**
- Interoperabilidad con Simulink, el Blockset de Procesamiento de Señales, Embedded Matlab y el Toolbox de Diseño de Filtros.
- Compatibilidad de Simulink con el espacio de trabajo y de los bloques del espacio de trabajo.

Ahora bien, para poder utilizar esta herramienta es necesario comprender algunos conceptos de la aritmética de punto fijo que los tipos de datos del Fixed Point Toolbox emplea.

4.1.1. Tipos de Datos de Punto Fijo

En el hardware digital, los números se almacenan en palabras binarias. Una palabra binaria es una secuencia de bits (1's y 0's) de longitud definida. El como son interpretadas estas secuencias de bits por los componentes de hardware o funciones de software depende de el tipo de datos. [Mat06]

Los números binarios pueden ser representados como tipos de datos de punto fijo o como punto flotante. En este documento presentaremos algunos términos y conceptos relacionados con los números, tipos de dato y aritmética de punto fijo.

Un tipo de dato de punto fijo se caracteriza por la longitud de bits, la posición del punto binario y si es o no signado. La posición de el punto binario es la manera mediante la cual los valores de punto fijo son escalados e interpretados.

Por ejemplo, la representación de un número en punto fijo cualquiera (sea signado o no signado) se muestra a continuación:

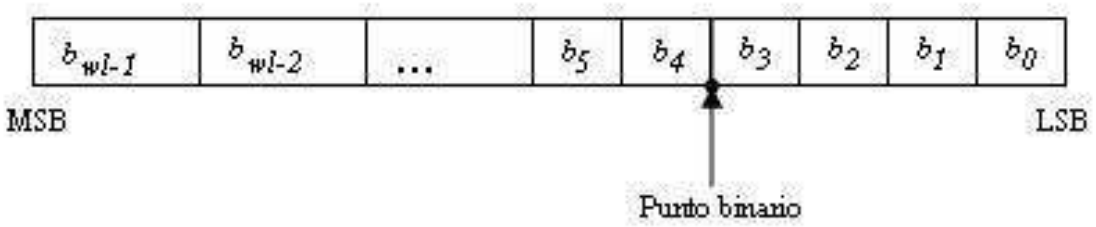


Fig. 4.1: Representación de un número en punto fijo

en donde

- b_i es el i-ésimo dígito binario.
- wl es la longitud de palabra en bits.
- b_{wl-1} es la ubicación del bit más significativo (MSB)
- b_0 es la ubicación del bit menos significativo (LSB)
- El punto binario se muestra cuatro posiciones a la izquierda del bit menos significativo. Por consiguiente, este ejemplo, decimos que el número tiene cuatro bits de parte fraccional, o una longitud de fracción de cuatro bits

Los tipos de datos de punto fijo pueden ser signados o no signados. Los números binarios de punto fijo signados se representan comúnmente en una de estas tres maneras:

- Signo/magnitud
- Complemento a uno
- Complemento a dos

La forma de complemento a dos es la representación más común para los números de punto fijo signados y es la única representación utilizada por el Fixed Point Toolbox.

4.1.2. Escalamiento

Los números de punto fijo pueden ser codificados de acuerdo al siguiente esquema

$$valor\ real = (slope\ x\ entero) + sesgo$$

en donde el slope se puede expresar como

$$slope = slope\ fraccional\ x\ 2^{exponente}$$

El entero es en ocasiones llamado *entero almacenado*. Este es el número binario que sirve de base, en el cual asumimos que el punto binario se encuentra en la extrema derecha de la palabra. En el Fixed Point Toolbox, el negativo del exponente es comúnmente conocido como la *longitud fraccionaria*.

El slope y el sesgo en conjunto representan el escalamiento para el número de punto fijo. En un número con sesgo cero, es únicamente el slope el que afecta el escalamiento. Un número de punto fijo que es escalado solo por la posición del punto binario equivale a un número en representación [Slope Sesgo], que tiene un sesgo igual a cero y un slope fraccional igual a uno. Esto es conocido como escalamiento por punto binario o escalamiento en potencia a 2.

$$valor\ real = 2^{exponente}\ x\ entero$$

o

$$valor\ real = 2^{-longitud_fraccional}\ x\ entero$$

El Fixed Point Toolbox soporta ambos escalamientos.

4.1.3. Precisión y Rango

Se debe tener cuidado en la precisión y el rango de los tipos de datos de punto fijo y escalamientos que se eligen para conocer que métodos de redondeo se debe invocar o si sobreflujos o underflows pueden ocurrir.

Rango

El rango es el conjunto de números que un tipo de datos y escalamiento de punto fijo puede representar. El rango de los números representables para un número de punto fijo en complemento a dos de longitud de palabra wl, escalamiento S y sesgo B se ilustra a continuación:

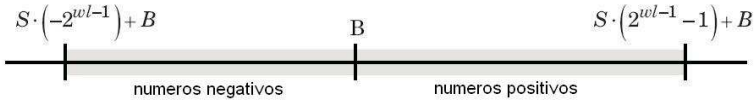


Fig. 4.2: Rango representable para un número de punto fijo

Tanto para los números en punto fijo signados como no signados de cualquier tipo de datos, el número de diferentes patrones de bits es 2^{wl}.

Manejo de Sobreflujos

Dado que los tipos de datos de punto fijo representan números en un rango finito, pueden ocurrir sobreflujos y underflows si el resultado de una operación es más largo o más pequeño que los números en dicho rango.

El Fixed Point Toolbox nos permite *saturar* o *wrap* los sobreflujos. La saturación representa sobreflujos positivos como el número positivo más grande en el rango a ser utilizado y los sobreflujos negativos como el número negativo más grande a ser utilizado dentro del rango. El Wrapping emplea aritmética modulo dos para acomodar un sobreflujo dentro del rango representable del tipo de datos.

Precisión

La precisión para un número de punto fijo es la diferencia entre valores sucesivos representables por su tipo de datos y escalamiento, que también es igual al valor de su bit menos significativo. El valor del bit menos significativo, así como la precisión del número, estan determinados por la cantidad de bits fraccionales.

Métodos de Redondeo

Una de las limitaciones de representar números con precisión finita es que no todos los números en el rango disponible pueden ser representados con exactitud. Cuando el resultado de un cálculo en punto fijo es un número que no puede ser representado exactamente por el tipo de datos y su escalamiento utilizado, se pierde precisión. Un método de redondeo debe ser empleado para hacer corresponder el resultado con un número representable.

Actualmente el Fixed Point Toolbox soporta los siguientes métodos de redondeo:

- **ceil** redondea hacia el número representable más cercano en la dirección del infinito positivo.
- **convergent** redondea hacia el entero representable más cercano. En caso de una misma distancia, este método redondea hacia el número entero más cercano almacenado. Este es el método de redondeo con menos sesgo que provee el Fixed Point Toolbox.
- **fix** redondea hacia el entero más cercano representable en la dirección del cero.
- **floor** es equivalente al truncamiento de complemento a dos, redondea hacia el número más cercano representable en dirección del infinito negativo.
- **nearest** redondea hacia el entero más cercano representable. En caso de que exista la misma distancia, redondea hacia el entero más cercano representable en la dirección del infinito positivo. Este es el método de redondeo por default para la creación de un objeto **fi** y la aritmética **fi**.
- **round** redondea hacia el entero más cercano representable. En caso de que exista la misma distancia, redondea los números positivos hacia el entero más cercano representable en la dirección del infinito positivo, y los números negativos los redondea hacia el entero más cercano representable en la dirección del infinito negativo.

4.1.4. Operaciones Aritméticas

Para tener claro como se llevan a cabo las operaciones aritméticas con el Fixed Point Toolbox es importante recordar dos tipos de aritmética que pueden ser empleadas cuando se trabaja con números binarios: como son la aritmética modulo y la aritmética de complemento a dos.

Aritmética Módulo

La aritmética módulo emplea únicamente un conjunto de números finitos, encapsulando a los resultados de cualquier cálculo que estén fuera de dicho conjunto de nuevo dentro de dicho conjunto.

Un ejemplo muy claro de este tipo de aritmética es el reloj de uso diario, el cual utiliza aritmética modulo 12. Los números en este sistema solo pueden estar entre 1 y 12. Por lo tanto, en este sistema, $9 + 9 = 6$

De manera similar, las matemáticas binarias únicamente pueden utilizar números 0 y 1, y cualquier resultado aritmético que este fuera del rango es traslapado *alrededor del circulo* ya sea como 0 o 1.

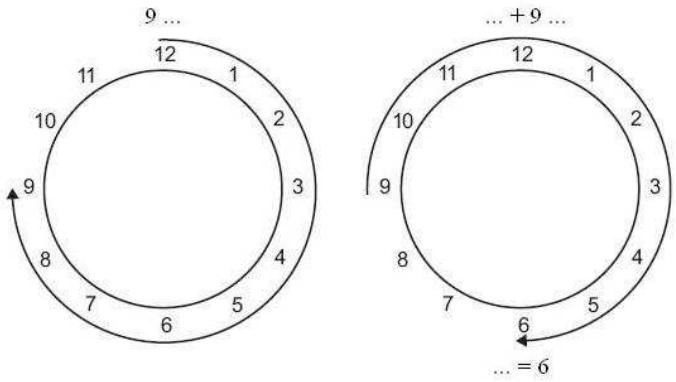


Fig. 4.3: Ejemplo de aritmética modulo

Complemento a Dos

El complemento a dos es una manera de interpretar un número binario. En complemento a dos, los números positivos siempre comienzan con 0 y los números negativos siempre comienzan con 1. Si el primer bit de un número complemento a dos es 0, el valor es obtenido calculando el valor binario del número de manera normal. Si el primer bit de un número complemento a dos es 1, el valor es obtenido asumiendo que el bit de extrema izquierda es negativo y posteriormente calculando el valor binario del número. Por ejemplo,

$$01 = \left(0 + 2^0\right) = 1$$

$$11 = \left(\left(-2^1\right) + \left(2^0\right)\right) = (-2 + 1) = -1$$

Para calcular el negativo de un número binario utilizando complemento a dos,

- Tomar el complemento a uno, o *cambiar los bits*.
- Sumar un 1 utilizando matemática binaria.
- Descartar cualquier acarreo de bits más allá de la longitud original de la palabra.

Por ejemplo, considere el número negativo 11010 (-6). Primero tomamos el complemento a uno del número o intercambiamos los bits

$$11010 \rightarrow 00101$$

A continuación, se suma un 1,

$$\begin{array}{r} 00101 \\ +1 \\ \hline 00110 \end{array} \quad (6)$$

Una vez comprendidos los tipos de aritmética con las que se trabajan los números binarios, a continuación se describen las operaciones aritméticas utilizadas por el Fixed Point Toolbox de Matlab:

4.1.5. Suma y Resta

La suma de números de punto fijo requiere que los puntos binarios de los sumandos sea alineada. La suma se realiza empleando aritmética binaria por lo cual ningún número que no sea 1 o 0 es utilizado.

Por ejemplo, considere la suma de 010010.1 (18.5) con 0110.110 (6.75):

$$\begin{array}{ll} 010010{,}1 & (18{,}5) \\ +00110{,}110 & (6{,}75) \\ \hline 011001{,}010 & (25{,}25) \end{array}$$

La suma de punto fijo es equivalente a sumar mientras se utiliza el valor de complemento a dos para cualquier valor negativo. En la resta, los sumandos deben ser de signo extendido para corresponder a cada una de sus longitudes de palabra. Por ejemplo, restar 0110.110 (6.75) a 010010.1 (18.5):

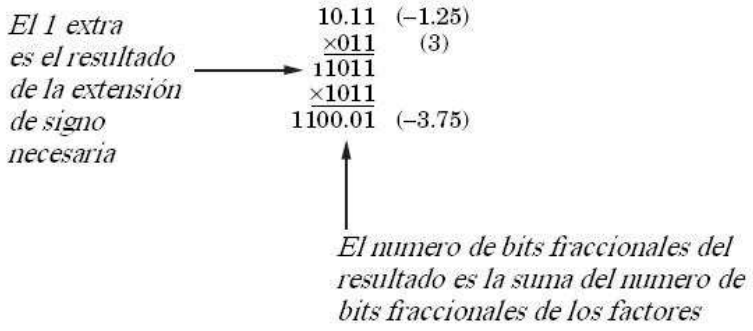
El objeto **fimath** default tiene un valor de 1 (verdadero) para la propiedad `CastBeforeSum`. Esto arregla los sumandos para el tipo de datos de suma antes de llevar a cabo la adición. Por lo tanto no es necesario realizar ningún corrimiento durante la suma para alinear el punto binario.

Si la propiedad CastBeforeSum tiene un valor de 0 (falso), los sumandos son adicionados manteniendo la precisión completa. Después de la operación la suma es cuantizada.

4.1.6. Multiplicación

La multiplicación complemento a dos de números en punto fijo es directamente análoga a la multiplicación decimal regular, excepto que los resultados intermedios deben ser extendidos en signo de manera que sus lados izquierdos con respecto al punto binario se alinien antes de sumarlos.

Por ejemplo, considere la multiplicación de 10.11 (-1.25) con 011 (3):



4.1.7. Tipos de Datos de Multiplicación

A continuación se ilustran los tipos de datos que se utilizan en la multiplicación de punto fijo. Loa siguientes diagramas ilustran las diferencias entre los tipos de datos utilizados para las multiplicaciones real-real, complejo-real y complejo-complejo.

Multiplicación Real-Real. El siguiente diagrama muestra los tipos de datos utilizados en la multiplicación de dos números reales en el Fixed Point Toolbox. La salida de esta multiplicación esta en el tipo de datos que es gobernada por la propiedad ProductMode del objeto **fimath**:

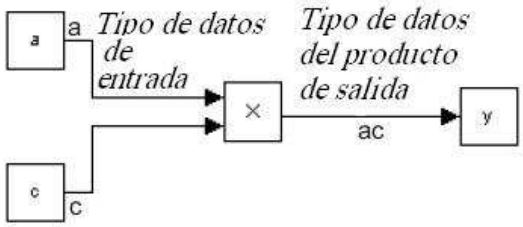


Fig. 4.4: Multiplicación de dos numeros reales.

Multiplicación Real-Complejo. El siguiente diagrama muestra el tipo de datos empleados en la multiplicación de un número real de punto fijo con un número complejo de punto fijo en el Fixed Point Toolbox. La salida de esta multiplicación esta en el tipo de datos de la multiplicación gobernado por la propiedad ProductMode del objeto **fimath**:

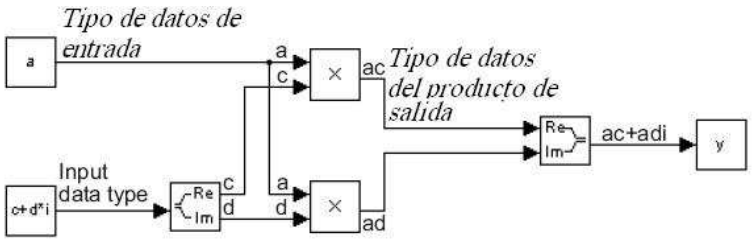


Fig. 4.5: Multiplicación de un numero real por un complejo.

Multiplicación Complejo-Complejo. El siguiente diagrama muestra el tipo de datos empleados en la multiplicación de dos números complejos de punto fijo en el Fixed Point Toolbox. Aquí es importante notar que la salida de esta multiplicación esta en el tipo de datos de la suma, la cual es gobernado por la propiedad Summae de objeto **fimath**. El tipo de datos de la multiplicación la determina la propiedad ProductMode del objeto **fimath**:

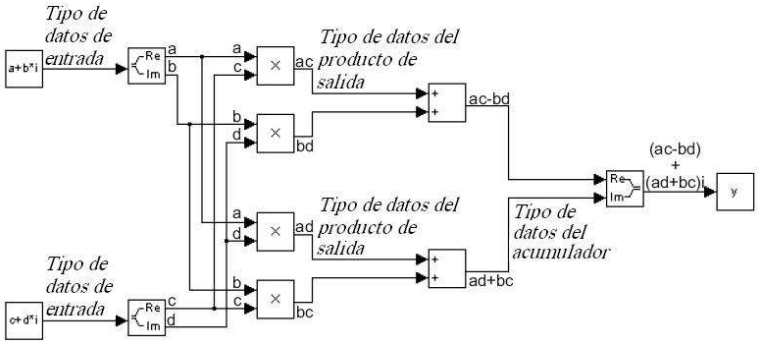


Fig. 4.6: Multiplicación de dos numeros complejos

4.2. Conclusiones

En este capítulo se describe la estrategia de implementación de los algoritmos en aritmética de punto fijo, en donde se detalla la manera en la que se deben de definir los objetos de punto fijo con la herramienta utilizada, los tipos de dato que soporta la herramienta, como realizar el escalamiento, las consideraciones que se deben tener en cuanto a la precisión y el rango de los objetos de punto fijo, como realizar las operaciones aritméticas con la herramienta de punto fijo tales como la suma y resta y la multiplicación.

El primer paso en esta estrategia es establecer el flujo de la simulación para llevar a cabo la implementación del algoritmo en punto flotante empleando Matlab de manera que se garantice la funcionalidad de cada uno de ellos enfocándose en la integridad algorítmica, para obtener así un buen funcionamiento de la implementación en punto flotante. Esto nos llevo a la realización de una biblioteca de algoritmos a evaluar en las simulaciones de punto flotante para validar la implementación en punto flotante ejecutando iterativamente dicha implementación para observar la relación costo/beneficio.

Después procedimos a la conversión del diseño a punto fijo, etapa que podemos dividir en dos tareas principales: la realización un estudio de dinámicas; en la cual se registran los valores máximos y mínimos de entrada y salida de cada una de las variables para establecer las longitudes de palabra necesarias para representar todos los valores del algoritmo con una longitud de palabra finita, y el empleo de la herramienta de punto fijo para definir el algoritmo al hacer todas las consideraciones pertinentes para su funcionamiento en punto fijo tales como: largos de palabra, métodos de redondeo y truncamiento y manejo de saturación. Concluyendo con la validación de las implementaciones en punto fijo al simular iterativamente para observar el costo/beneficio del algoritmo y validar los

resultados de acuerdo a los requerimientos originales.

Dentro de la definición del comportamiento de los algoritmos en aritmética de punto fijo, la longitud de palabra de la parte entera es muy importante puesto que es la parte de la palabra binaria en la que podemos evitar saturación de los acumuladores en la ejecución de la implementación, mientras que la longitud de palabra de la parte fraccional es la que nos va a proporcionar la mayor precisión posible.

5. IMPLEMENTACIÓN DE LOS ALGORITMOS FRLS

En este capítulo se presentan algunas de las simulaciones de la implementación en punto flotante de los algoritmos rápidos de los mínimos cuadrados descritos en los capítulos anteriores para la estimación de parámetros, identificación de sistemas e igualación de canal empleando los modelos de un conjunto de canales de comunicación de fase mínima y de fase no mínima. También se presentan las resultados de algunas de las implementaciones en punto fijo para la igualación de canal empleando la herramienta de punto fijo (**Fixed Point Toolbox**) de **Matlab**.

Cabe aclarar que los resultados de las simulaciones que se realizaron para los demás canales y algoritmos introducidos durante esta investigación se pueden encontrar en el Reporte de Simulaciones de la Implementación de Algoritmos en Punto Fijo con Matlab [Gal08], ya que el detalle de las simulaciones presentadas en esta sección se consideraron las más representativas.

5.1. Canales Empleados

Para validar el comportamiento de los algoritmos descritos anteriormente, se eligió un conjunto de modelos de canales de comunicación los cuales se pueden agrupar como:

- *Canales de fase mínima*
- *Canales de fase no mínima*

Estos canales están definidos por su función de transferencia mostrada en la tabla 5.1 y sus respuestas en frecuencia pueden verse en las figuras 5.1, 5.2, y 5.3. Tales canales fueron utilizados para la gran mayoría de los algoritmos programados, sin embargo solo se presentan los resultados más significativos que se obtuvieron.

Cabe mencionar que para los canales de fase no mínima y fase máxima existe una dificultad para obtener el filtro igualador, ya que debido a sus características, estos canales no son invertibles, razón por la cual solo es posible aproximar su respuesta inversa mediante un filtro de respuesta impulsional finita de orden excesivamente grande.

Canales de Fase Mínima	
Función de Transferencia	Referencia
$H_{C1} = 1 + 0{,}2z^{-1} - 0{,}6z^{-2}$	[CY03] [Hay91]
$H_{C2} = 1 + 0{,}51z^{-1} + 0{,}1997z^{-2}$	[CY03] [Hay91]
$H_{C3} = 1 + 0{,}536z^{-1} + 0{,}0718z^{-2}$	[CY03] [Hay91]
$H_{C4} = 1 - 1{,}6z^{-1} + 0{,}95z^{-2}$	[Hay84]
$H_{C5} = 1 - 1{,}9z^{-1} + 0{,}95z^{-2}$	[Hay84]
Canales de Fase No Mínima	
Función de Transferencia	Referencia
$H_{C6} = 0{,}407 + 0{,}815z^{-1} + 0{,}407z^{-2}$	[Pro01]
$H_{C7} = 0{,}04 - 0{,}05z^{-1} + 0{,}07z^{-2} - 0{,}21z^{-3} - 0{,}5z^{-4}$ $+0{,}72z^{-5} + 0{,}36z^{-6} + 0{,}21z^{-8} + 0{,}03z^{-9} + 0{,}07z^{-10}$	[Pro01] [Pro01]

Tab. 5.1: Canales empleados en las simulaciones.

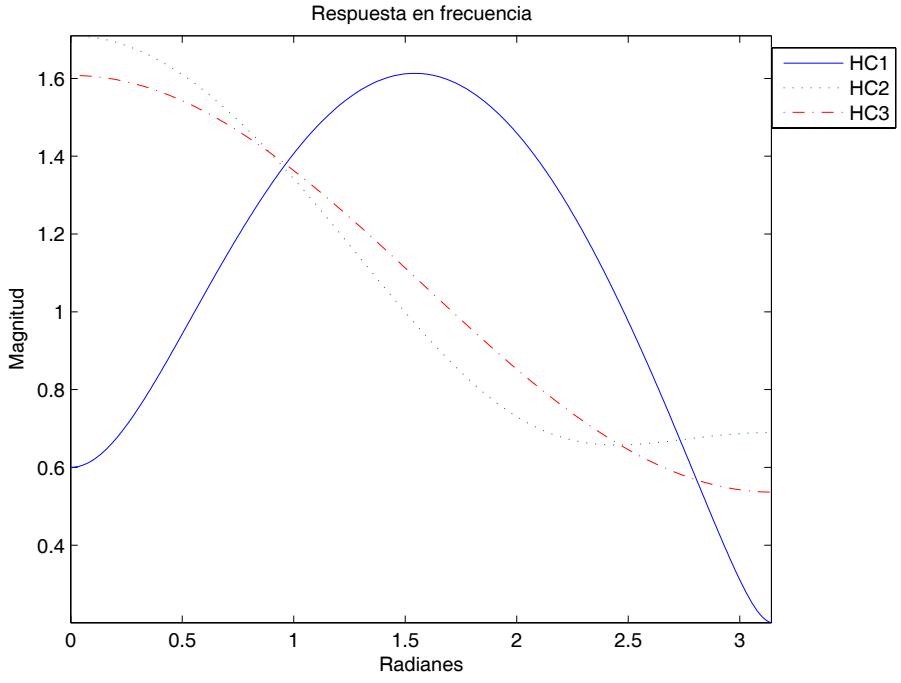


Fig. 5.1: Canales H_{C1}, H_{C2} y H_{C3} de fase mínima.

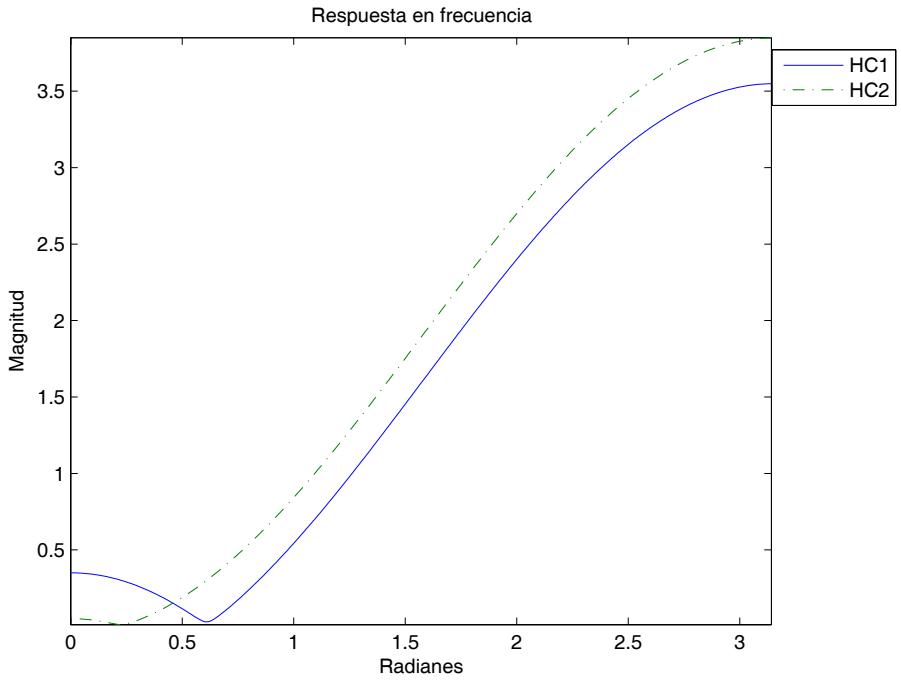


Fig. 5.2: Canales H_{C4} y H_{C5} de fase mínima.

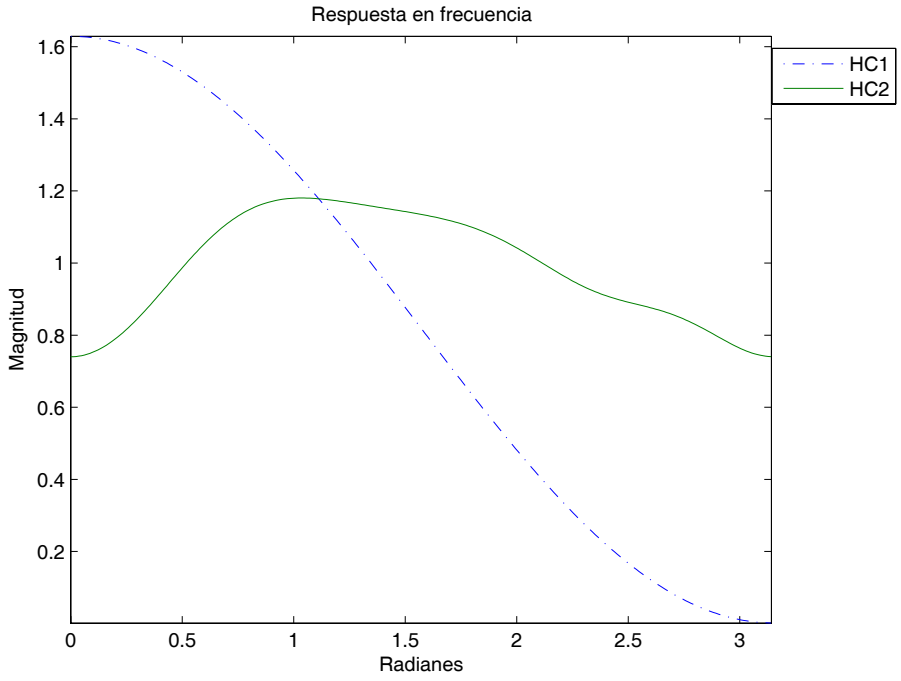


Fig. 5.3: Canales H_{C6} y H_{C7} de fase no mínima.

5.2. Implementaciones en Punto Flotante

5.2.1. Estimación de Parámetros

En esta sección se emplean los algoritmos rápidos descritos para atacar el problema de predicción. A continuación se muestra la tabla 5.2 en la que se realiza una comparación de los resultados obtenidos de la estimación de los parámetros de una señal de 2000 muestras tanto con el algoritmo matricial de los mínimos cuadrados como con los diferentes algoritmos de los mínimos cuadrados rápidos.

Cabe mencionar que para efectos de estas simulaciones la señal utilizada fue la señal resultante de aplicar un ruido blanco gaussiano a un sistema con dos polos ubicados en $p_1 = e^{\pi/4}$ y $p_3 = e^{3\pi/4}$ y sus complejos conjugados $p_2 = e^{-\pi/4}$ y $p_4 = e^{-3\pi/4}$, el parámetro lambda se mantuvo $\lambda = 1$, el parámetro $\delta = 0{,}01$ y se consideraron condiciones iniciales igual a cero.

A continuación se muestra en la figura 5.4 de la densidad espectral de potencia obtenida con los diferentes métodos antes mencionados así como con el algoritmo FK.

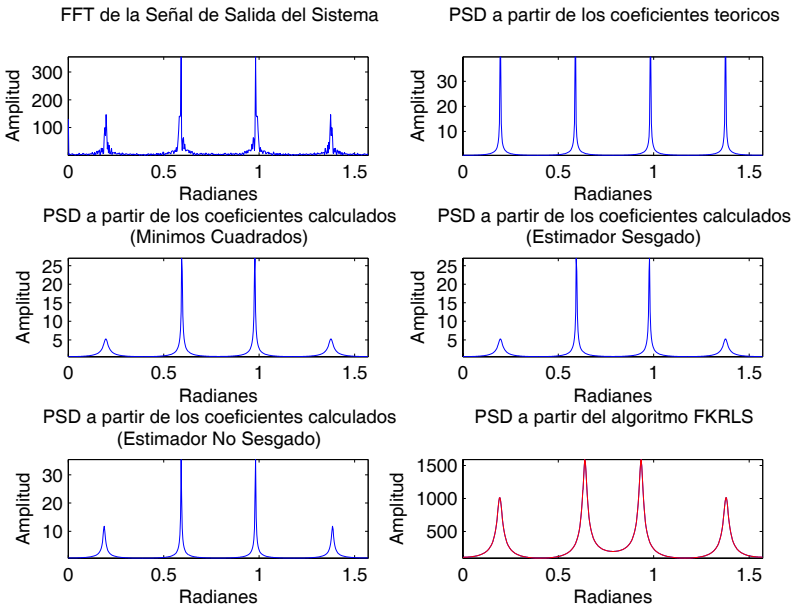


Fig. 5.4: Estimación de la PSD con los diferentes métodos de estimación de parámetros y con el algoritmo FK

Ahora en la figura 5.5 a) se muestra un espectrograma a partir de los coeficientes estimados para cada iteración.

