UNIVERSIDAD NACIONAL AUTONÓMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÁ ELÉCTRICA
DEPARTAMENTO DE INGENIERIA EN
TELECOMUNICACIONES

CONTROL ACTIVO DEL RUIDO ACÚSTICO

T E S I S

PARA OBTENER EL GRADO DE:

INGENIERO EN TELECOMUNIACIONES

P R E S E N T A:

PÉREZ BARRAGÁN LUIS MIGUEL

DIRECTOR DE TESIS:

DR. BOHUMIL PŠENIČKA

Ciudad Universitaria ABRIL 2010

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

A mi Papá: Sin tu ayuda y apoyo no hubiera logrado completar esta tesis, GRACIAS!!

A mi Mamá: Tu apoyo durante toda mi carrera fue vital, gracias por estar siempre conmigo. Te amo!

A mis Hermana Vero: Gracias por todos tus consejos. Suerte en la especialidad!

Al Dr. Buhomil: Gracias por darme la oportunidad de trabajar con usted.

Documento hecho en con LaTeX

Índice general

Índice de figuras

Resumen

El problema de ruido acústico se ha vuelto cada vez más evidente con el incremento de máquinas y equipos no solo en la industria, sino también en la calle y el hogar. La manera tradicional de controlar el ruido usa técnicas pasivas como barreras y atenuadores para poder disminuir el ruido, sin embrago, estos son costosos e ineficientes al momento de trabajar con ruidos de frecuencia baja.

Esta tesis tiene como objetivo introducir y simular varios esquemas de Cancelación Activa del Ruido para ofrecer una mejor solución a los problemas de ruido, que aquellos ofrecidos por atenuadores o silenciadores pasivos. El trabajo estudiará cuatro esquemas (o algoritmos) para ser simulados y al final implementar uno de ellos en un procesador de señales digitales (DSP) de Texas Instruments TMS320C25.

La Cancelación Activa de Ruido (CAR) incluye un sistema electrónico y acústico o electrónico y mecánico que cancela la fuente primaria de ruido (no deseado), basado en el principio de superposición; específicamente, se genera un ruido secundario de igual amplitud y de fase contraria y se combina con el ruido primario, dando como resultado la cancelación de ambas fuentes de ruido. El sistema de Cancelación Activa de Ruido es eficiente en baja frecuencia donde los métodos pasivos no lo son, o son muy costosos o voluminosos para implementar.

Hoy en día las computadoras, con su gran capacidad de procesamiento y recursos, son capaces de implementar sistemas complejos y simularlos; reduciendo costos y ofreciendo la posibilidad de probar los sistemas de una manera más controlada. Por esto, en esta tesis se simularán cuatro algoritmos de Control Activo del Ruido, dos que trabajan en el dominio del tiempo y dos trabajando en el dominio de la frecuencia. Para ello, utilizaremos el programa MATLAB y el ambiente SIMULINK para generar los modelos que representen estos esquemas y analizar su comportamiento. Finalmente, se usará el simulador SIM25 para implementar una solución de control activo del ruido en el DSP TMS320C25.

Abstract

Acoustic noise problems become more and more evident with the increased number of machines and equipment that generates noise, not only in the industry environment, but also at the streets and home. The traditional way of controlling noise uses passive techniques such as enclosures, barriers and silencers to attenuate the undesired noise; however, these are relatively large, costly and ineffective at low frequencies.

This thesis objective is to introduce and simulate a number of different algorithms of Active Noise Cancellation in an effort to offer a better solution to noise problems than those offered by passive techniques. This research studies four algorithms to be simulated and one of them is going to be implemented on a Texas Instruments Digital Signal Processor (DSP) TMS320C25.

Active Noise Control (ANC) involves electronics and acoustic or mechanical system which cancels the primary noise source, based in the principle of superposition, specifically; a secondary noise is generated of equal amplitude and opposite phase which is combined with the primary source of noise, thus resulting in the cancellation of both noises. ANC system is efficient at low frequencies where passive methods do not, or they are expensive or large.

Nowadays computers have great processing capacity and resources, and are capable of simulate complex systems, reducing costs y giving the possibility of testing systems in a controlled environment. In this thesis we will simulate four algorithms of Active Noise Control, two of them working in the Time Domain and two of them in Frequency Domain. To this end, we will use the scientific software MATLAB and the SIMULNK environment to generate the models representing these algorithms and analyze their behavior. Lastly, we will use SIM25 simulator to implement an Active Noise Control solution on the TMS320C25 DSP.

Introducción

El Control Activo del Ruido (CAR) o Active Noise Control (ANC) es una técnica de control de ruido basada en la interferencia destructiva. Es decir, se genera una onda de la misma magnitud y sentido que el ruido a cancelar con un desfasamiento de 180°, de manera que al superponerse las dos ondas en el campo en el que se propagan da lugar a una cancelación del ruido por medio de la interferencia destructiva.

El **objetivo** de este trabajo es analizar los algoritmos mas comunes de CAR (tanto en el dominio del tiempo como en frecuencia) y probar su desempeño usando herramientas computacionales como MATLAB y SIMULINK; para finalmente implementar un cancelador de ruido acústico monocanal haciendo uso del DSP TMS320C25.

Capítulo 1

Antecedentes

Comúnmente las ondas no deseadas en un medio de propagación son canceladas usando técnicas pasivas que consisten principalmente en el uso de materiales resistivos para eliminar la misma. El Control Activo de Ruido (CAR) es una técnica diferente, que consiste en aprovechar el principio de superposición, generando una onda *anti-ruido* con la misma amplitud y de fase contraria a la onda que se desea atenuar, eliminando el ruido con una interferencia destructiva (o cancelación en fase) de manera eficiente.

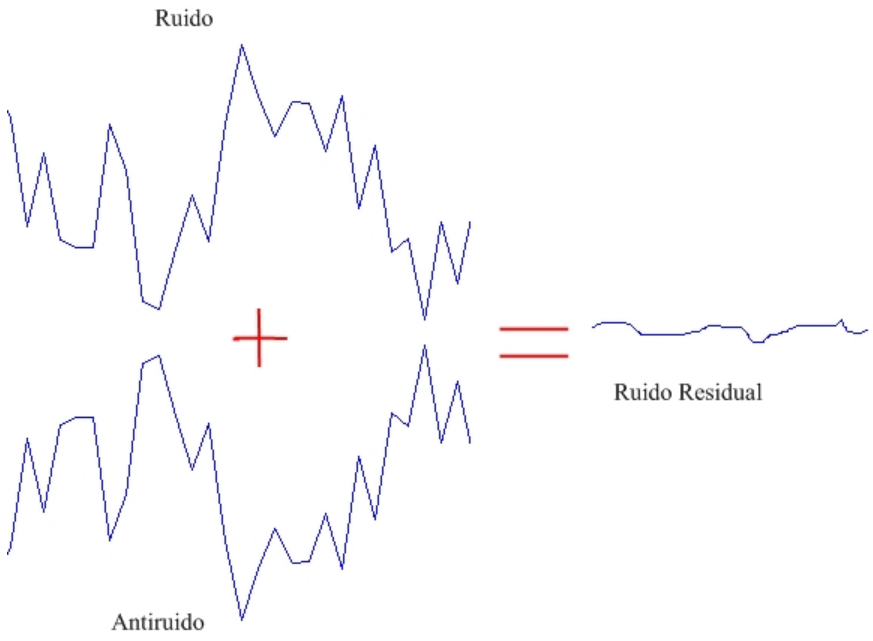


Figura 1.1: Concepto básico de CAR

La forma común de cancelar el ruido por medio de elementos pasivos tiene la gran desventaja de no ser tan efectivos en las frecuencias bajas. Las bajas frecuencias generalmente involucran una gran cantidad de energía y son muy difíciles de cancelar con técnicas ordi-

narias. CAR hace posible cancelar ruidos de baja frecuencia a un bajo costo. En el campo de procesamiento digital de señales existen sistemas adaptivos que implementan CAR. En estos sistemas los coeficientes de un filtro digital son ajustados para minimizar una señal de error que es la señal deseada menos una señal de control. Estos filtros adaptables pueden ser diseñados como filtros transversales o FIR (Finite Impulse Response), que son los mas usados.

1.1. Filtrado digital

El filtrado digital es parte del *Procesamiento digital de señales* y consiste de un sistema que realiza operaciones matemáticas en una señal muestreada, para alterar diversos aspectos de esta señal.

A diferencia de los filtros analógicos que operan con señales continuas y son típicamente conformados con elementos pasivos (resistores, capacitores e inductores), los filtros digitales operan con muestras de la señal o secuencias discretas y se implementan con lógica digital o microprocesadores enfocados al procesamiento digital de señales (DSP's).

Las ventajas de los filtros digitales sobre los analógicos son principalmente las siguientes:

- Las características de los filtros analógicos, particularmente los que contienen componentes activos, están sujetos a alteraciones y dependen de la temperatura. Los filtros digitales no sufren estos problemas y son extremadamente estables ante factores externos.
- Los filtros digitales pueden ser fácilmente diseñados, probados e implementados en un ordenador. Los analógicos pueden ser simulados, pero siempre hay que implementarlos a través de componentes discretos para ver su funcionamiento real.
- A diferencia de los filtros analógicos, los digitales pueden manejar con mucha precisión las bajas frecuencias. Como la tecnología de los Procesadores Digitales de Señales (DSP) va mejorando, el aumento de su velocidad permite que también sean aplicados en el campo de la radio frecuencia (muy altas frecuencias), la cual en el pasado era exclusivamente dominio de la tecnología analógica.
- Un filtro digital es programable, es decir, su funcionamiento está terminado por un programa almacenado en la memoria contigua al procesador. Esto significa que puede ser variado fácilmente sin afectar al hardware, mientras que la única manera de variar un filtro analógico es alterando el circuito.

El filtrado digital comúnmente consiste de un convertidor analógico-digital, un microprocesador (el DSP) y un convertidor digital-analógico. El software en el procesador implementa las operaciones matemáticas en las muestras proporcionadas por el convertidor A/D.

Existen dos clases de filtros digitales: filtros con respuesta infinita al impulso (IIR) y filtros con respuesta finita al impulso (FIR).

1.2. Transformada Z

La Transformada Z es usada para el *análisis de señales en tiempo discreto* de manera similar al uso de la Transformada de Laplace para analizar señales continuas en el tiempo. Así como la transformada de Laplace es usada para resolver ecuaciones diferenciales que representan un filtro analógico, la transformada Z puede ser empleada para resolver una ecuación en diferencias que represente un filtro digital.

Si consideramos una señal continua $x(t)$ idealmente muestreada:

$$x_s(t) = \sum_{k=-\infty}^{\infty} x(t)\delta(t - kT) \tag{1.1}$$

Donde $\delta(t - kT)$ es la función de impulso con un retraso kT y $T = 1/F_s$ es el periodo de muestreo. La función $x_s(t)$ será cero en cualquier instante excepto cuando $t = kT$. Si aplicamos la transformada de Laplace a $X_s(t)$ tenemos:

$$X_s(s) = \int_{-\infty}^{\infty} x_s(t)e^{-st}dt \tag{1.2}$$

$$= \int_{-\infty}^{\infty} \{x(t)\delta(t) + x(t)\delta(t - T) + \cdots\}e^{-st}dt \tag{1.3}$$

Usando la propiedad fundamental de la Delta de Dirac:

$$\int_{-\infty}^{\infty} f(t)\delta(t - kT) = f(kT) \tag{1.4}$$

Entonces $X_s(s)$ queda como

$$X_s(s) = \cdots + x(0) + x(T)e^{-s2t} + x(2T)e^{-s3t} + \cdots = \sum_{n=-\infty}^{\infty} x(nT)e^{-snT} \tag{1.5}$$

Si definimos $z = e^{sT}$, entonces:

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n} \tag{1.6}$$

Que es la ecuación que define la transformada Z *bilateral*. Observamos entonces que la transformada Z convierte una señal en tiempo discreto, que es una secuencia de valores reales, en

una representación compleja en el dominio de la frecuencia. La transformada Z unilateral de una secuencia $x[n]$ se define como

$$X(z) = \sum_{n=0}^{\infty} x[n] z^{-n} \tag{1.7}$$

Como la transformada Z es una función de variable compleja, es conveniente describirla e interpretarla usando el plano z. En este plano, el contorno correspondiente a $|z| = 1$ es un circulo de radio unitario, como se ilustra en la figura 1.2; este circulo es conocido como el circulo unitario. La transformada Z evaluada en el circulo unitario corresponde a la transformada de Fourier. Note que ω es el angulo entre el vector de un punto z en el circulo unitario y el eje real del plano z

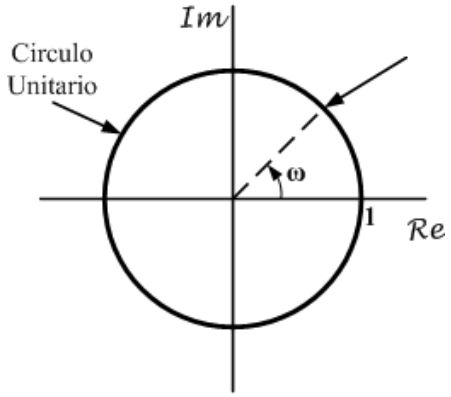


Figura 1.2: El circulo unitario en el plano z

La transformada Z de una secuencia puede ser representada como la relación de dos polinomios en el dominio z

$$X(z) = \frac{N(z)}{D(z)} \tag{1.8}$$

Los polos de $X(z)$ son las raíces del polinomio del denominador (valores de z que aseguran que $D(z) = 0$), así como los ceros de $X(z)$ son las raíces del polinomio del numerador (valores de z que aseguran que $N(z) = 0$). La región de convergencia (ROC) de la transformada Z es el rango de valor en en que la serie converge, la trasformada Z de una función $x[n]$ converge si es absolutamente sumable y la región de convergencia de la transformada Z se define

$$\sum_{n=-\infty}^{\infty} |x[n]||z|^{-n} < \infty \tag{1.9}$$

para todos los valores de z donde se cumple la desigualdad anterior. De manera que, si un valor $z = z_1$ está dentro de la ROC, todos los valores de z en el circulo definido por $|z| = |z_1|$ también estarán dentro del ROC.

1.3. Transformada Discreta de Fourier

La transformada discreta de Fourier (TDF) es un caso particular de la *Transformada de Fourier*, usada principalmente para el análisis de señales discretas en el dominio de la frecuencia. Transforma una función de entrada que sea de naturaleza discreta y con duración finita. Al usar la TDF, se debe de tomar en cuenta que la señal analizada sea un periodo de una señal periódica que se extiende de manera infinita; en caso de que esto no sea posible, se debe de usar una función ventana para reducir errores en el espectro de la señal.

La transformada de Fourier de una señal discreta y periódica esta dada por la siguiente ecuación:

$$X(k) = \sum_{n=0}^{N-1} x[n] W_N^{kn} \tag{1.10}$$

Y para la transformada inversa de Fourier (TDFI) tenemos:

$$x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X(k) W_N^{-kn} \tag{1.11}$$

Donde

$$W_N = e^{-j\frac{2\pi}{N}} \tag{1.12}$$

La relación entre estas dos ecuaciones se puede representar como

$$x[n] \overset{DFT}{\Longleftrightarrow} X(k) \tag{1.13}$$

Una descripción sencilla de las ecuaciones anteriores, es que los números complejos $X(k)$ representan la amplitud y la fase de las componentes senoidales de la señal de entrada $x[n]$. La TDF calcula $X(k)$ a partir de los valores discretos de $x[n]$; mientras que con la TDFI calculamos $x[n]$ como la suma de las componentes senoidales $X(k) * e^{-j\frac{2\pi kn}{N}}$ con frecuencia k/N ciclos por muestra. Si se escriben las ecuaciones como la forma anterior, hacemos uso extensivo de la formula de Euler para expresar senoidales en términos de exponenciales complejos, los cuales son mucho más fáciles de manipular.

Las principales propiedades de la TDF, son su periodicidad y simetría. Un periodo de una señal se extenderá de $f = 0$ a f_s, donde $f_s = 1/N$ es la frecuencia de muestreo. Además cuando la región entre 0 y f_s es examinada, podemos encontrar simetría alrededor del punto intermedio $0.5f_s$, es decir, la frecuencia de Nyquist. Esto se muestra en la figura 1.3.

Como ya se mencionó antes, la señal a transformar es una secuencia finita de números reales o complejos, lo cual hace de la transformada discreta de Fourier ideal para procesar información digital. La TDF es usada ampliamente en el procesamiento digital de señales para analizar y procesar señales digitalizadas, además de resolver ecuaciones diferenciales parciales, realizar operaciones de convolución o multiplicar números enteros muy grandes. Un factor clave que a permitido el desarrollo de todas estas aplicaciones es el hecho de que la TDF puede ser calculada de manera muy eficiente utilizando un algoritmo de **Transformada Rápida de Fourier**, algunos de los cuales discutiremos a continuación.

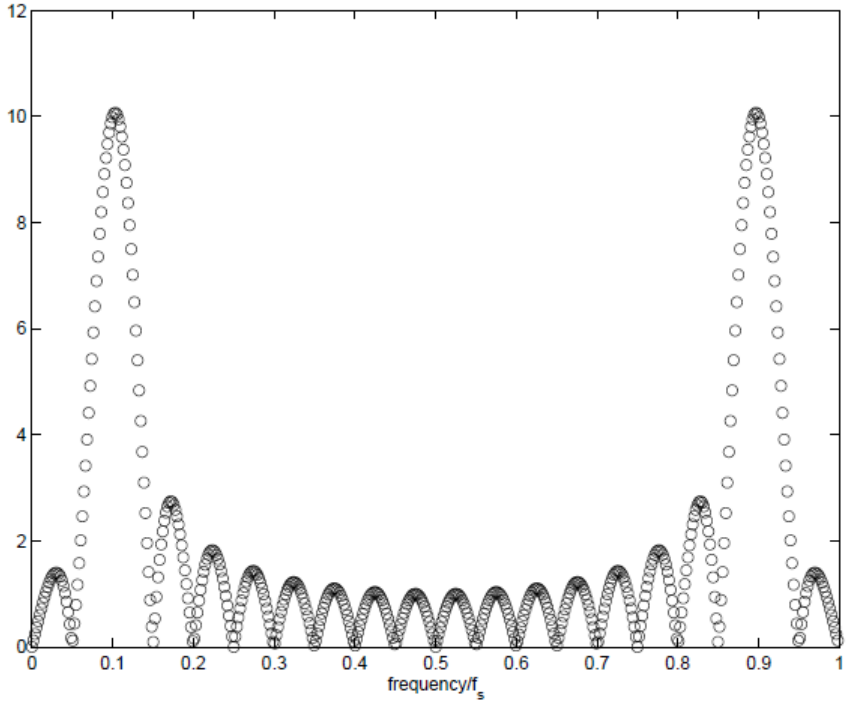


Figura 1.3: Simetría de la Transformada Discreta de Fourier.

1.3.1. Transformada Rápida de Fourier

Como vimos en la sección anterior, la Transformada Discreta de Fourier tiene un papel muy importante en el análisis, diseño e implementación de algoritmos y sistemas de procesamiento de señales discretos en el tiempo. Las propiedades básicas de la transformada discreta de Fourier la hacen muy útil al analizar y diseñar sistemas en el Dominio de la Frecuencia[20]. Esto se vuelve aun mas relevante debido a la existencia de algoritmos eficientes para el cálculo de de la TDF. De manera colectiva, estos algoritmos eficientes se les conoce como *algoritmos de Transformada Rápida de Fourier* o *Fast Fourier Transform (FFT).*

Debido a que la TDF es idéntica a las muestras de la transformada de Fourier en frecuencias igualmente espaciadas. Como consecuencia de esto, el cálculo de una TDF de N puntos[1] corresponde al cálculo de N muestras de la transformada de Fourier a N frecuencias igualmente espaciadas $\omega_k = 2\pi k/N$. Esto es a N puntos en el círculo unitario en el plano Z. Para alcanzar la máxima eficiencia, los algoritmos de la FFT deben de calcular todos los N valores de la TDF.

Una de las formas de obtener un gran incremento en la eficiencia del cálculo de la TDF, consiste en descomponer la transformada en una TDF de menor tamaño. En este proceso,

[1]En esta discusión sobre algoritmos FFT, usamos los términos "puntos" y "muestras" para denotar "el valor de la secuencia".

se hace uso de las propiedades de simetría y periodicidad del exponente complejo $W_N^{kn} = e^{-j(2\pi/N)kn}$. Este algoritmo, llamado **Decimación en el Tiempo**, consiste en descomponer la secuencia $x[n]$ en sub-secuencias más pequeñas.

Para ilustrar mejor este concepto, consideremos el caso donde N es una potencia entera de 2, por ejemplo $N = 2^v$. Como N es un número entero par, podemos calcular $X(k)$ separando a $x[n]$ en dos secuencias, una par $x[2n]$ y otra impar $x[2n+1]$. Con $X(k)$ determinado por:

$$X(k) = \sum_{n=0}^{N-1} x[n] W_N^{nk} \quad k = 0, 1, \cdots, N-1 \tag{1.14}$$

y separando x[n] en las muestras pares e impares obtenemos:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x[2n] W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x[2n+1] W_N^{k(2n+1)} \tag{1.15}$$

Recordamos que el término W_N se define como:

$$W_N = e^{-j\frac{2\pi}{N}} \tag{1.16}$$

que tiene la propiedad de periodicidad por lo que

$$W_N^{k+\frac{N}{2}} = -W_N^k \qquad W_N^{n(k+\frac{N}{2})} = W_{\frac{N}{2}}^{nk} \tag{1.17}$$

por lo que

$$X(k) = X(k + \frac{N}{2}) \tag{1.18}$$

Aplicando las condiciones anteriores a la ecuación (1.15) obtenemos:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x[2n] W_{\frac{N}{2}}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x[2n+1] W_N^{kn} \tag{1.19}$$

$$= G(k) + W_N^k H(k) \qquad k = 0, 1, \cdots, N-1 \tag{1.20}$$

Cada una de las sumatorias en la ecuación (1.19) es reconocida como una TDF de $N/2$ muestras. La primera correspondiendo a la TDF de la secuencia de los puntos pares y la segunda a los impares de la secuencia original. Aunque el índice k tiene un rango de N valores, $k = 0, 1, \cdots, N-1$, cada una de las sumas debe de ser calculada solo para k entre 0 y $(N/2) - 1$, debido a que tanto $G(k)$ como $H(k)$ son periódicos en k con un periodo de $N/2$. Después de que las dos transformadas discretas de Fourier son calculadas, se combinan de acuerdo con la ecuación (1.19), lo que da como resultado la TDF de N puntos $X(k)$. Un ejemplo de este cálculo se puede observar en la figura 1.4

Entonces tenemos que

$$G(k) = \sum_{n=0}^{\frac{N}{2}-1} x[2n] W_{\frac{N}{2}}^{nk} \tag{1.21a}$$

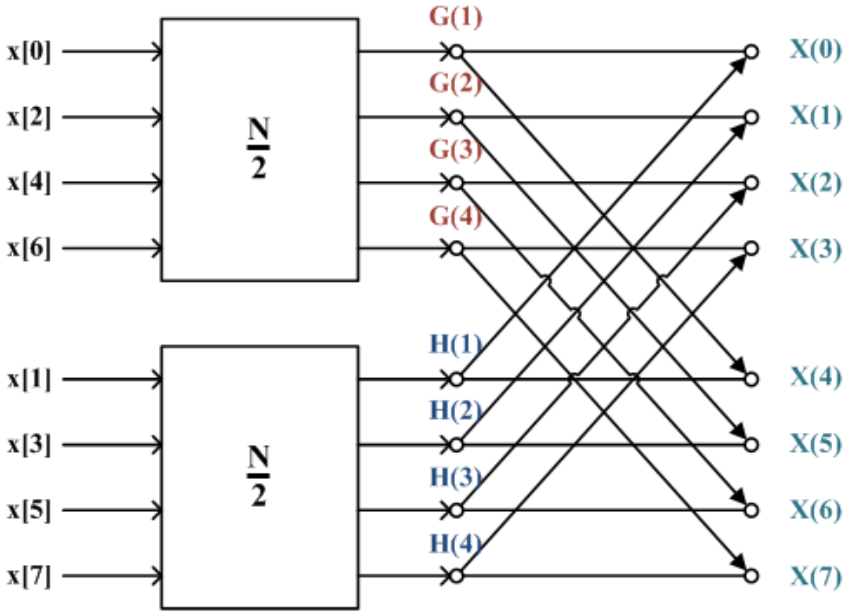


Figura 1.4: Diagrama de Flujo del algoritmo de Decimación en el Tiempo de una secuencia de $N = 8$ puntos dividida en dos.

$$H(k) = \sum_{n=0}^{\frac{N}{2}-1} x[2n+1]W_N^{kn} \tag{1.21b}$$

Con los cálculos organizados como en las ecuaciones (1.21a) y (1.21b), podemos comparar el número de adiciones y multiplicaciones realizados con aquellos requeridos por un cálculo directo de TDF. Para un cálculo directo de la Transformada Discreta de Fourier, sin hacer uso de las propiedades de simetría, se requerían N^2 adiciones y multiplicaciones complejas. En comparación, para las ecuaciones (1.21a) y (1.21b) requieren que se calculen dos transformadas discretas de Fourier de $N/2$ puntos, que utilizan $2(N/2)^2$ multiplicaciones complejas y $2(N/2)^2$ sumas complejas si se realizan las transformadas por medio del método directo. Despues, ambas transformadas deben de ser combinadas, por lo que se requieren N multiplicaciones (al multiplicar la segunda sumatoria por W_N^k) y N adiciones al sumar el resultado. De esta manera el cálculo de la ecuación (1.19) para todos los valores de k requiere de $N + N^2/2$ multiplicaciones complejas y adiciones complejas. Siendo fácil la verificación de que $\in N > 2$, el total $N + N^2/2$ será menor que N^2.

Observamos que la ecuación (1.19), resultado de descomponer la transformada discreta de Fourier de N puntos en dos ecuaciones que calculan $N/2$ puntos cada una; si $N/2$ resulta ser un número par (lo cual sucede cuando N es una potencia de 2), entonces podemos descomponer una vez más las ecuaciones (1.21a) y (1.21b) en TDFs de $N/4$ puntos. El procedimiento igual al mostrado para obtener la ecuación (1.19), separando $G(k)$ y $H(K)$, con lo cual obtenemos lo siguiente:

$$G(k) = \sum_{n=0}^{\frac{N}{4}-1} g[2n] W_{\frac{N}{4}}^{nk} + W_{\frac{N}{2}}^{k} \sum_{n=0}^{\frac{N}{4}-1} g[2n+1] W_{\frac{N}{4}}^{nk} \tag{1.22a}$$

$$H(k) = \sum_{n=0}^{\frac{N}{4}-1} h[2n] W_{\frac{N}{4}}^{nk} + W_{\frac{N}{2}}^{k} \sum_{n=0}^{\frac{N}{4}-1} h[2n+1] W_{\frac{N}{4}}^{nk} \tag{1.22b}$$

De esta manera, el calculo de las TDF de $G(k)$ y $H(k)$ ambas de $N/2$ puntos se lleva a cabo combinando las secuencias de $N/4$ puntos $g[2n]$ con $g[2n+1]$ y $h[2n]$ con $h[2n+1]$, respectivamente. El cálculo se realiza de acuerdo a las ecuaciones (1.22a) y (1.22b) como se ilustra en la figura 1.5. Insertando este cálculo en el diagrama de flujo de la figura 1.4 obtenemos la gráfica completa (figura 1.6) donde se expresan los coeficientes en términos de potencias de $W_{\frac{N}{2}}$, usando el hecho de que: $W_{\frac{N}{2}} = W_N^2$.

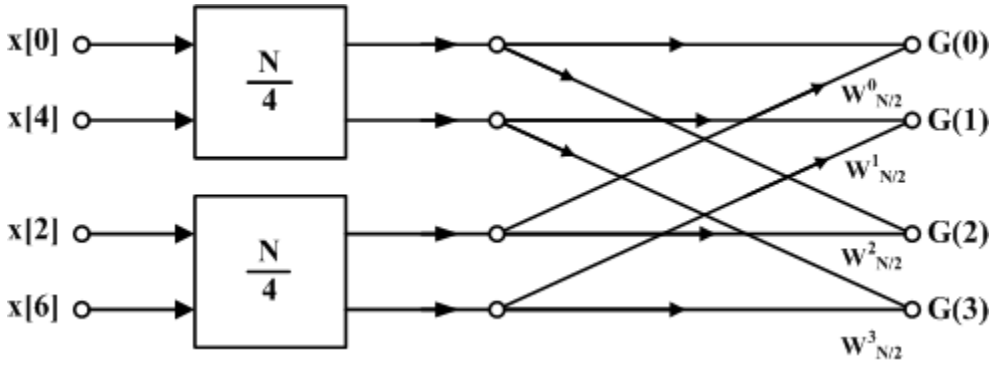


Figura 1.5: Descomposición de una TDF de $N/2$ en un cálculo de $N/4$ puntos.

Por consiguiente, el cálculo de la TDF de 8 puntos se ha reducido a una TDF de 2 puntos, la cual se puede simplificar aun más si usamos la figura 1.7. En los dos nodos correspondientes a la etapa $m-1$ se encontrarían las muestras de entrada $x[0]$ y $x[4]$, mientras que los coeficientes serían: $W_N^0 = 1$ y $W_N^{\frac{N}{2}} = -1$. Insertando lo anterior en la figura 1.6 obtenemos el diagrama de flujo completo para el cálculo de una Transformada Discreta de Fourier de 8 puntos, como se muestra en la figura 1.8.

El diagrama de flujo de la figura 1.8 muestra de manera explicita las operaciones realizadas. Tomando en cuenta cada rama de la gráfica, encontramos que cada etapa tiene N multiplicaciones complejas y N sumas complejas. Como existen $\log_2 N$ etapas, tenemos un total de $N \log_2 N$ multiplicaciones y sumas complejas. Esto representa una mejora importante en la cantidad de operaciones realizadas.

Los cálculos mostrados en la figura 1.8 pueden ser simplificados aun más aprovechando la simetría y periodicidad de los coeficientes W_N^k. Para esto observamos que el cálculo básico se realiza en la forma mostrada en la gráfica 1.7, esto es, se obtienen un par de valores a partir de un par de valores provenientes de la etapa anterior, donde los coeficientes son

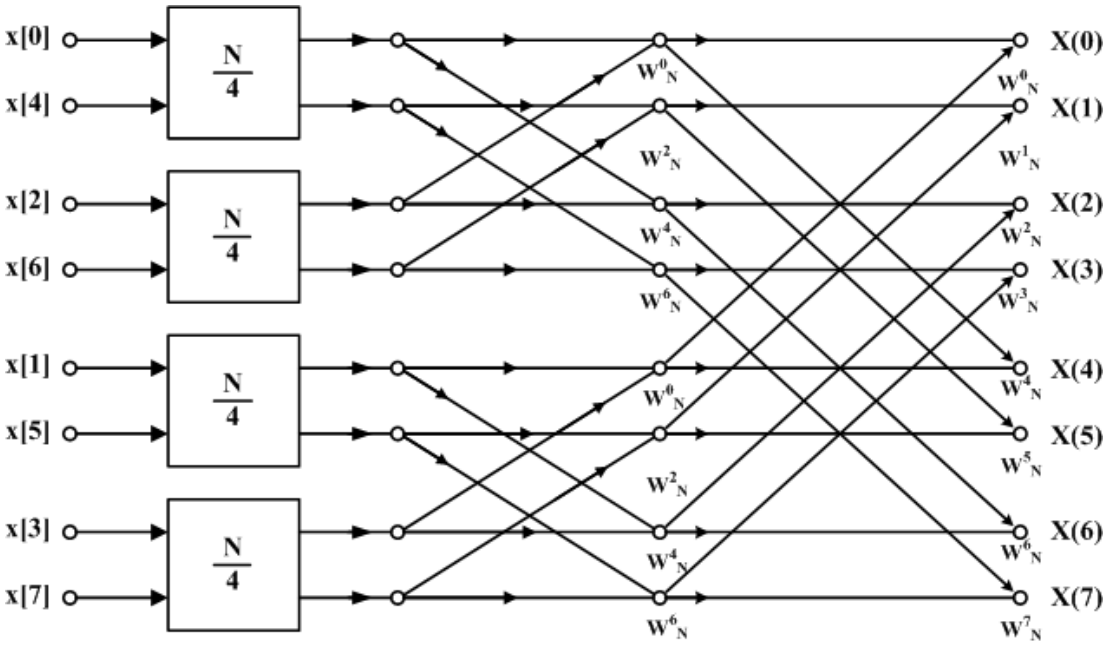


Figura 1.6: Diagrama de Flujo de una TDF de 8 puntos, resultado de sustituir 1.5 en 1.4.

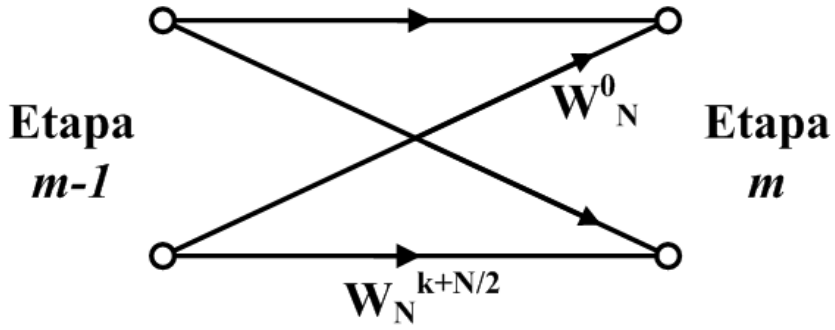


Figura 1.7: Diagrama de Flujo de una TDF de 2 puntos ó "butterfly".

siempre potencias de W_N y los exponentes están separados por $N/2$ Debido a la forma de esta gráfica, se le conoce como "butterfly" o "mariposa de FFT".

$$W_N^{N/2} = e^{-j(\frac{2\pi}{N})\frac{N}{2}} = e^{-j\pi} = -1 \tag{1.23}$$

El factor $W_N^{k+N/2}$ puede escribirse como:

$$W_N^{k+N/2} = W_N^{N/2} W_N^k = -W_N^k \tag{1.24}$$

Con esta observación, la gráfica de mariposa de la figura 1.7 puede ser simplificada a la forma mostrada en la figura 1.9, que solo requiere de una multiplicación en vez de dos.

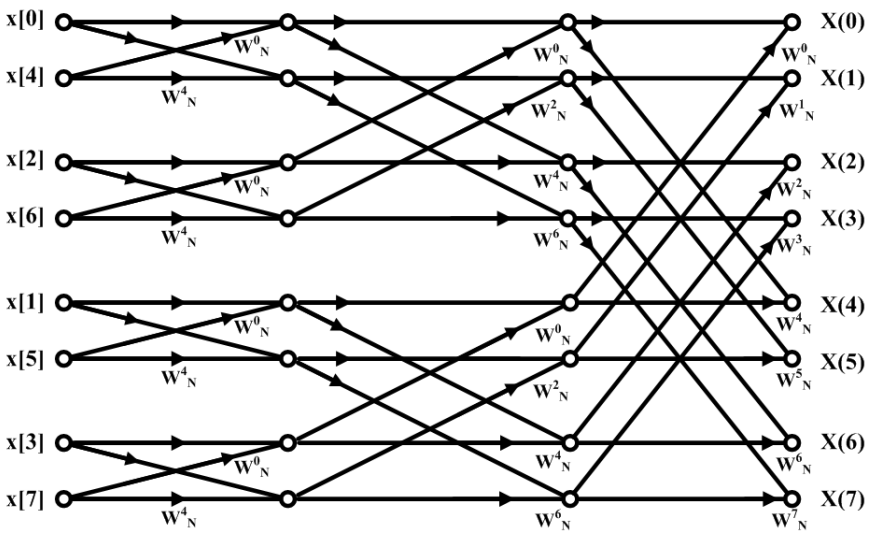


Figura 1.8: Diagrama de Flujo de la Decimación en Tiempo de una TDF de **8 puntos**.

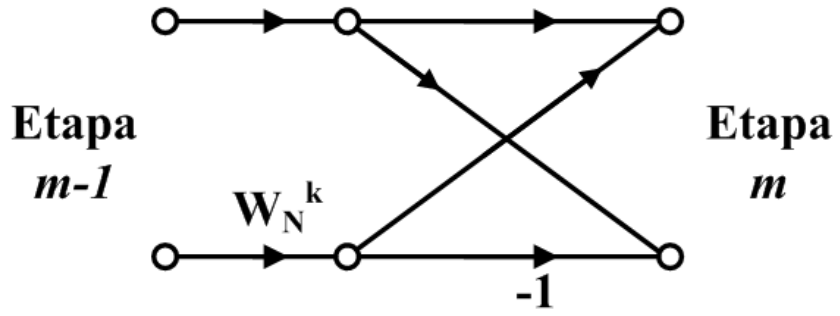


Figura 1.9: Diagrama de Flujo simplificado de la mariposa de FFT.

1.4. Filtros con respuesta finita al impulso (FIR).

Los filtros FIR (*Finite Impulse Response*),también llamados filtros transversales , son filtros digitales cuya respuesta a la señal impulso es finita, es decir, la salida tendrá un número finitos de términos no nulos. Esto es en contraste a los filtros IIR (*Infinite Impulse Response*), los cuales tienen realimentación y la salida tiene un número infinito de términos. La respuesta al impulso de un filtro FIR de orden N dura por $N+1$ muestras y después se reduce a cero.

Para obtener la salida del filtro se necesita la entrada actual y las anteriores. La siguiente ecuación en diferencias nos muestra como se relaciona la señal de entrada con la de salida:

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2[n-2] + \cdots + b_N x[n-N] \tag{1.25}$$

Donde $x[n]$ es la señal de entrada, $y[n]$ la señal de salida y b_i son los coeficientes del filtro. N es el orden del filtro, un filtro de orden N tiene $N+1$ términos en el lado derecho de

la ecuación. La salida también puede expresarse como la convolución de la señal de entrada $x[n]$ con la respuesta al impulso $h[n]$:

$$y[n] = \sum_{i=0}^{N} b_i x[n-i] \tag{1.26}$$

Con esto es posible encontrar la respuesta al impulso si $x[n] = \delta[n]$, donde $\delta[n]$ es la función impulso,y obtenemos:

$$h[n] = \sum_{i=0}^{N} b_i \delta[n-i] \tag{1.27}$$

La transformada Z de la respuesta al impulso se obtiene la función de transferencia del filtro FIR.

$$H(z) = \sum_{i=0}^{K} b_i z^{-i} = b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots \tag{1.28}$$

Con una aproximación adecuada podemos obtener los coeficientes b_i simétricos.

Como ya se mencionó antes, un filtro FIR de orden N es caracterizado por $N+1$ coeficientes y necesita $N+1$ multiplicadores y N sumadores para su implementación. Estructuras en que los coeficientes de los multiplicadores son precisamente los coeficientes de la función de transferencia son llamadas forma directa. Un ejemplo de la forma directa puede ser realizada con la ecuación (1.26) para $N = 4$ y el filtro resultante se muestra en la figura 1.10.

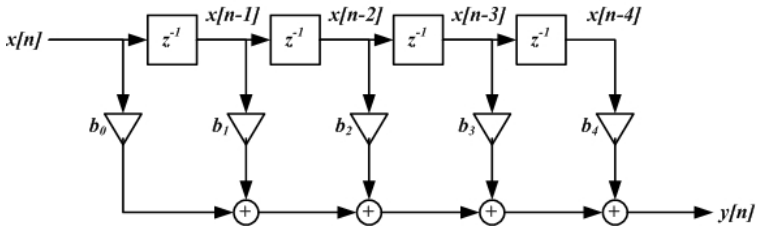


Figura 1.10: Filtro FIR en su forma directa transversal

El filtro calcula la salida $y[n]$ de una manera lineal. Usando el conjunto de coeficientes b_i con $i = 0, 1, 2, 3, 4$ y la secuencia de datos de entrada $x[n]$. En el caso de los filtros adaptables, los coeficientes b_i pasan de ser constantes a ser variables en el tiempo y actualizados por un algoritmo adaptable.