En la figura 5.5 b) se muestra la manera en la que converge la estimación de los parámetros para el algoritmo FKRLS y se puede apreciar que tiene una convergencia más rápida que el algoritmo RLS.

Coeficientes Estimados en Predicción con los algoritmos RLS y RLS Rápidos	
Teóricos	
a_0 = 1.00000000000000	a_1 = -0.00000000000000
a_2 = 0	a_3 = -0.00000000000000
a_4 = 1.00000000000000	
RLS	
a_0 = 1.00000000000000	a_1 = -0.11943632519157+0.000000000000001i
a_2 = -0.10143012138492-0.000000000000000i	a_3 = -0.06047809512611-0.000000000000002i
a_4 = 1.08674775534886+0.000000000000001i	
FK	
a_0 = 1.00000000000000	a_1 = -0.11943632513751+0.000000000000000i
a_2 = -0.10143012120827+0.000000000000000i	a_3 = -0.06047809503563-0.000000000000003i
a_4 = 1.08674775550411+0.000000000000001i	
FTF	
a_0 = 1.00000000000000	a_1 = -0.11943634446242+0.000000000004460i
a_2 = -0.10143008353374-0.000000000002203i	a_3 = -0.06047813464943-0.000000000002063i
a_4 = 1.08674777329745+0.000000000005528i	
FAEST	
a_0 = 1.00000000000000	a_1 = -1.70121561388336-0.000000000000001i
a_2 = 1.74377767482325+0.000000000000001i	a_3 = -1.73316973767450+0.000000000000003i
a_4 = 1.81788090608255-0.000000000000010i	
GFTF	
a_0 = 1.00000000000000	a_1 = -0.11943646235171+0.000000000002701i
a_2 = -0.10142985222446+0.000000000002312i	a_3 = -0.06047837623226-0.000000000001603i
a_4 = 1.08674788307137-0.000000000003422i	
NFTF	
a_0 = 1.00000000000000	a_1 = -0.11943632199903+0.000000000005838i
a_2 = -0.10143012764258-0.000000000003044i	a_3 = -0.06047808859055-0.000000000001002i
a_4 = 1.08674775237912+0.000000000003535i	
CFK	
a_0 = 1.00000000000000-0.000000000010687i	a_1 = -0.11943632527533-0.000000000010196i
a_2 = -0.10143012169787-0.000000000004379i	a_3 = -0.06047809536349+0.000000000009641i
a_4 = 1.08674775511513+0.000000000000221i	
CFTF	
a_0 = 1.00000000000000-0.000000000000001i	a_1 = -0.11943632518519-0.000000000000001i
a_2 = -0.10143012136887+0.000000000000001i	a_3 = -0.06047809510330+0.000000000000002i
a_4 = 1.08674775535416-0.000000000000002i	
SFTF	
a_0 = 1.00000000000000-0.000000000000001i	a_1 = -0.11943632518134-0.000000000000001i
a_2 = -0.10143012135189+0.000000000000001i	a_3 = -0.06047809510669+0.000000000000002i
a_4 = 1.08674775537602-0.000000000000002i	

Tab. 5.2: Parámetros estimados utilizando predicción RLS

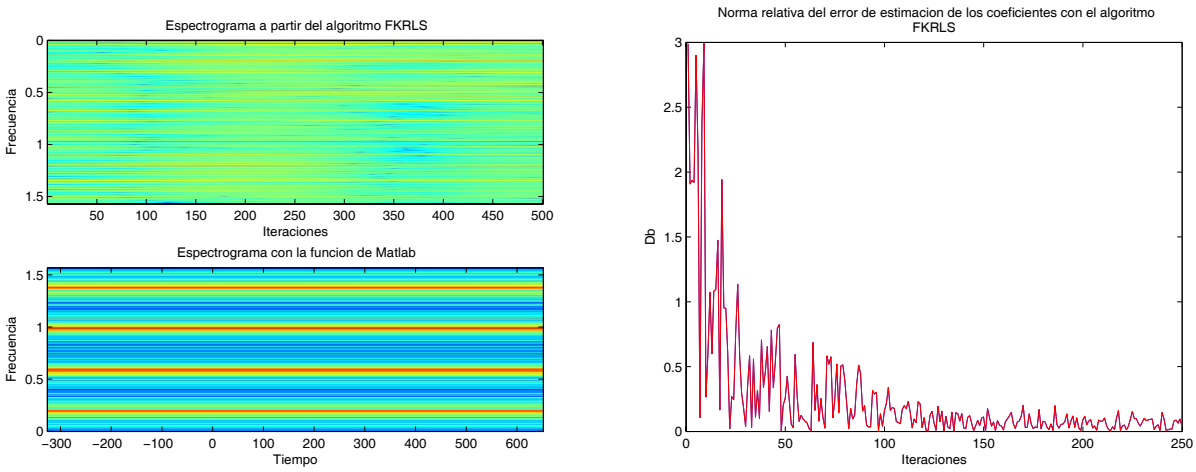


Fig. 5.5: a) Espectrograma con la estimación de predicción FK y con la función de Matlab. b) Norma relativa del error de la estimación de parámetros en predicción con el algoritmo FK

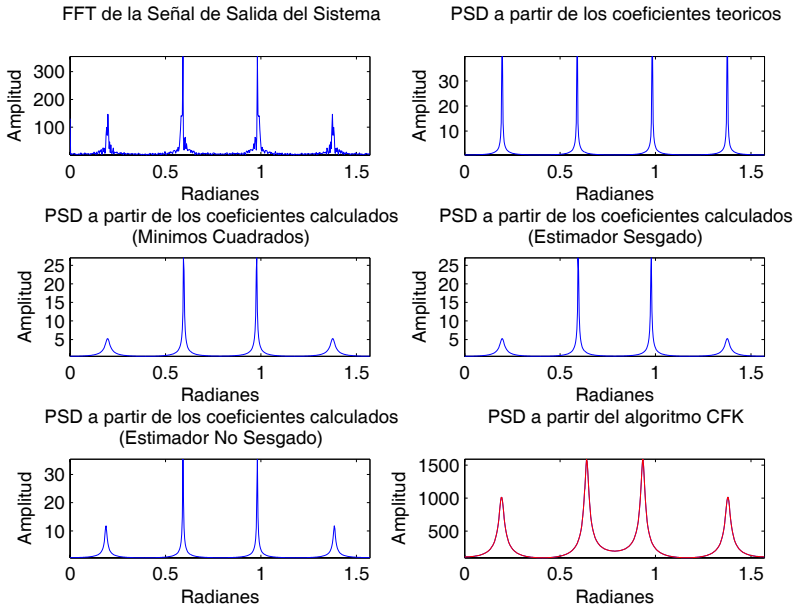


Fig. 5.6: Estimación de la PSD con los diferentes métodos de estimación de parámetros y con el algoritmo CFK

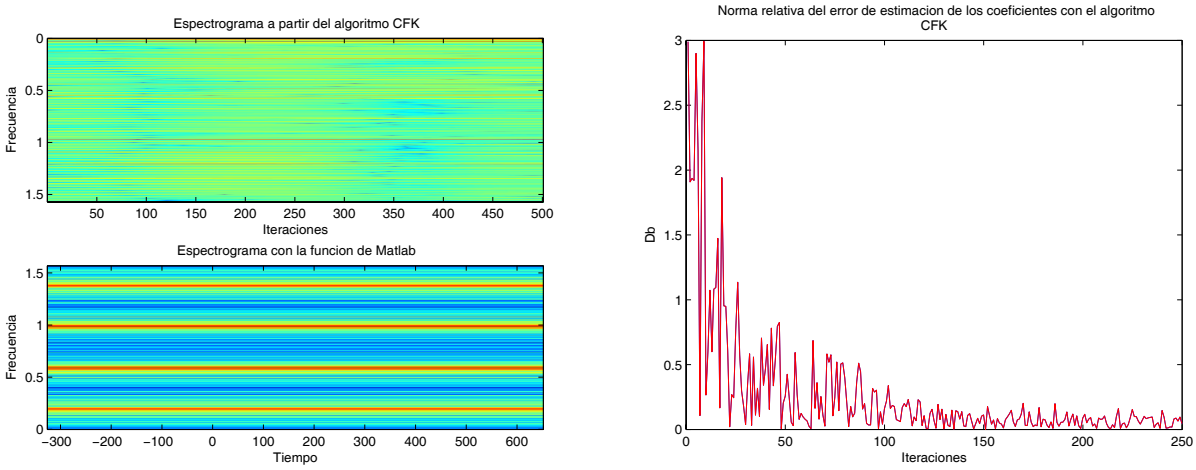


Fig. 5.7: a) Espectrograma con la estimación de predicción CFK y con la función de Matlab. b) Norma relativa del error de la estimación de parámetros en predicción con el algoritmo CFK

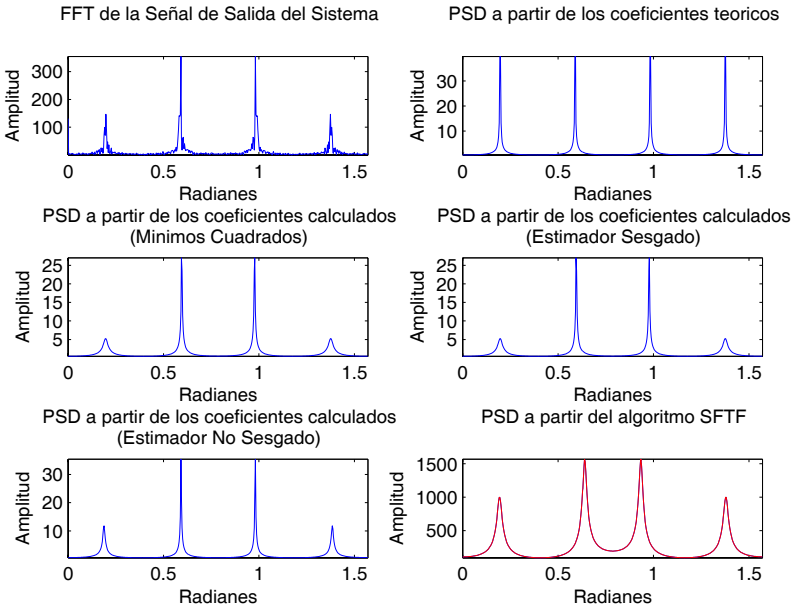


Fig. 5.8: Estimación de la PSD con los diferentes métodos de estimación de parámetros y con el algoritmo SFTF

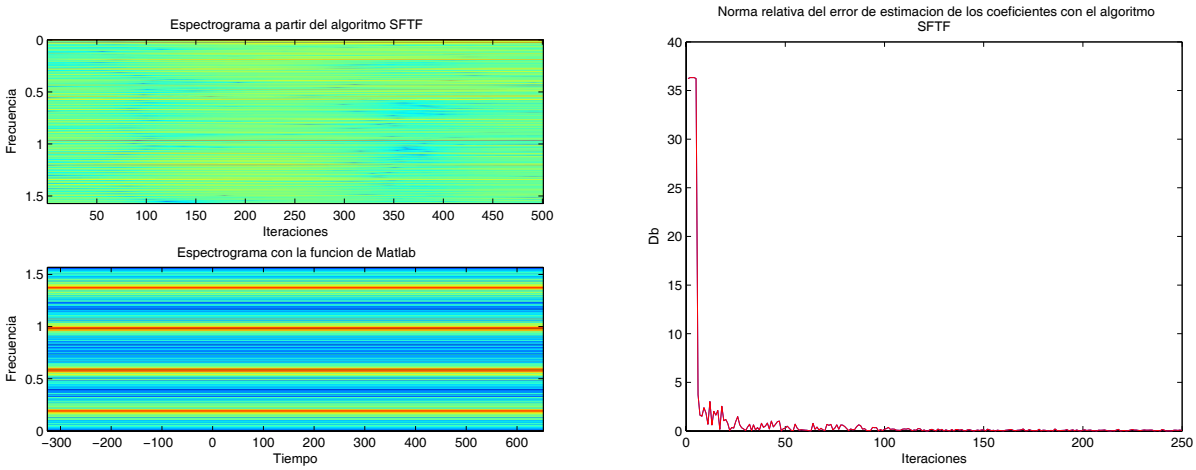


Fig. 5.9: a) Espectrograma con la estimación de predicción SFTF y con la función de Matlab. b) Norma relativa del error de la estimación de parámetros en predicción con el algoritmo SFTF

5.2.2. Identificación de Sistemas

A continuación se presentan las simulaciones en punto flotante de la identificación de los canales empleados utilizando los algoritmos rápidos.

En primera instancia en la tabla 5.3 se muestran los resultados obtenidos para cada uno de los algoritmos usados en la identificación del primer canal de fase mínima. Por otro lado en la figura 5.10 se muestran a) la norma relativa del error para cada uno de los algoritmos y b) la respuesta en frecuencia tanto del canal como de los sistemas identificados con cada uno de los algoritmos.

Canal	1	.2	-.6
RLS	.99998031592972048	.19999954400102324	-.59998116603524609
FK	.99998031592972048	.19999954400102324	-.59998116603524609
FTF	.99998031592972059	.19999954400102329	-.59998116603524621
FAEST	.99998031592972036	.19999954400102321	-.59998116603524598
GFTF	.99998031592972048	.19999954400102316	-.59998116603524609
NFTF	.99998031592972070	.19999954400102329	-.59998116603524643
CFK	.99998031592972036	.19999954400102321	-.59998116603524621
CFTF	.99998031592972048	.19999954400102324	-.59998116603524609
SFTF	.99998031592972036	.19999954400102321	-.59998116603524598

Tab. 5.3: Coeficientes identificados para el primer canal de fase mínima

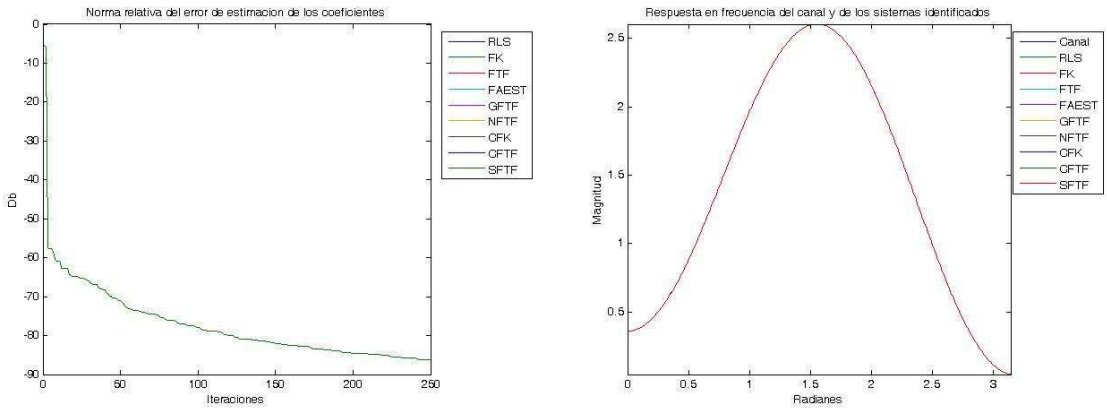


Fig. 5.10: Identificación del primer canal de fase mínima con los algoritmos a) Norma relativa del error b) Respuesta en frecuencia

Para las simulaciones de identificación de sistemas al emplear el cuarto canal de fase mínima se obtuvieron los resultados mostrados en la tabla 5.4 misma que muestra los valores de los coeficientes obtenidos tanto para el algoritmo RLS como para cada uno de los algoritmos rápidos.

Canal	1	-1.6	0.95
RLS	.99999404375152989	-1.5999886461196771	.94999433908878550
FK	.99999404375152989	-1.5999886461196771	.94999433908878550
FTF	.99999404375153034	-1.5999886461196775	.94999433908878583
FAEST	.99999404375152989	-1.5999886461196771	.94999433908878550
GFTF	.99999404375152989	-1.5999886461196771	.94999433908878550
NFTF	.99999404375153034	-1.5999886461196775	.94999433908878594
CFK	.99999404375152989	-1.5999886461196771	.94999433908878550
CFTF	.99999404375152989	-1.5999886461196771	.94999433908878550
SFTF	.99999404375152989	-1.5999886461196771	.94999433908878550

Tab. 5.4: Coeficientes identificados para el cuarto canal de fase mínima

Al igual que para los canales anteriores, en la figura 5.11 se muestran la curva de desempeño tomando como referencia la norma relativa de estimación de los parámetros en su inciso a, así como la respuesta en frecuencia tanto del canal como de cada uno de los juegos de coeficientes obtenidos para el sistema identificado con los diferentes algoritmos.

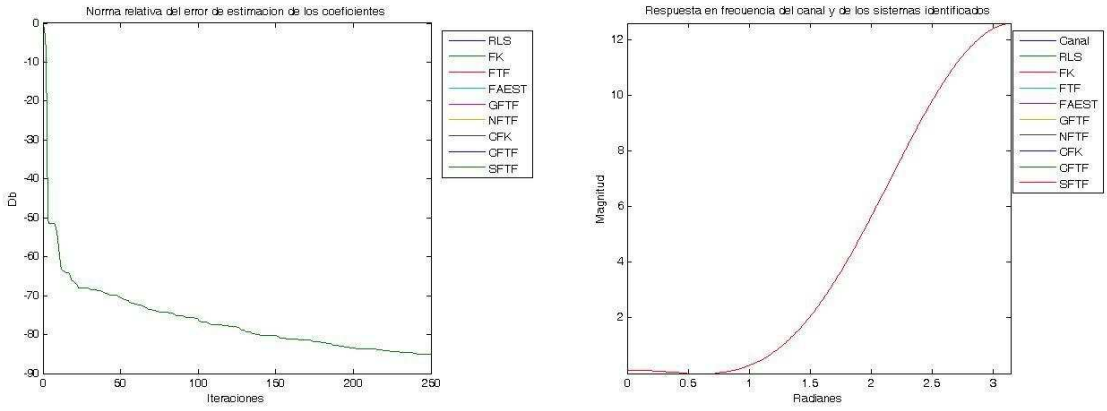


Fig. 5.11: Identificación del cuarto canal de fase mínima con los algoritmos a) Norma relativa del error b) Respuesta en frecuencia

La tabla 5.5 muestra los valores obtenido de los coeficientes en las simulaciones de punto flotante para la identificación del primer canal de fase no mínima para cada uno de los algoritmos

Y la figura 5.12 ilustra por un lado; el desempeño de los algoritmos en el inciso a) con la norma relativa de error de estimación de los parámetros, y la respuesta en frecuencia de los sistemas o canales identificados en el inciso b).

Canal	0.407	0.815	0.407
RLS	.40700245137332569	.81499145258953455	.40700244013889603
FK	.40700245137332569	.81499145258953455	.40700244013889603
FTF	.40700245137332564	.81499145258953488	.40700244013889597
FAEST	.40700245137332564	.81499145258953466	.40700244013889597
GFTF	.40700245137332569	.81499145258953455	.40700244013889603
NFTF	.40700245137332569	.81499145258953488	.40700244013889597
CFK	.40700245137332569	.81499145258953466	.40700244013889603
CFTF	.40700245137332569	.81499145258953455	.40700244013889603
SFTF	.40700245137332564	.81499145258953466	.40700244013889597

Tab. 5.5: Coeficientes identificados para el primer canal de fase no mínima

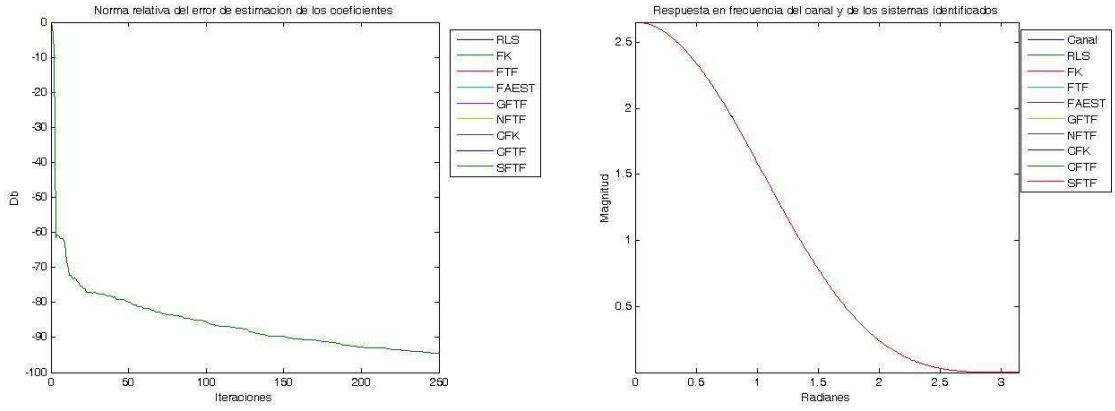


Fig. 5.12: Identificación del primer canal de fase no mínima con los algoritmos a) Norma relativa del error b) Respuesta en frecuencia

5.2.3. Igualación de Canal

A continuación se presentan los resultados de las simulaciones de igualación de canal con todos los algoritmos rápidos de los mínimos cuadrados para cada uno de los canales tanto de fase mínima, de fase no mínima y de fase máxima.

En la tabla 5.6 se presentan los resultados de la igualación del primer canal de fase mínima empleado, así mismo en las figura 5.13 se ilustran la respuesta en frecuencia del canal y de los filtros igualadores obtenidos con los diferentes algoritmos (inciso a) y las normas de error obtenidas con los diferentes algoritmos (inciso b); cabe apuntar aquí, que al ser las simulaciones en punto flotante la diferencia entre los coeficientes obtenidos con los diferentes algoritmos es muy pequeña y esta se puede detectar al rededor de la posición 15 decimal (ver tabla 5.6).

Coeficientes Calculados en la Igualación de Canal con los algoritmos			
Algoritmo	Coeficientes		
RLS	1.02203372851371	0.00879953635340	0.52620725408429
FK	1.02203372851371	0.00879953635338	0.52620725408432
FTF	1.02203372851397	0.00879953635361	0.52620725408458
FAEST	1.02203372851437	0.00879953635394	0.52620725408504
GFTF	1.02203372851397	0.00879953635361	0.52620725408459
NFTF	1.02203372851512	0.00879953635457	0.52620725408589
CFK	1.02203372851372	0.00879953635338	0.52620725408431
CFTF	1.02203372851371	0.00879953635340	0.52620725408429
SFTF	1.02203372851371	0.00879953635340	0.52620725408429

Tab. 5.6: Resultados de los coeficientes de igualación con los diferentes algoritmos para el canal de fase mínima $1 + 0{,}2z^{-1} - 0{,}6z^{-2}$

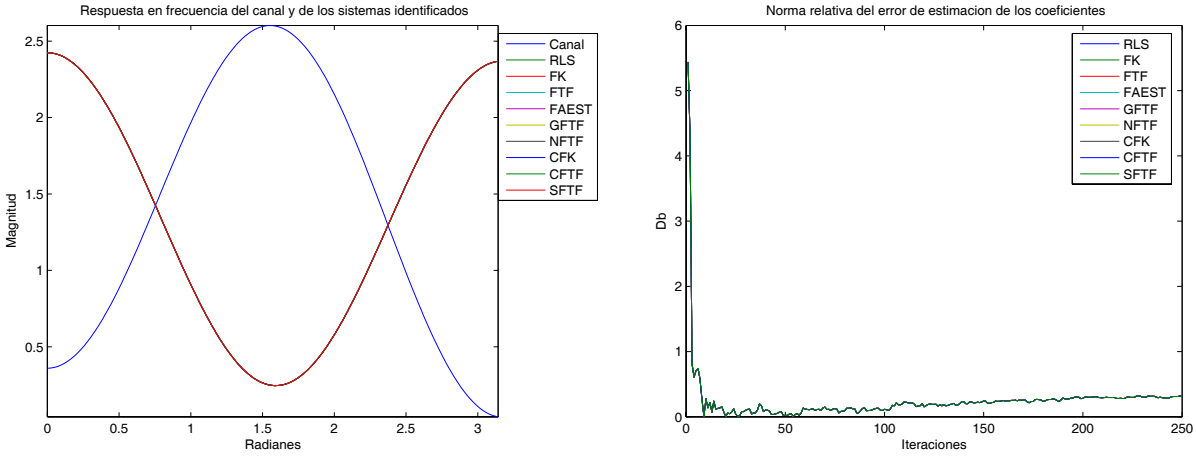


Fig. 5.13: a) Respuesta en frecuencia del canal de fase mínima $1 + 0{,}2z^{-1} - 0{,}6z^{-2}$ y de los filtros igualadores con los diferentes algoritmos. b) Norma relativa del error de la igualación del canal de fase mínima $1 + 0{,}2z^{-1} - 0{,}6z^{-2}$

Para el segundo canal de fase mínima, se obtuvieron los resultados mostrados en la tabla 5.7. Mientras que en la figura 5.14 se ilustran la respuesta en frecuencia de los filtros y las normas relativas de error obtenidas para este canal respectivamente.

Coeficientes Calculados en la Igualación de Canal con los algoritmos			
Algoritmo	Coeficientes		
RLS	1.00482298148070	-0..51556092652548	0.09494942155794
FK	1.00482298148068	-0..51556092652546	0.09494942155793
FTF	1.00482298148050	-0..51556092652559	0.09494942155728
FAEST	1.00482298148084	-0..51556092652541	0.09494942155840
GFTF	1.00482298148070	-0..51556092652548	0.09494942155794
NFTF	1.00482298148072	-0..51556092652547	0.09494942155801
CFK	1.00482298148070	-0..51556092652548	0.09494942155794
CFTF	1.00482298148070	-0..51556092652549	0.09494942155794
SFTF	1.00482298148070	-0..51556092652548	0.09494942155794

Tab. 5.7: Resultados de los coeficientes de igualación con los diferentes algoritmos para el canal de fase mínima $1 + 0{,}51z^{-1} + 0{,}1997z^{-2}$

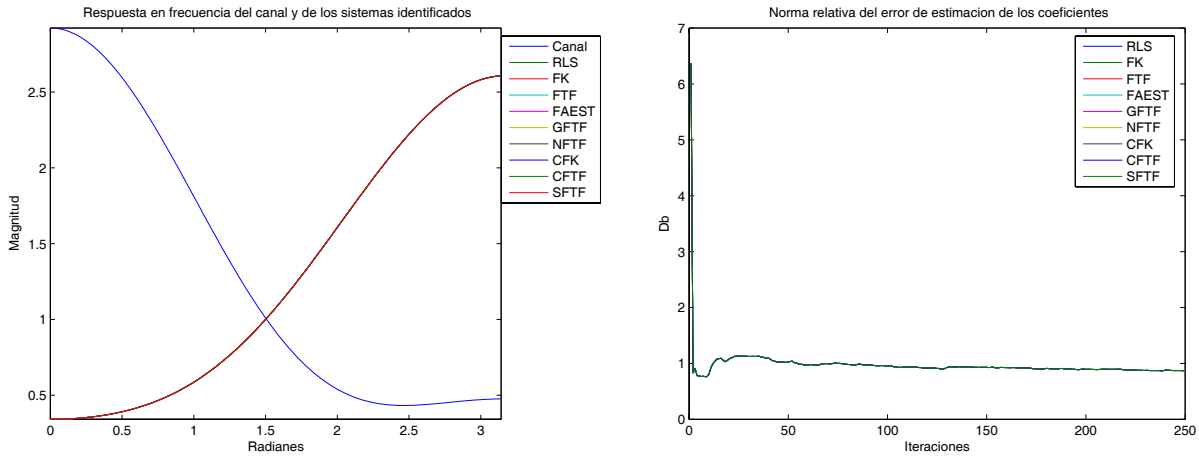


Fig. 5.14: a) Respuesta en frecuencia del canal de fase mínima $1 + 0{,}51z^{-1} + 0{,}1997z^{-2}$ y de los filtros igualadores con los diferentes algoritmos. b) Norma relativa del error de la igualación de canal de fase mínima $1 + 0{,}51z^{-1} + 0{,}1997z^{-2}$

De igual manera para el tercer canal de fase mínima se llevaron a cabo las simulaciones obteniendo los siguientes resultados mostrados en la tabla 5.8 e ilustradas la respuesta en frecuencia del canal y los filtros igualadores en la figura 5.15 (inciso a) y las normas relativas de error (inciso b):

Coeficientes Calculados en la Igualación de Canal con los algoritmos			
Algoritmo	Coeficientes		
RLS	0.98079832452469	-0..51917760667611	0..16242041691476
FK	0.98079832452470	-0..51917760667612	0..16242041691476
FTF	0.98079832452448	-0..51917760667619	0..16242041691424
FAEST	0.98079832452424	-0..51917760667627	0..16242041691363
GFTF	0.98079832452416	-0..51917760667631	0..16242041691341
NFTF	0.98079832452502	-0..51917760667599	0..16242041691563
CFK	0.98079832452469	-0..51917760667612	0..16242041691476
CFTF	0.98079832452468	-0..51917760667611	0..16242041691476
SFTF	0.98079832452469	-0..51917760667611	0..16242041691476

Tab. 5.8: Resultados de los coeficientes de igualación con los diferentes algoritmos para el canal de fase mínima $1 + 0{,}536z^{-1} + 0{,}0718z^{-2}$

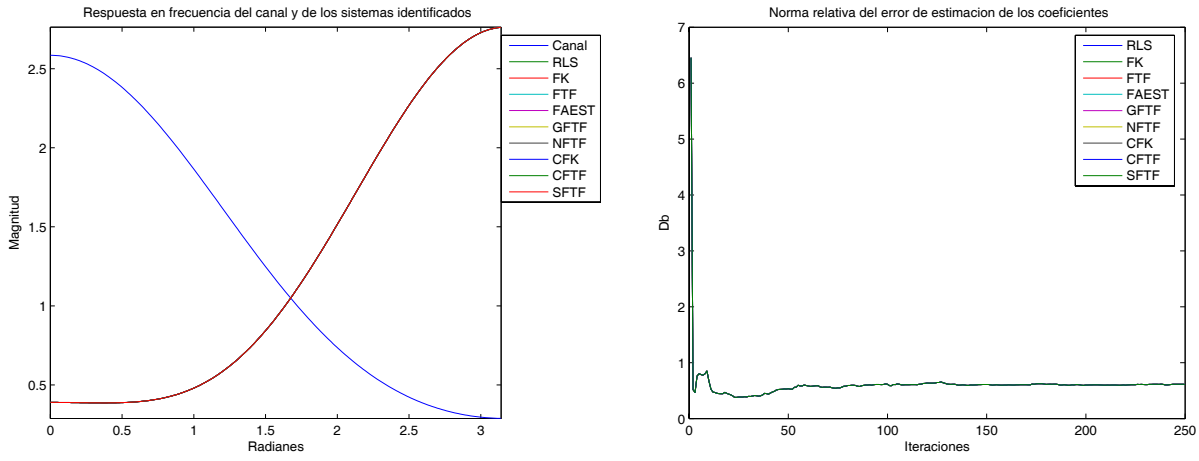


Fig. 5.15: a) Respuesta en frecuencia del canal de fase mínima $1 + 0{,}536z^{-1} + 0{,}0718z^{-2}$ y de los filtros igualadores con los diferentes algoritmos. b) Norma relativa del error de la igualación de canal de fase mínima $1 + 0{,}536z^{-1} + 0{,}0718z^{-2}$

Para el cuarto canal de fase mínima se muestran los resultados de las simulaciones de punto flotante en la tabla 5.9 y se ilustran en la gráfica de respuesta en frecuencia (inciso a) y normas relativas de error en las figuras 5.16.