1.4.1. Propiedades de los Filtros FIR

Como ya se mencionó antes, con una aproximación conveniente podemos obtener un filtro FIR con una respuesta en fase lineal, esto trae como ventajas mayor estabilidad y facilidad de implementación; ademas, el diseño de este tipo de filtros se simplifica al no tener que usar números complejos durante el procedimiento matemático y un número de operaciones

disminuida[11]. Para esto tenemos que observar las propiedades de la respuesta al impulso de los filtros FIR.

Sea $h[N]$, $0 \leq n \leq (M-1)$, la respuesta al impulso con duración M, entonces la función del sistema es:

$$H(z) = \sum_{n=0}^{M-1} h[n]z^{-n} = z^{-M-1} \sum_{n=0}^{M-1} h[n]z^{M-n-1} \tag{1.29}$$

Esta función tiene $M-1$ polos en el origen $z=0$ y $M-1$ ceros ubicados en cualquier lugar del plano z. La función de respuesta en frecuencia es:

$$H(e^{j\omega}) = \sum_{n=0}^{M-1} h[n]e^{j\omega n}, -\pi < \omega < \pi \tag{1.30}$$

Si imponemos una constante de fase lineal:

$$\angle H(e^{j\omega}) = -\alpha\omega, -\pi < \omega < \pi$$

Donde α es un *retraso lineal de fase*, entonces h[n] debe de ser simétrico; esto es:

$$h[n] = h[M-1-n], 0 \leq n \leq (M-1); \text{ con } \alpha = \frac{M-1}{2}$$

Entonces h[n] es simétrico alrededor de α, que es el índice de simetría. Existen dos tipos de simetría.

M impar En este caso $\alpha = (M-1)/2$ no es un número entero, la respuesta al impulso es como se muestra en la figura 1.11.

M par En este caso $\alpha = (M-1)/2$ es un número entero, la respuesta al impulso es como se muestra en la figura 1.12.

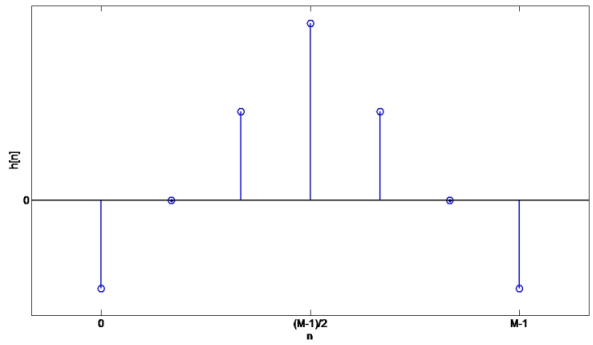


Figura 1.11: Respuesta al impulso simétrica (M impar)

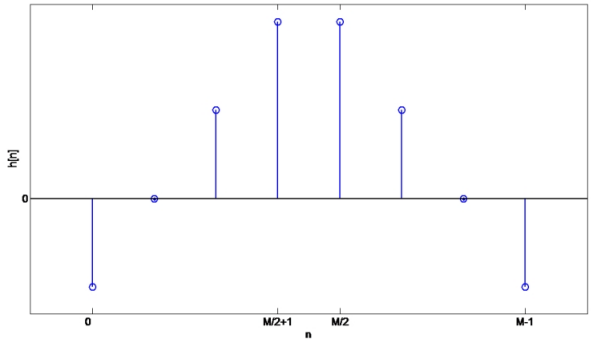


Figura 1.12: Respuesta al impulso simétrica (M par)

También existe otro tipo de filtro FIR de fase lineal, si requerimos que la respuesta en fase $\angle H(e^{j\omega})$ satisfaga la condición:

$$\angle H(e^{j\omega}) = \beta - \alpha\omega$$

Que es la ecuación de una recta pero no atraviesa el origen, en este caso α no es el retraso lineal de fase, si no:

$$\frac{d\angle H(e^{j\omega})}{d\omega} = -\alpha$$

es una constante, que es retraso grupal. Entonces α es conocida como *constante de retraso grupal*.En este caso, las frecuencias en grupo son retrasadas a un ritmo constante, pero algunas frecuencias pueden ser retrasadas mas que otras. Para este tipo de fase lineal se puede mostrar que:

$$h[n] = h[M-1-n],\ 0 \leq n \leq (M-1);\ \text{con}\ \alpha = \frac{M-1}{2}, \beta = \pm\frac{\pi}{2}$$

Esto significa que la respuesta al impulso $h[n]$ es anti-simétrica. El índice de simetría sigue siendo $\alpha = (M-1)/2$. De nuevo tenemos dos posibles tipos, uno para M par y otro para M impar.

M impar En este caso $\alpha = (M-1)/2$ no es un número entero, la respuesta al impulso es como se muestra en la figura 1.13.

M par En este caso $\alpha = (M-1)/2$ es un número entero, la respuesta al impulso es como se muestra en la figura 1.14.

Al combinar los casos de simetría con M par e impar, se obtienen cuatro tipos de filtros FIR con fase lineal. Las funciones de respuesta en frecuencia de estos filtros se estudian a continuación. Escribimos $H(e^{j\omega})$ como:

$$H(e^{j\omega}) = H(\omega)e^{j(\beta-\alpha\omega)}; \beta = \pm\frac{\pi}{2}, \alpha = \frac{M-1}{2} \tag{1.31}$$

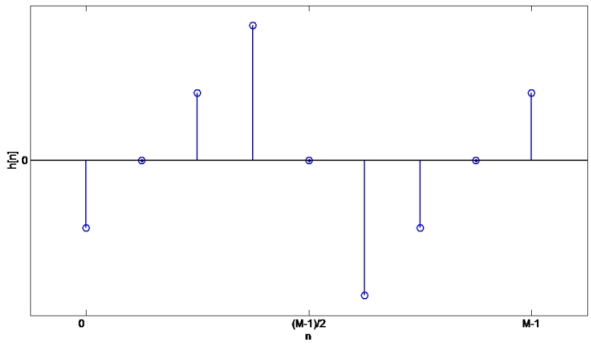


Figura 1.13: Respuesta al impulso anti-simétrica (M impar)

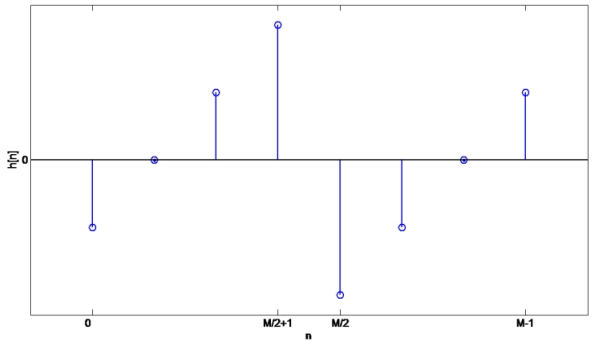


Figura 1.14: Respuesta al impulso anti-simétrica (M par)

Donde $H(\omega)$ es la *función de respuesta en amplitud* y no en magnitud. La función en amplitud es una función real, con partes positivas y negativas. La fase asociada con la respuesta en magnitud es una función discontinua, mientras que la respuesta asociada con la amplitud es una función lineal continua.

Tipo 1. Filtro FIR de fase lineal simétrico, M impar. En este caso $\beta = 0$ y $\alpha = (M-1)/2$ es un entero, $h[n] = h[M-1-n], 0 \leq n \leq M-1$ entonces:

$$H(e^{j\omega}) = \left[\sum_{n=0}^{(M-1)/2} a[n]\cos\omega n\right] e^{-j\omega(M-1)/2} \tag{1.32}$$

Donde la secuencia $a[n]$ es obtenida de $h[n]$ como sigue:

$$a(0) = h\left(\frac{M-1}{2}\right); \text{muestra de enmedio}$$

$$a(n) = 2h\left(\frac{M-1}{2} - n\right); \text{para } 1 \leq n \leq \frac{M-3}{2}$$

Entonces comparando (1.31) con (1.32) obtenemos:

$$H(\omega) = \sum_{n=0}^{(M-1)/2} a[n]\cos\omega n \tag{1.33}$$

Tipo 2. Filtro FIR de fase lineal simétrico, M par. En este caso $\beta = 0$, $h[n] = -h[M-1-n], 0 \leq n \leq M-1$ pero y $\alpha = (M-1)/2$ no es un entero, entonces:

$$H(e^{j\omega}) = \left[\sum_{n=0}^{M/2} b[n] \left\{\cos\omega\left(n - \frac{1}{2}\right)\right\}\right] e^{-j\omega(M-1)/2} \tag{1.34}$$

Donde:

$$b(n) = 2h\left(\frac{M}{2} - n\right);\ n = 1, 2, ..., \frac{M}{2}$$

Entonces comparando (1.31) con (1.34) obtenemos:

$$H(\omega) = \sum_{n=0}^{M/2} b[n] \cos\left\{\omega\left(n - \frac{1}{2}\right)\right\} \tag{1.35}$$

Nótese que con $\omega = \pi$ se tiene

$$H(\omega) = \sum_{n=0}^{M/2} b[n] \cos\left\{\omega\left(n - \frac{1}{2}\right)\right\} = 0$$

Por lo tanto no podemos usar este filtro para implementar filtros paso altas y supresor de banda[23].

Tipo 3. Filtro FIR de fase lineal anti-simétrico, M impar. Para este filtro $\beta = \pi/2$, $\alpha = (M-1)/2$ es un número entero, $h[n] = -h[M-1-n], 0 \leq h \leq M-1$ y $h[(M-1)/2] = 0$; entonces:

$$H(e^{j\omega}) = \left[\sum_{n=1}^{(M-1)/2} c[n] \sin\omega n\right] e^{j[\frac{\pi}{2} - (\frac{M-1}{2})\omega]} \tag{1.36}$$

Donde:

$$c[n] = 2h\left(\frac{M-1}{2} - n\right);\ n = 1, 2, ..., \frac{M-1}{2}$$

Obtenemos:

$$H(\omega) = \sum_{n=1}^{(M-1)/2} c[n] \sin\omega n \tag{1.37}$$

Véase que con $\omega = 0$ y $\omega = \pi$ obtenemos $H(\omega) = 0$, además $e^{j\pi/2} = j$, lo que nos indica que $jH(\omega)$ es puramente imaginario. Entonces este filtro no puede ser usado para implementar filtros paso bajas o paso altas. Sin embargo, este comportamiento sirve para hacer aproximaciones de transformadores Hilbert[23].

Tipo 4. Filtro FIR de fase lineal anti-simétrico, M par. Este filtro es similar al Tipo 2. Tenemos:

$$H(e^{j\omega}) = \left[\sum_{n=1}^{M/2} d[n] \sin\left\{\omega\left(n - \frac{1}{2}\right)\right\}\right] e^{j[\frac{\pi}{2} - \omega(M-1)/2]} \tag{1.38}$$

Donde:

$$d[n] = 2h\left(\frac{M}{2} - n\right),\ n = 1, 2, ..., \frac{M}{2}$$

y

$$H_r(\omega) = \sum_{n=1}^{M/2} d[n] \sin\left\{\omega\left(n - \frac{1}{2}\right)\right\} \tag{1.39}$$

1.5. Filtros FIR en su forma cruzada

Los filtros FIR en cruz ó *Lattice Filters* son usados extensivamente en el procesamiento digital de voz y en el filtrado adaptativo. Es preferido sobre otras estructuras debido al pequeño número de coeficientes permite que sea implementado de manera muy fácil y ademas puede ser usado en aplicaciones que requieren ser modelados en tiempo real. A continuación analizaremos como pasar un filtro FIR de una estructura común (transversal) a una estructura en cruz.

Para los filtros FIR podemos expresar la función de transferencia de la siguiente forma:

$$H_m(z) = A_m(z) = 1 + \sum_{k=1}^{m} a_m(k) z^{-k} \text{ para } m = 0, 1, 2, ... \tag{1.40}$$

o su ecuación en diferencias:

$$y[n] = x[n] + \sum_{k=1}^{m} a_m[k] x[n-k] \text{ para } m = 0, 1, 2, ... \tag{1.41}$$

En la figura 1.10 observamos un filtro FIR de orden n en su forma transversal, este filtro también puede ser dibujado como se muestra en la figura 1.15. De manera que a la salida de esta nueva configuración tenemos:

$$y[n] = x[n] - \hat{x}[n] \tag{1.42}$$

Donde:

$$\hat{x}[n] = -\sum_{k=1}^{m} a_m[k] x[n-k] \tag{1.43}$$

Siendo $\hat{x}[n]$ una predicción del valor de $x[n]$, esta estructura se llama *filtro de predicción de error* (prediction error filter). Suponiendo que el filtro FIR es de primer orden. En la salida tenemos la siguiente respuesta:

$$y[n] = x[n] - a_1(1) x[n-1] \tag{1.44}$$

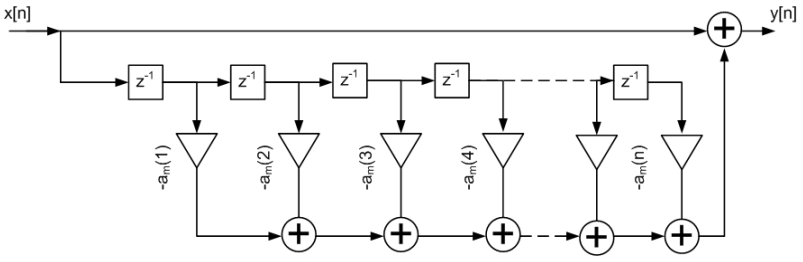


Figura 1.15: Prediction Error Filter

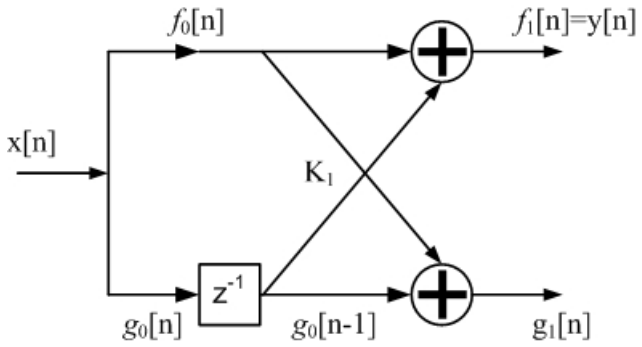


Figura 1.16: Filtro FIR en cruz de primer orden

La misma respuesta en la salida se obtiene mediante la estructura de la cruz mostrada en la figura 1.16. Si alimentamos ambas entradas de la estructura en cruz con la señal $x[n]$ obtenemos las salidas:

$$f_0[n] = g_0[n] = x[n] \tag{1.45}$$
$$f_1[n] = x[n] + K_1 x[n-1] \tag{1.46}$$
$$g_1[n] = K_1 x[n] + x[n-1] \tag{1.47}$$

En la salida superior del filtro en cruz si hacemos $K_1 = a_1$ obtenemos la ecuación (1.44). El parámetro K_1 se conoce como coeficiente de reflexión[14].

Si conocemos los coeficientes de reflexión del filtro FIR $a_m(k)$ es posible calcular los coeficientes del correspondiente filtro FIR en estructura de cruz. El vector $\{a_m\}$ de los coeficientes es:

$$\{a_m\} = \{1\ a_m(1)\ a_m(2)\ a_m(3)\ ...\} \tag{1.48}$$

La función $f_m[n]$ de la salida del filtro en cruz se puede describe mediante la siguiente ecuación:

$$f_m[n] = \sum_{k=0}^{m} a_m(k) x[n-k] \tag{1.49}$$

Para la salida inferior del filtro en cruz se tiene lo siguiente:

$$g_m[n] = \sum_{k=0}^{m} b_m(k)x[n-k] \tag{1.50}$$

Los coeficientes del vector $g_m[n]$ son representados de esta manera:

$$\{g_m\} = \{... \; b_m(3) \; b_m(2) \; b_m(1) \; 1\} \tag{1.51}$$

Los coeficientes b_m son ordenados en orden decreciente, por lo tanto:

$$b_m(k) = a_m(m-k) \tag{1.52}$$

Y si aplicamos la transformada inversa Z a la ecuación (1.50) obtenemos:

$$G_m(z) = B_m(z) \cdot X(z) \tag{1.53}$$

$$G_m(z) = \frac{G_m(z)}{X(z)} \tag{1.54}$$

Notamos que $B_m(z)$ es la función de transferencia y sus coeficientes son $b_m(k)$, por lo que:

$$B_m(z) = \sum_{k=0}^{m} b_m(k)z^{-k} \tag{1.55}$$

Utilizando la ecuación (1.52) y sustituyendo en la ecuación (1.55) de la forma:

$$\begin{aligned} B_m(z) &= \sum_{k=0}^{m} a_m(m-k)z^{-k} = \sum_{l=0}^{m} a_m(l)z^{l-m} \\ &= z^{-m}\sum_{l=0}^{m} a_m(l)z^{l} = z^{-m}A_m(z^{-1}) \end{aligned} \tag{1.56}$$

Usando las ecuaciones generales.

$$\begin{aligned} f_0[n] &= g_0[n] = x[n] \\ f_m[n] &= f_{m-1}[n] + K_m g_{m-1}[n-1] m = 1, 2, 3, ..., M-1 \\ g_m[n] &= K_m f_{m-1}[n] + g_{m-1}[n-1] m = 1, 2, 3, ..., M-1 \end{aligned} \tag{1.57}$$

Y utilizando la transformada Z obtenemos:

$$\begin{aligned} F_0(z) &= G_0(z) = X(z) \\ F_m(z) &= F_{m-1}(z) + K_m G_{m-1}(z) \\ G_m(z) &= K_m F_{m-1}(z) + G_{m-1}(z) \end{aligned} \tag{1.58}$$

Dividiendo las ecuaciones anteriores entre $X(z)$ obtenemos al final:

$$A_0(z) = B_0(z) = 1 \tag{1.59}$$
$$A_m(z) = A_{m-1}(z) + K_m B_{m-1}(z) z^{-1} \tag{1.60}$$
$$B_m(z) = K_m A_{m-1}(z) + z^{-1} B_{m-1}(z) \tag{1.61}$$

Con las ecuaciones (1.60) y (1.61) pudiendose expresar en forma matricial.

$$\begin{bmatrix} A_m(z) \\ B_m(z) \end{bmatrix} = \begin{bmatrix} 1 & K_m \\ K_m & 1 \end{bmatrix} \times \begin{bmatrix} A_{m-1}(z) \\ z^{-1} B_{m-1}(z) \end{bmatrix} \tag{1.62}$$

1.6. Diseño de Filtros FIR

Diseñar un filtro FIR significa seleccionar los coeficientes de manera que el sistema tenga ciertas características. Estas características son dictadas por las especificaciones del filtrado, que generalmente responden a la respuesta en frecuencia deseada del filtro.

El diseño de filtros FIR consta de 3 pasos:

Especificaciones Como ya se mencionó antes, las especificaciones están dictadas por las necesidades de filtrado.

Aproximación Una vez que se tienen las especificaciones, usamos varios conceptos matemáticos para llegar a una descripción del filtro que se aproxime a esas especificaciones.

Implementación Como resultados de las aproximaciones obtendremos el filtro en forma de una expresión matemática o la función del sistema $H(z)$ o respuesta al impulso $h[n]$. Con esto se implementa el filtro en hardware o a través de un software de computadora.

Existen varios métodos para la encontrar los coeficientes de los filtros de acuerdo a la respuesta en frecuencia:

- Método de las Ventanas
- Muestreo en frecuencia
- Mínimos cuadrados
- Rizado constante (Aproximación de Chevyshov y algoritmo de intercambio de Remez)

De los varios métodos propuestos para el diseño de filtros FIR, una forma directa y sencilla de hacerlo se basa en truncar (con una función ventana) la representación en Series de Fourier de la respuesta en frecuencia deseada del filtro. Este método se basa en la observación de que para un filtro de orden N, existen N muestras en frecuencia igualmente espaciadas que

constituyen la Transformada Discreta de Fourier (DFT) de su respuesta al impulso, por lo tanto,la respuesta al impulso del filtro puede ser calculada aplicando la transformada inversa a esas muestras de frecuencia.

La parte importante de este método consiste en seleccionar la función ventana correcta y un filtro ideal apropiado. El filtro ideal se denotara como $H_d(e^{j\omega})$, filtro que tiene magnitud unitaria y fase lineal en su banda de paso y respuesta cero en su banda de detención.

$$H_d(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h_d[n]e^{-j\omega n} \tag{1.63}$$

A partir de $H_d(e^{j\omega})$ podemos obtener la respuesta al impulso de este filtro que se puede expresar como:

$$\begin{aligned} h_d[n] &= \mathcal{F}^{-1}\{H_d(e^{j\omega})\} \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\omega})e^{j\omega n}d\omega \end{aligned} \tag{1.64}$$

Muchos sistemas ideales son discontinuos en los bordes de sus bandas de transición o definidos como sumas de funciones *Heaviside*. Lo que resulta en que las respuestas al impulso de estos sistemas son no causales y de duración infinita. Para obtener un sistema causal FIR, la solución más fácil es truncar la respuesta en frecuencia. La ecuación (1.63) puede verse como una representación en serie de Fourier de la respuesta periódica en frecuencia $H_d(e^{j\omega})$ con $h[n]$ como los coeficientes de dicha serie de Fourier.

La forma más sencilla de obtener un filtro FIR causal de $h_d[n]$ de longitud M es definir un nuevo sistema con respuesta al impulso $h[n]$[2] :

$$h[n] = \begin{cases} h_d[n], & 0 \leq n \leq M \\ 0, & \text{otro caso} \end{cases} \tag{1.65}$$

De manera general, podemos representar $h[n]$ como el producto de la respuesta al impulso deseada y una 'ventana' de duración finita $w[n]$:

$$h[n] = h_d[n] \cdot w[n] \tag{1.66}$$

En un ejemplo sencillo, la ventana puede ser *rectangular*, como la de la ecuación (1.65):

[2]En este caso M es el orden del sistema y por lo tanto $M+1$ es la longitud o duración de la respuesta al impulso.

$$w[n] = \begin{cases} 1, & 0 \leq n \leq M \\ 0, & \text{otro caso} \end{cases} \tag{1.67}$$

Gracias al teorema de modulación[20] sabemos que:

$$H(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(e^{j\theta}) W(e^{j(\theta-\omega)}) d\theta \tag{1.68}$$

Esto muestra que $H(e^{j\omega})$ es la convolución periódica de la respuesta en frecuencia ideal con la transformada de Fourier de la ventana. Entonces la respuesta en frecuencia $H(e^{j\omega})$ será una versión 'expandida' de la respuesta en frecuencia $H_d(e^{j\omega})$

Si $w[n] = 1$ para todo n (no se trunca la respuesta al impulso),$W(e^{j\omega})$ es un tren de impulsos con periodo 2π y por lo tanto $H(e^{j\omega}) = H_d(e^{j\omega})$. Esto nos sugiere que $w[n]$ es elegido de manera que $W(e^{j\omega})$ se concentre en una banda de frecuencias muy estrecha alrededor de $\omega = 0$, entonces $H(e^{j\omega})$ se parecerá a $H_d(e^{j\omega})$ excepto donde $H(e^{j\omega})$ cambia muy rápido. Por lo tanto, al elegir la ventana se debe de considerar que $w[n]$ sea de una duración muy corta (lo más que se pueda), de manera que se minimicen los cálculos en la implementación de filtro; mientras se busca que $W(e^{j\omega})$ se aproxime a un impulso, de forma que $W(e^{j\omega})$ se concentre en una banda muy estrecha de frecuencias de manera que la convolución de la ecuación (1.68) reproduzca de manera muy exacta la respuesta en frecuencia deseada.

Algunas de las ventanas más usadas son mostradas en la figura 1.17. Estas ventanas son definidas por las siguientes ecuaciones:

Rectangular

$$w[n] = \begin{cases} 1, & 0 \leq n \leq M \\ 0, & \text{otro caso} \end{cases} \tag{1.69}$$

Bartlett (triangular)

$$w[n] = \begin{cases} 2n/M, & 0 \leq n \leq M/2 \\ 2 - 2n/M, & M/2 < n \leq M \\ 0, & \text{otro caso} \end{cases} \tag{1.70}$$

Hanning

$$w[n] = \begin{cases} 0.5 - 0.5\cos(2\pi n/M), & 0 \leq n \leq M \\ 0, & \text{otro caso} \end{cases} \tag{1.71}$$

Hamming

$$w[n] = \begin{cases} 0.54 - 0.46\cos(2\pi n/M), & 0 \leq n \leq M \\ 0, & \text{otro caso} \end{cases} \tag{1.72}$$

Blackman

$$w[n] = \begin{cases} 0.42 - 0.5\cos(2\pi n/M) + 0.08\cos(4\pi n/M), & 0 \leq n \leq M \\ 0, & \text{otro caso} \end{cases} \tag{1.73}$$

Gauss

$$w[n] = \begin{cases} e^{-\frac{1}{2}\left(\alpha \frac{n}{M/2}\right)^2}, & 0 \leq n \leq M \\ 0, & \text{otro caso} \end{cases} \tag{1.74}$$

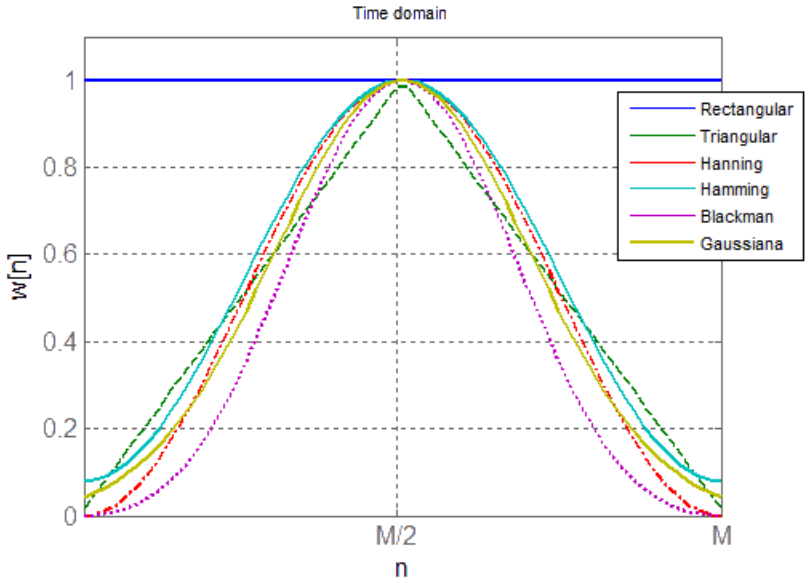


Figura 1.17: Ventanas más comunes.

(Por conveniencia las gráficas de la figura 1.17 fueron graficadas como funciones continuas. Sin embargo, como son definidas por sus ecuaciones, las ventanas son definidas solamente en valores enteros de n).

Las ventanas mostradas de la ecuación (1.69) a (1.74) se pueden usar para análisis espectral y diseño de filtros FIR. Tienen la propiedad de que sus transformadas de Fourier están centradas alrededor de $\omega = 0$, y tienen una forma que permite que sean calculadas fácilmente. La transformada de Fourier de la Ventana de Bartlett (1.70) puede ser expresada como producto de transformadas de Fourier de Ventanas Rectangulares y las transformadas de Fourier de otras ventanas pueden ser expresadas como sumas de transformadas de Fourier de la ventana rectangular desplazadas en frecuencia.

Capítulo 2

Control Activo del Ruido (CAR)

Como ya se menciono anteriormente el *Control Activo del Ruido (CAR)* se logra produciendo una señal 'anti-ruido' de manera que al interactuar con el ruido estas dos ondas se cancelen (figura 1.1), esto se logra a través de sistemas electrónicos que usan un algoritmo de procesamiento de señales muy particular para poder realizar esta cancelación.

Existen dos tipos de ruido acústico en el ambiente. Uno es causado por turbulencias y es totalmente aleatorio. Este tipo de ruido distribuye su energía de manera uniforme a través de varias bandas de frecuencias. Se le conoce como ruido de banda ancha y ejemplos de este tipo de ruidos son los sonidos de bajas frecuencias de los aviones o el sonido de una explosión.

Otro tipo de ruido, llamado ruido de banda angosta, concentra gran parte de su energía en frecuencias específicas; este tipo de ruido se relaciona con maquinas repetitivas o rotatorias, de manera que son periódicas o casi periódicas. Los motores de combustión interna, turbinas, compresores y bombas son un buen ejemplo de este tipo de ruido.

En la actualidad hay dos formas de controlar el ruido acústico: de forma pasiva o activa. La forma tradicional de controlar el ruido es de forma pasiva, usando absorbentes superficiales, silenciadores reactivos, materiales porosos y aisladores, entre otros; absorbiendo la energía y/o modificando la impedancia del medio de manera que se atenúen los sonidos no deseados. Este tipo de técnica es valiosa por su alta atenuación en un rango amplio de frecuencias, especialmente en las frecuencias altas y medias, con un costo no excesivamente elevado. Sin embargo en ruidos de baja frecuencia su efectividad disminuye[15] debido a que el tamaño del dispositivo esta relacionado con la longitud de onda del ruido a cancelar, esto provoca que este tipo de técnica sea poco rentable o inviable, ya que ocupan mucho espacio y son costosos.

Por otro lado, las técnicas activas, han llamado últimamente la atención debido a que su principio de trabajo es usar un sistema electroacústico o electromecánico que cancela el sonido no deseado a través de una *fuente secundaria* usando el principio de superposición, generando un anti-ruido de igual amplitud y fase opuesta, de manera que al combinarse con el ruido, ambos se cancelan. La ventaja de CAR es su mayor efectividad al eliminar el ruido

ya que si se usan métodos pasivos estos pueden resultar muy caros y difíciles de controlar.

CAR tiene aplicaciones en una gran cantidad de problemas en operaciones industriales y de manufactura ademas de productos para consumo general. Esto es debido a que el concepto de CAR puede ser aplicado a la reducción de cualquier clase de ruido, ya sea eléctrico, acústico, vibratorio o en cualquier otro tipo de medio.

El diseño básico de CAR utilizando un micrófono y una bocina controlada electrónicamente para generar el anti-ruido fue propuesto por primera vez en una patente de 1936 por Paul Lueg[10]. En la figura 2.1 se muestra el diagrama original propuesto por Lueg.

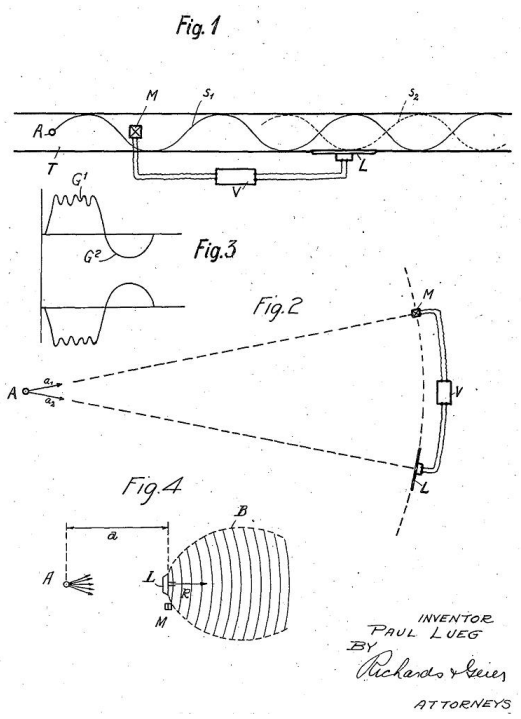


Figura 2.1: Diagrama de la patente de 1936 por el físico alemán Paul Lueg.

Si bien la patente estableció la idea básica de CAR, no fue si no hasta el desarrollo de las técnicas de procesamiento digital de señales que el desarrollo de este tipo de sistemas fue posible. Esto debido a que las características de la señal y el medio en el que esta se propaga varían con el tiempo, el contenido espectral, la amplitud, fase y velocidad de propagación no son estacionarios; de manera que un sistema de Control Activo del Ruido debe de ser *adaptivo*.

Un problema fundamental en CAR y que debe de ser considerado; es el requerimiento de un control de alta precisión, estabilidad temporal y confiabilidad. Para que sea pueda lograr un alto nivel de atenuación del ruido, la amplitud y la fase del ruido primario y secundario deben de coincidir con mucha precisión. De manera que es preferible que el cancelador del ruido sea digital, de manera que las señales provenientes de transductores electroacústicos o electromecánicos sean muestreadas y procesadas con un Procesador Digital de Señales (DSP)

lo suficiente velocidad y preciso de manera que ejecute complejas funciones matemáticas en tiempo real.

2.1. Principios Acústicos

Todas las estrategias de control activo del ruido se basan en el principio de superposición, el cual tiene aplicaciones en cualquier sistema lineal. La propagación de una onda sonora, con amplitud correspondiente a la de un ruido extremadamente intenso, es un proceso relativamente lineal. Los problemas de no-linealidad aparecen cuando la bocina utilizada para generar la interferencia destructiva actúa como fuente secundaria de ruido. Al momento de controlar ruido, por ejemplo, un tono de baja frecuencia debido a la interferencia destructiva muy poco ruido será audible en dicha frecuencia. Sin embargo, las armónicas generadas por la bocina podrían ser notablemente audibles.

El efecto de interferencia destructiva debido a la superposición de ondas acústicas es bastante simple de realizar, si el volumen y frecuencia de un tono puro emitido por una bocina se ajustan de manera relativa al tono emitido por una segunda bocina, entonces la presión acústica en un micrófono de monitoreo, el cual puede ser colocado en cualquier punto del campo resultante, sera casi cero.

Desgraciadamente, también es probable que en otros puntos del campo acústico , las dos componentes de la presión acústica estén en fase y ocurra una interferencia constructiva, incrementando el ruido en esos puntos.

2.2. Aplicaciones generales

La decisión de aplicar o no sistemas CAR, será determinada dependiendo de la efectividad del sistema en comparación de técnicas pasivas. Como ya se mencionó anteriormente CAR es un método muy efectivo para atenuar ruidos de baja frecuencia con un dispositivo de pequeño tamaño. El desempeño en altas frecuencias es limitado debido a varios factores, incluyendo la necesidad de una mayor tasa de muestreo y la existencia de modos de alto orden que resultan en cálculos más complicados. En bajas frecuencias, las tasas de muestreo son bajas y las longitudes de onda grandes. Por otro lado, las técnicas pasivas tienden a ser efectivas en altas frecuencias ofreciendo la alternativa de complementar ambas técnicas.

Desde un punto de vista geométrico CAR puede ser clasificado en las siguientes cuatro categorías[16]:

1. Ruido en conductos: Tuberías de una sola dimensión; tales como tubos de ventilación, aire acondicionado, escapes, etc.

2. Ruido interior: Ruido dentro de un espacio cerrado.

3. Ruido en el espacio libre: Ruido que se propaga en un espacio libre.

4. Protección auditiva personal: Un caso de ruido interior en pequeña escala.

Con la alza en atención y estudio que han recibido los sistemas CAR en los últimos 30 años, se han desarrollado aplicaciones específicas, que incluyen atenuación de fuentes de ruido inevitable en los siguientes escenarios:

Automotriz Atenuación de sonidos dentro de la cabina de pasajeros, reducción del sonido del escape, etc.

Electrodomésticos Aires acondicionados, refrigeradores, lavadoras, secadoras, etc.

Industrial Ventiladores, conductos de aire, chimeneas, compresores, bombas, particiones de cubículos en oficinas, protección de oídos, etc.

Transportación Aviones, helicópteros, barcos, motocicletas, etc.

Los algoritmos usados para el Control de Ruido Acústico también pueden ser usados en el Control Activo de Vibraciones; el cual se puede emplear para aislar variaciones de una variedad de máquinas y estabilizar plataformas en presencia de perturbaciones vibratorias. Debido a que el desempeño y fiabilidad de este tipo de sistemas son continuamente mejorados, a la vez que los costos siguen disminuyendo, los sistemas activos se convertirán en la solución preferida para una gran variedad de problemas de control vibratorio.

Existen también dos maneras de implementar el Control Activo del Ruido, la primera usa control a priori (*feedforward control*), que puede ser interpretada como un **Cancelador Activo de Ruido** o un identificador adaptivo de sistemas. La segunda forma consiste en usar control de realimentación (*feedback control*), que se puede realizar como un esquema adaptivo de predicción.

2.3. Cancelación Adaptiva del Ruido

Como se ilustra en la figura 2.2 el concepto básico de la Cancelación Adaptiva del Ruido es procesar dos señales provenientes de dos sensores y reducir el nivel de ruido con técnicas de filtrado adaptivas. El sensor primario es colocado cerca de la señal origen para obtener así la señal deseada; sin embargo el sensor primario también captará ruido. El sensor de referencia es colocado cerca de la señal origen del ruido para sensar esta señal solamente. Esta estructura usa a su favor la correlación entre las señales de ruido captadas por los sensores primario y de referencia.

Un diagrama de bloques de un sistema de cancelación adaptiva del ruido se ilustra en la figura 2.3, donde $P(z)$ representa la función de transferencia entre la fuente de ruido y el sensor primario. El cancelador tiene dos entradas, la entrada primaria $d[n]$ y la señal

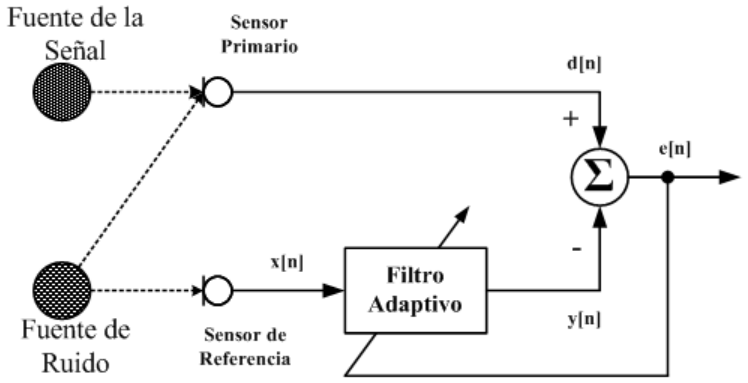


Figura 2.2: Concepto básico de la cancelación adaptiva del ruido.

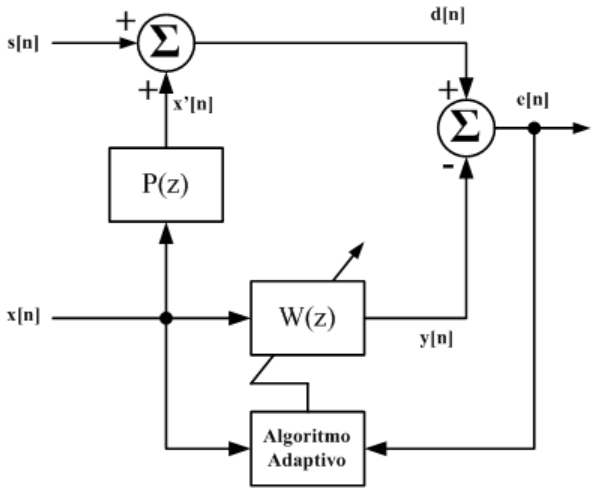


Figura 2.3: Diagrama de bloques de un cancelador adaptivo de ruido.

de referencia $x[n]$. La señal primaria $d[n]$ consiste de la señal original $s[n]$ más ruido $x'[n]$ (resultando $d[n] = s[n] + x'[n]$), la cual tiene un alto grado de correlación a $x[n]$ ya que ambas se originan de la misma fuente de ruido. La señal de referencia $x[n]$ consiste en ruido 'puro'. El objetivo del filtro adaptivo es usar la entrada de referencia $x[n]$ para estimar el ruido $x'[n]$. La salida del filtro $y[n]$ que es un estimado del ruido $x'[n]$, se resta de la señal de canal primario $d[n]$, produciendo la señal $e[n]$, que es la señal deseada con ruido reducido.