Coeficientes Calculados en la Igualación de Canal con los algoritmos			
Algoritmo	Coeficientes		
RLS	0.87821211004109	1.03903391706792	0.58748882154702
FK	0.87821211004130	1.03903391706789	0.58748882154679
FTF	0.87821211004210	1.03903391706791	0.58748882154545
FAEST	0.87821211004290	1.03903391706792	0.58748882154419
GFTF	0.87821211004307	1.03903391706792	0.58748882154392
NFTF	0.87821211004520	1.03903391706792	0.58748882154056
CFK	0.87821211004101	1.03903391706792	0.58748882154712
CFTF	0.87821211004118	1.03903391706791	0.58748882154693
SFTF	0.87821211004115	1.03903391706791	0.58748882154696

Tab. 5.9: Resultados de los coeficientes de igualación con los diferentes algoritmos para el canal de fase mínima $1 - 1{,}6z^{-1} + 0{,}95z^{-2}$

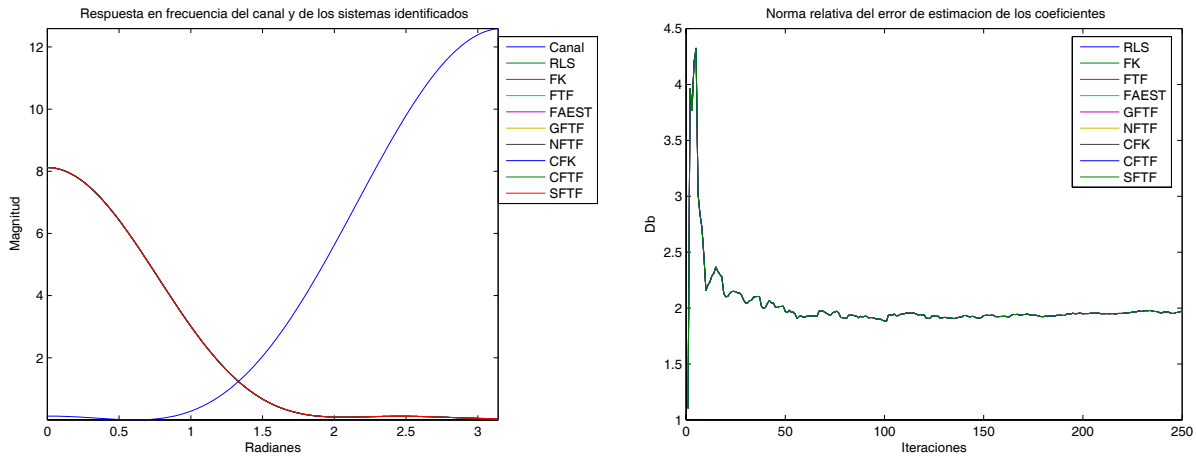


Fig. 5.16: a) Respuesta en frecuencia del canal de fase mínima $1 - 1{,}6z^{-1} + 0{,}95z^{-2}$ y de los filtros igualadores con los diferentes algoritmos. b) Norma relativa del error de la igualación de canal de fase mínima $1 - 1{,}6z^{-1} + 0{,}95z^{-2}$

En la siguiente tabla (5.10) se muestran los resultados de las simulaciones en punto flotante para el quinto canal de fase mínima, y en la figura 5.17 se muestran las gráficas tanto de la respuesta en frecuencia (inciso a) como de la norma relativa del error de los coeficientes (inciso b) obtenidas con esos resultados.

Coeficientes Calculados en la Igualación de Canal con los algoritmos			
Algoritmo	Coeficientes		
RLS	0.62938579657553	0.71307860113309	0.40373594981254
FK	0.62938579657548	0.71307860113291	0.40373594981240
FTF	0.62938579657635	0.71307860113515	0.40373594981500
FAEST	0.62938579657434	0.71307860113007	0.40373594980896
GFTF	0.62938579657547	0.71307860113292	0.40373594981235
NFTF	0.62938579657645	0.71307860113540	0.40373594981530
CFK	0.62938579657577	0.71307860113333	0.40373594981280
CFTF	0.62938579657553	0.71307860113309	0.40373594981254
SFTF	0.62938579657553	0.71307860113309	0.40373594981254

Tab. 5.10: Resultados de los coeficientes de igualación con los diferentes algoritmos para el canal de fase mínima $1 - 1{,}9z^{-1} + 0{,}95z^{-2}$

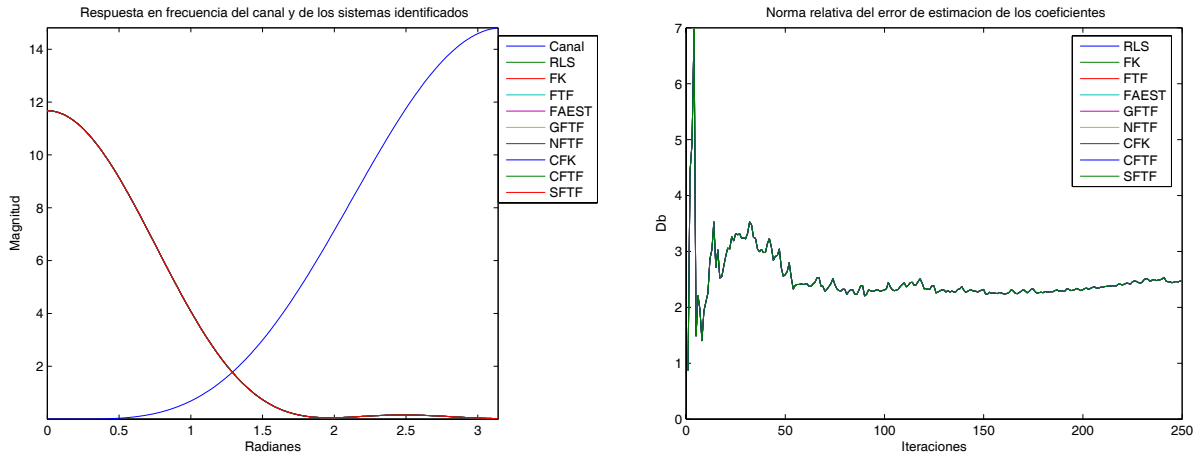


Fig. 5.17: a) Respuesta en frecuencia del canal de fase mínima $1 - 1{,}9z^{-1} + 0{,}95z^{-2}$ y de los filtros igualadores con los diferentes algoritmos. b) Norma relativa del error de la igualación de canal de fase mínima $1 - 1{,}9z^{-1} + 0{,}95z^{-2}$

A continuación se presentan los resultados para los canales de fase no mínima. En la tabla 5.11, se muestran los resultados obtenidos para el primer canal de fase no mínima.

Coeficientes Calculados en la Igualación de Canal con los algoritmos			
Algoritmo	Coeficientes		
RLS	1.02662149079864	-0.97376319768582	0.55044490464024
FK	1.02662149079862	-0.97376319768576	0.55044490464020
FTF	1.02662149078690	-0.97376319767839	0.55044490461663
FAEST	1.02662149080256	-0.97376319768837	0.55044490464811
GFTF	1.02662149079585	-0.97376319768402	0.55044490463460
NFTF	1.02662149078809	-0.97376319767909	0.55044490461908
CFK	1.02662149079868	-0.97376319768571	0.55044490464030
CFTF	1.02662149079864	-0.97376319768583	0.55044490464025
SFTF	1.02662149079864	-0.97376319768583	0.55044490464025

Tab. 5.11: Resultados de los coeficientes de igualación con los diferentes algoritmos para el canal de fase no mínima $0{,}407 + 0{,}825z^{-1} + 0{,}407z^{-2}$

Mientras que en la figura 5.18 a continuación se muestran las gráficas de la respuesta en frecuencia del canal y los filtros igualadores así como las normas relativas de error (incisos a y b respectivamente).

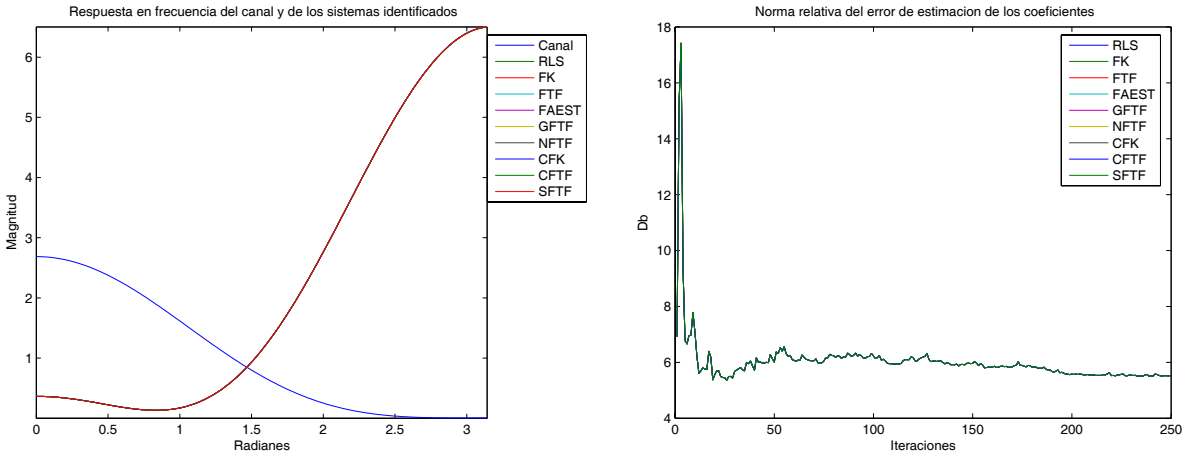


Fig. 5.18: a) Respuesta en frecuencia del canal de fase mínima $0{,}407 + 0{,}825z^{-1} + 0{,}407z^{-2}$ y de los filtros igualadores con los diferentes algoritmos. b) Norma relativa del error de la igualación de canal de fase mínima $0{,}407 + 0{,}825z^{-1} + 0{,}407z^{-2}$

Y para el segundo canal de fase no mínima, se muestran los siguientes resultados en la tabla 5.12.

Coeficientes Calculados en la Igualación de Canal con los algoritmos			
Algoritmo	Coeficientes		
RLS	0.20317643264060	0.13297822231498	0.12061876557871
	0.12914836637633	0.17130417540649	0.11654136130364
	0.15610199307060	0.00406728597202	0.19521166648953
	0.07239493890991		
FK	0.20317643264071	0.13297822231513	0.12061876557889
	0.12914836637645	0.17130417540675	0.11654136130381
	0.15610199307092	0.00406728597227	0.19521166648972
	0.07239493891011		
FTF	0.20317643263949	0.13297822231235	0.12061876557453
	0.12914836637071	0.17130417540016	0.11654136129445
	0.15610199306153	0.00406728596068	0.19521166648057
	0.07239493889712		
FAEST	0.20317643264113	0.13297822231603	0.12061876558045
	0.12914836637883	0.17130417540973	0.11654136130793
	0.15610199307497	0.00406728597741	0.19521166649430
	0.07239493891640		
GFTF	0.20317643263901	0.13297822231137	0.12061876557295
	0.12914836636853	0.17130417539734	0.11654136129066
	0.15610199305761	0.00406728595565	0.19521166647611
	0.07239493889146		
NFTF	0.20317643263721	0.13297822230733	0.12061876556653
	0.12914836635968	0.17130417538692	0.11654136127607
	0.15610199304326	0.00406728593762	0.19521166646116
	0.07239493887035		
CFK	0.20317643264056	0.13297822231491	0.12061876557863
	0.12914836637624	0.17130417540637	0.11654136130348
	0.15610199307042	0.00406728597176	0.19521166648934
	0.07239493890971		
CFTF	0.20317643264061	0.13297822231499	0.12061876557871
	0.12914836637631	0.17130417540649	0.11654136130362
	0.15610199307058	0.00406728597200	0.195211666489511
	0.07239493890991		
SFTF	0.20317643264061	0.13297822231499	0.12061876557871
	0.12914836637631	0.17130417540649	0.11654136130362
	0.15610199307058	0.00406728597200	0.19521166648951
	0.07239493890990		

Tab. 5.12: Resultados de los coeficientes de igualación con los diferentes algoritmos para el canal de fase no mínima $0{,}04 - 0{,}05z^{-1} + 0{,}07z^{-2} - 0{,}21z^{-3} - 0{,}5z^{-4} + 0{,}72z^{-5} + 0{,}36z^{-6} + 0{,}21z^{-8} + 0{,}03z^{-9} + 0{,}07z^{-10}$

Mientras que en las figuras 5.19 y 5.20 se ilustran la respuesta en frecuencia y las normas relativas de error obtenidas para este canal.

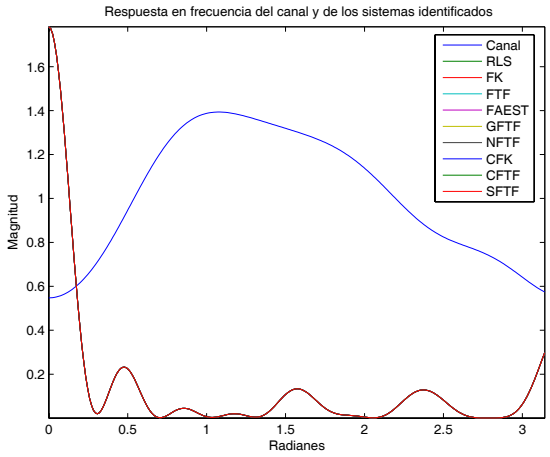


Fig. 5.19: Respuesta en frecuencia del canal de fase mínima $0{,}04 - 0{,}05z^{-1} + 0{,}07z^{-2} - 0{,}21z^{-3} - 0{,}5z^{-4} + 0{,}72z^{-5} + 0{,}36z^{-6} + 0{,}21z^{-8} + 0{,}03z^{-9} + 0{,}07z^{-10}$ y de los filtros igualadores con los diferentes algoritmos

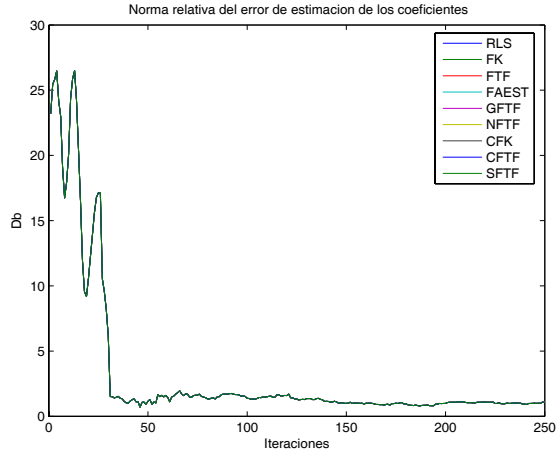


Fig. 5.20: Norma relativa del error de la igualación de canal de fase mínima $0{,}04 - 0{,}05z^{-1} + 0{,}07z^{-2} - 0{,}21z^{-3} - 0{,}5z^{-4} + 0{,}72z^{-5} + 0{,}36z^{-6} + 0{,}21z^{-8} + 0{,}03z^{-9} + 0{,}07z^{-10}$

5.3. Implementaciones en Aritmética de Punto Fijo

5.3.1. Estudio de las dinámicas de los algoritmos.

El estudio de las dinámicas de los algoritmos se llevo a cabo monitoreando los valores máximos y mínimos de las variables obtenidos al realizar las simulaciones en punto flotante de todos los algoritmos con cada uno de los diferentes canales de fase mínima y fase no mínima.

A continuación se muestran los resultados del este estudio de las dinámicas de las variables para los canales de fase mínima empleados.

Estudio de las dinámicas del algoritmo FK.

En las tablas 5.13 y 5.14 se presentan los resultados del estudio de dinámicas para el algoritmo FK al correr las simulaciones con los canales de fase mínima.

	Canal 1		Canal 2		Canal 3	
Variable	Máximo	Mínimo	Máximo	Mínimo	Máximo	Mínimo
α_p^b	63.314	0	57.753	0	56.873	0
α_p^f	63.199	0.001	49.039	0.001	49.101	0.001
a	1	-0.6	1	0.1997	1	0.0718
$\mathbf{B_p}$	0.82793	-0.52043	1.2685	-0.61172	1.2461	-0.668
$\mathbf{k}_p$	1.0788	-0.53275	1.0788	-0.70122	1.0788	-0.65262
d	0.99664	0.0037844	0.99664	0.0037844	0.99664	0.0037844
e	0.9259	-0.25094	0.9259	-0.4708	0.9259	-0.49484
ε_p^b	1.6121	-0.61835	2.8672	0	2.7469	0
e_p^b	1.0005	-0.81236	0.93676	-0.76118	0.99601	-0.79562
e_p^f	1.0922	-0.80501	1.2119	-0.82934	1.236	-0.83609
ε^f	1.0885	-0.79606	0.9259	-0.67889	0.9259	-0.6654
$\mathbf{A_p}$	1.0937	-0.81502	1.3563	-0.64884	1.3385	-0.74785
$\mathbf{m}$	1.0788	-0.61347	1.0788	-0.72166	1.0788	-0.71506
μ	1.0752	-0.36454	1.0762	-0.24831	1.0756	-0.27716
$\hat{w}_p$	1.0579	-0.19837	1.0069	-0.53042	1.0037	-0.5478
x	1.1086	-0.44989	1.5744	0.12861	1.502	0.074691
y	1.2145	-0.24076	1.2105	-0.028857	1.2345	-0.016248

Tab. 5.13: Dinámica de las variables del algoritmo FK para los canales 1,2 y 3 de fase mínima.

	Canal 4		Canal 5	
Variable	Máximo	Mínimo	Máximo	Mínimo
α_p^b	82.349	0	77.383	0
α_p^f	77.808	0.001	68.707	0.001
a	1	-1.6	1	-1.9
$\mathbf{B_p}$	1.0024	-1.1214	0.30554	-1.3868
$\mathbf{k}_p$	1.0788	-0.64812	1.0788	-0.91112
d	0.99664	0.0037844	0.99664	0.0037844
e	1.4885	-0.73132	3.5262	-0.36666
ε_p^b	2.6356	-2.0432	2.5197	-2.6329
e_p^b	0.9553	-1.1048	1.0199	-0.79804
e_p^f	1.1915	-1.0178	1.0039	-1.0732
ε^f	1.1701	-0.99782	0.96911	-0.74393
$\mathbf{A_p}$	1.2574	-1.1389	0.38016	-1.5931
$\mathbf{m}$	1.0788	-0.29814	1.0788	-0.67603
μ	1.0736	-0.75003	1.0745	-0.90812
$\hat{w}_p$	1.6043	0	3.6025	0
x	1.6216	-1.225	1.6064	-1.5118
y	1.415	-0.91451	1.2977	-2.7951

Tab. 5.14: Dinámica de las variables del algoritmo FK para los canales 4 y 5 de fase mínima.

Enseguida en la tabla 5.15 se muestra el número de bits necesario para representar los máximos y mínimos de cada una de las variables del algoritmo FK para los canales de fase mínima. Mientras que en la figura 5.21 se ilustra el comportamiento de dichas variables.

	Canal 1		Canal 2		Canal 3		Canal 4		Canal 5	
Variable	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
α_p^b	7	1	7	1	7	1	8	1	8	1
α_p^f	7	11	7	11	7	11	8	11	8	11
$\mathbf{B_p}$	2	6	2	6	2	6	2	6	3	6
$\mathbf{k}_p$	2	6	2	6	2	6	2	6	2	6
d	2	10	2	10	2	10	2	10	2	10
e	2	6	2	6	2	6	2	6	3	6
ε_p^b	2	6	3	1	3	1	3	6	3	6
e_p^b	2	6	2	6	2	6	2	6	2	6
e_p^f	2	6	2	6	2	6	2	6	2	6
ε^f	2	6	2	6	2	6	2	6	2	6
$\mathbf{A_p}$	2	6	2	6	2	6	2	6	3	6
$\mathbf{m}$	2	6	2	6	2	6	2	6	2	6
μ	2	6	2	6	2	6	2	6	2	6
$\hat{w}_p$	2	7	2	6	2	6	2	1	3	1
x	2	6	2	4	2	5	2	6	2	6
y	2	6	2	8	2	9	2	6	2	6

Tab. 5.15: Número de bits necesarios para representar los máximos de la dinámica de las variables del algoritmo FK para los canales de fase mínima.

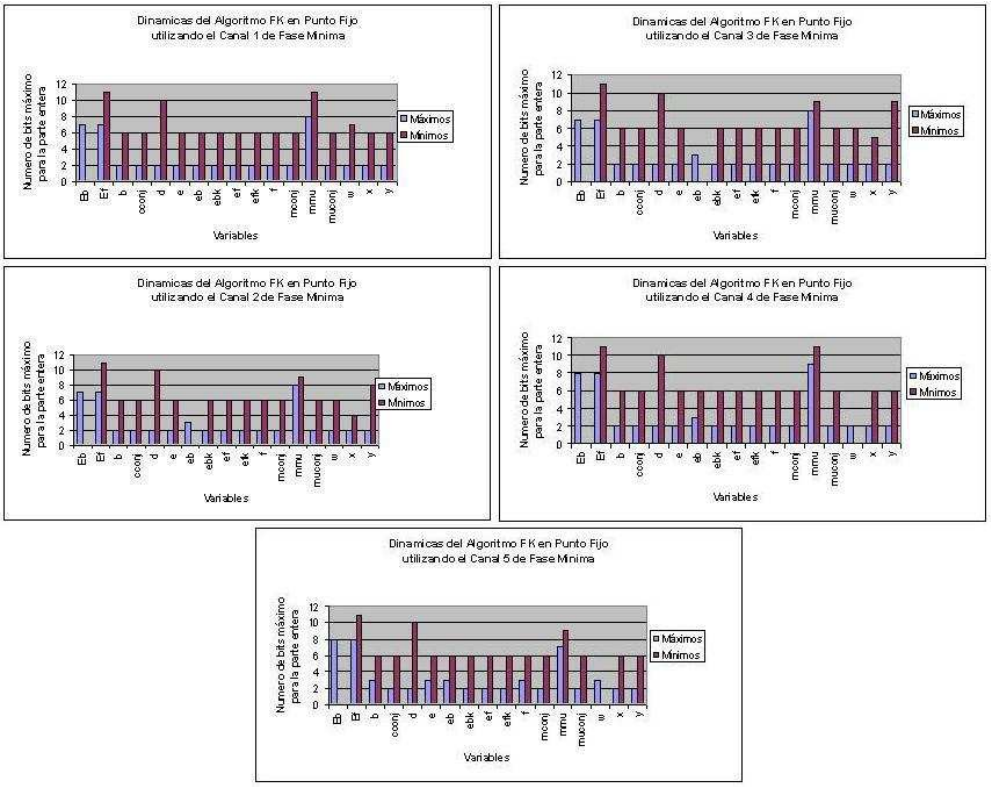


Fig. 5.21: Gráfica del número de bits necesarios para las dinámicas de los canales de fase mínima.

En la tabla 5.16 se muestran los valores máximos y mínimos de las variables del algoritmo FK para los canales de fase no mínima, y en la tabla 5.17 se muestran el número de bits para la representación de los valores máximos y mínimos de las variables del algoritmo.

	Canal 6		Canal 7	
Variable	Máximo	Mínimo	Máximo	Mínimo
α_p^b	21.767	0	47.21	0
α_p^f	17.088	0.001	40.488	0.001
a	0.825	0	0.72	-0.5
$\mathbf{B_p}$	1.8068	-1.636	0.96899	-1.0883
$\mathbf{k}_p$	2.6351	-1.4959	15.616	-4.7099
d	0.99664	0.0037844	0.99664	0.0037844
e	2.6489	-2.8783	14.756	-3.1564
ε_p^b	2.9393	0	1.7788	-1.1618
e_p^b	0.65211	-0.46438	1.0453	-0.84599
e_p^f	2.4633	-2.6294	9.2079	-3.4009
ε^f	0.57652	-0.42043	0.68761	-0.68188
$\mathbf{A_p}$	5.4082	-4.7869	4.8336	-9.3349
$\mathbf{m}$	2.6351	-1.8562	15.616	-4.7099
μ	1.8437	-0.29432	1.0005	-0.54455
$\hat{w}_p$	5.8159	-5.2402	16.041	-16.839
x	1.5052	0.092811	1.1587	-0.58128
y	3.6352	-2.075	3.7853	-14.058

Tab. 5.16: Dinámica de las variables del algoritmo FK para los canales 6 y 7 de fase no mínima.

	Canal 6		Canal 7	
Variable	Máximo	Mínimo	Máximo	Mínimo
α_p^b	6	1	7	1
α_p^f	6	11	7	11
$\mathbf{B_p}$	2	6	2	6
$\mathbf{k}_p$	3	6	5	7
d	2	10	2	10
e	3	6	5	6
ε_p^b	3	1	2	6
e_p^b	2	6	2	6
e_p^f	3	6	5	6
ε^f	2	6	2	6
$\mathbf{A_p}$	4	7	4	7
$\mathbf{m}$	3	6	5	7
μ	2	6	2	6
$\hat{w}_p$	4	7	6	8
x	2	5	2	6
y	3	6	3	7

Tab. 5.17: Número de bits necesarios para representar los máximos de la dinámica de las variables del algoritmo FK para los canales de fase no mínima.

En la figura 5.22 se ilustra la dinámica de las variables del algoritmo en términos de el número de bits necesarios para representar los máximos y mínimos empleando la herramienta de punto fijo de matlab.

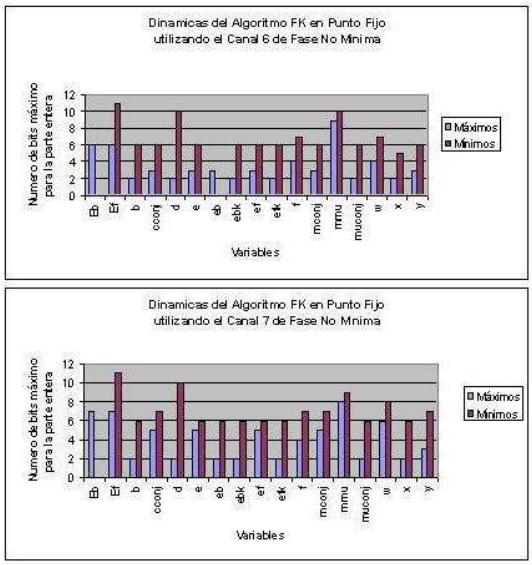


Fig. 5.22: Gráfica del número de bits necesarios para las dinámicas de los canales de fase mínima.

	Canal 1		Canal 2		Canal 3	
Variable	Máximo	Mínimo	Máximo	Mínimo	Máximo	Mínimo
α_p^f	62.342	0.001	48.182	0.001	48.243	0.001
a	1	-0.6	1	0.1997	1	0.0718
$\mathbf{B}_p$	1.297	-1.0608	3.191	-4.1667	2.0868	-2.7187
$\mathbf{k}_p^*$	1.0788	-0.53275	1.0788	-0.70122	1.0788	-0.65262
$\mathbf{k}_{pi}^*$	0.9777	-0.97678	3.0592	-4.013	1.7606	-2.353
d	0.99664	0.0037844	0.99664	0.0037844	0.99664	0.0037844
d2	1.0788	-1.075	1.0788	-1.4075	1.0788	-1.4354
e	0.9259	-0.25094	0.9259	-0.4708	0.9259	-0.49484
e_p^b	0.99831	-0.8144	0.93815	-0.76493	0.99677	-0.7998
e_p^f	1.0922	-0.80501	1.2119	-0.82934	1.236	-0.83609
ε_p^f	1.0885	-0.79606	0.74627	-0.67889	0.78256	-0.6654
$\mathbf{A}_p$	1.0937	-0.81502	1.3563	-0.64884	1.3385	-0.74785
$\mathbf{m}^*$	1.537	-1.5325	5.2104	-3.9921	4.769	-3.5538
μ^*	1.7036	-0.45529	1.7753	-0.22795	2.2581	-0.28111
r	1	0	1	0	1	0
$\hat{\mathbf{w}}_p$	1.0579	-0.19837	1.0069	-0.53042	1.0037	-0.5478
x	1.1086	-0.44989	1.5744	0.12861	1.502	0.074691
y	1.2145	-0.24076	1.2105	-0.028857	1.2345	-0.016248

Tab. 5.18: Dinámica de las variables del algoritmo CFK para los canales 1,2 y 3 de fase mínima.

	Canal 4		Canal 5	
Variable	Máximo	Mínimo	Máximo	Mínimo
α_p^f	76.951	0.001	67.849	0.001
a	1	-1.6	1	-1.9
$\mathbf{B}_p$	1.1048	-1.2436	1.3159	-3.7343
$\mathbf{k}_p^*$	1.0788	-0.64812	1.0788	-0.91112
$\mathbf{k}_{pi}^*$	1.0035	-1.0335	1.5919	-4.1448
d	0.99664	0.0037844	0.99664	0.0037844
d2	1.0788	-1.3059	1.2434	0
e	1.4885	-0.73132	3.5262	-0.36666
e_p^b	0.95632	-1.1034	1.0196	-1.1399
e_p^f	1.1915	-1.0178	1.0039	-1.0732
ε_p^f	1.1701	-0.99782	0.96911	-0.74393
$\mathbf{A}_p$	1.2574	-1.1389	0.38016	-1.5931
$\mathbf{m}^*$	1.7606	-2.1368	1.5968	-3.3554
μ^*	1.156	-2.2715	1.064	-0.78877
r	1	0	1	0
$\hat{w}_p$	1.6043	0	3.6025	0
x	1.6216	-1.225	1.6064	-1.5118
y	1.415	-0.91451	1.2977	-2.7951

Tab. 5.19: Dinámica de las variables del algoritmo CFK para los canales 4 y 5 de fase mínima.

En la tabla 5.20 se muestra el número de bits para la representación de los valores máximos y mínimos obtenidos en el estudio de las dinámicas. Mientras que en la figura 5.23 se ilustran los valores del número de bits necesarios para la adecuada representación de los valores de las variables del algoritmo al emplear la herramienta de punto fijo de matlab.

	Canal 1		Canal 2		Canal 3		Canal 4		Canal 5	
Variable	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
α_p^f	7	11	7	11	7	11	8	11	8	11
$\mathbf{B}_p$	2	6	3	6	3	6	2	6	2	6
$\mathbf{k}_p^*$	2	6	2	6	2	6	2	6	2	6
$\mathbf{k}_{pi}^*$	2	6	3	6	2	6	2	6	2	6
d	2	10	2	10	2	10	2	10	2	10
d2	2	6	2	6	2	6	2	6	2	1
e	2	6	2	6	2	6	2	6	3	6
e_p^b	2	6	2	6	2	6	2	6	2	6
e_p^f	2	6	2	6	2	6	2	6	2	6
ε_p^f	2	6	2	6	2	6	2	6	2	6
$\mathbf{A}_p$	2	6	2	6	2	6	2	6	3	6
$\mathbf{m}^*$	2	6	4	6	4	6	2	6	2	6
μ^*	2	6	2	7	3	6	2	6	2	6
r	1	1	1	1	1	1	1	1	1	1
$\hat{w}_p$	2	7	2	6	2	6	2	1	3	1
x	2	6	2	4	2	5	2	6	2	6
y	2	6	2	8	2	9	2	6	2	6

Tab. 5.20: Número de bits necesarios para representar los máximos de la dinámica de las variables del algoritmo CFK para los canales de fase mínima.

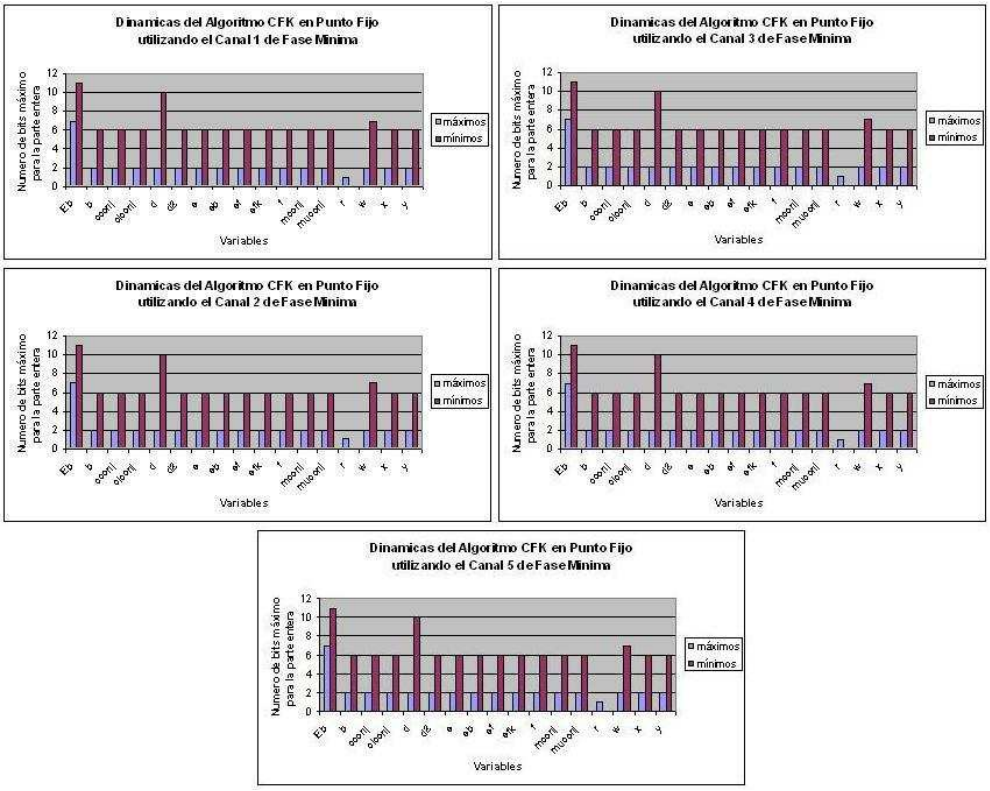


Fig. 5.23: Gráfica del número de bits necesarios para las dinámicas de los canales de fase mínima.

Ahora en la tabla 5.21 se muestran los valores obtenidos para los máximos y mínimos de las variables del algoritmo CFK al emplear los canales de fase no mínima.

	Canal 6		Canal 7	
Variable	Máximo	Mínimo	Máximo	Mínimo
α_p^f	16.946	0.001	40.487	0.001
a	0.825	0	0.72	-0.5
$\mathbf{B}_p$	1.8817	-1.8259	0.9691	-1.0884
$\mathbf{k}_p^*$	2.6351	-1.4959	15.616	-4.7099
$\mathbf{k}_{pi}^*$	1.8817	-1.7294	13.971	-6.303
d	0.99664	0.0037844	0.99664	0.0037844
d2	9.6892	-9.8461	15.616	-7.6245
e	2.6489	-2.8783	14.756	-3.1564
e_p^b	0.6524	-0.46476	1.0453	-0.846
e_p^f	2.4633	-2.6294	9.2079	-3.4009
e_p^f	0.57652	-0.42043	0.68761	-0.68188
$\mathbf{A}_p$	5.4082	-4.7869	4.8336	-9.3349
$\mathbf{m}^*$	2.0054	-2.1637	13.971	-6.303
μ^*	1.9373	-0.29419	1.0005	-0.54454
r	1	0	1	0
$\hat{w}_p$	5.8159	-5.2402	16.041	-16.839
x	1.5052	0.092811	1.1587	-0.58128
y	3.6352	-2.075	3.7853	-14.058

Tab. 5.21: Dinámica de las variables del algoritmo CFK para los canales 6 y 7 de fase no mínima.

Mientras que en la tabla 5.22 se muestran el número de bits necesarios para la representación de los valores máximos y mínimos de las variables del algoritmo mostrados en la tabla anterior (tabla 5.21). Y en la figura 5.24 se ilustra el número de bits antes mencionado.

	Canal 6		Canal 7	
Variable	Max	Min	Max	Min
α_p^f	6	11	7	11
$\mathbf{B}_p$	2	6	2	6
$\mathbf{k}_p^*$	3	6	5	7
$\mathbf{k}_{pi}^*$	2	6	5	7
d	2	10	2	10
d2	5	7	5	7
e	3	6	5	6
e_p^b	2	6	2	6
e_p^f	3	6	5	6
ε_p^f	2	6	2	6
$\mathbf{A}_p$	4	7	4	7
$\mathbf{m}^*$	3	6	5	7
μ^*	2	6	2	6
r	1	1	1	1
$\hat{w}_p$	4	7	6	8
x	2	5	2	6
y	3	6	3	7

Tab. 5.22: Número de bits necesarios para representar los máximos de la dinámica de las variables del algoritmo CFK para los canales de fase no mínima.

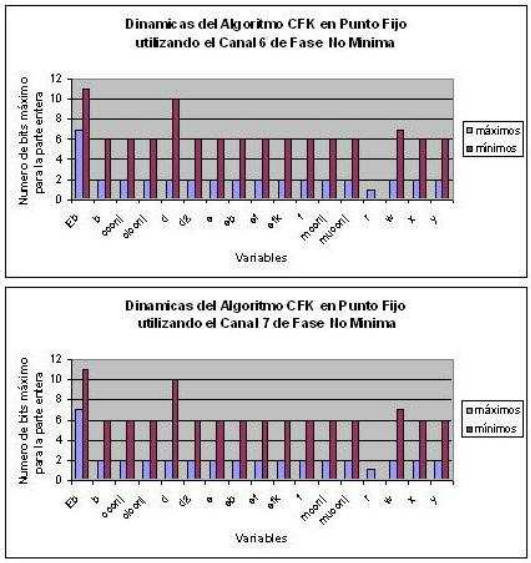


Fig. 5.24: Gráfica del número de bits necesarios para las dinámicas de los canales de fase mínima.

Estudio de las dinámicas del algoritmo SFTF.

Por último para el algoritmo SFTF se muestran los valores máximos y mínimos de las variables obtenidos para los canales de fase mínima; en la tabla 5.23 se muestran los valores para los primeros tres canales de fase mínima mientras que en la tabla 5.24 se muestran los valores para los otros dos canales de fase mínima.

	Canal 1		Canal 2		Canal 3	
Variable	Máximo	Mínimo	Máximo	Mínimo	Máximo	Mínimo
α_p^b	62.823	0.001	48.736	0.001	48.847	0.001
α_p^f	63.199	0.001	49.039	0.001	49.101	0.001
a	1	-0.6	1	0.1997	1	0.0718
$\mathbf{B}_p$	0.82793	-0.52043	1.2685	-0.61172	1.2461	-0.668
$\mathbf{k}_p$	925.9	-2.0577	925.9	-1.1339	925.9	-1.5607
d	0.99664	0.0037844	0.99664	0.0037844	0.99664	0.0037844
e	0.9259	-0.25094	0.9259	-0.4708	0.9259	-0.49484
e_p^b	1.0005	-0.81236	0.93676	-0.76118	0.99601	-0.79562
e_p^β	1.0005	-0.81236	0.93676	-0.76118	0.99601	-0.79562
e_p^ε	1.0005	-0.81236	0.93676	-0.76118	0.99601	-0.79562
e_p^f	1.0922	-0.80501	1.2119	-0.82934	1.236	-0.83609
ε^f	1.0885	-0.79606	0.9259	-0.67889	0.9259	-0.6654
e_p^γ	1.0005	-0.81236	0.93676	-0.76118	0.99601	-0.79562
$\mathbf{A}_p$	1.0937	-0.81502	1.3563	-0.64884	1.3385	-0.74785
γ	1	0.0011629	1	0.0011625	1	0.0011621
γ_+	0.99924	0	0.99855	0	0.99881	0
$\mathbf{m}$	925.9	-2.6511	925.9	-1.56	925.9	-1.7321
μ	925.9	-0.63091	925.9	-0.42809	925.9	-0.45857
$\hat{w}_p$	1.0579	-0.19837	1.0069	-0.53042	1.0037	-0.5478
x	1.1086	-0.44989	1.5744	0.12861	1.502	0.074691
ξ	1.6626E-14	-4.1744E-14	1.8319E-15	-2.1788E-15	9.0372E-14	-1.7103E-13
y	1.2145	-0.24076	1.2105	-0.028857	1.2345	-0.016248

Tab. 5.23: Dinámica de las variables del algoritmo SFTF para los canales 1,2 y 3 de fase mínima.

	Canal 4		Canal 5	
Variable	Máximo	Mínimo	Máximo	Mínimo
α_p^b	76.933	0.001	67.941	0.001
α_p^f	77.808	0.001	68.707	0.001
a	1	-1.6	1	-1.9
$\mathbf{B}_p$	1.0024	-1.1214	0.30554	-1.3868
$\mathbf{k}_p$	925.9	-3.3168	925.9	-4.0465
d	0.99664	0.0037844	0.99664	0.0037844
e	1.4885	-0.73132	3.5262	-0.36666
e_p^b	0.9553	-1.1048	1.0199	-0.79804
e_p^β	0.9553	-1.1048	1.0199	-0.79804
e_p^ε	0.9553	-1.1048	1.0199	-0.79804
e_p^f	1.1915	-1.0178	1.0039	-1.0732
ε^f	1.1701	-0.99782	0.96911	-0.74393
e_p^γ	0.9553	-1.1048	1.0199	-0.79804
$\mathbf{A}_p$	1.2574	-1.1389	0.38016	-1.5931
γ	1	0.0011617	1	0.0011605
γ_+	0.99765	0	0.9978	0
$\mathbf{m}$	925.9	-3.3168	925.9	-4.0465
μ	925.9	-4.0711	925.9	-4.113
$\hat{w}_p$	1.6043	0	3.6025	0
x	1.6216	-1.225	1.6064	-1.5118
ξ	1.4794E-13	-1.4133E-13	2.971E-13	-2.7783E-13
y	1.415	-0.91451	1.2977	-2.7951

Tab. 5.24: Dinámica de las variables del algoritmo SFTF para los canales 4 y 5 de fase mínima.

En la tabla 5.25 se muestran el número de bits necesario para la representación de los valores máximos y mínimos obtenidos en el estudio de las dinámicas del algoritmo SFTF para los canales de fase mínima ilustrados anteriormente en las tablas 5.23 y 5.24.

	Canal 1		Canal 2		Canal 3		Canal 4		Canal 5	
Variable	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
α_p^b	7	11	7	11	7	11	8	11	8	11
α_p^f	7	11	7	11	7	11	8	11	8	11
$\mathbf{B}_p$	2	6	2	6	2	6	2	6	3	6
$\mathbf{k}_p$	11	6	11	6	11	6	11	6	11	6
d	2	10	2	10	2	10	2	10	2	10
e	2	6	2	6	2	6	2	6	3	6
e_p^b	2	6	2	6	2	6	2	6	2	6
e_p^β	2	6	2	6	2	6	2	6	2	6
e_p^ε	2	6	2	6	2	6	2	6	2	6
e_p^f	2	6	2	6	2	6	2	6	2	6
ε^f	2	6	2	6	2	6	2	6	2	6
e_p^γ	2	6	2	6	2	6	2	6	2	6
$\mathbf{A}_p$	2	6	2	6	2	6	2	6	3	6
γ	1	11	1	11	1	11	1	11	1	11
γ_+	2	1	2	1	2	1	2	1	2	1
$\mathbf{m}$	11	6	11	6	11	6	11	6	11	6
μ	11	6	11	6	11	6	11	6	11	6
$\hat{w}_p$	2	7	2	6	2	6	2	1	3	1
x	2	6	2	4	2	5	2	6	2	6
ξ	1	1	1	1	1	1	1	1	1	1
y	2	6	2	8	2	9	2	6	2	6

Tab. 5.25: Número de bits necesarios para representar los máximos de la dinámica de las variables del algoritmo SFTF para los canales de fase mínima.

Mientras que en la figura 5.25 se ilustra el número de bits necesario para la representación de los valores de las variables para cada uno de los canales de fase mínima.

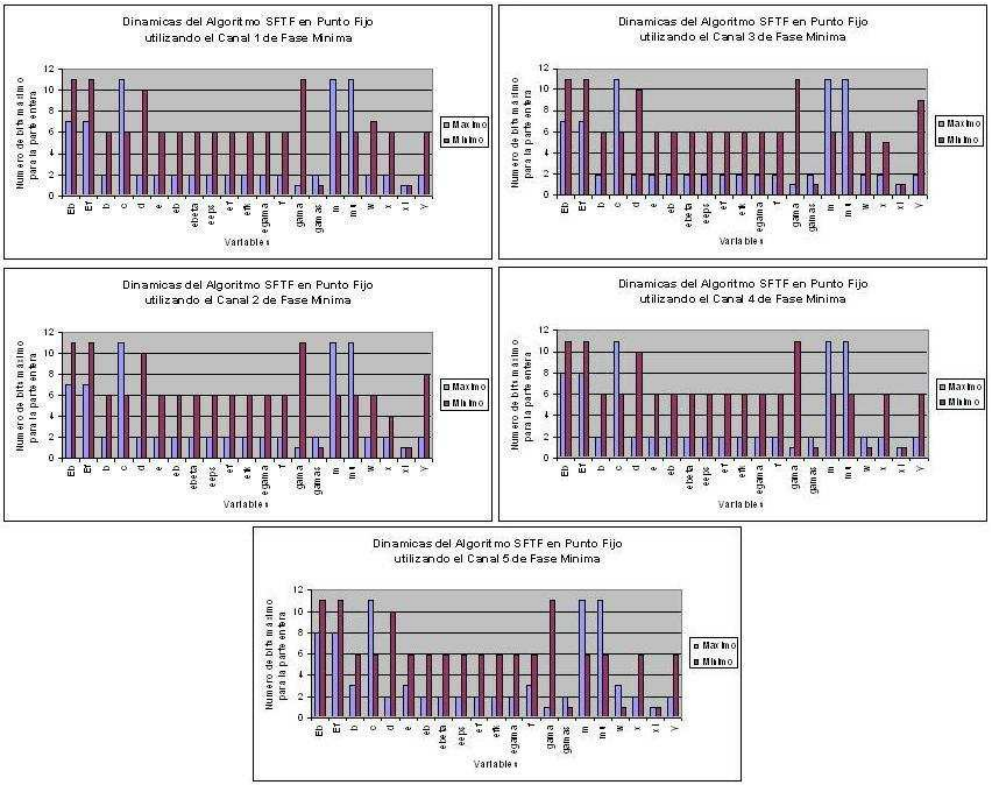


Fig. 5.25: Gráfica del número de bits necesarios para las dinámicas de los canales de fase mínima.

A continuación los valores de los máximos y mínimos de las variables al emplear los canales de fase no mínima se muestran en la tabla 5.26, mientras que en la tabla 5.27 se muestran el número de bits necesarios para la representación de los valores de las variables.

Por otro lado en la figura 5.26 se ilustra, para cada canal, este número de bits necesario para la representación de los valores de las variables del algoritmo al emplear los canales de fase no mínima.

	Canal 6		Canal 7	
Variable	Máximo	Mínimo	Máximo	Mínimo
α_p^b	17.011	0.001	38.682	0.001
α_p^f	17.088	0.001	40.488	0.001
a	0.825	0	0.72	-0.5
$\mathbf{B}_p$	1.8068	-1.636	0.96899	-1.0883
$\mathbf{k}_p$	376.84	-99.442	662.12	-476.58
d	0.99664	0.0037844	0.99664	0.0037844
e	2.6489	-2.8783	14.756	-3.1564
e_p^b	0.65211	-0.46438	1.0453	-0.84599
e_p^β	0.65211	-0.46438	1.0453	-0.84599
e_p^ε	0.65211	-0.46438	1.0453	-0.84599
e_p^f	2.4633	-2.6294	9.2079	-3.4009
ε^f	0.57652	-0.42043	0.68761	-0.68188
e_p^γ	0.65211	-0.46438	1.0453	-0.84599
$\mathbf{A}_p$	5.4082	-4.7869	4.8336	-9.3349
γ	1	0.0048311	1	0.0011938
γ_+	0.99758	0	0.96931	0
$\mathbf{m}$	376.84	-133.65	658.5	-476.58
μ	376.84	-0.39422	613.85	-437.9
$\hat{w}_p$	5.8159	-5.2402	16.041	-16.839
x	1.5052	0.092811	1.1587	-0.58128
ξ	1.1878E-13	-9.9198E-14	1.0317E-12	-9.1388E-13
y	3.6352	-2.075	3.7853	-14.058

Tab. 5.26: Dinámica de las variables del algoritmo SFTF para los canales 6 y 7 de fase no mínima.

	Canal 6		Canal 7	
Variable	Max	Min	Max	Min
α_p^b	6	11	7	11
α_p^f	6	11	7	11
$\mathbf{B}_p$	2	6	2	6
$\mathbf{k}_p$	10	10	11	11
d	2	10	2	10
e	3	6	5	6
e_p^b	2	6	2	6
e_p^β	2	6	2	6
e_p^ε	2	6	2	6
e_p^f	3	6	5	6
ε^f	2	6	2	6
e_p^γ	2	6	2	6
$\mathbf{A}_p$	4	7	4	7
γ	1	9	1	11
γ_+	2	1	2	1
$\mathbf{m}$	10	10	11	11
μ	10	6	11	11
$\hat{w}_p$	4	7	6	8
x	2	5	2	6
ξ	1	1	1	1
y	3	6	3	7

Tab. 5.27: Número de bits necesarios para representar los máximos de la dinámica de las variables del algoritmo SFTF para los canales de fase no mínima.

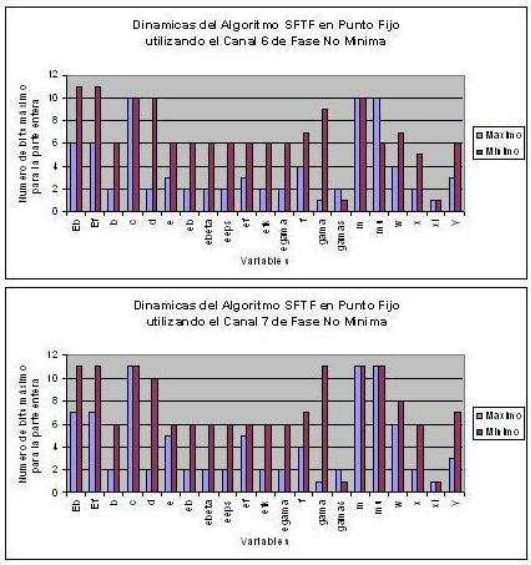


Fig. 5.26: Gráfica del número de bits necesarios para las dinámicas de los canales de fase no mínima.

5.3.2. Estimación de Parámetros

En esta sección se emplean los algoritmos rápidos ya descritos para atacar el problema de la identificación de sistemas, la cual en el area de sistemas es considerada como una aplicación del modelado de sistemas.

Cabe mencionar que para efectos de estas simulaciones la señal utilizada fue la señal resultante de aplicar un ruido blanco gaussiano a un sistema con dos polos ubicados en $p_1 = e^{\pi/4}$ y $p_3 = e^{3\pi/4}$ y sus complejos conjugados $p_2 = e^{-\pi/4}$ y $p_4 = e^{-3\pi/4}$, el parámetro lambda se mantuvo $\lambda = 1$, el parámetro $\delta = 0{,}0001$ y se consideraron condiciones iniciales igual a cero.

Coeficientes Calculados en la Identificación de Sistemas con los algoritmos RLS y RLS Rapidos	
Teóricos	1.00000000000000 - 0.00000000000000 + 0 - 0.00000000000000i - 0.00000000000000 + 1.00000000000000
RLS	1.00000000000000+0.00000000000000i - 0.11943632519157+0.00000000000001i - 0.10143012138492-0.00000000000000i - 0.06047809512611-0.00000000000002i + 1.08674775534886+0.00000000000001i
FKRLS	1.00000000000000+0.00000000000000i - 0.11943632513751+0.00000000000000i - 0.10143012120827+0.00000000000000i - 0.06047809503563-0.00000000000003i + 1.08674775550411+0.00000000000001i
FTFRLS	1.00000000000000+0.00000000000000i - 0.11943634446242+0.00000000004460i - 0.10143008353374-0.00000000002203i - 0.06047813464943-0.00000000002063i + 1.08674777329745+0.00000000005528i
FAEST	1.00000000000000+0.00000000000000i - 1.70121561388336-0.00000000000001i + 1.74377767482325+0.00000000000001i - 1.73316973767450+0.00000000000003i + 1.81788090608255-0.00000000000010i
GFTF	1.00000000000000+0.00000000000000i - 0.11943646235171+0.00000000002701i - 0.10142985222446+0.00000000002312i - 0.06047837623226-0.00000000001603i + 1.08674788307137-0.00000000003422i
NFTF	1.00000000000000+0.00000000000000i - 0.11943632199903+0.00000000005838i - 0.10143012764258-0.00000000003044i - 0.06047808859055-0.00000000001002i + 1.08674775237912+0.00000000003535i
CFK	1.00000000000000-0.00000000010687i - 0.11943632527533-0.00000000010196i - 0.10143012169787-0.00000000004379i - 0.06047809536349+0.00000000009641i + 1.08674775511513+0.00000000000221i
CFTF	1.00000000000000-0.00000000000001i - 0.11943632518519-0.00000000000001i - 0.10143012136887+0.00000000000001i - 0.06047809510330+0.00000000000002i + 1.08674775535416-0.00000000000002i
SFTF	1.00000000000000-0.00000000000001i - 0.11943632518134-0.00000000000001i - 0.10143012135189+0.00000000000001i - 0.06047809510669+0.00000000000002i + 1.08674775537602-0.00000000000002i

Tab. 5.28: Parámetros estimados utilizando predicción RLS empleando aritmética de punto fijo

A continuación se muestra en la figura 5.27 de la densidad espectral de potencia ob-

tenida con los diferentes métodos antes mencionados así como con el algoritmo FKRLS.

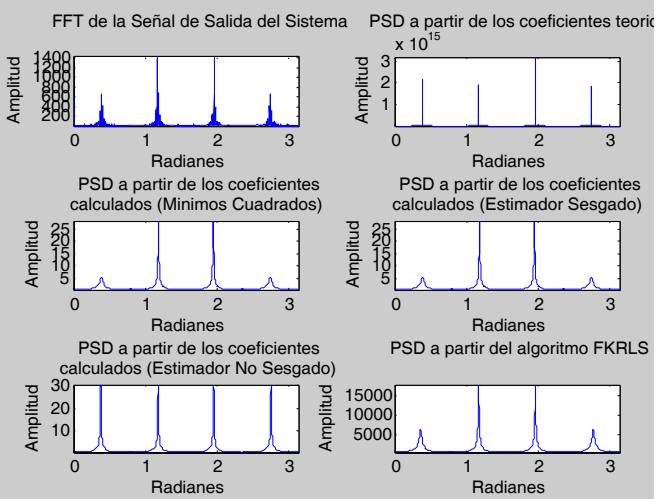


Fig. 5.27: Estimación de la PSD con los diferentes métodos de estimación de parámetros y con el algoritmo FKRLS

De igual manera que para el algoritmo RLS en la figura 5.28 se muestra un espectrograma a partir de los coeficientes estimados para cada iteración.

En la figura 5.29 se muestra la manera en la que converge la estimación de los parámetros para el algoritmo FKRLS y se puede apreciar que tiene una convergencia más rápida que el algoritmo RLS.