Para minimizar el error residual $e[n]$, el filtro adaptivo $W(z)$ generará una salida $y[n]$ que es una aproximación de $x'[n]$. Entonces el filtro adaptivo convergerá al bloque desconocido $P(z)$. Esto es conocido como un esquema de identificación de sistemas que se discutirá a continuación.

2.3.1. Identificación Adaptiva de Sistemas

La identificación de sistemas es un acercamiento experimental de modelado de procesos o sistemas. La idea básica es medir las señales producidas por el sistema y usarlas para construir un modelo.

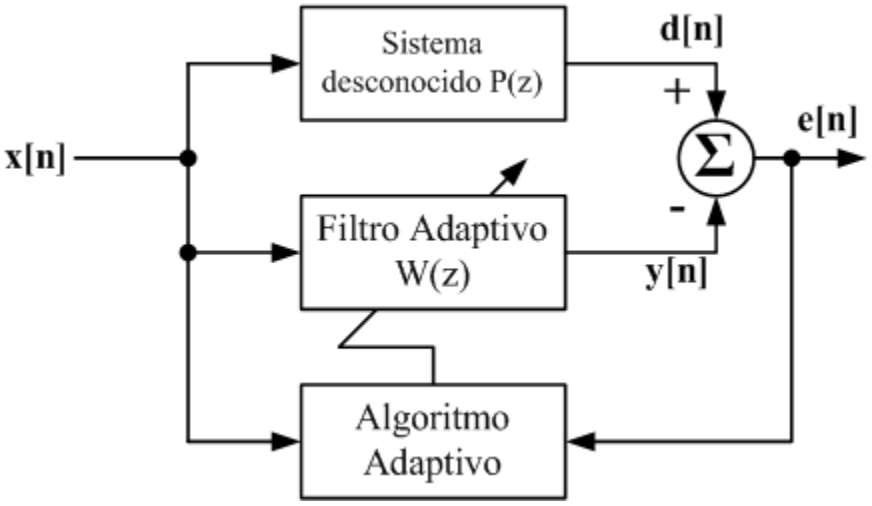


Figura 2.4: Identificación adaptiva de sistemas.

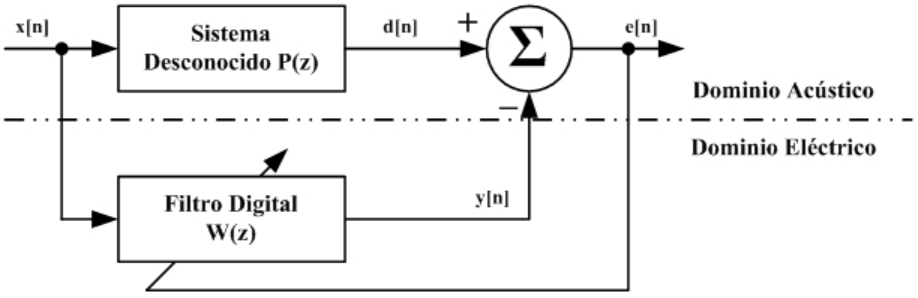


Figura 2.5: Identificación adaptiva de sistemas para CAR

En la figura 2.4, donde el sistema desconocido $P(z)$ y el filtro digital $W(z)$ modelará $P(z)$ basándose en un algoritmo de minimización de errores predeterminado. Al introducir una señal $x[n]$ tanto en el sistema $P(z)$ como en el filtro $W(z)$ y midiendo las señales de salida $y[n]$ y $d[n]$, podremos determinar las características de $P(z)$ ajustando el filtro $W(z)$ para que minimice la diferencia entre ambas salidas.

Con esto en mente se puede llegar a una forma simplificada de un sistema de Control Activo de Ruido, mostrada en la figura 2.5 , en donde un filtro adaptivo $W(z)$ se usa para estimar un sistema desconocido $P(z)$. Se asume que el sistema y el filtro reciben la misma entrada $x[n]$, el sistema proporciona la respuesta "deseada"[1]al filtro adaptivo. Si el sistema es dinámico, el modelo será variante en el tiempo. El algoritmo adaptivo tiene entonces la

tarea de mantener el error de modelado al mínimo detectando las variaciones en el tiempo del sistema desconocido. El camino primario $P(z)$ consiste de la respuesta acústica del sensor de referencia al sensor de error donde se lleva acabo la atenuación del ruido. En aplicaciones actuales, otro número de funciones de transferencias deben de ser incluidas, las cuales serán discutidas en secciones siguientes.

Existen tres puntos que deben de ser considerados en la identificación adaptiva de sistemas: la señal de excitación, la estructura del filtro y el mecanismo de adaptación. Si la señal de excitación $x[n]$ tiene un contenido frecuencial muy amplio y el ruido interno del sistema a identificar es pequeño, el filtro adaptivo convergerá a un modelo muy aproximado del sistema desconocido. El filtro podrá tener una estructura de *solo ceros*, *solo polos* o *polos y ceros*. Una estructura de solo ceros es representada por un filtro **FIR**, mientras que las estructuras de solo polos ó polos y ceros pueden ser representadas por un filtro **IIR**. El mecanismo de adaptación puede ser un algoritmo adaptable como el LMS o RLS y sus variantes.

El objetivo del filtro $W(z)$ será entonces minimizar el error residual $e[n]$. De la figura 2.5 la *transformada z* de $e[n]$ se denota como $E(z)$ y se expresa como:

$$\begin{aligned} E(z) &= D(z) - Y(z) \\ &= P(z)X(z) - W(z)X(z) \end{aligned} \tag{2.1}$$

De manera ideal $E(z) = 0$ después de que el filtro adaptable converge. De la ecuación anterior tenemos:

$$W(z) = P(z) \tag{2.2}$$

Para un $X(z) \neq 0$, lo que significa que

$$y[n] = d[n] \tag{2.3}$$

La ecuación (2.3) nos dice que la salida del filtro adaptivo $y[n]$ es idéntica a la señal primaria $d[n]$. De esta manera cuando $d[n]$ y $y[n]$ son combinadas acústicamente, el error residual es

$$e[n] = d[n] - y[n] = 0 \tag{2.4}$$

Que basándonos en el principio de superposición resulta en la cancelación de ambos sonidos.

De esta manera, si el filtro $W(z)$ converge a un buen modelo del sistema $P(z)$, el acercamiento a priori es capaz de cancelar ruidos de banda ancha (aleatorios) de manera efectiva. Lo anterior supone que se tiene suficiente tiempo para que el sistema complete los cálculos del modelo, determine la inversa de la señal y genere el sonido requerido para cancelar el ruido antes de que este pase a través del sistema. Como las señales pasan a través del sistema y el filtro en paralelo, los cambios en la fuente primaria no afectan el nivel del sonido cancelado si el modelo generado por el filtro representa de manera adecuada al sistema desconocido.

[1]En realidad, en el contexto de CAR, la salida del sistema desconocido es una señal **no** deseada, el término 'deseada' es un término genérico usado con filtros adaptivos para denotar una señal piloto o de prueba[22]

2.3.2. Efectos de la Trayectoria Secundaria

La sección anterior describió de forma simplificada el principio de CAR en un ambiente ideal. El uso de un algoritmo adaptable, asume que una señal de error esta disponible y es la diferencia entre la señal proveniente de la fuente primaria y la salida del filtro adaptable.

El uso del filtro adaptivo en la figura 2.5 se complica debido a que la señal eléctrica de referencia se debe de obtener por medio de presión acústica usando un micrófono. De la misma manera, la señal eléctrica de error debe de ser obtenida usando otro micrófono. Finalmente, el sonido cancelador o anti-ruido, debe de ser generado a partir de la señal eléctrica de salida por medio de una bocina. Por lo tanto, existen un número de funciones de transferencia que deben de ser consideradas. La suma de sonidos en la figura 2.5 representa superpocisón acústica en el espacio de la bocina canceladora al micrófono de error, donde el ruido primario se combina con la salida del filtro adaptivo. Esto deriva la necesidad de compensar por la función de transferencia de la trayectoria secundaria $S(z)$ desde $y[n]$ a $e[n]$, lo cual incluye: el convertidor A/D (analógico a digital), filtro de reconstrucción, amplificadores de potencia, bocina, micrófono de error, preamplificador y convertidor D/A (digital a analógico).

Como se ilustra en la figura 2.6, la función de transferencia de la trayectoria secundaria S(z) puede ser separada en dos funciones de transferencia en cascada:

$$S(z) = R(s)S'(z) \tag{2.5}$$

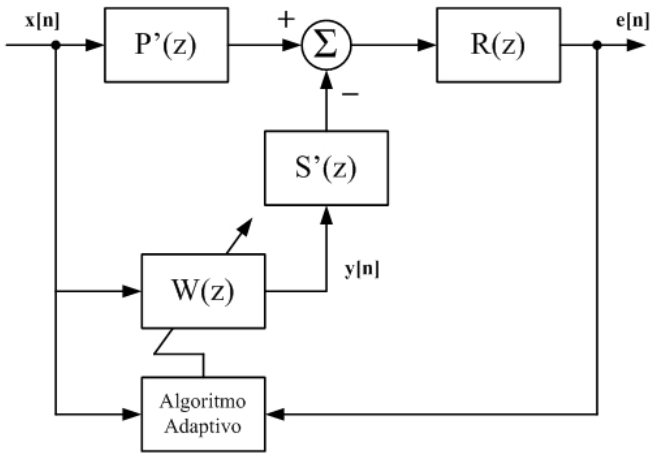


Figura 2.6: Diagrama de bloques de un sistema CAR con trayectoria secundaria.

Donde $S'(z)$ representa la función de transferencia secundaria de la salida del filtro adaptivo a la sumatoria de señales y $R(z)$ representa la función de transferencia residual de la sumatoria de señales a la señal de error. De manera similar, la función de transferencia primaria $P(z)$ que va del sensor de entrada a la señal de error puede ser separada como:

$$P(z) = R(z)P'(z) \tag{2.6}$$

Donde $P'(z)$ es la función de transferencia del sistema acústico desconocido desde el micrófono de referencia a la sumatoria de señales. Los coeficientes del filtro adaptivo deben de converger a los valores apropiados para minimizar la señal de error aun con la presencia de las funciones de transferencias de los diferentes transductores y filtros. De la figura 2.6, la función de transferencia de la señal de error es:

$$E(z) = R(z)\big[P'(z) - S'(z)W(z)\big]X(z) \tag{2.7}$$

Asumiendo que $W(z)$ tiene un orden mucho mayor a cero. Entonces, después de la convergencia del filtro adaptivo, el error residual resulta ser cero, $E(z) = 0$, lo que significa que $W(z)$ ha llegado a la función de transferencia óptima

$$W^o(z) = \frac{P'(z)}{S'(z)} \tag{2.8}$$

Observamos que el filtro adaptivo tiene que invertir la función de transferencia $S'(z)$; sin embargo, es imposible compensar por el retardo en el canal secundario provocado por $S'(z)$ si el canal primario $P'(z)$ no contiene un retardo de por lo menos igual duración.

En la ecuación (2.7), el sistema de control es inefectivo si existe una frecuencia para la que $R(\omega) = 0$; en otras palabras, un cero en la función de transferencia del sensor de error causa una componente en frecuencia fuera del alcance del sistema. Incluso si se realiza un filtro adaptable estable en estas circunstancias, no es posible obtener una cancelación de ruido significativa a dicha frecuencia: $P'(\omega)S'(\omega)W(\omega)$ puede tomar cualquier valor pero el controlador nunca sabrá si $R(\omega) = 0$. Esto no lleva a que el ruido primario solamente es controlable si el sistema es observable por medio del sensor de error. Un sistema *CAR* será inefectivo para frecuencias fuera de rango e inestable para frecuencias incontrolables sin importar el algoritmo adaptivo o la estructura de control.

Para la ecuación (2.8), el filtro adaptable $W(z)$ realiza un modelado directo de la función acústica primaria $P'(z)$ para generar el ruido cancelador, y un modelado inverso de $S'(\omega)$ para compensar el efecto de la trayectoria secundaria. En términos de procesamiento de señales en tiempo real, la introducción de la función de la trayectoria secundaria en un controlador usando un algoritmo adaptable estándar, causará de manera general inestabilidad. Esto es porque la señal de error no está correctamente alineada en el tiempo con la señal de referencia, debido a la presencia de $S(z)$.

Como las funciones de transferencia primaria y secundaria tienen a $R(z)$ en común, es conveniente introducir este término en el modelo, lo que resulta en el diagrama equivalente de la figura 2.7.

De las ecuaciones (2.5) a la (2.7), la función de estado estable del filtro adaptivo puede ser expresada de manera mas sencilla, quedando la *transformada z* de la señal de error como:

$$E(z) = \big[P(z) - S(z)W(z)\big]X(z) \tag{2.9}$$

Despues de que el filtro converge, el error residual es cero $[E(z) = 0]$, que requiere que $W(z)$

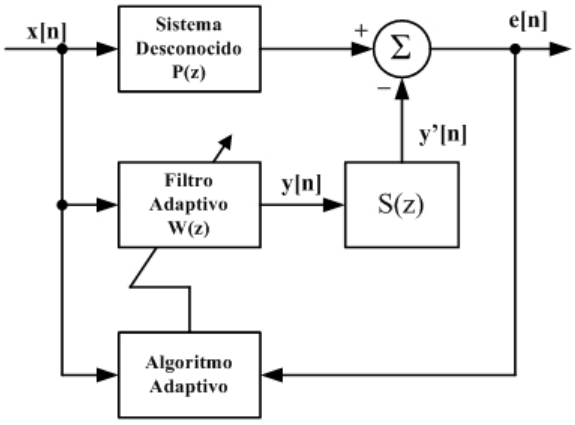


Figura 2.7: Diagrama de bloques simplificado de un sistema *CAR*.

llegue a la función de transferencia óptima:

$$W^o(z) = \frac{P(z)}{S(z)} \tag{2.10}$$

En otras palabras, el filtro adaptivo tiene que modelar $P(z)$ y realizar el modelado inverso de $S(z)$. Una vez conocida las respuestas de las funciones primarias y secundarias se puede calcular los coeficientes óptimos en el filtro adaptable. Debido a factores no previstos en las características del sistema acústico, no es práctico determinar el modelo de la trayectoria secundaria antes de poner en marcha el sistema. Esto es debido a que el filtro $W(z)$ nos provee con los medios necesarios para determinar el modelo del sistema desconocido $P(z)$ y la trayectoria secundaria $S(z)$, siempre y cuando se tenga una fuente de ruido que necesite ser minimizado. Una ventaja clara de esta técnica es que con el correcto modelado del sistema, el filtro puede responder instantaneamente a cambios en la señal de entrada provocados por cambios en la fuente de ruido.

Podemos concluir por consiguiente que el desempeño de un sistema de Control Activo del Ruido depende principalmente de la función de transfrencia de la trayectoria secundaria

2.4. Control de Realimentación

Como vimos en la sección anterior los sistemas *Feedforward* de CAR utilizan un sensor de error que obtiene la señal residual de error, la cual se utiliza para actualizar los coeficientes del filtro adaptivo. Los sistemas CAR de *Feedback* (o control de realimentación) utilizan un solo sensor cuya salida es usada por sistema adaptivo para generar la señal secundaria. Un diagrama de un sistema CAR con control de realimentación se muestra en la figura 2.8

Esta técnica se puede considerar como un sistema a priori (feedforward) que sintetiza o regenera su propia señal de referencia, basándose solamente en la señal que recibe del único

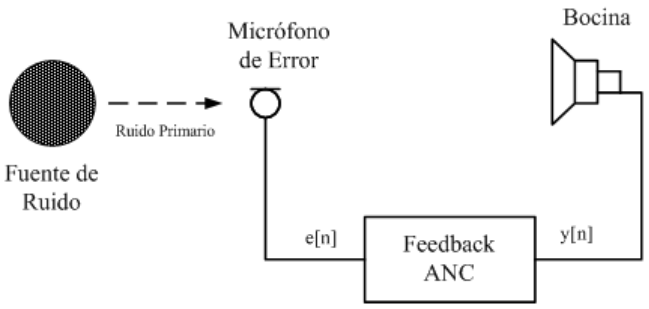


Figura 2.8: Sistema CAR con control de realimentación de un solo canal.

sensor de error y la salida del filtro adaptable[8]. Bajo ciertas condiciones, un sistema CAR con control de realiementación puede ser considerado como un predictor adaptable.

En la figura 2.9 se muestra un diagrama de bloques del sistema de la figura 2.8, donde $d[n]$ es el ruido primario en el sensor de error, $e[n]$ es el ruido residual censado por el micrófono de error, y la señal de control secundaria es denotada como $y[n]$. Como en el caso de CAR a priori, $W(z)$ es la función de transferencia del filtro adaptable y $S(z)$ la función de transferencia de la trayectoria secundaria.

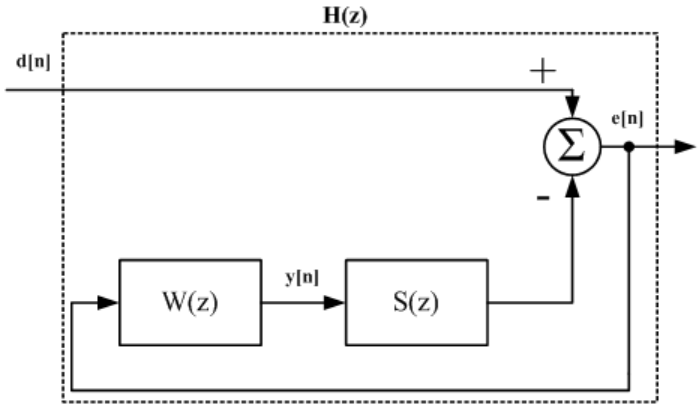


Figura 2.9: Diagrama de bloques del sistema CAR con control de realimentación de un solo canal.