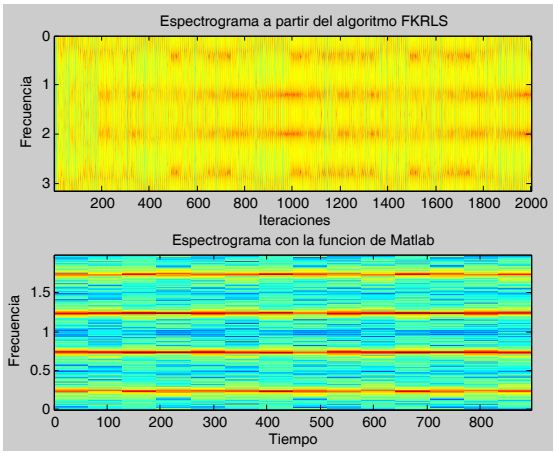


Fig. 5.28: Espectrograma con la estimación de predicción FKRLS y con la función de Matlab

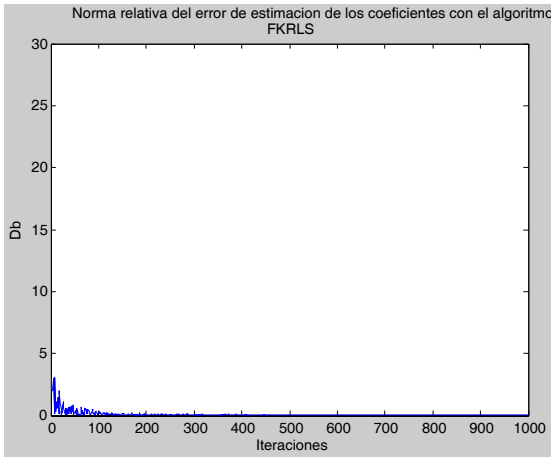


Fig. 5.29: Norma relativa del error de la estimación de parámetros en predicción con el algoritmo FRLS

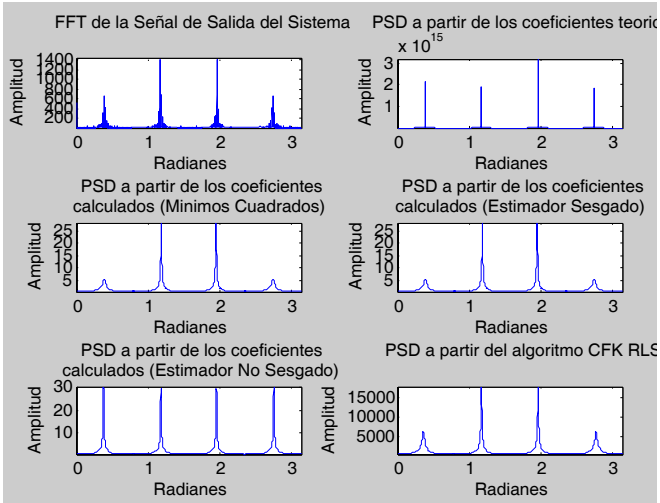


Fig. 5.30: Estimación de la PSD con los diferentes métodos de estimación de parámetros y con el algoritmo CFK RLS

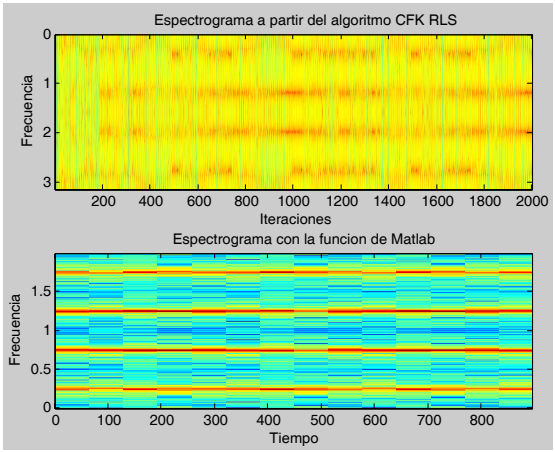


Fig. 5.31: Espectrograma con la estimación de predicción CFK RLS y con la función de Matlab

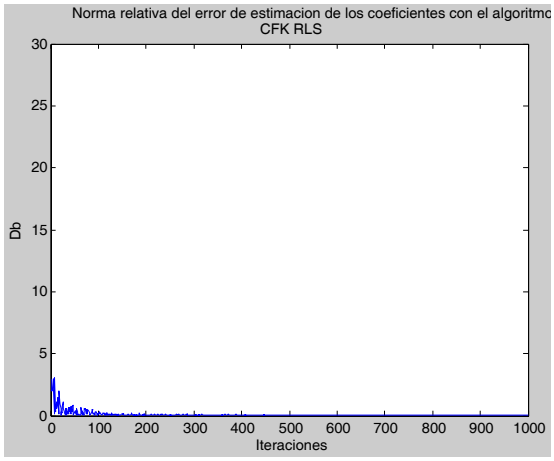


Fig. 5.32: Norma relativa del error de la estimación de parámetros en predicción con el algoritmo CFK RLS

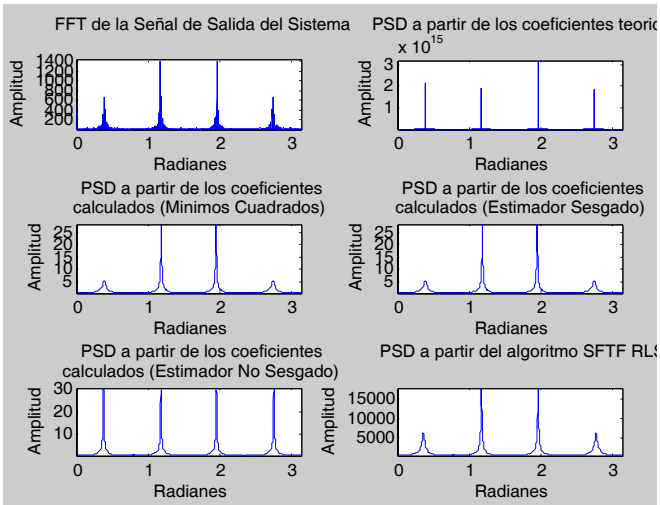


Fig. 5.33: Estimación de la PSD con los diferentes métodos de estimación de parámetros y con el algoritmo SFTF RLS

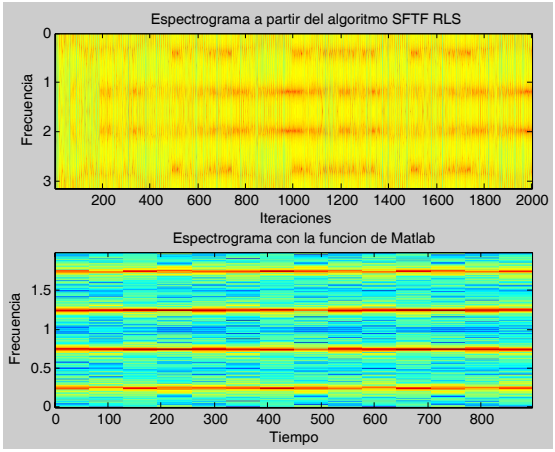


Fig. 5.34: Espectrograma con la estimación de predicción SFTF RLS y con la función de Matlab

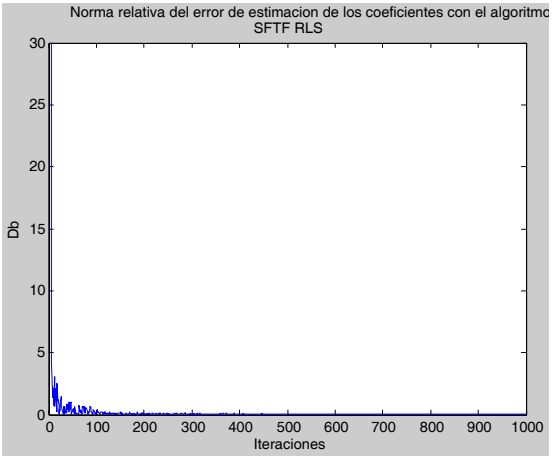


Fig. 5.35: Norma relativa del error de la estimación de parámetros en predicción con el algoritmo SFTF RLS

5.3.3. Identificación de Sistemas

A continuación se presentan las simulaciones en punto fijo de la identificación de los canales empleados utilizando los algoritmos rápidos.

En primera instancia en la tabla 5.29 se muestran los resultados obtenidos para cada uno de los algoritmos para la identificación del primer canal de fase mínima. Por otro lado en la figura 5.36 se muestran a) la norma relativa del error para cada uno de los algoritmos y b) la respuesta en frecuencia tanto del canal como de los sistemas identificados con cada uno de los algoritmos.

Canal	1	.2	-.6
RLS	0.99980329908431	0.19999555777758	-0.59981158096343
FK	0.91075960919261	0.13628481701016	-0.48836286738515
FTF	0.91191820427775	0.13838371075690	-0.49308050982654
FAEST	0.86850887537003	0.17919385433197	-0.48203390836716
GFTF	0.99385053105652	0.19915535300970	-0.59512227587402
NFTF	0.90872728824615	0.13205409049988	-0.48345744609833
CFK	0.99979585409164	0.19998806715012	-0.59981900453568
CFTF	0.99999999254942	0.20000000111759	-0.59999999403954
SFTF	0.99980297312140	0.19999520853162	-0.59981193207204

Tab. 5.29: Coeficientes identificados para el primer canal de fase mínima

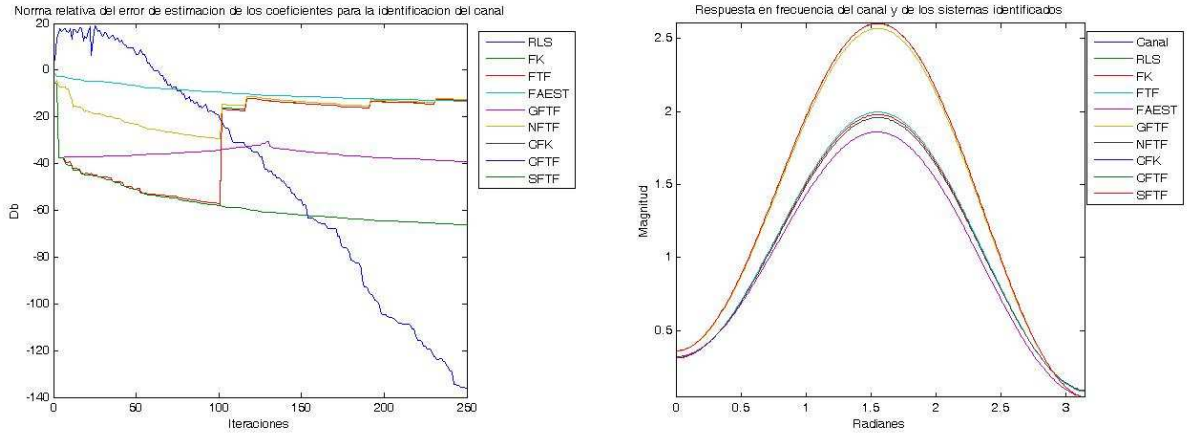


Fig. 5.36: Identificación del primer canal de fase mínima con los algoritmos a) Norma relativa del error b) Respuesta en frecuencia

Para las simulaciones de identificación de sistemas al emplear el cuarto canal de fase mínima se obtuvieron los resultados mostrados en la tabla 5.30 misma que muestra los valores de los coeficientes obtenidos tanto para el algoritmo RLS como para cada uno de los algoritmos rápidos.

Canal	1	-1.6	0.95
RLS	0.99976378120482	-1.59954878687859	0.94977449811995
FK	0.22858106344938	-0.48391117341816	0.27346478775144
FTF	0.99976654537022	-1.59965960867703	0.94956259429455
FAEST	3.73578258417547	3.59014553576708	5.31943822652102
GFTF	0.99352094531059	-1.58595278672874	0.95781696215272
NFTF	0.99976009130478	-1.59955236315727	0.94977062940598
CFK	0.99976342171431	-1.59954923018813	0.94977410510182
CFTF	1.02064808085561	-1.59713906235993	0.93282430619001
SFTF	0.99976363405585	-1.59954900294542	0.94977432303131

Tab. 5.30: Coeficientes identificados para el cuarto canal de fase mínima

Al igual que para los canales anteriores, en la figura 5.37 se muestran la curva de desempeño tomando como referencia la norma relativa de estimación de los parámetros en su inciso a, así como la respuesta en frecuencia tanto del canal como de cada uno de los juegos de coeficientes obtenidos para el sistema identificado con los diferentes algoritmos.

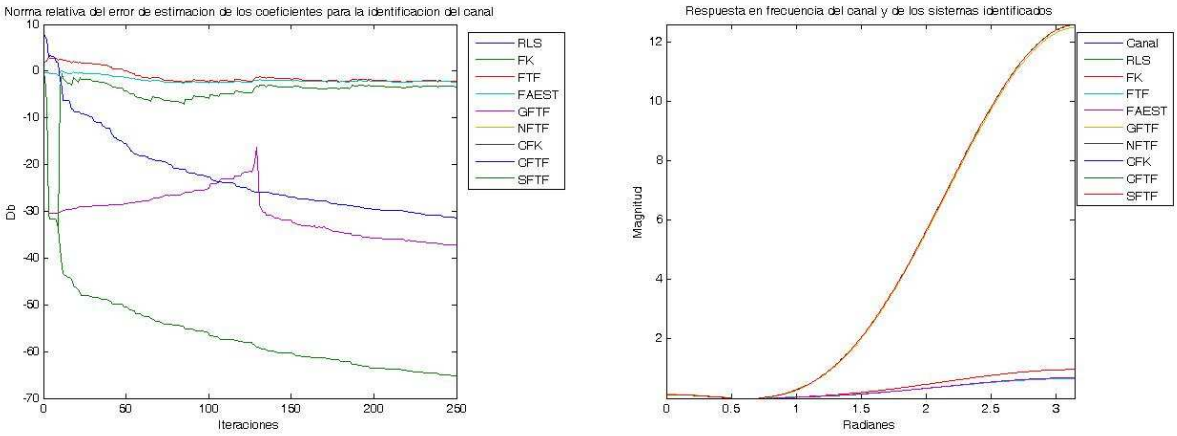


Fig. 5.37: Identificación del cuarto canal de fase mínima con los algoritmos a) Norma relativa del error b) Respuesta en frecuencia

La tabla 5.31 muestra los valores obtenido de los coeficientes en las simulaciones de punto flotante para la identificación del primer canal de fase no mínima para cada uno de los algoritmos

Canal	0.407	0.815	0.407
RLS	0.40702421870083	0.81491690315306	0.40702407434583
FK	0.40702358633280	0.81491626892239	0.40702347271144
FTF	0.40704687312245	0.81482650060207	0.40683535672724
FAEST	1.57540380582213	0.29473111405969	-0.49117404036224
GFTF	0.40888869017363	0.81258314847946	0.40441331267357
NFTF	0.40701651573181	0.81490880250931	0.40701586008072
CFK	0.40702186524868	0.81491442024708	0.40702179074287
CFTF	0.41899794898927	0.80830291658640	0.39968814514577
SFTF	0.40701645612717	0.81490892171860	0.40701603889465

Tab. 5.31: Coeficientes identificados para el primer canal de fase no mínima

Y la figura 5.38 ilustra por un lado; el desempeño de los algoritmos en el inciso a) con la norma relativa de error de estimación de los parámetros, y la respuesta en frecuencia de los sistemas o canales identificados en el inciso b).

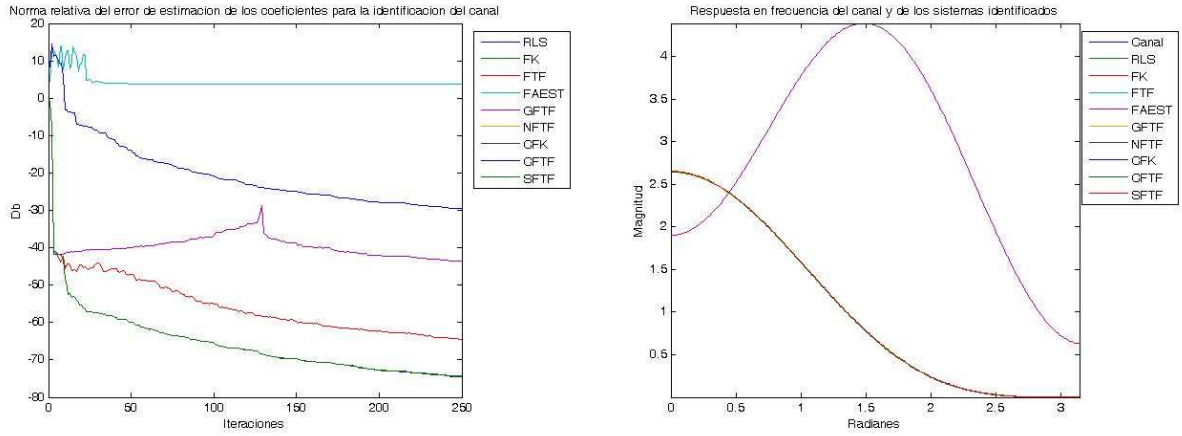


Fig. 5.38: Identificación del primer canal de fase no mínima con los algoritmos a) Norma relativa del error b) Respuesta en frecuencia

5.3.4. Igualación de Canal

A continuación se presentan los resultados de las simulaciones en aritmética de punto fijo para la igualación de canal con todos los algoritmos rápidos de los mínimos cuadrados para cada uno de los canales tanto de fase mínima, de fase no mínima y de fase máxima.

En la tabla 5.32 se presentan los resultados de la igualación del primer canal de fase mínima empleado, así mismo en las figura 5.39 se ilustran la respuesta en frecuencia del canal y de los filtros igualadores obtenidos con los diferentes algoritmos (inciso a) y las normas de error obtenidas con los diferentes algoritmos (inciso b); cabe apuntar aquí, que al ser las simulaciones en punto flotante la diferencia entre los coeficientes obtenidos con los diferentes algoritmos es muy pequeña y ésta se puede detectar alrededor de la posición 15 decimal (ver tabla 5.32).

Coeficientes Calculados en la Igualación de Canal con los algoritmos			
Algoritmo	Coeficientes		
RLS	1.02203372851371	0.00879953635340	0.52620725408429
FK	1.02203372851371	0.00879953635338	0.52620725408432
FTF	1.02203372851397	0.00879953635361	0.52620725408458
FAEST	1.02203372851437	0.00879953635394	0.52620725408504
GFTF	1.02203372851397	0.00879953635361	0.52620725408459
NFTF	1.02203372851512	0.00879953635457	0.52620725408589
CFK	1.02203372851372	0.00879953635338	0.52620725408431
CFTF	1.02203372851371	0.00879953635340	0.52620725408429
SFTF	1.02203372851371	0.00879953635340	0.52620725408429

Tab. 5.32: Resultados de los coeficientes de igualación con los diferentes algoritmos para el canal de fase mínima $1 + 0{,}2z^{-1} - 0{,}6z^{-2}$

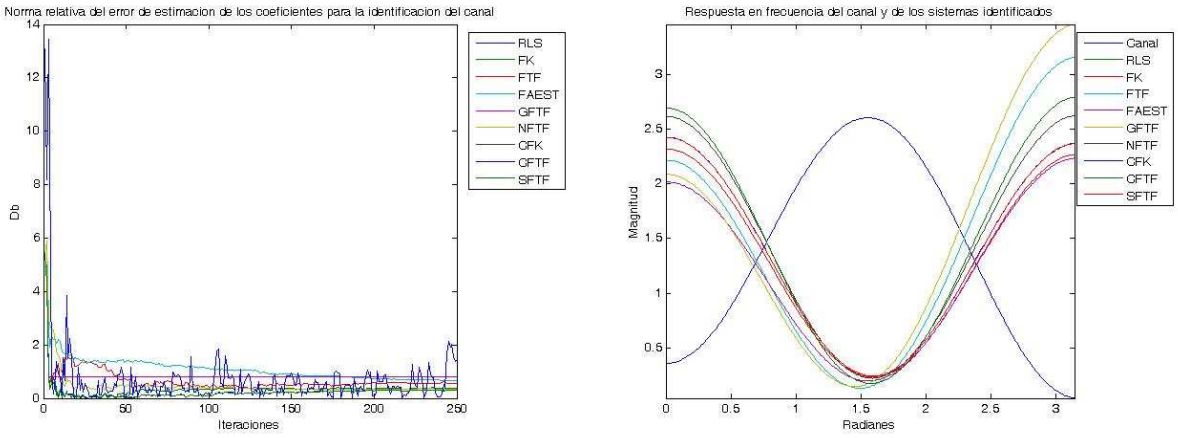


Fig. 5.39: a) Norma relativa del error de la igualación del canal. b) Respuesta en frecuencia del canal de fase mínima $1 + 0{,}2z^{-1} - 0{,}6z^{-2}$ y de los filtros igualadores con los diferentes algoritmos.

Para el segundo canal de fase mínima, se obtuvieron los resultados mostrados en la tabla 5.33. Mientras que en la figura 5.40 se ilustran la respuesta en frecuencia de los filtros y las normas relativas de error obtenidas para este canal respectivamente.

Coeficientes Calculados en la Igualación de Canal con los algoritmos			
Algoritmo	Coeficientes		
RLS	1.00482298148070	-0.51556092652548	0.09494942155794
FK	1.00482298148068	-0.51556092652546	0.09494942155793
FTF	1.00482298148050	-0.51556092652559	0.09494942155728
FAEST	1.00482298148084	-0.51556092652541	0.09494942155840
GFTF	1.00482298148070	-0.51556092652548	0.09494942155794
NFTF	1.00482298148072	-0.51556092652547	0.09494942155801
CFK	1.00482298148070	-0.51556092652548	0.09494942155794
CFTF	1.00482298148070	-0.51556092652549	0.09494942155794
SFTF	1.00482298148070	-0.51556092652548	0.09494942155794

Tab. 5.33: Resultados de los coeficientes de igualación con los diferentes algoritmos para el canal de fase mínima $1 + 0{,}51z^{-1} + 0{,}1997z^{-2}$

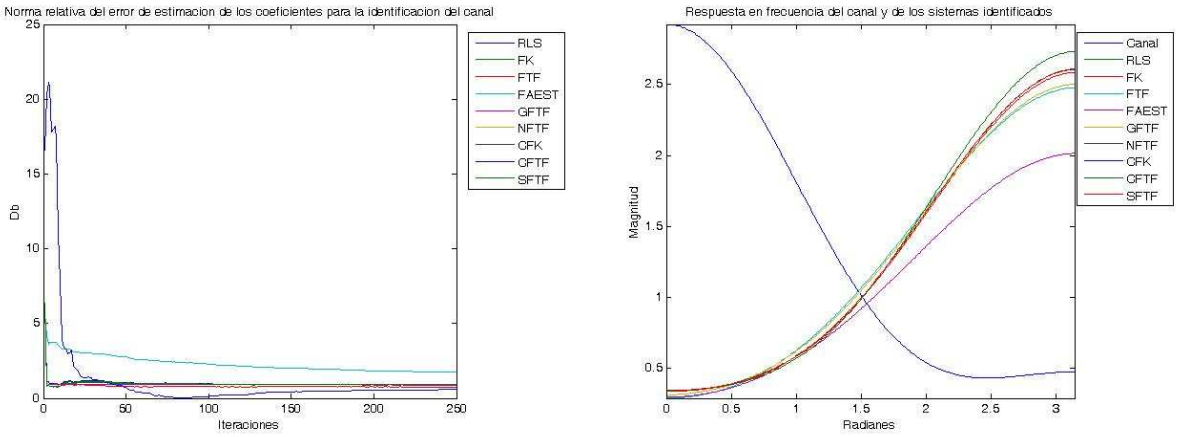


Fig. 5.40: a) Norma relativa del error de la igualación de canal. b) Respuesta en frecuencia del canal de fase mínima $1 + 0{,}51z^{-1} + 0{,}1997z^{-2}$ y de los filtros igualadores con los diferentes algoritmos.

De igual manera para el tercer canal de fase mínima se llevaron a cabo las simulaciones obteniendo los siguientes resultados mostrados en la tabla 5.34 e ilustradas la respuesta en frecuencia del canal y los filtros igualadores en la figura 5.41 (inciso a) y las normas relativas de error (inciso b):

Coeficientes Calculados en la Igualación de Canal con los algoritmos			
Algoritmo	Coeficientes		
RLS	0.98079832452469	-0.51917760667611	0.16242041691476
FK	0.98079832452470	-0.51917760667612	0.16242041691476
FTF	0.98079832452448	-0.51917760667619	0.16242041691424
FAEST	0.98079832452424	-0.51917760667627	0.16242041691363
GFTF	0.98079832452416	-0.51917760667631	0.16242041691341
NFTF	0.98079832452502	-0.51917760667599	0.16242041691563
CFK	0.98079832452469	-0.51917760667612	0.16242041691476
CFTF	0.98079832452468	-0.51917760667611	0.16242041691476
SFTF	0.98079832452469	-0.51917760667611	0.16242041691476

Tab. 5.34: Resultados de los coeficientes de igualación con los diferentes algoritmos para el canal de fase mínima $1 + 0{,}536z^{-1}+0{,}0718z^{-2}$

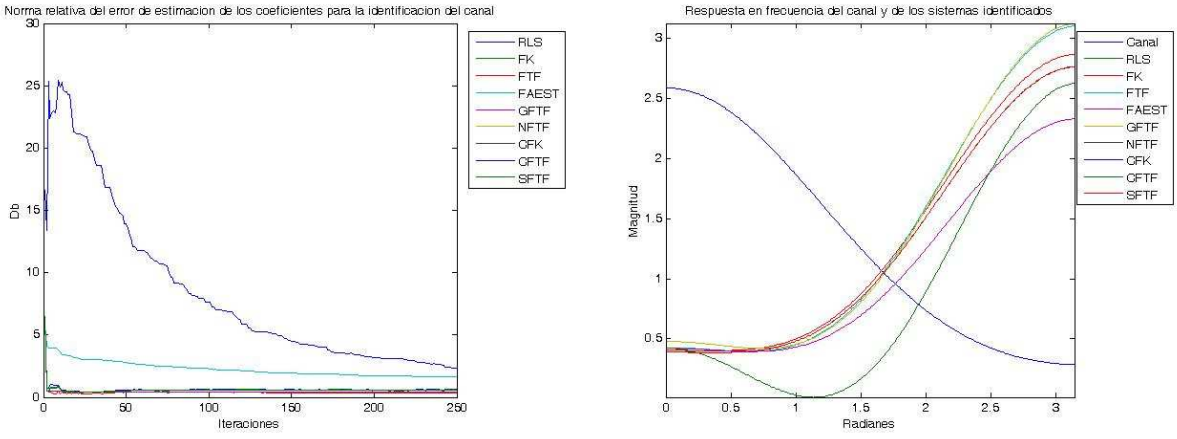


Fig. 5.41: a) Norma relativa del error de la igualación de canal. b) Respuesta en frecuencia del canal de fase mínima $1 + 0{,}536z^{-1}+0{,}0718z^{-2}$ y de los filtros igualadores con los diferentes algoritmos.

Para el cuarto canal de fase mínima se muestran los resultados de las simulaciones de punto flotante en la tabla 5.35 y se ilustran en la gráfica de respuesta en frecuencia (inciso a) y normas relativas de error en las figuras 5.42.

Coeficientes Calculados en la Igualación de Canal con los algoritmos			
Algoritmo	Coeficientes		
RLS	0.87821211004109	1.03903391706792	0.58748882154702
FK	0.87821211004130	1.03903391706789	0.58748882154679
FTF	0.87821211004210	1.03903391706791	0.58748882154545
FAEST	0.87821211004290	1.03903391706792	0.58748882154419
GFTF	0.87821211004307	1.03903391706792	0.58748882154392
NFTF	0.87821211004520	1.03903391706792	0.58748882154056
CFK	0.87821211004101	1.03903391706792	0.58748882154712
CFTF	0.87821211004118	1.03903391706791	0.58748882154693
SFTF	0.87821211004115	1.03903391706791	0.58748882154696

Tab. 5.35: Resultados de los coeficientes de igualación con los diferentes algoritmos para el canal de fase mínima $1 - 1{,}6z^{-1} + 0{,}95z^{-2}$

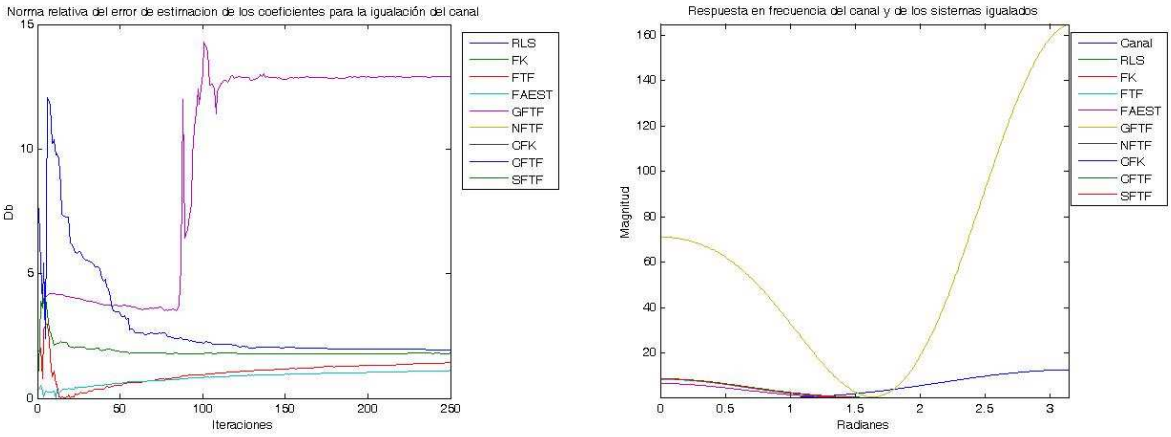


Fig. 5.42: a) Norma relativa del error de la igualación de canal. b) Respuesta en frecuencia del canal de fase mínima $1 - 1{,}6z^{-1} + 0{,}95z^{-2}$ y de los filtros igualadores con los diferentes algoritmos.

En la siguiente tabla (5.36) se muestran los resultados de las simulaciones en punto flotante para el quinto canal de fase mínima, y en la figura 5.43 se muestran las gráficas tanto de la respuesta en frecuencia (inciso a) como de la norma relativa del error de los coeficientes (inciso b) obtenidas con esos resultados.

Coeficientes Calculados en la Igualación de Canal con los algoritmos			
Algoritmo	Coeficientes		
RLS	0.62938579657553	0.71307860113309	0.40373594981254
FK	0.62938579657548	0.71307860113291	0.40373594981240
FTF	0.62938579657635	0.71307860113515	0.40373594981500
FAEST	0.62938579657434	0.71307860113007	0.40373594980896
GFTF	0.62938579657547	0.71307860113292	0.40373594981235
NFTF	0.62938579657645	0.71307860113540	0.40373594981530
CFK	0.62938579657577	0.71307860113333	0.40373594981280
CFTF	0.62938579657553	0.71307860113309	0.40373594981254
SFTF	0.62938579657553	0.71307860113309	0.40373594981254

Tab. 5.36: Resultados de los coeficientes de igualación con los diferentes algoritmos para el canal de fase mínima $1 - 1{,}9z^{-1} + 0{,}95z^{-2}$

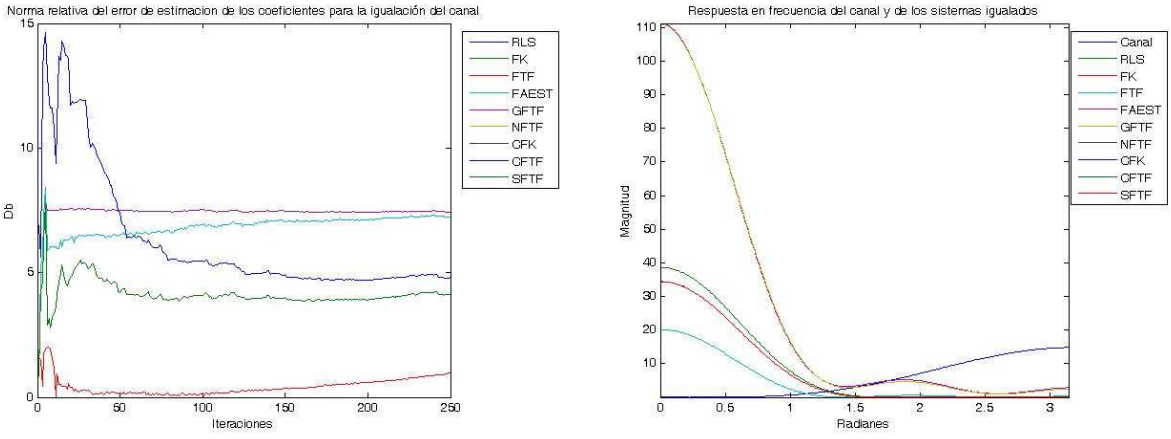


Fig. 5.43: a) Norma relativa del error de la igualación de canal. b)Respuesta en frecuencia del canal de fase mínima $1 - 1{,}9z^{-1} + 0{,}95z^{-2}$ y de los filtros igualadores con los diferentes algoritmos.

A continuación se presentan los resultados para los canales de fase no mínima. En la tabla 5.37, se muestran los resultados obtenidos para el primer canal de fase no mínima.

Coeficientes Calculados en la Igualación de Canal con los algoritmos			
Algoritmo	Coeficientes		
RLS	1.02662149079864	-0.97376319768582	0.55044490464024
FK	1.02662149079862	-0.97376319768576	0.55044490464020
FTF	1.02662149078690	-0.97376319767839	0.55044490461663
FAEST	1.02662149080256	-0.97376319768837	0.55044490464811
GFTF	1.02662149079585	-0.97376319768402	0.55044490463460
NFTF	1.02662149078809	-0.97376319767909	0.55044490461908
CFK	1.02662149079868	-0.97376319768571	0.55044490464030
CFTF	1.02662149079864	-0.97376319768583	0.55044490464025
SFTF	1.02662149079864	-0.97376319768583	0.55044490464025

Tab. 5.37: Resultados de los coeficientes de igualación con los diferentes algoritmos para el canal de fase no mínima $0{,}407 + 0{,}825z^{-1} + 0{,}407z^{-2}$

Mientras que en la figura 5.44 a continuación se muestran las gráficas de la respuesta en frecuencia del canal y los filtros igualadores así como las normas relativas de error (incisos a y b respectivamente).

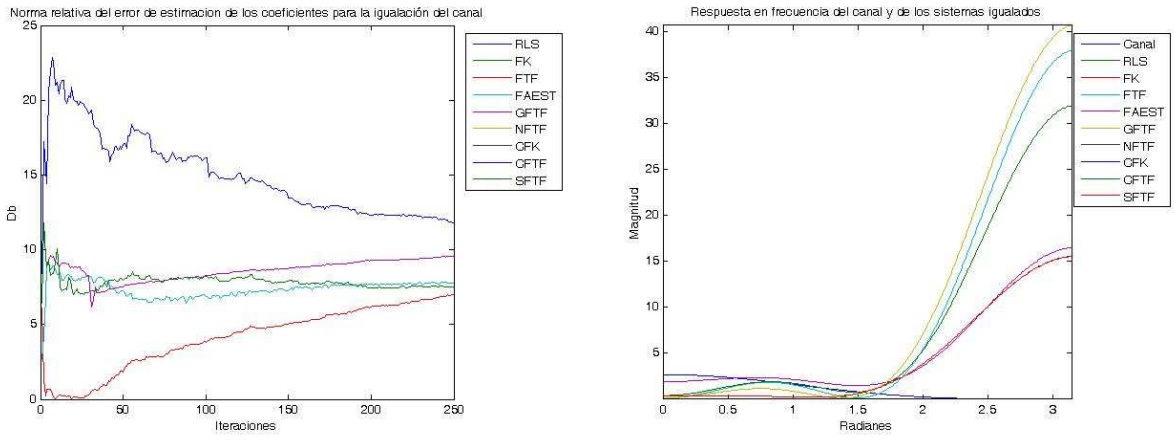


Fig. 5.44: a) Norma relativa del error de la igualación de canal. b) Respuesta en frecuencia del canal de fase mínima $0{,}407 + 0{,}825z^{-1} + 0{,}407z^{-2}$ y de los filtros igualadores con los diferentes algoritmos.

Y para el segundo canal de fase no mínima, se muestran los siguientes resultados en la tabla 5.38.

Coeficientes Calculados en la Igualación de Canal con los algoritmos			
Algoritmo	Coeficientes		
RLS	0.20317643264060	0.13297822231498	0.12061876557871
	0.12914836637633	0.17130417540649	0.11654136130364
	0.15610199307060	0.00406728597202	0.19521166648953
	0.07239493890991		
FK	0.20317643264071	0.13297822231513	0.12061876557889
	0.12914836637645	0.17130417540675	0.11654136130381
	0.15610199307092	0.00406728597227	0.19521166648972
	0.07239493891011		
FTF	0.20317643263949	0.13297822231235	0.12061876557453
	0.12914836637071	0.17130417540016	0.11654136129445
	0.15610199306153	0.00406728596068	0.19521166648057
	0.07239493889712		
FAEST	0.20317643264113	0.13297822231603	0.12061876558045
	0.12914836637883	0.17130417540973	0.11654136130793
	0.15610199307497	0.00406728597741	0.19521166649430
	0.07239493891640		
GFTF	0.20317643263901	0.13297822231137	0.12061876557295
	0.12914836636853	0.17130417539734	0.11654136129066
	0.15610199305761	0.00406728595565	0.19521166647611
	0.07239493889146		
NFTF	0.20317643263721	0.13297822230733	0.12061876556653
	0.12914836635968	0.17130417538692	0.11654136127607
	0.15610199304326	0.00406728593762	0.19521166646116
	0.07239493887035		
CFK	0.20317643264056	0.13297822231491	0.12061876557863
	0.12914836637624	0.17130417540637	0.11654136130348
	0.15610199307042	0.00406728597176	0.19521166648934
	0.07239493890971		
CFTF	0.20317643264061	0.13297822231499	0.12061876557871
	0.12914836637631	0.17130417540649	0.11654136130362
	0.15610199307058	0.00406728597200	0.19521166648951
	0.07239493890991		
SFTF	0.20317643264061	0.13297822231499	0.12061876557871
	0.12914836637631	0.17130417540649	0.11654136130362
	0.15610199307058	0.00406728597200	0.19521166648951
	0.07239493890990		

Tab. 5.38: Resultados de los coeficientes de igualación con los diferentes algoritmos para el canal de fase no mínima $0{,}04 - 0{,}05z^{-1} + 0{,}07z^{-2} - 0{,}21z^{-3} - 0{,}5z^{-4} + 0{,}72z^{-5} + 0{,}36z^{-6} + 0{,}21z^{-8} + 0{,}03z^{-9} + 0{,}07z^{-10}$

Mientras que en la figura 5.45 se ilustran: a) las normas relativas de error obtenidas para este canal y b)la respuesta en frecuencia.

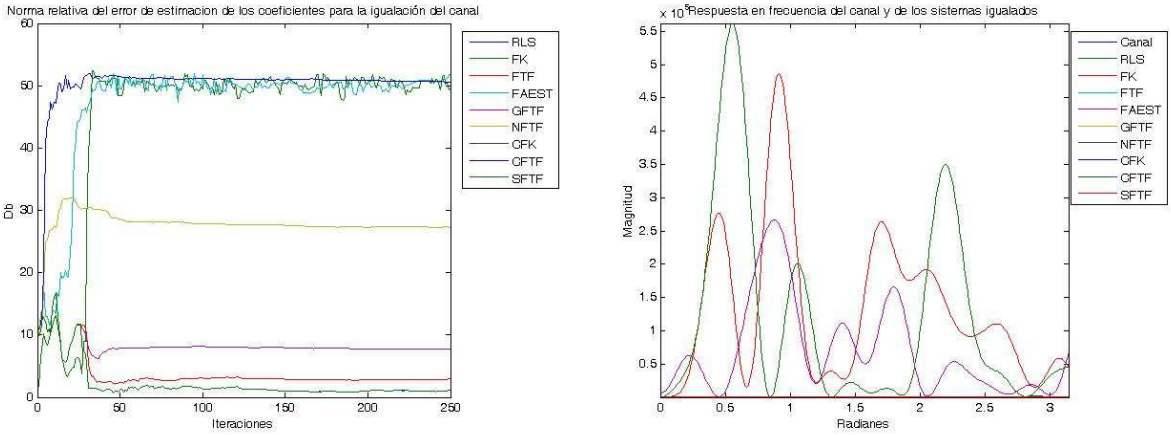


Fig. 5.45: a) Norma relativa del error de la igualación de canal. b) Respuesta en frecuencia del canal de fase no mínima $0{,}04 - 0{,}05z^{-1} + 0{,}07z^{-2} - 0{,}21z^{-3} - 0{,}5z^{-4} + 0{,}72z^{-5} + 0{,}36z^{-6} + 0{,}21z^{-8} + 0{,}03z^{-9} + 0{,}07z^{-10}$ y de los filtros igualadores con los diferentes algoritmos

5.4. Comparativo de la Implementación en Longitudes de Palabra Mínima

A continuación (tabla 5.39) se presenta una comparativo de la señal de error obtenida con los algoritmos para el primer canal, para tres diferentes valores del factor de olvido manteniendo el parámetro delta en un valor de 0.01.

Canal 1						
	P. Flotante	P. Fijo	P. Flotante	P. Fijo	P. Flotante	P. Fijo
lambda	1		0.99		0.98	
RLS	9.289e-02	9.288e-02	9.289e-02	9.288e-02	9.289e-02	9.288e-02
FKRLS	9.289e-02	9.289e-02	9.158e-02	9.159e-02	9.177e-02	9.177e-02
FTFRLS	1.418e-01	1.234e-01	2.763e-01	5.345e-02	2.891e-01	1.778e-02
FAEST	9.289e-02	8.699e-03	7.216e-02	1.086e-03	4.196e-02	7.981e-04
GFTF	9.289e-02	1.011e-01	6.959e-02	1.054e-01	3.686e-02	5.254e-02
NFTF	9.289e-02	8.869e-02	9.289e-02	8.869e-02	9.289e-02	8.869e-02
CFK	9.289e-02	9.289e-02	9.158e-02	9.158e-02	9.177e-02	9.177e-02
CFTF	1.835e-01	7.358e-05	5.754e-01	1.999e-02	6.811e-01	1.893e-02
SFTF	9.289e-02	9.234e-02	8.085e-01	4.707e-06	1.810e-01	8.157e-07

Tab. 5.39: Valores finales de la señal de error para las simulaciones de igualación de canal en punto flotante y en punto fijo con diferentes factores de olvido

En la tabla anterior se puede observar que los algoritmos que presentan inestabilidad son el FAEST y el GFTF a pesar de que dicha inestabilidad no resulta en un gran error en la estimación final de los coeficientes.

En seguida se muestra la tabla 5.40, en la cual podemos observar los valores obtenidos de la señal de error para las simulaciones de igualación de canal con los diferentes algoritmos al utilizar el segundo de los canales ya mencionados.

Canal 2						
	P. Flotante	P. Fijo	P. Flotante	P. Fijo	P. Flotante	P. Fijo
lambda	1		0.99		0.98	
RLS	2.344e-02	2.344e-02	2.344e-02	2.344e-02	2.344e-02	2.344e-02
FKRLS	2.344e-02	2.344e-02	2.213e-02	2.213e-02	2.093e-02	2.093e-02
FTFRLS	7.341e-02	3.560e-02	1.440e-01	2.273e-03	1.488e-01	2.333e-03
FAEST	2.344e-02	4.436e-03	5.284e-03	2.219e-04	1.556e-03	3.239e-04
GFTF	2.344e-02	6.766e-02	4.789e-03	1.590e-01	1.980e-03	1.704e-02
NFTF	2.344e-02	6.022e-02	2.344e-02	6.022e-02	2.344e-02	6.022e-02
CFK	2.344e-02	2.344e-02	2.213e-02	2.213e-02	2.093e-02	2.093e-02
CFTF	9.358e-02	1.049e-01	1.654e-01	1.918e-05	2.312e-01	3.736e-05
SFTF	2.344e-02	2.330e-02	2.824e-01	6.165e-05		1.421e-05

Tab. 5.40: Valores finales de la señal de error para las simulaciones de igualación de canal en punto flotante y en punto fijo con diferentes factores de olvido

En la tabla anterior (5.40) se pueden observar las variaciones en los valores, las cuales son mayores para el algoritmo FAEST pero aun así se manifiesta una correspondencia consistente entre las simulaciones de punto flotante y las de punto fijo al igual que el algoritmo CFK. Mientras que el algoritmo SFTF presenta una mejora en el desempeño cuando el factor de olvido toma valores menores a la unidad.

Para el tercer canal de fase mínima se encontraron los resultados para la señal de error mostrados en la tabla 5.41; la cual se muestra a continuación:

Canal 3						
	P. Flotante	P. Fijo	P. Flotante	P. Fijo	P. Flotante	P. Fijo
lambda	1		0.99		0.98	
RLS	3.499e-02	3.501e-02	3.499e-02	3.501e-02	3.499e-02	3.501e-02
FKRLS	3.499e-02	3.499e-02	3.436e-02	3.436e-02	3.420e-02	3.420e-02
FTFRLS	3.874e-02	3.470e-02	1.321e-01	3.243e-03	1.424e-01	3.474e-04
FAEST	3.499e-02	4.952e-03	7.030e-01	7.421e-05	5.549e-01	4.482e-04
GFTF	3.499e-02	1.103e-01	1.212e-01	2.153e-01	7.556e-01	2.330e-01
NFTF	3.499e-02	4.765e-02	3.499e-02	4.765e-02	3.499e-02	4.765e-02
CFK	3.499e-02	3.500e-02	3.436e-02	3.436e-02	3.420e-02	3.420e-02
CFTF	1.243e-01	6.066e-02	1.678e-01	1.079e-08		8.618e-05
SFTF	3.499e-02	3.482e-02	3.929e-01	1.339e-04	3.035e-01	1.789e-04

Tab. 5.41: Valores finales de la señal de error para las simulaciones de igualación de canal en punto flotante y en punto fijo con diferentes factores de olvido

Al igual que para los dos canales anteriores, podemos observar que existen ligeras variaciones entre los resultado obtenidos para los diferentes valores del factor de olvido, sin embargo también se puede observar una correspondencia entre las simulaciones de punto flotante y las simulaciones de punto fijo. Destacando el comportamiento de los algoritmos SFTF y FAEST, los cuales mejoran el error obtenido al tomar valores de $\lambda < 1$.

Para el caso de las simulaciones de igualación del cuarto canal, podemos observar ahora si un mayor número de inconsistencias entre los valores de punto flotante y punto fijo, tal y como se muestra en la tabla 5.42.

Canal 4						
	P. Flotante	P. Fijo	P. Flotante	P. Fijo	P. Flotante	P. Fijo
lambda	1		0.99		0.98	
RLS		1.234e-01		1.234e-01		1.234e-01
FKRLS	1.234e-01	1.234e-01	1.205e-01	1.205e-01	1.152e-01	1.695e-01
FTFRLS	5.904e+00	1.214e-01	3.614e-02	1.140e-01	4.253e-02	1.140e-01
FAEST	1.234e-01	7.169e-03	4.243e-02	9.579e-04	8.226e-02	1.786e-03
GFTF	1.234e-01	7.296e+00	5.854e-02	5.339e-02	8.504e-02	1.395e-01
NFTF	1.234e-01	1.234e-01	1.234e-01	1.234e-01	1.234e-01	1.234e-01
CFK	1.234e-01	1.234e-01	1.205e-01	1.205e-01		1.152e-01
CFTF		1.310e-01	6.151e-01	8.519e-05	7.052e-01	6.534e-06
SFTF	1.234e-01	1.220e-01	6.708e-01	1.658e-05	3.700e-01	3.055e-04

Tab. 5.42: Valores finales de la señal de error para las simulaciones de igualación de canal en punto flotante y en punto fijo con diferentes factores de olvido

No obstante las variaciones mencionadas, se mantiene la tendencia de mejora de los algoritmos SFTF y FAEST al tomar valores menores a 1 para el factor de olvido lambda, resultando en una disminución de la señal de error obtenida en las simulaciones de punto flotante.

Ahora bien en la tabla 5.43 que se observa enseguida, podemos observar los valores obtenidos para el quinto canal empleado en las simulaciones.

Canal 5						
	P. Flotante	P. Fijo	P. Flotante	P. Fijo	P. Flotante	P. Fijo
lambda	1		0.99		0.98	
RLS	3.719e-01	3.721e-01	3.719e-01	3.721e-01	3.719e-01	3.721e-01
FKRLS	3.719e-01	3.719e-01	3.886e-01	3.886e-01	4.021e-01	4.021e-01
FTFRLS	2.412e-02	3.675e-01	1.322e-01	7.345e-02	1.596e-01	7.769e-02
FAEST	3.719e-01	9.856e-03	1.827e-01	1.104e-03	1.165e-01	9.389e-05
GFTF	3.719e-01	4.596e-01	1.805e-01	3.537e-01	1.172e-01	6.561e-01
NFTF	3.719e-01	3.719e-01	3.719e-01	3.719e-01		3.719e-01
CFK	3.719e-01	3.719e-01	3.886e-01	3.886e-01	4.021e-01	4.021e-01
CFTF		3.265e-01	7.702e-01	2.585e-05	7.827e-01	2.496e-05
SFTF	3.719e-01	3.678e-01	1.057e+01	9.296e-05	1.595e+01	2.406e-05

Tab. 5.43: Valores finales de la señal de error para las simulaciones de igualación de canal en punto flotante y en punto fijo con diferentes factores de olvido

Después de observar la tabla 5.43, una vez más se puede apreciar la mejora en el comportamiento de los algoritmos SFTF y FAEST al disminuir el valor del factor de olvido a valores cercanos a la unidad.

En la siguiente tabla (5.44) podemos observar los resultados de la señal de error obtenidos para las simulaciones de igualación del sexto canal, en la cual podemos observar que el mayor valor obtenido lo presenta el algoritmo GFTF, el cual sin embargo los presenta tanto en las simulaciones de punto flotante como en las simulaciones de punto fijo. Mientras que el algoritmo FAEST nuevamente muestra un comportamiento de disminución del error al variar el factor de olvido, tal como en los casos anteriores y el algoritmo CFTF para este caso especifico muestra una mejora significativa de la implementación en punto fijo con respecto a la implementación de punto flotante.

Canal 6						
	P. Flotante	P. Fijo	P. Flotante	P. Fijo	P. Flotante	P. Fijo
lambda	1		0.99		0.98	
RLS	3.408e-01	3.409e-01	3.408e-01	3.409e-01	3.408e-01	3.409e-01
FKRLS	3.408e-01	3.408e-01	3.458e-01	3.458e-01	3.550e-01	3.550e-01
FTFRLS	3.364e-01	4.450e-01	4.126e-01	4.450e-01	4.614e-01	4.450e-01
FAEST	3.408e-01	7.018e-02	3.146e+00	1.731e-03	1.695e+01	8.598e-04
GFTF	3.408e-01	4.501e-01	3.127e+00	2.346e+00	2.793e+01	4.164e-01
NFTF	3.408e-01	3.415e-01	3.408e-01	3.415e-01	3.408e-01	3.415e-01
CFK	3.408e-01	3.408e-01	3.458e-01	3.458e-01	3.550e-01	3.550e-01
CFTF	2.977e-01	1.892e-01	2.801e-01	3.459e-05	3.252e-01	1.667e-05
SFTF	3.408e-01	3.392e-01	3.426e-01	3.134e-01	3.541e-01	3.747e-01

Tab. 5.44: Valores finales de la señal de error para las simulaciones de igualación de canal en punto flotante y en punto fijo con diferentes factores de olvido

Por último, en cuanto a las señales de error, se muestran en la siguiente tabla 5.45 los valores obtenidos por los diferentes algoritmos al realizar las simulaciones de igualación del séptimo canal, empleando los diferentes valores del factor de olvido. Aquí se puede observar que el algoritmo FAEST tiene cierta mejora para los valores del factor de olvido de 1 y 0,98, mientras que el algoritmo SFTF presenta una mejora para los tres casos del factor de olvido.

Canal 7						
	P. Flotante	P. Fijo	P. Flotante	P. Fijo	P. Flotante	P. Fijo
lambda	1		0.99		0.98	
RLS	2.111e-01	2.081e-01	2.111e-01	2.081e-01	2.111e-01	2.081e-01
FKRLS	2.111e-01	2.690e+01	7.209e-02	6.829e+00	6.336e-02	3.795e+01
FTFRLS	2.880e-01	5.481e-02	4.283e-01	9.930e-02	5.188e-01	1.591e-01
FAEST	2.111e-01	2.479e-03	1.245e-01	2.016e-01	6.206e-02	1.629e-02
GFTF	2.111e-01	2.388e-01	1.375e-01	1.373e-01	5.962e-02	4.620e-02
NFTF	2.111e-01	3.186e+00	2.111e-01	3.186e+00	2.111e-01	3.186e+00
CFK	2.111e-01	2.123e-01	7.209e-02	7.289e-02	6.336e-02	6.337e-02
CFTF	2.365e-01	8.995e-01	2.626e-01	0.000e+00	4.087e-01	1.137e-05
SFTF	2.111e-01	2.017e-01	2.182e-01	1.507e-02	4.942e-02	3.577e-05

Tab. 5.45: Valores finales de la señal de error para las simulaciones de igualación de canal en punto flotante y en punto fijo con diferentes factores de olvido

A continuación se presenta un comparativo de la diferencia entre los valores obtenido con las simulaciones de punto fijo con respecto de las simulaciones de punto flotante, y podemos observar en la siguiente tabla 5.46, que la diferencia en general es muy pequeña, y que aun para los casos en los cuales la diferencia es menor todavía se trata de una cantidad del orden de $1e^{-2}$.

MSE de los Coeficientes							
	Canal 1	Canal 2	Canal 3	Canal 4	Canal 5	Canal 6	Canal 7
RLS	1.550e-08	1.550e-08	1.550e-08	1.550e-08	1.550e-08	1.550e-08	1.550e-08
FK	1.743e-08	1.743e-08	1.743e-08	1.743e-08	1.743e-08	1.743e-08	1.743e-08
FTF	3.069e-03	3.069e-03	3.069e-03	3.069e-03	3.069e-03	3.069e-03	3.069e-03
FAEST	1.474e-02	1.474e-02	1.474e-02	1.474e-02	1.474e-02	1.474e-02	1.474e-02
GFTF	3.071e-02	3.071e-02	3.071e-02	3.071e-02	3.071e-02	3.071e-02	3.071e-02
NFTF	1.093e-04	1.093e-04	1.093e-04	1.093e-04	1.093e-04	1.093e-04	1.093e-04
CFK	1.703e-08	1.703e-08	1.703e-08	1.703e-08	1.703e-08	1.703e-08	1.703e-08
CFTF	2.722e-02	2.722e-02	2.722e-02	2.722e-02	2.722e-02	2.722e-02	2.722e-02
SFTF	1.693e-08	1.693e-08	1.693e-08	1.693e-08	1.693e-08	1.693e-08	1.693e-08

Tab. 5.46: Valores de error de las simulaciones en punto fijo con respecto a las simulaciones de punto flotante.

Por último se muestra un comparativo de las señales de error obtenidas con la implementación en punto fijo de los diferentes algoritmos al emplear longitudes de palabra de 16 y 32 bits para la problemática de igualación de canal.

En la tabla 5.47 se puede apreciar que los algoritmos cuyas las implementaciones presentan un buen desempeño tanto en longitudes de palabra de 32 bits como de 16 bits son las de los algoritmos FKRLS, FAEST, CFK y SFTF.

	16 bits	32 bits
RLS	0.041748046875000	0.035005874931810
FKRLS	0.079101562500000	0.034994266927240
FTFRLS	0.037433624267580 *	0.034697718918320
FAEST	0.002716064453130	0.004952018614860
GFTF	0.248474121093750 **	0.110280307009820
NFTF	0.142578125000000	0.047653220593930
CFK	0.018066406250000	0.034996822476390
CFTF	0.475272417068480	0.060659663548410
SFTF	0.000307895243170	0.034818230272760
* Implementación a 20 bits		
** Implementación a 18 bits		

Tab. 5.47: Valores de error de las simulaciones en punto fijo con longitudes de palabra de 16 y 32 bits.

5.5. Conclusiones

La implementación de algoritmos en punto fijo, requiere de un formateo y almacenamiento de los valores de los datos y coeficientes en un formato binario signado que se ajuste a una longitud de palabra finita. Los efectos del cálculo con una longitud de palabra fija y su representación son fuentes significativas de imprecisión y ruido en sistemas digitales. La cuantización tanto de los datos de entrada como de los coeficientes previamente calculados resulta en una ligera perdida de precisión. Las operaciones aritméticas internas también contribuyen a la perdida de precisión. Sin embargo la mayoría de estas fuentes de error pueden ser reducidas a través de un análisis de implementación del algoritmo y modificaciones en la implementación.

Por lo anteriormente analizado se puede concluir que la implementación de los diferentes algoritmos es lo suficientemente consistente con respecto a la implementación de los algoritmos en punto flotante, lo cual enfoca la mayor parte de las ventajas de estas implementaciones en punto fijo en la posibilidad de aprovechar plataformas de menor costo para que el esfuerzo y la complejidad de emplear la metodología de implementación en punto fijo de los algoritmos tenga un rédito los suficientemente importante.

Por último se puede apreciar que los algoritmos más robustos después de realizar las implementaciones son: el algoritmo rápido de Kalman (Fast Kalman) [LMF78], el algoritmo rápido de estimación del error a priori (FAEST) [CMK83], el algoritmo rápido de kalman de covarianza (Covariance Fast Kalman) [Lin84] y el algoritmo estabilizado FTF (Stabilized Fast Transversal Filter) [BG88], los cuales arrojaron el menor error en la mayoría de las simulaciones de igualación de canal con los diferentes valores de factor de olvido que se emplearon. Sin embargo al realizar las implementaciones en longitudes de palabra de 16 bits destacan el SFTF y el FAEST como los algoritmos que mejor comportamiento tienen en cuando al menor error obtenido.

6. CONCLUSIONES Y PERSPECTIVAS

Como resultado de este trabajo se desarrollaron una biblioteca de algoritmos rápidos de filtrado adaptable en aritmética de punto flotante y una biblioteca de algoritmos rápidos de filtrado adaptable en aritmética de punto fijo para poder realizar las simulaciones de la implementación de los algoritmos. Con estas bibliotecas de algoritmos se llevo a cabo la evaluación de la dinámica y la precisión de las variables con un enfoque heurístico basado en la simulación, así como la evaluación de las implementaciones en punto fijo con diferentes valores del factor de olvido obteniendo un buen desempeño en longitudes de palabra de 32 bits para todos los algoritmos y un desempeño que se puede considerar todavía bueno para algunos de ellos al llevarlos a longitudes de palabra mínimas de 16 bits.

Después de la realización de este trabajo, se puede concluir que aun cuando la implementación de algoritmos en aritmética de punto fijo requiere de un trabajo más exhaustivo en la programación y la implementación, también presenta grandes ventajas si un procesador o arquitectura de punto fijo puede realizar una función en aplicaciones que requieran su implementacion en altos volúmenes sensibles en costos, el utilizar una arquitectura de punto fijo puede justificarse al evaluar un análisis de costo general del sistema.

También vale la pena mencionar que hasta este punto el proceso de implementación se valida únicamente en simulación, sin embargo, dado que se está considerando que la herramienta de punto fijo de Matlab nos permite controlar las variables con las que se define la manera en la cual se van a realizar las operaciones de punto fijo para todas las variables involucradas en los algoritmos; los resultados al ser llevados a una arquitectura no deben de alejarse en gran medida de los presentados aquí.

Es prudente acotar que el trabajo presente es, bajo la visión de la metodología de implementación de la herramienta elegida, el punto de partida para emplear las demás herramientas complementarias proveeidas por Mathworks (Code Composer, Real Time Workshop, etc...) para la implementación de los algoritmos en diversas arquitecturas.

APÉNDICES

A. FILTRADO DE WIENER

El filtro de Wiener tiene la siguiente estructura

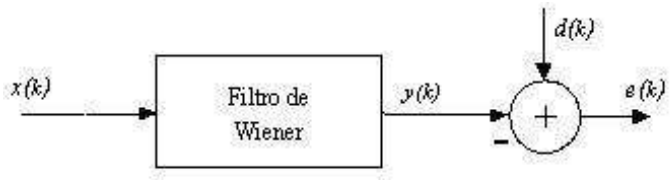


Fig. A.1: Estructura del Filtro de Wiener.