En la aplicación de la figura 2.9, el ruido primario $d[n]$ no esta disponible durante la operación de Control Activo del Ruido debido a que debe de ser cancelado por el ruido secundario. Por lo tanto, la idea básica de un esquema de realimentación aplicado a CAR es estimar el ruido primario y usarlo como señal de referencia $x[n]$ para el filtro adaptable $W(z)$. La transformada Z del ruido primario se puede expresar como:

$$D(z) = E(z) + S(z)Y(z) \tag{2.11}$$

Donde $E(z)$ es la señal obtenida por el sensor de error y $Y(z)$ es la señal secundaria

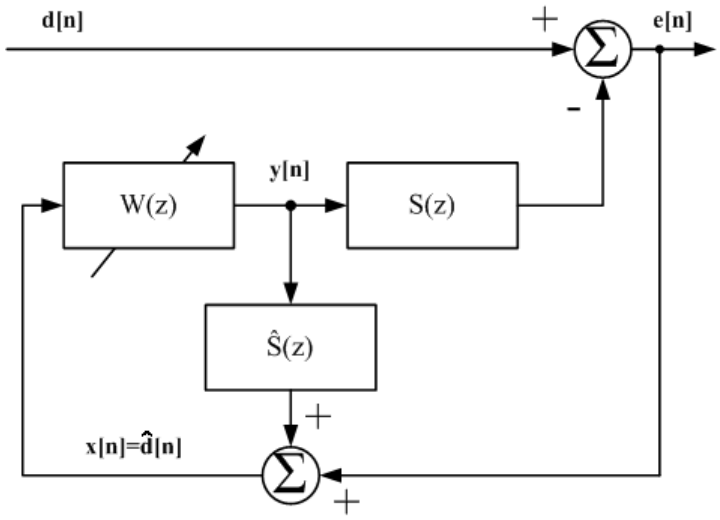


Figura 2.10: Feedback CAR usando la señal de referencia sintetizada de las señales disponibles ($y[n]$ y $e[n]$)

generada por el filtro adaptivo. De esta manera, tanto $E(z)$ como $Y(z)$ están disponibles para el proceso de adaptación. Si la función de transferencia de la trayectoria secundaria es medible y aproximada a $\hat{S}(z)$, de modo que $\hat{S}(z) \approx S(z)$, podemos estimar el ruido primario $d[n]$ y usar este como una señal sintética de referencia $x[n]$:

$$X(z) \equiv \hat{D}(z) = E(z) + \hat{S}(z)Y(z) \tag{2.12}$$

Esta técnica de síntesis (regeneración) de la señal de referencia se ilustra en la figura 2.10, donde la señal $y[n]$ es filtrada por el estimado de la trayectoria secundaria $\hat{S}(z)$ y después sumada a $e[n]$ para regenerar el ruido primario.Al mismo tiempo, se necesita compensar el canal secundario por medio de $\hat{S}(z)$. La señal de referencia $x[n]$ se genera como un estimado de $d[n]$, y se puede expresar como:

$$x[n] \equiv \hat{d}[n] = e[n] + \sum_{m=0}^{M-1} \hat{s}_m y[n-m] \tag{2.13}$$

Donde $\hat{s}_m$, con $m = 0, 1, \cdots, M-1$ son los coeficientes del filtro $\hat{S}(z)$ de orden M, usado para estimar la trayectoria secundaria. La señal secundaria generada es:

$$y[n] = \sum_{l=0}^{L-1} w_l[n]x[n-1] \tag{2.14}$$

Donde w_l, con $l = 0, 1, \cdots, L-1$ son los coeficientes de $W(z)$ en el tiempo n y L es el orden del filtro adaptivo. Estos coeficientes serán actualizados por el algoritmo adaptivo y

obtendremos la señal de referencia filtrada que la siguiente:

$$x'[n] \equiv \sum_{m=0}^{M-1} \hat{s}x[n-m] \tag{2.15}$$

2.4.1. Esquema de predicción adaptiva

De la ecuación (2.11) y (2.12) tenemos que $x[n] = \hat{d}[n]$ si $\hat{S}(z) = S(z)$. Cuando estas condiciones se satisfacen, el sistema CAR con control de realimentación (feedback) toma la forma de un sistema CAR con control a priori (feedforward), como se muestra en la figura 2.11. Si además asumimos que la trayectoria secundaria puede ser modelado como un simple retraso ($S(z) = z^{-\Delta}$), entonces el esquema mostrado en la figura 2.12 es idéntico a un esquema de predicción adaptiva.

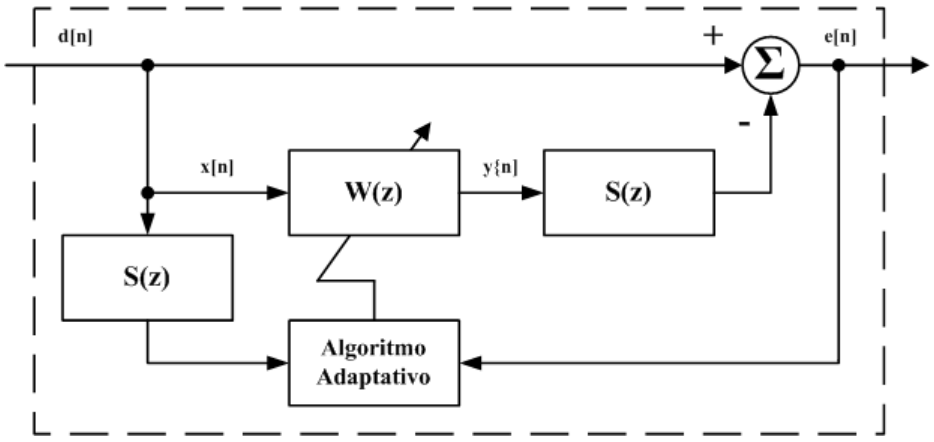


Figura 2.11: Feedback CAR toma la forma de un sistema CAR con control a priori (feedforward)

Aquí, la señal de referencia es $x[n] = d[n-\Delta]$, $y[n]$ es la señal primaria pronosticada y $e[n]$ es el error predictivo (o residual). La respuesta del sistema desde $d[n]$ a $e[n]$ se conoce como **Filtro de Predicción de Errores** (*prediction error filter*). El sistema adaptivo de control de realimentación usa la señal $x[n]$ sintetizada en la ecuación (2.13) y es equivalente a un predictor adaptivo asumiendo que la velocidad de convergencia del algoritmo adaptivo sea muy lenta y $S(z) = \hat{S}(z) = z^{-\Delta}$. El filtro adaptivo $W(z)$ del este tipo de sistema actúa como un predictor adaptivo del ruido primario $d[n]$ para minimizar el ruido residual $e[n]$, de modo que el desempeño de un sistema de control activo del ruido con control de realimentación depende en que tan predecible sea el ruido primario $d[n]$.

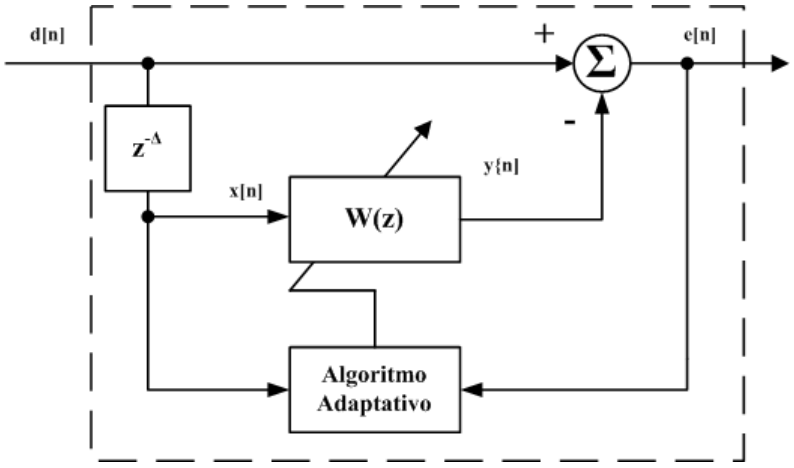


Figura 2.12: Diagrama de Bloques de un Predictor Adaptivo

2.5. Consideraciones Prácticas

El Control Activo del Ruido nunca proveerá una solución universal a **todos** los problemas de ruido[4]. Si bien el desempeño de los sistemas *CAR* pueden ser mejorados aumentando el número de sensores de error (microfonos) y salidas del sistema (bocinas); o incrementando la eficiencia de los algoritmos y del hardware para que sean mas rápidos y estables, existen limitaciones físicas fundamentales que limitan el desempeño y que no pueden ser superadas con mejoras en el procesamiento de la señal. La limitación más importante es que el Control Activo del Ruido está limitado a situaciones en las que la separación entre las fuentes primaria y secundaria es por lo menos del mismo orden que la longitud de onda acústica.

Cuando un sistema CAR es implementado en aplicaciones reales, surgen una gran cantidad de problemas que necesitan ser considerados. Es recomendable realizar un análisis de desempeño del sistema, el cual incluye una serie de técnicas; empezando con un problema simplificado e ideal y progresivamente se agregan restricciones prácticas y otras complejidades. Este análisis de desempeño resuelve los siguientes problemas:

1. Las limitaciones fundamentales de desempeño.
2. Las restricciones prácticas que limitan el desempeño.
3. Un balance de desempeño contra complejidad.
4. Como determinar un diseño de arquitectura práctico.

Por otro lado, si se desea usar el sistema CAR para uso industrial, se deben de garantizar ciertas propiedades:

1. Máxima eficiencia sobre la mayor banda de frecuencias posible para lograr cancelar una gran variedad de ruidos.

2. Autonomía con respecto a la instalación, de manera que sistema pueda ser construido y calibrado en un area diferente al sitio donde va a operar.

3. Adaptabilidad del sistema para poder operar correctamente al variar varios parámetros ambientales (temperatura)

4. Los componentes deben de ser lo suficientemente robustos y confiables, de manera que se simplifiquen los elementos electrónicos.

Capítulo 3

Filtros adaptables

3.1. Introducción

En Procesamiento Digital de Señales, el término filtrar se refiere al proceso lineal diseñado para alterar el contenido espectral de una señal de entrada de una manera específica. Los filtros convencionales son lineales e invariantes en el tiempo, realizan un conjunto constante de operaciones lineales en una secuencia de datos $x[n]$ para proporcionar una salida basada en el valor de los coeficientes del filtro.

En el caso de los filtros adaptables esta restricción de invariancia en el tiempo es eliminada. Un filtrado adaptable implica que los parámetros del filtro tales como ancho de banda y frecuencia de resonancia varían con el tiempo y son ajustados por un algoritmo especial.

Un sistema adaptable consta de 2 partes: un filtro digital adapteble que realiza el procesamiento deseado de la señal, y un algoritmo adaptable para ajustar los coeficientes del filtro.

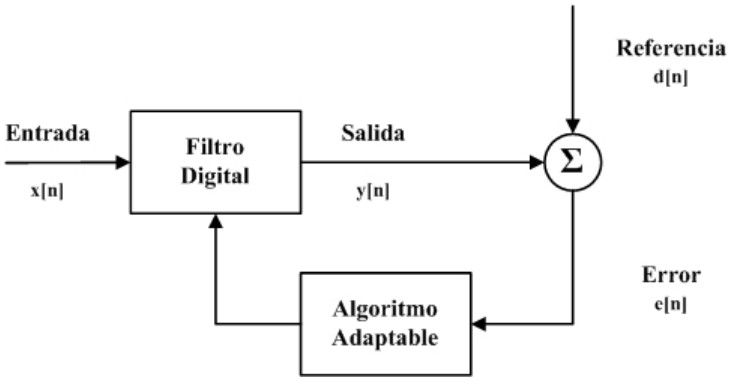


Figura 3.1: Filtro adaptable alimentado a priori *(feedforward)*

La característica principal de los filtros adaptables es que sus coeficientes varían con el tiempo. Como observamos en la figura 3.1 el filtro genera la señal de control $y[n]$ a partir de

la señal de referencia $x[n]$, la cual debe de estar correlacionada con $d[n]$. Al restartle a la señal $d[n]$ la señal $y[n]$ se genera la señal de error $e[n]$ y entonces el algoritmo adaptable ajusta los coeficientes del filtro digital, muestra con muestra, de forma que el error es progresivamente minimizado.

En general, hay dos estructuras para implementar un filtro digital adaptable: los filtros de respuesta finita al impulso (FIR) y los filtros de respuesta infinita al impulso (IIR). Los filtros más usados en el desarrollo de sistemas CAR son los filtros FIR, ya que son mas estables y proporcionan una respuesta lineal en fase.

El filtrado adaptativo es muy útil para el eliminar el ruido en sistemas cuya salida deba seguir a una entrada, como lo son amplificadores o servomecanismos. Esto se ejemplifica en la figura 3.2. El ruido en la entrada del sistema es modelado como ruido aditivo insertado en la salida del sistema. La señal contaminada proporciona la entrada el filtro adaptativo. Esta señal es filtrada para producir una salida sin ruido en y. La salida en y es comparada con la señal deseada, que en este caso es la señal de entrada x pero retrasada, de manera que se compensen los retardos propios del sistema y del filtro adaptable. Después de que el filtro se ha adaptado y iguala a d y el error e tiende a cero.

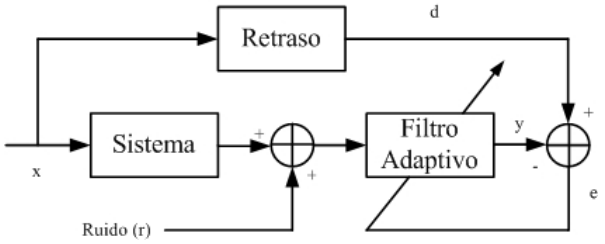


Figura 3.2: Cancelación de ruido a la salida

Otra aplicación es que el filtro funcione como un cancelador de ruido a la entrada. Como vemos en la figura 3.3 la señal x esta contaminada con ruido r. La entrada a el filtro es r', que está correlacionada con el ruido r en la señal. Cuando el sistema se ha adaptado, y se acerca al ruido aditivo n y la señal de error e se acerca a la señal de entrada x. Si x no esta correlacionado con r, lo que se busca entonces es minimizar $E^2(e)$, donde $E(\bullet)$ es el valor esperado de la señal de error.

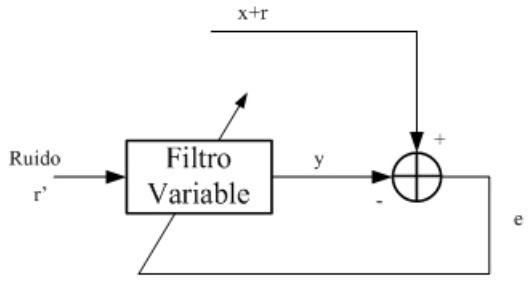


Figura 3.3: Cancelación de ruido a la entrada

La forma transversal del filtro FIR, que se puede apreciar en la figura 1.10, es la más usada, debido a su estabilidad y respuesta lineal en fase. Este tipo de filtros calculan su salida en forma lineal. Esto es, dado un conjunto L de coeficientes

$$w[n] = \{w_0[n], w_1[n], ..., w_{L-1}[n]\}^T \tag{3.1}$$

y la señal de entrada se define como:

$$x[n] = \{x[n], x[n-1], ..., x[n-L+1]\}^T \tag{3.2}$$

Por lo tanto la señal de salida será:

$$y[n] = \sum_{i=0}^{L-1} w_i[n]x[n-i] \tag{3.3}$$

La señal de salida en la ecuación (3.3) puede ser expresada con vectores de la siguiente manera:

$$\begin{aligned} y[n] &= w^T[n]x[n] \\ &= x^T[n]w[n] \end{aligned} \tag{3.4}$$

Regularmente la salida $y[n]$ calculada en la ecuación (3.4) se compara con la señal deseada $d[n]$, de donde resulta una señal de error o diferencia. La señal de error está definida por la ecuación (3.5):

$$\begin{aligned} e[n] &= d[n] - y[n] \\ &= d[n] - w^T[n]x[n] \end{aligned} \tag{3.5}$$

A partir de la cual podremos determinar las modificaciones a los coeficientes del filtro para poder minimizar el error residual, utilizando algoritmos adaptivos.

3.2. Algoritmos Adaptivos

Muchas aplicaciones prácticas incluyen la reducción de ruido y distorsión para la extracción de la información de la señal recibida. La degradación de la señal en algunos casos es desconocida, cambiante en el tiempo o ambas. Los filtros adaptivos sugieren un excelente comportamiento en este tipo de aplicaciones, ya que pueden modificar sus características para lograr ciertos objetivos y generalmente acompañan la modificación automáticamente.

3.2.1. Algoritmo MSE

Para poder minimizar la señal de error debemos considerar como *estacionarias y estáticas* las señales $d[n]$ y $x[n]$, para realizar esta reducción de la señal de error es necesario modificar

los valores de los coeficientes del filtro también conocidos como 'pesos' del filtro; para hacer esto contamos con diversos algoritmos de adaptación. La señal de error sirve como referencia para empezar a deducir los métodos de adaptación de los coeficientes del filtro, sin embargo, algo que se debe tener en consideración antes es la superficie de error cuadrático medio. La superficie de *error cuadrático medio (MSE)* es un concepto fundamental en el desarrollo de algoritmos de adaptación, la cual es una función de los coeficientes del filtro. Con el fin de obtener una expresión para la superficie de error cuadrático medio considere la salida del filtro dada por la ecuación (3.3) y la señal de error dada por la ecuación (3.5), y si tomamos la definición del error cuadrático medio $\xi[n]$ dada por la siguiente ecuación:

$$\xi[n] = E\{e^2[n]\} \tag{3.6}$$

Donde $E[\bullet]$ representa la esperanza matemática o valor esperado, y $e[n]$ es el error definido por la ecuación (3.5). Si consideramos que el vector $w[n]$ es una secuencia determinista, podemos sustituir la ecuación (3.5) en (3.6)

$$\xi[n] = E\{(d[n] - w^T[n]x[n])(d[n] - w^T[n]x[n])\} \tag{3.7}$$

Entonces la función MSE puede ser determinada por:

$$\xi[n] = E\{d^2[n]\} - 2\mathbf{p}^T w[n] + w^T[n]\mathbf{R}w[n] \tag{3.8}$$

Donde $\mathbf{p}$ es el vector de correlación cruzada y $\mathbf{R}$ la matriz de autocorrelación, que se definen como sigue:

$$\begin{aligned} \mathbf{p} &= E\{d[n]x[n]\} \\ &= [r_{dx}(0)\ r_{dx}(1)\ ...\ r_{dx}(L-1)]^T \end{aligned} \tag{3.9}$$

Donde

$$r_{dx}(k) = E\{d[n]x[n-k]\} \tag{3.10}$$

Es la función de correlación entre $d[n]$ y $x[n]$, por otro lado:

$$R = E\{x[n]x^T[n]\} \tag{3.11}$$

$$= \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(L-1) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(L-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(L-1) & r_{xx}(L-2) & \cdots & r_{xx}(0) \end{bmatrix} \tag{3.12}$$

Donde:

$$r_{xx}(k) = E\{x[n]x[n-k]\} \tag{3.13}$$

Es la función de autocorrelación de $x[n]$. El filtro óptimo $w^o[n]$ minimiza la función $\xi[n]$. Utilizando el gradiente e igualando a cero para obtener dicha función encontramos:

$$\nabla\xi[n] = 2\mathbf{R}w[n] - 2\mathbf{p} = 0 \tag{3.14}$$

$$\mathbf{R}w^o[n] = \mathbf{p} \tag{3.15}$$

$$w^o[n] = \mathbf{p}\mathbf{R}^{-1} \tag{3.16}$$

La ecuación anterior, conocida como la ecuación Wiener-Hopf[17], provee en principio, una solución al problema de filtrado adaptivo; sin embargo, esta solución requiere un continuo calculo de la matriz de correlación y el vector de correlación cruzada. En muchos casos, la señal será no estacionaria y la estimación de $\mathbf{R}$ y $\mathbf{p}$ requiere de un gran número de cálculos. Si bien estos cálculos pueden ser realizados eficientemente usando varios algoritmos computacionales, cuando las dimensiones de la matriz de correlación es grande, los cálculos requeridos para obtener el vector óptimo pueden llegar a ser bastante grandes.

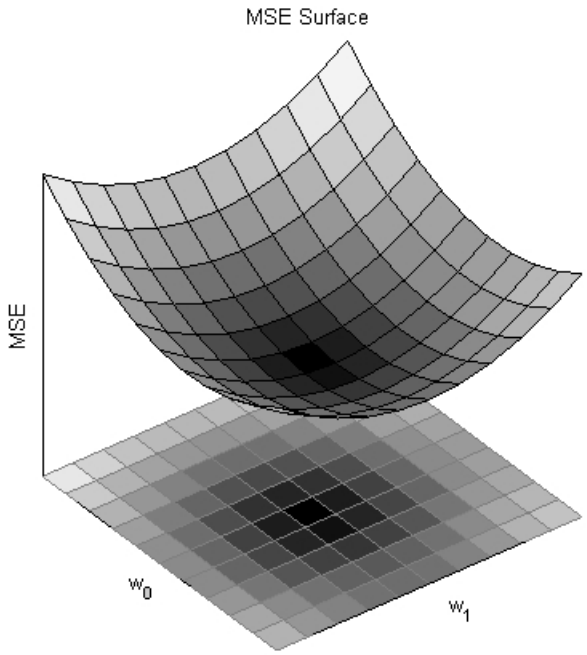


Figura 3.4: Superficie de desempeño tridimensional para el caso $L = 2$

Para obtener el mínimo MSE, sustituimos el vector de coeficientes óptimo $w^o[n]$ de la ecuación (3.15) por $w[n]$ en la ecuación (3.8)

$$\xi_{min} = E\{d^2[n]\} - \mathbf{p}w^o[n] \tag{3.17}$$

Combinando ente resultado con las ecuaciones (3.8) y (3.15), expresamos el error cuadrático medio como:

$$\xi[n] = \xi_{min} + \{w[n] - w^o[n]\}^T\mathbf{R}\{w[n] - w^o[n]\} \tag{3.18}$$

$$= \xi_{min} + v^T[n]\mathbf{R}v[n] \tag{3.19}$$

Donde

$$v[n] = w[n] - w^o[n]$$

Es el vector de desviación de coeficientes, esto es, la diferencia entre los coeficientes del filtro adaptivo y la solución óptima.