En donde x(k) es una señal real de observación la cual es estacionaria en sentido amplio WSS, d(k) es la señal real deseada, WSS, y(k) es una estimación de d(k) y e(k) es la señal de error entre d(k) e y(k). El filtro de Wiener cuando se calcula en su forma matricial da como resultado un filtro lineal invariante en el tiempo con respuesta al impulso finita (FIR). Si se define que los M coeficientes de este filtro son $w_0, w_1, \ldots, w_{M-1}$, entonces la salida del filtro es

$$y(k) = \sum_{l=0}^{M-1} w_l x(k-l) \tag{A.1}$$

y la señal de error

$$e(k) = d(k) - y(k) = d(k) - \sum_{l=0}^{M-1} w_l x(k-l) \tag{A.2}$$

Si se definen los vectores

$$\mathbf{x}(k) = \begin{bmatrix} x(k) & x(k-1) & \cdots & x(k-M+1) \end{bmatrix}^T$$
$$\mathbf{w} = \begin{bmatrix} w_0 & w_1 & \cdots & w_{M-1} \end{bmatrix}^T$$

entonces la ecuación A.1 puede reescribirse como

$$y(k) = \mathbf{w}^T\mathbf{x}(k) = \mathbf{x}^T(k)\mathbf{w} \tag{A.3}$$

y la señal de error

$$e(k) = d(k) - y(k) = d(k) - \mathbf{w}^T\mathbf{x}(k) = d(k) - \mathbf{x}^T(k)\mathbf{w} \tag{A.4}$$

El error cuadrático promedio es el siguiente

$$\begin{aligned} J(\mathbf{w}) &\triangleq E\left\{|e(k)|^2\right\} = E\left\{\left|d(k) - \mathbf{x}^T(k)\mathbf{w}\right|^2\right\} \\ &= E\left\{|d(k)|^2\right\} - 2\mathbf{w}^T E\left\{d(k)\mathbf{x}(k)\right\} + \mathbf{w}^T E\left\{\mathbf{x}(k)\mathbf{x}^T(k)\right\} \end{aligned} \tag{A.5}$$

Si se define

$$R_d(0) = E\left\{|d(k)|^2\right\} \tag{A.6}$$

Vector de Correlación Cruzada de Mx1

$$\mathbf{p} = E\left\{d(k)\mathbf{x}(k)\right\} \tag{A.7}$$

Matriz de Autocorrelación de MxM

$$\mathbf{R} = E\left\{\mathbf{x}(k)\mathbf{x}^T(k)\right\} \tag{A.8}$$

La ecuación A.5 queda como

$$J(\mathbf{w}) = R_d(0) - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\mathbf{w} \tag{A.9}$$

en donde J(w) también se define como función de costo. La idea del filtro de Wiener es encontrar los coeficientes w tales que minimicen el error cuadrático promedio, la función de costo, definida en A.9. La manera convencional para encontrar el mínimo de la función de costo es calcular el gradiente de J(w) e igualarlo a cero. Esto es

$$\nabla J(\mathbf{w}) = \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \left[\begin{array}{cccc} \frac{\partial J(\mathbf{w})}{\partial w(0)} & \frac{\partial J(\mathbf{w})}{\partial w(1)} & \cdots & \frac{\partial J(\mathbf{w})}{\partial w(M-1)} \end{array}\right]^T \tag{A.10}$$

El resultado es

$$\nabla J(\mathbf{w}) = 2\mathbf{R}\mathbf{w} - 2\mathbf{p} \tag{A.11}$$

Igualando A.11 a cero y definiendo el vector

$$\mathbf{w_o} = \left[\begin{array}{cccc} w_{o0} & w_{o1} & \cdots & w_{oM-1} \end{array}\right]^{\mathbf{T}}$$

como el vector de coeficientes óptimos que minimizan el error cuadrático promedio, se tiene el siguiente sistema de ecuaciones

$$\mathbf{R}\mathbf{w}_o = \mathbf{p} \tag{A.12}$$

El sistema de ecuaciones definido en A.12 es comúnmente conocido como ecuación normal o ecuaciones de Wiener-Hopf. La solución a este sistema de ecuaciones se le conoce como la solución de Wiener-Hopf y esta dada por la siguiente expresión

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p} \tag{A.13}$$

Sustituyendo A.12 en A.9 se obtiene el error cuadrático promedio mínimo, o sea

$$J_{\text{mín}}(\mathbf{w}) = R_d(0) - \mathbf{p}^T\mathbf{w}_o = R_d(0) - \mathbf{w}_o^T\mathbf{p} \tag{A.14}$$

También sustituyendo A.13 en A.14

$$J_{\text{mín}}(\mathbf{w}) = R_d(0) - \mathbf{p}^T\mathbf{R}^{-1}\mathbf{p} \tag{A.15}$$

A.1. Propiedades de la Matriz de Autocorrelación

La función de autocorrelación de una señal x(k) real WSS se define como

$$R_x(m) \triangleq E\{x(k+m)x(k)\} \tag{A.16}$$

y su matriz de autocorrelación

$$\mathbf{R} = E\left\{\mathbf{x}(k)\mathbf{x}^T(k)\right\} \tag{A.17}$$

en donde

$$\mathbf{x}(k) = \begin{bmatrix} x(k) & x(k-1) & \cdots & x(k-M+1) \end{bmatrix}^T$$

entonces la matriz de autocorrelación es una matriz de MxM con la siguiente forma

$$\mathbf{R} = \begin{bmatrix} R_x(0) & R_x(1) & \cdots & R_x(M-2) & R_x(M-1) \\ R_x(-1) & R_x(0) & \cdots & R_x(M-3) & R_x(M-2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ R_x(-M+2) & R_x(-M+3) & \cdots & R_x(0) & R_x(1) \\ R_x(-M+1) & R_x(-M+2) & \cdots & R_x(-1) & R_x(0) \end{bmatrix} \tag{A.18}$$

A continuación se presentarán algunas de las propiedades más importantes de la matriz de autocorrelación [Hay96].

- **Propiedad 1**. La matriz de autocorrelación de un proceso aleatorio real y WSS es simétrica.

$$\mathbf{R} = \mathbf{R}^T \tag{A.19}$$

Dada esta propiedad la matriz de autocorrelación puede reescribirse como

$$\mathbf{R} = \begin{bmatrix} R_x(0) & R_x(1) & \cdots & R_x(M-2) & R_x(M-1) \\ R_x(1) & R_x(0) & \cdots & R_x(M-3) & R_x(M-2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ R_x(M-2) & R_x(M-3) & \cdots & R_x(0) & R_x(1) \\ R_x(M-1) & R_x(M-2) & \cdots & R_x(1) & R_x(0) \end{bmatrix} \tag{A.20}$$

- **Propiedad 2**. La matriz de Autocorrelación de un proceso aleatorio real WSS es una matriz de Toeplitz.

  Esto significa que los elementos de la diagonal principal y los elementos en cualquier otra diagonal paralela a la principal son iguales.

- **Propiedad 3**. La matriz de autocorrelación es siempre no negativa definida y casi siempre positiva definida.

  Esto significa que una matriz no negativa definida $\mathbf{R}$ cumple

$$\mathbf{c}^T\mathbf{R}\mathbf{c} \geq 0 \tag{A.21}$$

  y una matriz positiva definida R cumple

$$\mathbf{c}^T\mathbf{R}\mathbf{c} > 0 \tag{A.22}$$

  Para cualquier vector real $\mathbf{c}$.

A.1.1. Forma Normal de la Matriz de Autocorrelación

Los eigenvalores y eigenvectores de la matriz de autocorrelación se pueden obtener mediante el siguiente sistema de ecuaciones

$$[\mathbf{R} - \lambda\mathbf{I}_M]\,\mathbf{q}_i = \mathbf{0} \tag{A.23}$$

en donde λ es una variable escalar y q_i es un vector columna. Para encontrar los eigenvalores de la matriz se desarrolla la ecuación característica de $\mathbf{R}$ y se iguala a cero

$$\det\left(\mathbf{R} - \lambda\mathbf{I}_M\right) = 0 \tag{A.24}$$

La ecuación anterior es un polinomio en λ con M soluciones: $\lambda_0, \lambda_1, \ldots, \lambda_{M-1}$. Para cada eigenvalor $\lambda_i, i = 0, 1, \ldots, M-1$ existe por lo menos un vector solución en A.23 que se encuentra de la siguiente manera

$$\mathbf{R}\mathbf{q}_i = \lambda_i\mathbf{q}_i \tag{A.25}$$

donde q_i es el eigenvector correspondiente al eigenvalor λ_i. Entonces, extendiendo A.25 se tiene

$$\mathbf{R}\begin{bmatrix} \mathbf{q}_0 & \mathbf{q}_1 & \cdots & \mathbf{q}_{M-1} \end{bmatrix} = \begin{bmatrix} \mathbf{q}_0 & \mathbf{q}_1 & \cdots & \mathbf{q}_{M-1} \end{bmatrix} \begin{bmatrix} \lambda_0 & 0 & 0 & 0 \\ 0 & \lambda_1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \lambda_{M-1} \end{bmatrix} \tag{A.26}$$

Y definiendo las matrices

$$\begin{aligned} \mathbf{Q} &\triangleq \begin{bmatrix} \mathbf{q}_0 & \mathbf{q}_1 & \cdots & \mathbf{q}_{M-1} \end{bmatrix} \\ \Lambda &\triangleq \begin{bmatrix} \lambda_0 & 0 & 0 & 0 \\ 0 & \lambda_1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \lambda_{M-1} \end{bmatrix} \end{aligned} \tag{A.27}$$

la ecuación A.26 se reescribe como

$$\mathbf{RQ} = \mathbf{Q}\Lambda \tag{A.28}$$

y la matriz de autocorrelación queda como

$$\mathbf{R} = \mathbf{Q}\Lambda\mathbf{Q}^{-1} \tag{A.29}$$

Se puede demostrar [Wid85] que los eigenvectores son ortogonales. Y escalando cada eigenvector adecuadamente los eigenvectores pueden ser ortonormales esto es

$$\mathbf{q}_i\mathbf{q}_j = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases} \tag{A.30}$$

Normalizar los eigenvectores no afecta a A.25. Ya que los eigenvectores son un conjunto de vectores ortonormales se tendra que

$$\mathbf{QQ}^T = \mathbf{I}_M \tag{A.31}$$

lo que da como consecuencia que

$$\mathbf{Q}^T = \mathbf{Q}^{-1} \tag{A.32}$$

y A.29 queda como

$$\mathbf{R} = \mathbf{Q}\Lambda\mathbf{Q}^T \tag{A.33}$$

B. LOS CANALES DE COMUNICACIÓN Y SUS CARACTERÍSTICAS

Si consideramos a el canal de comunicación como el medio que provee la conexión entre el transmisor y el receptor, podemos considerar como canales de comunicación desde un par de alambres para la transmisión de una señal eléctrica, la fibra óptica para la transmisión de un haz de luz modulado, hasta medios de almacenamiento de información ópticos o magnéticos.

Uno de los problemas más comunes en la transmisión de señales es el ruido aditivo. En general, el ruido aditivo es generado por los componentes internos que se utilizan para la implementación de el sistema de comunicación y es llamado en algunas ocasiones *Ruido Térmico*, sin embargo existen otras fuentes externas de ruido o interferencia, como por ejemplo la interferencia de otros usuarios del canal.

A continuación se describen ciertas características de algunos canales de comunicación.

Canales Alámbricos

.

Las redes de telefonía utilizan en su mayoría líneas de alambre para la transmisión de la señal, datos y video. Las líneas de par trenzado y de cable coaxial son básicamente canales electromagnéticos con anchos de banda relativamente pequeños. El alambre telefónico utilizado para conectar al usuario con la central tiene un ancho de banda de algunos cientos de kilohertz (Khz), mientras que el cable coaxial tiene un ancho de banda de algunos megahertz (Mhz)[Pro95].

Las señales transmitidas por este tipo de canales son distorsionadas tanto en amplitud como en fase y corrompidas por ruido aditivo y son propensos a interferencias de canales adyacentes. Para mitigar este efecto se realizan torceduras físicas en cada par de alambres dentro del cable. Generalmente hay de dos a doce torceduras por pie en un par de alambres. A este tipo de configuración se le conoce como par trenzado [Fre99].

Otro canal de comunicación alámbrico es el cable coaxial, el cual es una linea de transmisión que consiste de un par no balanceado de conductores dispuestos en forma concentrica y separados por un material dieléctrico. Generalmente el conductor externo, que envuelve al material dieléctrico, es tierra y provee al conductor interno de inmunidad a interferencias externas. Por su parte el dielectrico puede ser polietileno, espuma, aire, etc [CY03].

Principales Limitaciones en Canales Alámbricos

Existen básicamente tres limitantes en los canales de comunicación alámbricos:

1. **Distorsión en amplitud**. Esta se presenta cuando la respuesta del canal no es "plana"sobre el intervalo de frecuencias para el cual el espectro de entrada no es cero, de esta manera las diferentes componentes espectrales de la señal de entrada se modifican de forma diferente, unas sufren atenuación y otras ganancia [Sha79].

2. **Distorsión en fase**. Si el cambio de fase producido por el canal es de manera arbitraria, entonces las diversas componentes espectrales de la señal de entrada sufren distintos retardos, dando por resultado la *distorsión en fase* o *retardo* de la señal transmitida [Sha79]. Dicho tipo de distorsión resulta ser un problema en la transmisión por pulsos, ya que estos experimentan un efecto de ensanchamiento en el tiempo.

3. **Ruido**. Este se refiere, básicamente, a cualquier señal no deseada en el sistema de comunicación que limita el desempeño del mismo. Típicamente tenemos el *ruido* térmico, el cual esta presente en cualquier medio de transmisión y en todos los circuitos integrados [Fre99].

Canales de Fibra Óptica

.

Los canales de fibra óptica ofrecen un ancho de banda considerablemente más grande que los canales de cable coaxial, lo que ha permitido ofrecer a los usuarios de telefonía una amplia variedad de servicios de telecomunicación tales como voz, datos y video.

El transmisor o modulador en un sistema de comunicación de fibra óptica es una fuente de luz, LED o Laser. La información es transmitida variando la intensidad de la fuente de luz con el mensaje de la señal, propagándose a través de la fibra como una onda de luz la cual es amplificada periódicamente para compensar la atenuación. En el receptor la intensidad de la señal es detectada por un fotodiodo cuya salida es una señal eléctrica que varia en proporción directa a la potencia de la luz recibida por el fotodiodo. Las fibras ópticas poseen características únicas que las hacen altamente atractivas como medio de transmisión[CY03], ofreciendo las siguientes ventajas:

- **Gran banda de paso**. Con una frecuencia de portadora óptica de casi $2x10^{14}Hz$, es posible tener un ancho de banda teórico de alrededor de $2x10^{13}$.

- **Baja atenuación**. Típicamente de $0{,}2dB/Km$, permitiendo de esta manera, acrecentar la distancia entre las repetidoras en un sistema de comunicación por fibra óptica.

- **Aislamiento eléctrico**. Las fibras se hacen de materiales aislantes eléctricos (vidrios, plásticos). Esto hace que las interferencias electromagnéticas externas no perturben la transmisión de la fibra.

- **Peso y dimensiones**. Un cable de fibra óptica es por lo menos diez veces más ligero y compacto que un cable coaxial.

Canales Inalámbricos Electromagnéticos

.

En los sistemas de comunicación inalámbricos la energía electromagnética es acoplada al medio de propagación mediante una antena que la radia. El tamaño físico y la configuración de la antena dependen primordialmente de la frecuencia de operación. Para obtener una propagación eficiente de la energía electromagnética, la antena debe ser mayor que 1/10 de la longitud de onda.

B.1. Caracterización de los Canales

Bello fue el primero en introducir el modelo de un canal de propagación como un "tapped delay line filter"del tipo que se muestra en la figura B.1[Bel76]. Bello mostró que prácticamente cualquier canal de propagación real puede ser adecuadamente modelado mediante la selección adecuada de los espaciamientos del retardo y los coeficientes de ponderación. En la selección de los espaciamientos del retardo, cabe hacer notar que dichos espaciamientos deben ser menores al inverso del ancho de banda de la señal transmitida, es decir, se debe de satisfacer el teorema de Nyquist [TFJ96]. Como resultado de esto es común modelar al canal de propagación como un filtro de respuesta impulsional infinita (FIR), cuyos retardos están espaciados por el periodo de muestreo y los coeficientes de ponderación son seleccionados para modelar de una manera adecuada la respuesta al impulso del canal. [Tor97].

Bello introdujo dos conceptos de utilidad: el efecto de retardo y el efecto Doppler [Bel76]. El efecto de retardo de un canal es el soporte temporal de un canal FIR y mide el grado del ensanchamiento temporal de un pulso al ser transmitido a través del canal. El efecto Doppler mide el grado del ensanchamiento espectral de la señal transmitida. El efecto de retardo puede ser visto como la duración de la respuesta del canal a la función impulso, mientras que el efecto Doppler puede ser visto como el ancho del espectro de la señal de salida del canal cuando la entrada al canal es una senoide.

En este trabajo, se llevaron a cabo la implementación de algunos algoritmos de filtrado adaptable teniendo como resultado de este trabajo el contar con una biblioteca de algoritmos rápidos de filtrado adaptable en aritmética de punto flotante, la validación de la metodología de implementación de algoritmos mediante el estudio de las dinámicas de los algoritmos empleando método basado en la simulación, el desarrollo de una biblioteca de algoritmos de filtrado adaptable en aritmética de punto fijo, además de la evaluación

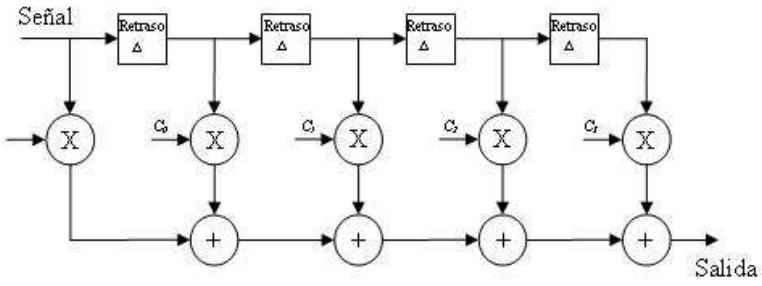


Fig. B.1: Modelo para un Canal de Propagación Lineal.

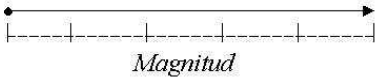
de las simulaciones de identificación de sistemas, simulaciones de igualación de canal con el fin de validar las implementaciones en aritmética de punto fijo, obteniéndose desempeños en punto fijo a 32 y a 16 bits lo suficientemente buenos en comparación con las implementaciones de punto flotante.

C. CONCEPTOS BÁSICOS DE ALGEBRA LINEAL

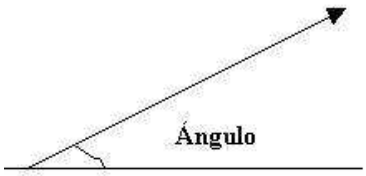
Definición de vector

La definición clásica de vector lo define como aquella cantidad en que cumple con las siguientes características:

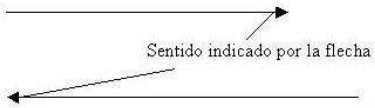
1. a) Tiene magnitud $|A|$



2. b) Dirección. Indicado el ángulo con respecto a un eje (por ejemplo, la horizontal).



3. c) Sentido. Indicado por la dirección de la flecha.



Notación con vectores

Las siguientes notaciones son las más típicas para representar a los vectores:

$\vec{A} = A_x\hat{x} + A_y\hat{y} + A_z\hat{z}$ con $\hat{x}, \hat{y}$ y $\hat{z}$ vectores unitarios.
$\vec{A} = A_x\hat{i} + A_y\hat{j} + A_z\hat{k}$ con $\hat{i}, \hat{j}$ y $\hat{k}$ vectores unitarios.
$\vec{A} = A_x e_x + A_y e_y + A_z e_z$ con e_x, e_y y e_z vectores unitarios.

Operaciones básicas entre vectores

La suma de vectores

Sean los vectores

$$\vec{A} = A_x\hat{i} + A_y\hat{j} + A_z\hat{k}$$

la suma se define como

$$\begin{aligned}\vec{A} + \vec{B} &= \left(A_x\hat{i} + A_y\hat{j} + A_z\hat{k}\right) + \left(B_x\hat{i} + B_y\hat{j} + B_z\hat{k}\right) \\ &= \left(A_x + B_x\right)\hat{i} + \left(A_y + B_y\right)\hat{j} + \left(A_z + B_z\right)\hat{k}\end{aligned}$$

La resta de vectores

$$\begin{aligned}\vec{A} - \vec{B} &= \left(A_x\hat{i} + A_y\hat{j} + A_z\hat{k}\right) - \left(B_x\hat{i} + B_y\hat{j} + B_z\hat{k}\right) \\ &= \left(A_x - B_x\right)\hat{i} + \left(A_y - B_y\right)\hat{j} + \left(A_z - B_z\right)\hat{k}\end{aligned}$$

El producto escalar o producto punto

$$\begin{aligned}\vec{A}\cdot\vec{B} &= \left(A_x\hat{i} + A_y\hat{j} + A_z\hat{k}\right)\cdot\left(B_x\hat{i} + B_y\hat{j} + B_z\hat{k}\right) \\ &= \left(A_xB_x\right) + \left(A_yB_y\right) + \left(A_zB_z\right)\end{aligned}$$

donde para este producto hay que considerar la siguiente convención

$$\begin{aligned}\vec{A}\cdot\vec{B} &= \left(A_x\hat{i} + A_y\hat{j} + A_z\hat{k}\right)\cdot\left(B_x\hat{i} + B_y\hat{j} + B_z\hat{k}\right) \\ &= A_x\hat{i}\left(B_x\hat{i} + B_y\hat{j} + B_z\hat{k}\right) + A_y\hat{j}\left(B_x\hat{i} + B_y\hat{j} + B_z\hat{k}\right) + A_z\hat{k}\left(B_x\hat{i} + B_y\hat{j} + B_z\hat{k}\right) \\ &= A_xB_x\left(\hat{i}\cdot\hat{i}\right) + A_xB_y\left(\hat{i}\cdot\hat{j}\right) + A_xB_z\left(\hat{i}\cdot\hat{k}\right) + A_yB_x\left(\hat{j}\cdot\hat{i}\right) + A_yB_y\left(\hat{j}\cdot\hat{j}\right) + A_yB_z\left(\hat{j}\cdot\hat{k}\right) + \\ &\quad + A_zB_x\left(\hat{k}\cdot\hat{i}\right) + A_zB_y\left(\hat{k}\cdot\hat{j}\right) + A_zB_z\left(\hat{k}\cdot\hat{k}\right) \\ &= \left(A_xB_x\right) + \left(A_yB_y\right) + \left(A_zB_z\right)\end{aligned}$$

donde para este producto hay que considerar la siguiente convención

$\hat{i}\cdot\hat{i} = 1, \hat{j}\cdot\hat{j} = 1, \hat{k}\cdot\hat{k} = 1$

En principio podemos observar que bajo esta definición el producto escalar entre dos vectores se realiza como si estuviéramos multiplicando dos polinomios

$$\begin{aligned}\vec{A}\cdot\vec{B} &= \left(A_x\hat{i} + A_y\hat{j} + A_z\hat{k}\right)\cdot\left(B_x\hat{i} + B_y\hat{j} + B_z\hat{k}\right) \\ &= A_x\hat{i}\left(B_x\hat{i} + B_y\hat{j} + B_z\hat{k}\right) + A_y\hat{j}\left(B_x\hat{i} + B_y\hat{j} + B_z\hat{k}\right) + A_z\hat{k}\left(B_x\hat{i} + B_y\hat{j} + B_z\hat{k}\right) \\ &= A_xB_x\left(\hat{i}\cdot\hat{i}\right) + A_xB_y\left(\hat{i}\cdot\hat{j}\right) + A_xB_z\left(\hat{i}\cdot\hat{k}\right) + A_yB_x\left(\hat{j}\cdot\hat{i}\right) + A_yB_y\left(\hat{j}\cdot\hat{j}\right) + A_yB_z\left(\hat{j}\cdot\hat{k}\right) + \\ &\quad + A_zB_x\left(\hat{k}\cdot\hat{i}\right) + A_zB_y\left(\hat{k}\cdot\hat{j}\right) + A_zB_z\left(\hat{k}\cdot\hat{k}\right) \\ &= \left(A_xB_x\right) + \left(A_yB_y\right) + \left(A_zB_z\right)\end{aligned}$$

El producto vectorial

Una operación de gran utilidad dentro de algunas áreas de ciencias e ingenierías. El producto vectorial permite encontrar un vector perpendicular a los dos vectores involucrados:

$$\begin{gathered}\vec{A}x\vec{B} = \left(A_x\hat{i} + A_y\hat{j} + A_z\hat{k}\right) x \left(B_x\hat{i} + B_y\hat{j} + B_z\hat{k}\right) \\ = A_x\hat{i}\mathrm{x}\left(B_x\hat{i} + B_y\hat{j} + B_z\hat{k}\right) + A_y\hat{j}\mathrm{x}\left(B_x\hat{i} + B_y\hat{j} + B_z\hat{k}\right) + A_z\hat{k}\mathrm{x}\left(B_x\hat{i} + B_y\hat{j} + B_z\hat{k}\right) \\ = A_xB_x\left(\hat{i}\mathrm{x}\hat{i}\right) + A_xB_y\left(\hat{i}\mathrm{x}\hat{j}\right) + A_xB_z\left(\hat{i}\mathrm{x}\hat{k}\right) + A_yB_x\left(\hat{j}\mathrm{x}\hat{i}\right) + A_yB_y\left(\hat{j}\mathrm{x}\hat{j}\right) + A_yB_z\left(\hat{j}\mathrm{x}\hat{k}\right) + \\ + A_zB_x\left(\hat{k}\mathrm{x}\hat{i}\right) + A_zB_y\left(\hat{k}\mathrm{x}\hat{j}\right) + A_zB_z\left(\hat{k}\mathrm{x}\hat{k}\right)\end{gathered}$$

ahora las restricciones son presentadas como sigue:

$$\begin{gathered}\hat{i}\mathrm{x}\hat{i} = 0; \hat{j}\mathrm{x}\hat{j} = 0; \hat{k}\mathrm{x}\hat{k} = 0 \\ \hat{i}\mathrm{x}\hat{j} = \hat{k}; \hat{j}\mathrm{x}\hat{k} = \hat{i}; \hat{k}\mathrm{x}\hat{i} = \hat{j} \\ \hat{j}\mathrm{x}\hat{i} = -\hat{k}; \hat{k}\mathrm{x}\hat{j} = -\hat{i}\hat{i}\mathrm{x}\hat{k} = -\hat{j}\end{gathered}$$

aplicando esto tendremos:

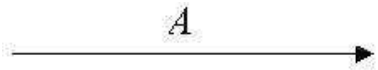
$$\begin{gathered}\vec{A}x\vec{B} = A_xB_y\hat{k} - A_xB_z\hat{j} - A_yB_x\hat{k} + A_yB_z\hat{i} + A_zB_x\hat{j} - AzBy\hat{i} \\ = \left(A_yB_z - A_zB_y\right)\hat{i} - \left(A_xB_z - A_zB_x\right)\hat{j} + \left(A_xB_y - A_yB_x\right)\hat{k}\end{gathered}$$

Esta expresión vectorial se puede también se puede expresar mediante el siguiente determinante:

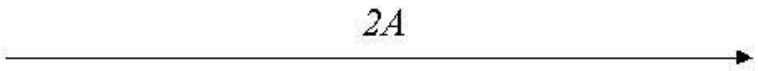
$$\begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ A_x & A_y & A_z \\ B_x & B_y & B_z \end{vmatrix} = \left(A_yB_z - A_zB_y\right)\hat{i} - \left(A_xB_z - A_zB_x\right)\hat{j} + \left(A_xB_y - A_yB_x\right)\hat{k}$$

Producto de vectores por escalares

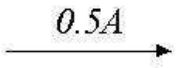
Cuando un vector es multiplicado por una cantidad escalar lo que se modifica es la magnitud del vector, haciéndolo más grande o más pequeño. Por ejemplo, si este es el vector A:



dos veces el vector, 2A tendríamos:



únicamente aumento de tamaño. Por el contrario, si multiplicamos por un escalar r¡1, donde r es el escalar, tendríamos un vector más pequeño, por ejemplo si multiplicamos por r = 1/2



El Lema de Inversión de Matriz

Sean A y B dos matrices positivas definidas de dimensiones M x M relacionadas por

$$A = B^{-1} + CD^{-1}C^H \tag{C.1}$$

en donde D es otra matriz definida positiva de N x M, y C es una matriz de M x N. de acuerdo con el lema de inversión de matriz, podemos expresar la inversa de la matriz A de la siguiente manera:

$$A^{-1} = B - BC(D + C^H BC)^{-1}C^H B \tag{C.2}$$

El lema de inversión de matriz establece que si estamos usando una matriz A definida por la ecuación C.1 se puede determinar su inversa A^{-1} utilizando la ecuación C.2.