Es importante notar que por cada valor del vector de coeficientes $w[n]$, existe un correspondiente valor escalar del MSE, por lo tanto, estos valores asociados con $w[n]$ forman un espacio de $(L+1)$ dimensiones, que es comúnmente llamada **superficie de desempeño**. Para $L=2$, esto corresponde a una superficie de error en un espacio de tres dimensiones, como la mostrada en la figura 3.4.

3.2.2. Método de pasos descendentes

El método de pasos descendentes (MSD - method of steepest descent) es una técnica iterativa que sirve para obtener un algoritmo adaptivo debido a que la superficie de error es cuadrática con respecto a los coeficientes w_i. Su funcionamiento puede ser fácilmente explicado con ayuda de la gráfica mostrada en la figura 3.4, la cual al ser una función cuadrática de los coeficientes del filtro, podemos imaginarla como una superficie hiperbólica o un tazón. Al ajustar los coeficientes para disminuir el error significa descender a lo largo de la superficie hasta llegar al "fondo del tazón"de esta forma obtener el mínimo ξ_{min}, en este momento, los componentes del vector de coeficientes obtienen sus valores óptimos.

Supongamos que $\xi[0]$ representa el valor del MSE en el tiempo $n=0$ con una elección arbitraria de coeficientes $w[0]$, el método de pasos descendentes nos permite descender al fondo del tazón, w^o, de una forma sistemática. La idea es moverse en la superficie de error en dirección a la tangente de dicho punto, los coeficientes del filtro son actualizados en cada iteración en dirección negativa al gradiente de la superficie de error.

El desarrollo matemático de este método es evidenciado de manera muy facil, si se usa el acercamiento geométrico descrito en los párrafos anteriores. Como se observa en la figura 3.4, cada selección de un filtro $w[n]$ corresponde a un solo punto en la superficie MSE, $\{w[n], \xi[n]\}$. Continuando con los coeficientes arbitrariamente escogidos ($w[0]$) en la superficie MSE, tendremos que en el punto $\{w[0], \xi[0]\}$ existe una orientación específica a la superficie, descrita por las derivadas direccionales de la superficie en dicho punto. Estas derivadas direccionales cuantifican la taza de cambio de la superficie MSE con respecto a los ejes coordinados $w[n]$. Esto es, en dicho punto $\{w[0], \xi[0]\}$, existe una pendiente a la superficie a lo largo de una línea paralela a cada eje w_i. Estas pendientes tienen valores definidos por las derivadas direccionales $\partial\xi[n]/\partial w_i$. El gradiente de la superficie de error $\nabla\xi[n]$ es definido como el vector de esas derivadas direccionales.

De esta manera, el concepto de descenso mas corto puede ser expresado de la siguiente manera:

$$w[n+1] = w[n] - \frac{\mu}{2}\nabla\xi[n] \tag{3.20}$$

Donde μ es el factor de convergencia que controla la estabilidad y el ritmo de descenso al fondo del tazón; mientras más grande sea el valor de μ, más rápido será la velocidad de descenso. El vector $\nabla\xi[n]$ denota el gradiente de la función de error con respecto a $w[n]$ y el signo negativo dirige el vector de coeficientes en dirección negativa al gradiente.

De la ecuación (3.8), podemos calcular el gradiente de la función de error:

$$\nabla\xi[n] = -2\mathbf{p} + 2\mathbf{R}w[n] \tag{3.21}$$

Substituyendo en la ecuación (3.20) obtenemos la forma final del algoritmo de pasos descendentes

$$w[n+1] = w[n] - \mu[\mathbf{p} - \mathbf{R}w[n]] \tag{3.22}$$

Cuando $w[n]$ converge a w^o, llegando al mínimo de la superficie de error, el gradiente $\nabla\xi[n] = 0$. En este momento, la adaptación en la ecuación (3.22) se detiene y el vector de coeficientes se mantiene en la solución óptima.

3.3. Algoritmo LMS

El algoritmo LMS (Least Mean Square algorithm), desarrollado por Bernard Widrow y Edward Hopf en el año de 1959, usa el descenso del gradiente para estimar una señal variable en el tiempo. El método de descenso del gradiente determina el mínimo (si existe) tomando pasos en la dirección descendente del gradiente. Esto lo realiza ajustando los coeficientes del filtro transversal de manera que se minimice el error.Este algoritmo resulta ser el más ampliamente usado debido a su baja complejidad computacional y estabilidad.

Observamos de la ecuación (3.20) que el incremento de $w[n]$ a $w[n+1]$ se lleva a cabo en dirección contraria a la dirección del gradiente, de tal manera que los coeficientes seguirán aproximadamente el camino de mayor descenso en la superficie de error. Sin embargo, en muchas aplicaciones prácticas las estadísticas de $d[n]$ y $x[n]$ son desconocidas, de tal manera que el método del descenso mas corto no puede ser usado de forma directa, ya que este necesita conocer exactamente el vector gradiente en cada iteración. Una solución a este problema fue propuesta por Widrow[16], usando el error cuadrático instantáneo $e^2[n]$, para estimar el error cuadrático medio de modo que:

$$\xi[n] = e^2[n] \tag{3.23}$$

De esta manera, la estimación del gradiente usada por el algoritmo LMS es simplemente el gradiente instantáneo de una muestra del error cuadrático:

$$\nabla\xi[n] = 2\{\nabla e[n]\}e[n] \tag{3.24}$$

Como $e[n] = d[n] - w^T[n]x[n]$,

$$\nabla e[n] = -x[n] \tag{3.25}$$

Y la estimación del gradiente se vuelve:

$$\nabla\hat{\xi}[n] = -2x[n]e[n] \tag{3.26}$$

Sustituyendo esta estimación del gradiente en la ecuación del algoritmo de pasos descendentes (ecuación (3.20)) tenemos

$$w[n+1] = w[n] + \mu x[n]e[n] \tag{3.27}$$

Esta es la ecuación del algoritmo LMS o algoritmo de gradiente estocástico. El algoritmo es simple y no requiere de operaciones complejas. El diagrama de bloques de la siguiente figura muestra el algoritmo LMS para filtros transversales:

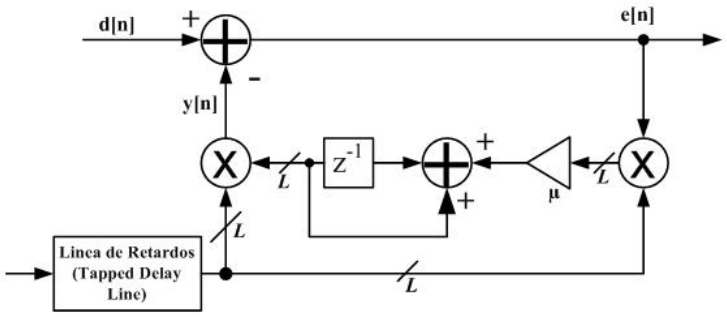


Figura 3.5: Diagrama de bloques de el algoritmo LMS

El algoritmo puede ser resumido como:

1. Se escogen los parámetros y las condiciones iniciales: L, μ y $w[0]$. Donde L es el orden del filtro, μ es el parámetro de rapidez y $w[0]$ el vector inicial de coeficientes en el tiempo $n = 0$.

2. Se calcula la salida del filtro adaptable.

$$y[n] = \sum_{l=0}^{L-1} w_l[n]x[n-l] \tag{3.28}$$

3. Cálculo de la señal de error.

$$e[n] = d[n] - y[n] \tag{3.29}$$

4. Se actualiza el vector de coeficientes de $w[n]$ a $w[n+1]$ usando el algoritmo LMS.

$$w_l[n+1] = w_l[n] + \mu x[n-l]e[n],\ l = 0, 1, \cdots, L-1 \tag{3.30}$$

Observamos que la ecuación (3.28) requiere de L multiplicaciones y $L-1$ sumas. La operación de actualización de coeficientes (3.30) requiere $L+1$ multiplicaciones y L adiciones.

3.3.1. Algoritmo FXLMS

Como vimos en el capitulo anterior, la función de transferencia $S(z)$ de la trayectoria secundaria se encuentra generalmente despues del filtro adaptivo; es por esto que el algoritmo LMS debe de ser modificado para asegurar la convergencia, de esta manera surge el algoritmo FXLMS o *Filtered-X Least Mean Square algorithm.*

Existen varias formas de compensar el efecto de $S(z)$. Entre todas ellas dos destacan, la primera consiste en colocar un filtro inverso $1/S(z)$ en serie con $S(z)$ para contrarrestar su efecto. La segunda forma consiste en colocar un filtro idéntico en la trayectoria de la señal de referencia al filtro adaptivo de manera que realice el algoritmo de *filtrado x* (FXLMS). Debido a que la función inversa de $S(z)$ no siempre existe, el algoritmo FXLMS es el método más efectivo.

Este algoritmo asegura la convergencia, la entrada es filtrada por una copia (un estimado) de la trayectoria secundaria. La posición de la trayectoria secundaria despues del filtro adaptable puede observarse en la figura 2.7. La señal residual se puede expresar como:

$$\begin{aligned} e[n] &= d[n] - y'[n] \\ &= d[n] - s[n] * y[n] \\ &= d[n] - s[n] * \left\{ w^T[n] x[n] \right\} \end{aligned} \tag{3.31}$$

Donde $s[n]$ es la respuesta al impulso de la trayectoria secundaria y el operador $*$ es la convolución lineal, además

$$w[n] = \Big\{ w_0[n] \; w_1[n] \; \cdots \; w_{L-1}[n] \Big\}^T \tag{3.32}$$

es el vector de coeficientes $W(z)$ en el tiempo n y

$$x[n] = \Big\{ x[n] \; x[n-1] \; \cdots \; x[n-L+1] \Big\}^T \tag{3.33}$$

es el vector que representa la señal en el tiempo n y L es el orden del filtro $W(z)$. El objetivo del filtro es minimizar el error cuadrático instantaneo, $\hat{\xi} = e^2[n]$. Como se discutió en la sección anterior, el algoritmo LMS actualiza el vector de coeficientes en la dirección de gradiente negativo con un tamaño de paso de μ:

$$w[n+1] = w[n] - \frac{\mu}{2} \nabla \hat{\xi}[n] \tag{3.34}$$

Donde $\nabla \hat{\xi}[n]$ es el estimado instantaneo del gradiente MSE en el tiempo n y puede ser expresado como:

$$\nabla \hat{\xi}[n] = \nabla e^2[n] = 2 \Big\{ \nabla e[n] \Big\} e[n] \tag{3.35}$$

De la ecuación (3.31) tenemos que

$$\nabla e[n] = -s[n] * x[n] = -x'[n] \tag{3.36}$$

Donde:

$$x'[n] = \Big\{ x'[n] \; x'[n-1] \; \cdots \; x'[n-L+1] \Big\}^T \tag{3.37}$$

y

$$x'[n] = s[n] * x[n] \tag{3.38}$$

Por lo tanto. el estimado del gradiente se vuelve

$$\nabla\hat{\xi}[n] = -2x'[n]e[n] \tag{3.39}$$

Substituyendo la ecuación anterior en (3.34), obtenemos el algoritmo FXLMS

$$w[n+1] = w[n] + \mu x'[n]e[n] \tag{3.40}$$

El resultado anterior muestra que cuando la función de transferencia de la trayectoria secundaria $S(z)$, es antecedida por el filtro adaptivo la misma función debe de ser colocada en la trayectoria de la actualización de los coeficientes, de ahi el nombre de **algoritmo FXLMS**

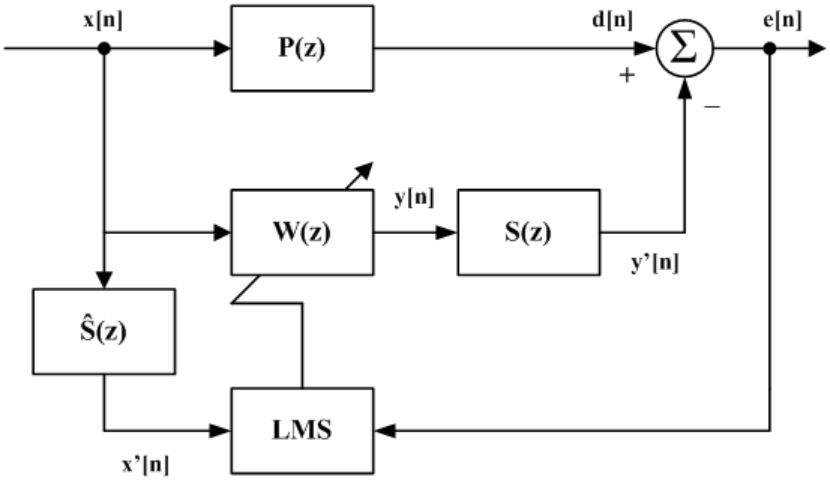


Figura 3.6: Diagrama de bloques de el algoritmo FXLMS

En aplicaciones prácticas, $S(z)$ es desconocido y debe de ser estimado por un filtro adicional, $\hat{S}(z)$. De esta manera, el vector de referencia filtrado es generado al pasar la señal de referencia por un estimado de la trayectoria secundaria, como se ilustra en la figura 3.6; el algoritmo FXLMS puede ser expresado como:

$$w[n+1] = w[n] + \mu\hat{s}[n] * x[n]e[n] \tag{3.41}$$

3.4. Algoritmo RLS

El algoritmo recursivo de mínimos cuadrados o RLS (Recursive Least Square) es un algoritmo que provee mayor velocidad de convergencia y menor error que el algoritmo LMS. Sin embargo, el algoritmo RLS requiere de L^2 operaciones (siendo L el orden del filtro) a diferencia de las $4L$ operaciones que requiere el algoritmo LMS. Sin embargo, con la continua mejora en capacidad de procesamiento y memoria de los DSP's este algoritmo se ha vuelto una opción viable en muchas aplicaciones actuales.

Como en el algoritmo LMS, el algoritmo RLS inicia el proceso de adaptación con el filtro en ciertas condiciones iniciales y se usan las muestras siguientes para adaptar los coeficientes del filtro. La función de costo en el tiempo n consiste de la suma de el error cuadrático estimado expresado por:

$$\xi[n] = \sum_{i=1}^{n} \lambda^{n-i} e^2[i] \tag{3.42}$$

Donde $0 \leq \lambda \leq 1$ es el factor de estimación, que estima los datos mas recientes asignandoles un mayor 'peso' para acomodar señales no estacionarias. La derivación del algorítmo RLS asume que el vector de estimación $w[n]$ es constante durante el intervalo $(1, n)$. De esta manera, la señal de error en el tiempo i se forma como:

$$e[i] = d[i] - w^T[n]x[i] \quad 1 \leq i \leq n \tag{3.43}$$

Donde $w[n]$ es el vector de estimación actual del filtro adaptivo orden L

$$w[n] \equiv \{w_0[n]\ w_1[n]\ \cdots\ w_{L-1}[n]\}^T \tag{3.44}$$

Además $x[i]$ es el vector señal de referencia con dimensiones $L \times 1$ al tiempo i.

$$x[i] \equiv \{x[i]\ x[i-1]\ \cdots\ x[i-L+1]\}^T \tag{3.45}$$

Substituyendo la ecuación (3.43) en la ecuación (3.42) obtenemos los siguiente:

$$\xi[n] = \sum_{i=1}^{n} \lambda^{n-i} d^2[i] - 2w^T[n] \left\{ \sum_{i=1}^{n} \lambda^{n-i} d[i]x[i] \right\} + w^T[n] \left\{ \sum_{i=1}^{n} \lambda^{n-i} x[i]x^T[i] \right\} w[n] \tag{3.46}$$

Además definimos la matriz de correlación de las muestras como:

$$\mathbf{R}[n] = \sum_{i=1}^{n} \lambda^{n-i} x[i]x^T[i] \tag{3.47}$$

y el vector de correlación cruzada de las muestras es:

$$\mathbf{p}[n] = \sum_{i=1}^{n} \lambda^{n-i} d[i]x[i] \tag{3.48}$$

La ecuación (3.46) puede ser expresada como:

$$\xi[n] = \sum_{i=1}^{n} \lambda^{n-i} d^2[i] - 2\mathbf{p}^T[n]w[n] + w^T[n]\mathbf{R}[n]w[n] \tag{3.49}$$

Esta ecuación describe el error cuadrático acumulado y puede ser minimizado con respecto a $w[n]$ en cada muestra n haciendo que el gradiente $\xi[n]$ con respecto a $w[n]$ sea cero. Esto dá como resultado:

$$\mathbf{R}[n]w^o[n] = \mathbf{p}[n] \tag{3.50}$$

Donde $w^o[n]$ es el vector de coeficientes óptimo que en el tiempo n minimiza la suma de los errores cuadráticos $\xi[n]$ definida en la ecuación (3.42). Si la matriz inversa $\mathbf{R}^{-1}[n]$ existe, podemos resolver esta ecuación para obtener una solución unica para $w^o[n]$.