Asumiendo que la matriz de correlación $\Phi(k)$ es definida, positiva y por lo tanto no singular, podríamos aplicar el lema de inversión de matriz a la ecuación recursiva C.2. Haciendo primero las siguientes definiciones:

$$\begin{aligned} A &= \Phi(k) \\ B^{-1} &= \lambda\Phi(k-1) \\ C &= u(k) \\ D &= 1 \end{aligned} \tag{C.3}$$

Por conveniencia de cálculo, hagamos

$$P(k) = \Phi^{-1}(k) \tag{C.4}$$

y

$$\mathrm{k}(k) = \frac{\lambda^{-1}P(k-1)u(k)}{1 + \lambda^{-1}u^H(k)P(k-1)u(k)} \tag{C.5}$$

Entonces, sustituyendo dichas definiciones en el lema de inversión de matriz de la ecuación C.4 obtenemos la siguiente ecuación recursiva para la inversa de la matriz de correlación:

$$\Phi^{-1}(k) = \lambda^{-1}\Phi^{-1}(k-1) = \frac{\lambda^{-2}\Phi^{-1}(k-1)u(k)u^H(k)\Phi^{-1}(k-1)}{1 + \lambda^{-1}u^H(k)\Phi^{-1}(k-1)u(k)} \tag{C.6}$$

Utilizando estas definiciones, podemos reescribir la ecuación C.5 de la siguiente manera:

$$P(k) = \lambda^{-1}P(k-1) - \lambda^{-1}\mathrm{k}(k)u^H(k)P(k-1) \tag{C.7}$$

Esta matriz P(k) de M x M es conocida como la matriz de correlación inversa. El vector k(k) de M x 1 se conoce como el vector de ganancia.

D. CONSIDERACIONES DE LA HERRAMIENTA DE PUNTO FIJO DE MATLAB

*D.1. Comparación de los objetos **fi** con los tipos de datos enteros en C*

A continuación se comparan el rango numérico de los tipos de datos enteros de objetos **fi** con los rangos numéricos mínimos de los tipos de datos enteros del ANSI C [HJ91].

D.1.1. Tipos de Datos Enteros de ANSI C

La siguiente tabla D.1 muestra los rangos mínimos de los tipos de datos enteros del ANSI C. Los rangos enteros pueden ser más grandes o iguales a los mostrados a continuación, pero no pueden ser menores. El rango de un tipo *long* debe ser mayor o igual a el rango de un tipo *int*, el cual debe ser mayor o igual a el rango de un tipo *short*.

Los rangos mínimos del ANSI C son lo suficientemente largos para incluir representaciones de números complemento a uno o signo/magnitud, pero no para incluir números complemento a dos. En las representaciones de complemento a uno y signo/magnitud, un entero con signo de n bits tiene un rango desde -2^{n-1} hasta $2^{n-1}-1$. En ambas representaciones un número igual de números negativos y positivos son representados, y el cero es representado dos veces.

Tipo Entero	Mínimo	Máximo
signed char	-127	127
unsigned char	0	255
short int	-32,767	32,767
unsigned short	0	65,535
int	-32,767	32,767
unsigned int	0	65,535
long int	-2,147,483,647	2,147,483,647
unsigned long	0	4,294,967,295

Tab. D.1: Rangos mínimos y máximos de los Tipos de Datos Enteros del lenguaje de programación ANSI C

D.1.2. Tipos de Datos Enteros de fi

A continuación se muestran los rangos numéricos de los tipos de datos enteros de los objetos **fi**, en particular aquellos equivalentes a los tipos de datos enteros de C. Los rangos

son los suficientemente largos para poder representar números en complemento a dos, la cual es la única técnica de codificación binaria soportada por el Fixed Point Toolbox. En la representación de complemento a dos, un entero de n bits con signo tiene un rango desde -2^{n-1} hasta $2^{n-1}-1$. Un entero sin signo de n bits tiene un rango desde 0 hasta -2^{n-1}. El lado negativo del rango tiene un valor más que el lado positivo, y el cero es representado únicamente.

Constructor	Signo	Longitud de Palabra	Longitud Fraccional	Mínimo	Máximo	Equivalente ANSI C
fi(x,1,n,0)	Si	n (2 a 65,535) a	0	-2^{n-1}	$2^{n-1}-1$	N/A
fi(x,0,n,0)	No	n (2 a 65,535) a	0	0	$2^{n}-1$	N/A
fi(x,1,8,0)	Si	8	0	-128	127	signed char
fi(x,0,8,0)	No	8	0	0	255	unsigned char
fi(x,1,16,0)	Si	16	0	-32,768	32,767	short int
fi(x,0,16,0)	No	16	0	0	65,535	unsigned short
fi(x,1,32,0)	Si	32	0	-2,147,483,648	2,147,483,647	long int
fi(x,0,32,0)	No	32	0	0	4,294,967,295	unsigned long

Tab. D.2: Rangos mínimos y máximos de los Tipos de Datos Enteros de objetos **fi**

D.1.3. Manejo de Sobreflujo

Las sumas y multiplicaciones con objetos **fi** producen resultados que pueden representarse exactamente por un objeto **fi** con longitudes de palabra de hasta 65,535 bits o de la memoria disponible en la computadora. Esto no aplica para la división, sin embargo, muchas relaciones resultan en expresiones binarias infinitas. Nosotros podemos realizar divisiones con objetos **fi** utilizando la función divide, la cual requiere que se especifique explícitamente el tipo numérico del resultado.

Las condiciones bajo las cuales un objeto **fi** presenta un sobreflujo y por consecuencia el resultado producido es determinado por el objeto fimath asociado. Nosotros podemos especificar ciertas características individuales para las sumas (incluyendo las restas) y productos. Para hacerlo nos podemos orientar en la siguiente tabla [Mat06]:

D.2. Implementación en Punto Fijo del Algoritmo FK

Para demostrar como se emplea la herramienta de punto fijo de matlab se muestra a continuación la manera en que se implemento el algoritmo rápido de Kalman (FK) para el primer canal de fase mínima.

Propiedades del Objeto fimath para el Manejo de Sobreflujo	Valor de la Propiedad	Descripción
OverflowMode	'saturate'	Los sobreflujos son saturados hacia el valor máximo o mínimo dentro del rango
	'wrap'	Los sobreflujos se truncan utilizando aritmética modulo si no se utiliza signo y utilizando truncamiento complemento a dos si se emplea signo
ProductMode	'Fullprecisión'	Los resultados de precisión completa se mantienen. No ocurre sobreflujo. Se entrega un error si la longitud de palabra resultante es mayor que la propiedad 'MaxProductLength'.
	'KeepLSB'	Se mantiene la parte menos significativa del producto. Los resultados a precisión completa se mantienen, pero se puede presentar sobreflujo. La longitud de palabra resultante se determina por la propiedad ProductWordLength
	'KeepMSB'	Se mantienen los bits más significativos del producto. Se previene el sobreflujo, pero se puede perder precisión. La longitud de palabra resultante se determina por la propiedad ProductWordLength
	'Specifyprecisión'	Uno puede especificar tanto la longitud de palabra como la longitud de la parte fraccionaria del producto resultante

Como se ha mencionado, una vez que se cuenta con la implementación del algoritmo en punto flotante se procede a estudiar la dinámica de las variables que intervienen en el algoritmo para saber la cantidad necesaria de bits para representar tato la parte entera como la parte fraccionaria y evitar la aparición de sobreflujo. Este estudio de las dinámicas se realizo al rastrear los valores máximos y mínimos que se obtienen en la simulación de punto flotante obteniendo los resultados mostrados en la figura D.1, en la cual se puede apreciar el número de bits necesarios para representar los valores máximos y mínimos de cada una de las variables.

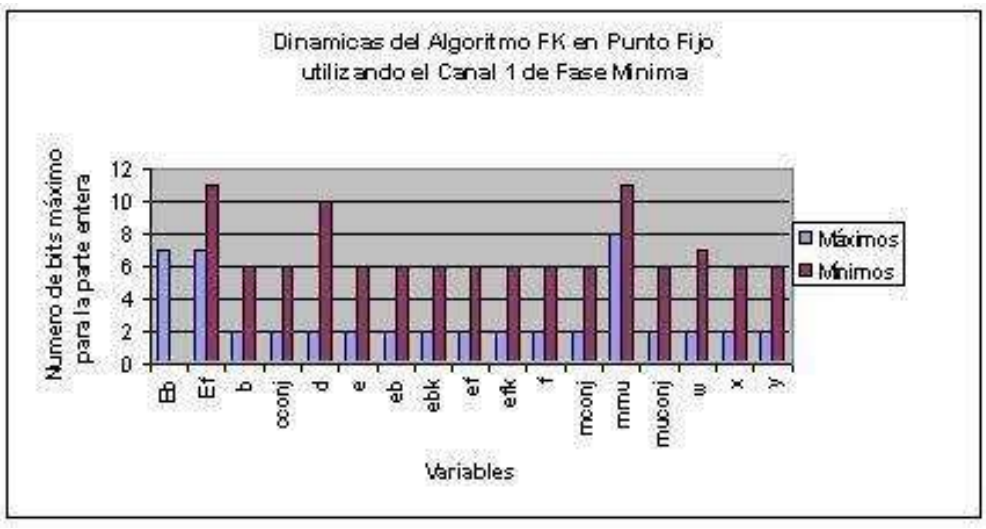


Fig. D.1: Gráfica del número de bits necesarios para las dinámicas de los canales de fase mínima.

Una vez que se tiene el estudio de las dinámicas, se procede a definir para cada una de las variables los objetos **fimath** y **numerictype**, que como se menciona en el presente apéndice controlan tanto la aritmética de punto fijo que se va a emplear en las operaciones que involucren las variables de punto fijo, como los atributos del tipo de datos y escalamiento de los objetos.

Para definir los atributos de cada objeto de punto fijo se emplea el comando **fi** de la siguiente manera:

- $[T, F] = definefi(1, bits, bits - 4, 0, 0)$;
- $[T_Eb, F_Eb]$ =definefi(1,bits,bits-7,0,0);
- $[T_Ef, F_Ef]$ =definefi(1,bits,bits-7,0,0);
- $[T_b, F_b]$ =definefi(1,bits,bits-2,0,0);
- $[T_cconj, F_cconj]$ =definefi(1,bits,bits-2,0,0);
- $[T_d, F_d]$ =definefi(1,bits,bits-1,0,0);
- $[T_e, F_e]$ =definefi(1,bits,bits-1,0,0);

- $[T_eb, F_eb]$ =definefi(1,bits,bits-3,0,0);
- $[T_ebk, F_ebk]$ =definefi(1,bits,bits-2,0,0);
- $[T_ef, F_ef]$ =definefi(1,bits,bits-2,0,0);
- $[T_efk, F_efk]$ =definefi(1,bits,bits-2,0,0);
- $[T_f, F_f]$ =definefi(1,bits,bits-2,0,0);
- $[T_mconj, F_mconj]$ =definefi(1,bits,bits-2,0,0);
- $[T_mmu, F_mmu]$ =definefi(1,bits,bits-1,0,0);
- $[T_muconj, F_muconj]$ =definefi(1,bits,bits-2,0,0);
- $[T_w, F_w]$ =definefi(1,bits,bits-2,0,0);
- $[T_y, F_y]$ =definefi(1,bits,bits-2,0,0);
- $[T_aux, F_aux]$ =definefi(1,bits,bits-9,0,0);

En ocasiones dado que tanto matlab como la herramienta de punto fijo requieren de mucha memoria para alojar todo el ambiente generado por la definición de los objetos de punto fijo, se ha encontrado conveniente generar un número menor de objetos **fimath** y **numerictype** de modo que cuando dos o más variables tengan la misma dinámica, estas compartan los objetos de punto fijo para regir su aritmética en las operaciones como los atributos de tipo de dato y escalamiento. Entonces siguiendo la recomendación citada anteriormente la definición de los objetos de punto fijo **fimath** y **numerictype** para el algoritmo quedaría de la siguiente manera:

- $[T, F] = definefi(1, bits, bits - 4, 0, 0)$;
- $[T7, F7]$ =definefi(1,bits,bits-7,0,0);
- $[T2, F2]$ =definefi(1,bits,bits-2,0,0);
- $[T1, F1]$ =definefi(1,bits,bits-1,0,0);
- $[T3, F3]$ =definefi(1,bits,bits-3,0,0);
- $[T_aux, F_aux]$ =definefi(1,bits,bits-9,0,0);

Como se puede observar claramente se reduce la cantidad de objetos de punto fijo que se van a emplear durante la ejecución del algoritmo; lo que implica un ahorro en espacio de memoria, pero aun así se garantiza el respeto de las reglas de aritmética y atributos de los objetos de punto fijo resultantes de las operaciones ya que estas se siguen rigiendo por los objetos **fimath** y **numerictype** creados, tal y como se muestra en el algoritmo mostrado en la tabla a continuación:

```
M = length(x);
u = fi(zeros(N,1),T,F);
lambda = fi(lambda,T2,F2);
cconj = fi(u,T2,F2);
w_fp = fi(zeros(1,N),T2,F2);
w = zeros(1,N);
W = zeros(M,N);
y = zeros(1,M);
e = y;
ef_fp = fi(0,T2,F2);
efk_fp = fi(0,T2,F2);
Db = zeros(1,M);
f = fi(u',T2,F2);
b = fi(f,T2,F2);
Ef_fp = fi(delta,T7,F7);
Efn_fp = fi(0,T7,F7);
Eb = fi(0,T7,F7);
for n=1:M
ef_fp = F2.sub(fi(x(k),T,F),producto(f,u));
f = F2.add(f,F_aux.mpy(cconj',ef_fp));
efk_fp = F2.sub(fi(x(k),T,F),producto(f,u));
Efn_fp = F7.add(F7.mpy(lambda,Ef_fp),F_aux.mpy(ef_fp,efk_fp));
[mmu] = F1.sub([0;cconj],F.mpy(T.divide(efk_fp,Efn_fp),[-1;f']));
mconj = mmu(1:N);
muconj = mmu(N+1);
if n ¿N
xb = fi(x(k-N),T,F);
else
xb = fi(0,T,F);
end
Ef_fp = Efn_fp;
u = [fi(x(k),T,F);u(1:N-1)];
ebk = F2.sub(xb,producto(b,u));
cconj = T2.divide(F.add(mconj,F.mpy(muconj,b')),F.sub(fi(1,T2,F2),F.mpy(muconj,ebk)));
b = F2.add(b,F_aux.mpy(cconj',ebk));
eb = F3.add(xb,producto(b,u));
Eb = F7.add(Eb,F.mpy(ebk,eb));
y_fp = producto(w_fp,u);
y(k) = y_fp.data;
e_fp = F1.sub(fi(d(k),T1,F1),y_fp);
e(k) = e_fp.data;
w_fp = F2.add(w_fp,F_aux.mpy(cconj',e_fp));
w = w_fp.data;
W(k,:) = w;
Db(k) = ((a-w)*(a-w)')/(a*a');
end
```

Tab. D.3: Algoritmo Rápido de Kalman (Fast Kalman) Implementado en Punto Fijo.

BIBLIOGRAFÍA

[Aam01] T. Aamodt. Floating point to fixed-point compilation and embedded architectural support. Master's thesis, University of Toronto, January 2001.

[AC99] Tor Aamodt and Paul Chow. Numerical error minimizing floating-point to fixed-point ansi c compilation. *In 1st Workshop on Media Processors and Digital Signal Processing*, November 1999.

[AC00] Tor Aamodt and Paul Chow. Embedded isa support for enhanced floating-point to fixed-point ansi c compilation. *In 3rd International Conference on Compilers Architectures and Synthesis for Embedded Systems*, November 2000.

[Alc86] R. Alcantara. *Implantation d´algorithmes Rapides sur des Processeurs de Traitement du Signal.* PhD thesis, ENTS, 1986.

[Ale86] S. T. Alexander. *Adaptive Signal Processing: Theory and Applications.* Springer, Berlin, 1986.

[Bel76] P. A. Bello. Characterization of randomly time invariant linear channels. *IEEE Trans. on Communication Systems*, vol. 23:1043–1047, Dec. 1976.

[BG88] A. Benallal and A. Gilloire. A new method to stabilize fast rls algorithms based on a first order model of the propagation of numerical error. *Proc. Internat. Conf. Acoust. Speech Signal Process*, pages 1373–1376, 1988.

[BM86] J. L. Botto and G. V. Moustakides. Stabilization of fast recursive least-squares transversal filters. *INRIA-IRISA Rapports de Recherche*, No. 570, September 1986.

[CG86] C. F.Ñ. Cowan and P. M. Grant. *Adaptive Filters.* Prentice Hall, 1986.

[CH06] RC Cofer and B. Harding. Fixed point dsp and algorithm implementation. *DSP Design Line*, 2006.

[CK84] J. M. Cioffi and T. Kailath. Fast recursive least-squares transversal filters for adaptive filtering. *IEEE Trans. Acoust. Speech Signal Process.*, ASSP-32:304–337, 1984.

[CMK83] G. Carayannis, G. Manolakis, and N. Kalouptsidis. A fast sequential algorithm for least-squares-filtering and prediction. *IEEE Trans. Acoust. Speech Signal Process.*, ASSP-31:1394–1402, 1983.

[CRS+99] R. Cmar, L. Rijnders, P. Schaumont, S. Vernalde, and I. Boslens. A methodology and design environment for dsp asic fixed point refinement. *IMEC*, 1999.

[CY03] N. Castañeda and J. A. Ybañez. Evaluación e implantación de algoritmos de filtrado adaptable para la igualación de canal en sistemas de comunicación digital. Master's thesis, UNAM, 2003.

[Esc97] L. Escobar. Evaluación e implantación de algoritmos de filtrado adaptable en sistemas de comunicación digital. Master's thesis, UNAM, 1997.

[FB98] B. Farhang-Boroujeny. *Adaptive Filters, Theory and Applications*. John Wiley and Sons, 1998.

[FG86] P. Fabre and C. Gueguen. Improvement for the fast recursive least-squares algorithm via normalization: A comparative study. *IEEE Trans. Acoust. Speech Signal Process.*, ASSP-34:296–308, December 1986.

[FL78] D. Falconer and L. Ljung. Application of fast kalman estimation to adaptive equalization. *IEEE Transactions on Communications*, COM-26:1439–1446, 1978.

[Fre99] Roger L. Freeman. *Fundamentals of Telecommunications*. John Wiley and Sons, New York, USA, 1st edition, 1999.

[Gal08] H. Galindo. Implementación de algoritmos de filtrado adaptable en aritmética de punto fijo con matlab. UNAM, Laboratorio de Procesamiento de Señales y Tratamiento de la Información, 2008.

[GM97] J. Gosling and H. McGilton. *The Java Language Environment*. Sun Microsystems, Inc., 1997.

[Hay84] Simon S. Haykin. *Introduction to Adaptive Filters*. Prentice-Hall, NY, 1a edition, 1984.

[Hay91] S. Haykin. *Adaptive Filter Theory*. McGraw-Hill,NY, 1a edition, 1991.

[Hay96] S. Haykin. *Adaptive Filter Theory*. Prentice-Hall, NJ, 3a edition, 1996.

[HJ91] Samuel P. Harbison and Guy L. Steele Jr. *C: A reference manual*. Prentice Hall, 1991.

[KCLM01] H. Keding, M. Coors, O. Luthje, and H. Meyr. Fast bit-true simulation. *Design Automation Conference 2001 (DAC-01)*, June 01.

[Kea96] R. Kearfott. Interval computations: Introduction, uses, and resources. *Eurromath Bulletin*, 2(1):95–112, 1996.

[KKS97] K. Kum, J. Kang, and W. Sung. A floating-point to fixed-point c converter for fixed-point digital signal processors. *In Proc. 2nd SUIF Compiler Workshop*, August 1997.

[KKW98] S. Kim, K. Kum, and S. Wonyong. Fixed-point optimization utility for c and c++ based digital signal processing programs. *IEEE Transactions on Circuits and Systems II*, 45(11), November 1998.

[Lee80] D. T. L. Lee. *Canonical Ladder Form Realizations and Fast Estimation Algorithms, Ph.D Dissertation.* PhD thesis, Department of Electrical Engineering, Stanford University, Stanford CA, August 1980.

[Lin84] D. Lin. On digital implementation of the fast kalman algorithms. *IEEE Trans. Acoust. Speech Signal Process.*, ASSP-32:998–1005, October 1984.

[LL85] S. Ljung and L. Ljung. Error propagation properties on recursive least-squares adaptation algorithms. *Automática*, Vol. 21, No. 2:157–167, 1985.

[LMF78] L. Ljung, M. Morf, and D. Falconer. Fast calculation of gain matrices for recursive estimation schemes. *International J. Control*, 27:1–19, 1978.

[LW90] Kevin W. Leary and William Waddington. Dsp/c: A standard high level language for dsp and numeric processing. *In Proceedings of the ICASSP*, pages 1065–1068, 1990.

[Mat06] Mathworks Inc., 3 Apple Hill Drive, Natick, MA. *Fixed Point Toolbox User's Guide*, 2004-2006.

[Men03] D. Menard. Méthodologies de conversion automatique en virgule fixe pour les applications de traitement du signal. Technical report, Équipe de recherche RD - IRISA/INRIA ENSSAT - Université de Rennes1, 31 mars - 4 avril 2003.

[MG73] J. D. Markel and A. H. Gray. On autocorrelation equations as applied to speech analysis. *IEEE Trans.*, AU-21, No. 5:69–79, April 1973.

[MG75] J. D. Markel and A. H. Gray. Fixed-point implementation algorithms for a class of orthogonal polynomial filter structures. *IEEE Trans.*, ASSP-23, No. 5:486–494, October 1975.

[MKH77] S. K. Mitra, P. S Kamat., and D. C. Huey. Cascaded lattice realization of digital filters. *Circuit Theory and Applications.*, Vol 5:3–11, 1977.

[MLNV77] M. Morf, D. T. Lee, J. R. Nickolls, and A. Vieira. A classification of algorithms for arma models and ladder realizations. *Proceedings IEEE International Conference Acoustics, Speech, and Signal Processing (ICASSP)*, pages 13–19, April 1977.

[Mor74] M. Morf. *Fast Algorithms for Multivariable Systems, PhD Dissertation.* PhD thesis, Stanford University, 1974.

[Mou89] G. V. Moustakides. Correcting the instability due to the finite precision of the fast kalman identification algorithms. *Signal Processing*, Vol. 18, No. 1:33–42, September 1989.

[MVH+97] M.Willems, V.Bursgens, H.Keding, T.Grotker, and H.Meyr. System level fixed-point design based on an interpolative approach. *Proc.of the DAC*, 1997.

[PM96] J.G. Proakis and D.G. Manolakis. *Digital Signal Processing: Principles, algorithms and applications.* Prentice-Hall Inc., USA, 1996.

[Pro95] J. G. Proakis. *Digital Communications*. McGraw-Hill, New York, USA, 3a. edition, 1995.

[Pro01] J. G. Proakis. *Digital Communications*. McGraw-Hill,NY, 4a. edition, 2001.

[Sha79] K. S. Shanmugan. *Digital and Analog Communication Systems*. John Wiley and Sons., 1a edition, 1979.

[SK88] D. T. M. Slock and T. Kailath. Numerically stable fast recursive least squares transversal filters. *Proceedings IEEE International Conference Acoustics, Speech, and Signal Processing (ICASSP) 88 Conf. (New York)*, pages 1365–1368, April 1988.

[SK91] D. T. M. Slock and T. Kailath. Numerically stable fast transversal filters for recursive least squares adaptive filtering. *IEEE Trans. Acoust. Speech Signal Process.*, ASSP-39:92–113, January 1991.

[SK95] W. Sung and K. Kum. Simulation-based word-length optimization method for fixed-point digital signal processing systems. *IEEE Transactions on Signal Processing*, 43:3087–3090, Dec.1995.

[SKW95] S.Kim, K.Kum, and W.Sung. Fixed-point optimization utility for c and c++ based digital signal processing programs. *Workshop on VLSI and Signal Processing*, pages pp.197–206, Nov. 1995.

[SR92] H. Schütze and Z. Ren. Numerical characteristics of fast recursive least squares transversal adaption algorithms - a comparative study. *Elsevier, Signal Processing.*, 0165-1684/92:317–332, 1992.

[Syn00] Synopsys Inc. *Synopsys Accelerates System-Level C-Based DSP Design With Co-Centric Fixed-Point Designer Tool*, April 2000.

[TFJ96] J. R. Treichler, I. Fijalkow, and C. R. Johnson Jr. Fractionally spaced equalizers. *IEEE Signal Processing Magazine*, pages 65–81, May. 1996.

[TJL87] J. R. Treichler, C. R. Johnson Jr., and M. G. Larimore. *Theory and design of adaptive filters*. John Wiley and Sons, 1987.

[Tor97] J. E. Torres. Estudio y comparación de algoritmos de filtrado adaptable para la igualación de canal. Master's thesis, UNAM, 1997.

[Tou99] J. Tourielles. *Conception d'architectures pour le traitement du signal en précision finite*. PhD thesis, Université de Rennes I, Jan 99.

[WBM97] M. Willems, V. Bursgens, and H. Meyr. Fridge: Floating-point programming of fixed-point digital signal processors. *International Conference on Signal Processing Applications and Technology, (ICSPAT'97)*, 1997.

[Wid85] B. Widrow. *Adaptive Signal Processing*. Prentice Hall, 1985.

[WM76] B. Widrow and J. M. McCool. Comparison of adaptive algorithms based on the methods of steepest descent and random search. *IEEE Trans. Antennas and Propagation*, AP-24:615–637, Sept. 1976.

[Zar06] Houman Zarrinkoub. Fixed-point signal processing with matlab® and simulink®. In *Signal Processing Toolboxes*. Mathworks, 2006.

INDICE DE FIGURAS

INDICE DE TABLAS

Glosario

A

algoritmo Del latín, dixit algorithmus y éste a su vez del matemático persa Muhammad ibn Musa al-Jwarizmi, es una lista bien definida, ordenada y finita de operaciones que permite hallar la solución a un problema.

ASIC Circuito Integrado para Aplicaciones Específicas, o ASIC por sus siglas en inglés, es un circuito integrado hecho a la medida para un uso en particular, en vez de ser concebido para propósitos de uso general.

C

CFK Algoritmo CFK del inglés *Covariance Fast Kalman propuesto por Lin. [Lin84].*

CFTF Algoritmo CFTF del inglés *Corrected Fast Transversal Filter propuesto por Moustakides [Mou89].*

D

Desajuste Del inglés *Misadjustment, es una medida cuantitativa de la cantidad del valor final del error medio cuadrático.*

DSP Del acrónimo de Digital Signal Processor significa Procesador Digital de Señal, es un sistema basado en un procesador o microprocesador que posee un juego de instrucciones, un hardware y un software optimizados para aplicaciones que requieran operaciones numéricas a muy alta velocidad. [PM96].

F

FAEST Algoritmo FAEST propuesto por Carayannis. [CMK83].

filtrado Proceso de eliminar alguna información especifica de la señal.

FK Algoritmo rápido de Kalman conocido como FK por su nombre en inglés *Fast Kalman. [LMF78].*

FPGA Del inglés *Field Programmable Gate Array es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad se puede programar.*

FRLS De su nombre en inglés *Fast Recursive Least Squares hace referencia a los algoritmos rápidos de los mínimos cuadrados.*

FTF Acrónimo proveniente del inglés fast transversal filter, la traducción propuesta al español de este anglicismo es: filtro rápido transversal.

Función de Transferencia Modelo matemático que entrega la respuesta de un sistema a una señal de entrada o excitación exterior.

G

GFTF Algoritmo GFTF (Gain-Normalized Fast Transversal Filter) propuesto por Cioffi y Kailath. [CK84].

H

hardware Neologismo proveniente del inglés utilizado generalmente para describir componentes físicos de una tecnología.

I

ISI Del inglés *intersymbol interference, en comunicaciones, la distorsión resultante de la interferencia de los símbolos con los símbolos adyacentes.*

J

Java Lenguaje de programación desarrollado por SUN Microsystems para el desarrollo de aplicaciones en ambientes de redes distribuidas [GM97].

L

LSB Del acrónimo de Least Significant Bit significa Bit Menos Significativo, se refiere al bit que representa el valor mas pequeño de una palabra de longitud finita.

M

MADPR Del inglés *Multiplication Addition and Division Per Recursion, acrónimo que incida la cantidad de operaciones básicas por iteración de un algoritmo.*

MATLAB Abreviatura de Matrix Laboratory (laboratorio de matrices). Es un programa de análisis numérico creado por The MathWorks.

N

NFTF Algoritmo NFTF (Normalized Fast Transversal Filter) propuesto por Fabre y Gueguen. [FG86].

P

PARCOR Coeficientes de correlación parcial, también conocidos como coeficientes de reflexión y son empleados en estructuras lattice.

PSD Del acrónimo *Power Spectral Density, se refiere a la Densidad Espectral de Potencia para una señal aleatoria.*

R

Rango de convergencia Se conoce así al número de iteraciones requeridos por un algoritmo, para converger lo suficiente a una solución óptima.

Robustez Del inglés *Robustness, es una medida cuantitativa de los errores de estimación de un algoritmo.*

S

Seguimiento Del inglés *Tracking, se refiere a la propiedad de un algoritmo de rastrear variaciones estadísticas al trabajar en un medio no estacionario.*

SFTF Algoritmo SFTF (Stabilized Fast Transversal Filter) propuesto por Banallal y Gilliore. [BG88].

Simulink Herramienta adicional de Matlab para modelado, simulación y análisis de sistemas dinámicos.

software Palabra de origen anglosajón con la cual se denomina al conjunto de programas y procedimientos necesarios para hacer posible la realización de una tarea específica.

systemC Ratificado como estándar IEEE SystemC es un lenguaje desarrollado en standard C++ para extender el lenguaje con el uso de librerías de clase y es particularmente adecuado para el modelado de particionamiento de sistemas, para evaluar y verificar la asignación de bloques tanto para implementaciones de hardware o software.

T

Test de Student También conocido como test-t se llama así a cualquier hipótesis estadística en la cual la prueba resulte en una distribución de Student si la hipótesis nula es verdadera.

W

WSS Del inglés Wide Sense Stationary se refiere a un proceso aleatorio estacionario en sentido amplio.