Debido a que el índice n puede incrementar a valores muy grandes en casos de procesamiento de tiempo real, el cálculo de $\mathbf{R}[n]$ como está definido en la ecuación (3.47) y $\mathbf{p}[n]$ de la ecuación (3.48) puede llegar a complicarse. Este problema se resuelve usando un algoritmo recursivo, que calcule $\mathbf{R}[n]$ y $\mathbf{p}[n]$ de la matriz previa $\mathbf{R}[n-1]$ y el vector $\mathbf{p}[n]$, incorporando el nuevo vector señal de referencia $x[n]$ y la señal primaria $d[n]$. De esta manera, de la ecuación (3.47), la señal de correlación de muestras puede ser expresada como:

$$\begin{aligned}\mathbf{R}[n] &= \sum_{i=1}^{n} \lambda^{n-i} x[i] x^T[i] \\ &= \sum_{i=1}^{n-1} \lambda^{n-i} x[i] x^T[i] + \lambda^o x[n] x^T[n] \\ &= \lambda \sum_{i=1}^{n-1} \lambda^{(n-1)-i} x[i] x^T[i] + x[n] x^T[n] \\ &= \lambda \mathbf{R}[n-1] + x[n] x^T[n] \end{aligned} \tag{3.51}$$

Esta ecuación muestra que la matriz $\mathbf{R}[n]$ actual se puede obtener de manera recursiva de la matriz previa $\mathbf{R}[n-1]$ y el vector actual $x[n]$. De manera similar podemos expresar el vector $\mathbf{p}[n]$ de la ecuación (3.48) en su forma recursiva:

$$\mathbf{p}[n] = \lambda \mathbf{p}[n-1] + d[n] x[n] \tag{3.52}$$

Al usar las dos ecuaciones anteriores, la complejidad del algoritmo RLS se reduce al orden de L^3, siendo el cálculo de la matriz inversa $\mathbf{R}^{-1}[n]$ la que mayor carga de cómputo requiere. Para obtener la matriz inversa en su forma recursiva en la forma:

$$\mathbf{R}[n] = \mathbf{R}[n-1] + \textit{Actualización}[n] \tag{3.53}$$

Podemos usar el siguiente teorema el cual establece que:

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^T\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{B} \tag{3.54}$$

Donde $\mathbf{A}$ y $\mathbf{B}$ son dos matrices $L \times L$ y definimos como sigue:

$$\mathbf{A} \equiv \lambda \mathbf{R}[n-1] \tag{3.55}$$
$$\mathbf{B} \equiv x[n] \tag{3.56}$$
$$\mathbf{C} \equiv \mathbf{I}^1 \tag{3.57}$$
$$\mathbf{D} \equiv x^T[n] \tag{3.58}$$

La ecuación (3.51) puede ser expresada como:

$$\begin{aligned}\mathbf{R}[n] &= \lambda\mathbf{R}[n-1] + x[n]x^T[n] \\ &= \mathbf{A} + \mathbf{BCD}\end{aligned} \tag{3.59}$$

De las ecuaciones (3.54) y (3.59) obtenemos:

$$\begin{aligned}\mathbf{R}^{-1}[n] &= (\mathbf{A} + \mathbf{BCD})^{-1} \\ &= \lambda^{-1}\mathbf{R}^{-1}[n-1] - \frac{\lambda^{-1}\mathbf{R}^{-1}[n-1]x[n]x^T[n]\lambda^{-1}\mathbf{R}^{-1}[n-1]}{x^T[n]\lambda^{-1}\mathbf{R}^{-1}[n-1]x[n] + 1}\end{aligned} \tag{3.60}$$

La ecuación anterior muestra que $\mathbf{R}^{-1}[n]$ puede ser calculada directamente de la matriz previa $\mathbf{R}^{-1}[n-1]$ usando el vector $x[n]$. Entonces no es necesario calcular o invertir $\mathbf{R}[n]$. Ahora definimos la matriz $L \times L$

$$\mathbf{Q}[n] = \mathbf{R}^{-1}[n] \tag{3.61}$$

Esto para evitar confundir que la inversión de la matriz realmente sucede y definimos el vetor *Kalman* de ganancia con dimensiones $L \times 1$ como:

$$\mathbf{k}[n] = \frac{\lambda^{-1}\mathbf{Q}[n-1]x[n]}{\lambda^{-1}x^T[n]\mathbf{Q}[n-1]x[n] + 1} \tag{3.62}$$

De modo que la ecuación (3.60) pueda ser reescrita como:

$$\mathbf{Q}[n] = \lambda^{-1}\mathbf{Q}[n-1] - \lambda^{-1}\mathbf{k}[n]x^T[n]\mathbf{Q}[n-1] \tag{3.63}$$

De esta manera la matriz $Q[n]$ es calculada de forma recursiva, reduciendo el número de cálculos necesarios. Reacomodando la ecuación (3.62) obtenemos:

$$\begin{aligned}\mathbf{k}[n] &= \lambda^{-1}\mathbf{Q}[n-1]x[n] - \lambda^{-1}\mathbf{k}[n]x^T[n]\mathbf{Q}[n-1]x[n] \\ &= \Big(\lambda^{-1}\mathbf{Q}[n-1] - \lambda^{-1}\mathbf{k}[n]x^T[n]\mathbf{Q}[n-1]\Big)x[n] \\ &= \mathbf{Q}[n]x[n]\end{aligned} \tag{3.64}$$

De las ecuaciones (3.50), (3.52) y (3.61) obtenemos la ecuación recursiva para actualizar el vector de coeficientes:

$$\begin{aligned}w[n] &= \mathbf{Q}[n]\mathbf{p}[n] \\ &= \mathbf{Q}[n]\Big(\lambda\mathbf{p}[n-1] + d[n]x[n]\Big) \\ &= \lambda\mathbf{Q}[n]\mathbf{p}[n-1] + d[n]\mathbf{Q}[n]x[n]\end{aligned} \tag{3.65}$$

Sustituyendo la ecuación (3.63) en la expresión anterior y usando (3.64).

$$\begin{aligned}w[n] &= \mathbf{Q}[n-1]\mathbf{p}[n-1] - \mathbf{k}[n]x^T[n]\mathbf{Q}[n-1]\mathbf{p}[n-1] + d[n]\mathbf{Q}[n]x[n] \\ &= w[n-1] - \mathbf{k}[n]x^T[n]w[n-1] + d[n]\mathbf{k}[n] \\ &= w[n-1] + \mathbf{k}[n]\Big(d[n] - w^T[n-1]x[n]\Big)\end{aligned} \tag{3.66}$$

[1]Siendo $\mathbf{I}$ la matriz identidad

Notamos que el termino $d[n] - w^T[n-1]x[n]$ en la ecuación anterior es en realidad el error *a priori*, que resulta de usar el vector de coeficientes previo $w[n-1]$. Sin embargo, para poder calcular este error es necesario esperar a que la nueva muestra n llegue, de manera que es necesario redefinir el vector de coeficientes actuales como si estuviera desplazado por una muestra:

$$\begin{aligned} w[n+1] &= w[n] + \mathbf{k}[n]\Big(d[n] - w^T[n]x[n]\Big) \\ &= w[n] + \mathbf{k}[n]e[n] \end{aligned} \tag{3.67}$$

Donde $e[n]$ se redefine como

$$e[n] = d[n] - w^T[n]x[n] \tag{3.68}$$

De esta manera el algoritmo RLS que resuelve para $w^o[n]$ de manera recursiva se resume de la siguiente manera:

$$e[n] = d[n] - w^T[n]x[n] \tag{3.69}$$

$$z[n] = \lambda^{-1}\mathbf{Q}[n-1]x[n] \tag{3.70}$$

$$\mathbf{k}[n] = \frac{z[n]}{x^T[n]z[n] + 1} \tag{3.71}$$

$$w[n+1] = w[n] + k[n]e[n] \tag{3.72}$$

Con

$$\mathbf{Q}[n] = \lambda^{-1}\mathbf{Q}[n-1] - k[n]z^T[n] \tag{3.73}$$

El vector $z[n]$ definido en la ecuación (3.70) es introducido por conveniencia dado que es repetidamente usado en las ecuaciones (3.71) y (3.72). El diagrama de bloques del algoritmo RLS se muestra en la figura 3.7, donde la actualización de los coeficientes es realizada como se expreso en la ecuación (3.72) y el vector de ganancias $\mathbf{k}[n]$ es actualizado por medio de las ecuaciones (3.70), (3.71) y (3.73).

Se puede deducir de las ecuaciones (3.69)-(3.73) que la complejidad del algoritmo RLS es del orden de $2L^2 + 4L$ multiplicaciones. Haciendo uso de ciertas propiedades de invariación al desplazamiento (*shift invariance*) del vector señal de referencia, la complejidad puede ser reducida aproximadamente a $7L$ usando el algoritmo de Filtro Transversal Rápido (*Fast Transversal Filter - FTF*). Cabe notar que si asumimos que:

$$\mathbf{Q}[n] = \mu\mathbf{I} \tag{3.74}$$

El vector de ganancias de Kalman obtenido en la ecuación (3.64) se simplifica a $\mu x[n]$. De modo que la ecuación de actualización de coeficientes (3.67) se reduce a:

$$w[n+1] = w[n] + \mu x[n]e[n] \tag{3.75}$$

Que es el algoritmo LMS obtenido en la sección anterior. Por lo tanto, en vez de calcular $\mathbf{Q}[n]$ usando un procedimiento muy sofisticado en las ecuaciones (3.70), (3.71) y (3.73), el algoritmo LMS simplemente asume que $\mathbf{Q}[n] = \mu\mathbf{I}$. Esta aproximación simplifica de manera considerable el algoritmo, con la desventaja de inferior desempeño en la velocidad de convergencia.

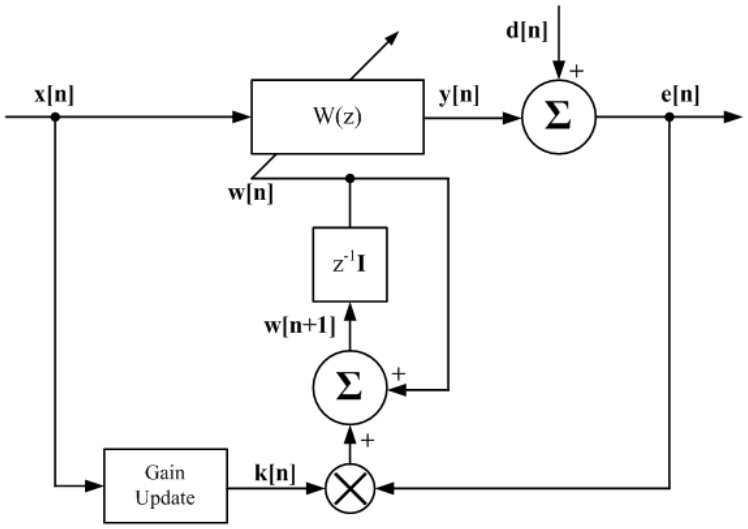


Figura 3.7: Diagrama de bloques de el algoritmo RLS.

3.4.1. Algoritmo RLS aplicado a CAR

Como se mencionó en la Sección 2.3, las aplicaciones de CAR generalmente tienen una trayectoria secundaria $S(z)$ despues del filtro adaptivo $W(z)$. Por lo tanto, la ecuación (3.43) se modifica de la siguiente manera:

$$\begin{aligned} e[i] &= d[i] - s[n] * \left(w^T[n]x[i]\right) \\ &\approx d[i] - w^T[n]x[i] \end{aligned} \tag{3.76}$$

Asumiendo que $w[n]$ varía lentamente y donde $s[n]$ es la respuesta al impulso de $S(z)$, además:

$$x'[i] = s[n] * x[i] \tag{3.77}$$

Es el vector señal de referencia filtrado por la trayectoria secundaria $S(z)$. Usando el mismo procedimiento que en la sección anterior para derivar el algoritmo RLS obtenemos que el algoritmo **RLS filtrado X (FXRLS)** puede ser resumido usando las siguientes ecuaciones:

$$e[n] = d[n] - s[n] * y[n] \tag{3.78}$$

$$y[n] = w^T[n]x[n] \tag{3.79}$$

$$z'[n] = \lambda^{-1}Q'[n-1]x'[n] \tag{3.80}$$

$$k'[n] = \frac{z'[n]}{z'^T[n]z'[n] + 1} \tag{3.81}$$

$$w[n+1] = w[n] + k'[n]e[n] \tag{3.82}$$

Con:

$$Q'[n] = \lambda^{-1}Q'[n-1] - k'[n]z'^T[n] \tag{3.83}$$

Donde el vector señal de referencia es

$$x'[n] \equiv \{x'[n]\ x'[n-1]\ \cdots\ x'[n-L+1]\}^T \tag{3.84}$$

Con los elementos:

$$x'[n] \equiv \hat{s}[n] * x[n] \tag{3.85}$$

Y $\hat{s}[n]$ es la respuesta al impulso de el estimado de la trayectoria secundaria $\hat{S}(z)$. Notamos que $y[n]$ en la ecuación (3.79) es la señal de cancelación que sale de la trayectoria secundaria y $e[n]$ en la ecuación (3.78) es el ruido residual censado por el micrófono de error. Un diagrama de bloques del algoritmo **FXRLS** se muestra a continuación.

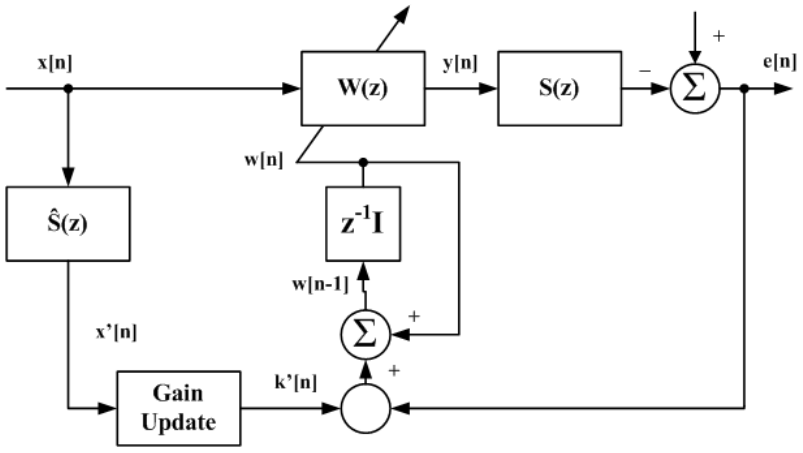


Figura 3.8: Diagrama de bloques de el algoritmo **FXRLS**.

3.5. Transformada Coseno Discreta Adaptiva

El filtro de transformada coseno discreta es la realización de un filtro con puros ceros en cascada con un banco de resonadores digitales *IIR*. El resultado de la configuración anterior es que cada uno de los resonadores IIR tiene un coeficiente de magnitud y un coeficiente de fase. De esta manera, cada coeficiente puede ser identificado con la amplitud o fase de la función de transferencia en una frecuencia particular cuando el filtro se usa como un filtro adaptivo.

El filtro de transformada coseno discreta puede ser implementado como un filtro a priori FXLMS que usa un solo micrófono de error y una sola bocina. Como se menciono anteriormente el filtro FIR es de la forma:

$$y[n] = \sum_{i=0}^{N-1} b_i x[n-i] \tag{3.86}$$

Y la función de transferencia es:

$$H(z) = \sum_{m=0}^{N-1} b_m z^{-m} \tag{3.87}$$

El paso clave en la formulación del filtro de transformada coseno discreta consiste en expresar los coeficientes b_m como la transformada de Fourier discreta de $B(k)$, de la siguiente manera:

$$b_m = \frac{1}{N} \sum_{k=0}^{N-1} B(k) e^{\frac{j2\pi mk}{N}} \tag{3.88}$$

Cuando la ecuación anterior es sustituida en la función de transferencia obtenemos:

$$H(z) = \sum_{m=0}^{N-1} \left(\frac{1}{N} \sum_{k=0}^{N-1} B(k) e^{\frac{j2\pi mk}{N}} \right) z^{-m} \tag{3.89}$$

Reacomodando el orden de las sumatorias resulta en

$$H(z) = \frac{1}{N} \sum_{k=0}^{N-1} B(k) \sum_{m=0}^{N-1} \left(e^{\frac{j2\pi k}{N}} z^{-1} \right)^m \tag{3.90}$$

La ecuación en la sumatoria interior puede ser resuelta usando series geométricas.

$$H(z) = \frac{1}{N} \sum_{k=0}^{N-1} B(k) \frac{[1 - e^{j2\pi k} z^{-N}]}{[1 - e^{\frac{j2\pi k}{n}} z^{-1}]} \tag{3.91}$$

que se simplifica a

$$H(z) = \frac{1}{N} (1 - z^{-N}) \sum_{k=0}^{N-1} \frac{B(k)}{1 - e^{\frac{j2\pi k}{N}} z^{-1}} \tag{3.92}$$

Una forma equivalente de la ecuación anterior es

$$H(z) = \frac{1}{N} (1 - z^{-N}) \sum_{k=0}^{\frac{N}{2}-1} \left(\frac{B(k)}{1 - e^{\frac{j2\pi k}{N}} z^{-1}} + \frac{B(N-k)}{1 - e^{-\frac{j2\pi k}{N}} z^{-1}} \right) \tag{3.93}$$

Como los valores de b_m son reales entonces $B(k)$ y $B(N-k)$ son complejos conjugados

$$B(N-k) = B^*(k) \tag{3.94}$$

Además, la forma general de $B(k)$ tiene magnitud y fase definidas como

$$B(k) = |B(k)| e^{j\phi(k)} \tag{3.95}$$

de manera que

$$B(N-k) = |B(k)| e^{j\phi(k)} \tag{3.96}$$

Sustituyendo en la ecuación (3.93) resulta en

$$H(z) = \frac{1}{N}(1 - z^{-N}) \sum_{k=0}^{\frac{N}{2}-1} |B(k)| \left(\frac{e^{j\phi(k)}}{1 - e^{\frac{j2\pi k}{N}} z^{-1}} + \frac{e^{-j\phi(k)}}{1 - e^{-\frac{j2\pi k}{N}} z^{-1}} \right) \tag{3.97}$$

Con un denominador común y agrupando términos tenemos, la ecuación se vuelve

$$\begin{aligned} H(z) =& \frac{1}{N}(1 - z^{-N}) \sum_{k=0}^{\frac{N}{2}-1} |B(k)| \\ & \cdot \left\{ \frac{e^{j\phi(k)} + e^{-j\phi(k)}}{1 - (e^{\frac{j2\pi k}{N}} + e^{\frac{j2\pi k}{N}}) z^{-1} + z^{-2}} \right. \\ & \left. - \frac{(e^{j(\frac{2\pi n}{N} - \phi(k))} + e^{-j(\frac{2\pi n}{N} - \phi(k))}) z^{-1}}{1 - (e^{\frac{j2\pi k}{N}} + e^{\frac{j2\pi k}{N}}) z^{-1} + z^{-2}} \right\} \end{aligned} \tag{3.98}$$

Simplificando la trigonometría da como resultado la siguiente función de transferencia:

$$\begin{aligned} H(z) =& \frac{2}{N}(1 - z^{N}) \times \\ & \sum_{k=1}^{\frac{N}{2}-1} |B(k)| \frac{\cos(\phi(k)) - z^{-1} \cos(\frac{2\pi k}{N} - \phi(k))}{1 - 2z^{-1} \cos(\frac{2\pi k}{N}) + z^{-2}} \end{aligned} \tag{3.99}$$

La cancelación de los polos y los ceros no será exacta y se llevará acabo en un circulo de radio $r < 1$. Con estas condiciones la función de transferencia se vuelve

$$\begin{aligned} H(z) =& \frac{2}{N}(1 - r^{N} z^{-N}) \times \\ & \sum_{k=1}^{\frac{N}{2}-1} |B(k)| \frac{\cos(\phi(k)) - r \cdot z^{-1} \cos(\frac{2\pi k}{N} - \phi(k))}{1 - 2rz^{-1} \cos(\frac{2\pi k}{N}) + r^2 z^{-2}} \end{aligned} \tag{3.100}$$

La implementación de la función de transferencia resulta en dos filtros en cascada: un filtro FIR con puros ceros con una banco de filtros pasobanda, los cuales son:

$$w[n] = \frac{2}{N} \Big[x[n] - r^{N} x[n - N] \Big] \tag{3.101}$$

$$\begin{aligned} y[n] =& \sum_{k=1}^{\frac{N}{2}-1} |B(k)| \Big\{ w[n] \cos(\phi(k)) - r \cdot \cos(\frac{2\pi k}{N} - \phi(k)) w[n-1] \Big\} \\ & + \sum_{k=1}^{\frac{N}{2}-1} 2r \cdot \cos(\frac{2\pi k}{N}) y_k[n-1] - r^2 \sum_{k=1}^{\frac{N}{2}-1} y_k[n-2] \end{aligned} \tag{3.102}$$

Donde la salida final, y es la suma de las salidas de los filtros pasabandas y_k. De modo que, aun usandose como filtro no adaptivo, el filtro de transformada coseno discreta es muy util debido a que la magnitud y fase en un rango uniforme de frecuencias son usados como parámetros de diseño.

Es importante tomar en cuenta que las condiciones impuestas en $B(k)$ nos llevan a pasar de la transformada de Fourier discreta a la transformada coseno discreta, para ver como se lleva a cabo esto, primero veamos la ecuación de la DFT como:

$$b_m = \frac{1}{N}\sum_{k=0}^{\frac{N}{2}-1} B(k)e^{\frac{j\pi mk}{N}} + B(N-k)e^{-\frac{j\pi mk}{N}} \tag{3.103}$$

Empleamos las condiciones de simetría en $B(k)$ para obtener

$$b_m = \frac{1}{N}\sum_{k=0}^{\frac{N}{2}-1} |B(k)|\left[e^{\frac{j\pi mk}{N}+\phi(k)} + e^{-\frac{j\pi mk}{N}-\phi(k)}\right] \tag{3.104}$$

que puede ser reescrito como

$$b_m = \frac{2}{N}\sum_{k=0}^{\frac{N}{2}-1} |B(k)|\cos\left(\frac{j\pi mk}{N}+\phi(k)\right) \tag{3.105}$$

Esta ecuación el la expresión de la transformada coseno discreta.

Para una implementación adaptiva del filtro, los coeficientes son actualizados de acuerdo a la magnitud y fase de un rango uniforme de frecuencias. Para encontrar las ecuaciones de actualización LMS, tomamos la derivada parcial con respecto a $|B(k)|$ de la salida $y[n]$ para obtener

$$\alpha_k[n] = (w[n]\cos(\phi(k)) - r\cos(\frac{2\pi k}{N}-\phi(k))w[n-1]) + 2r\cos(\frac{2\pi k}{N}\alpha_k[n-1] - r^2\alpha_k[n-2]) \tag{3.106}$$

Donde

$$\frac{\partial y[n]}{\partial |B(k)|} = \alpha_k[n] \tag{3.107}$$

A continuación, tomamos la derivada parcial de $y[n]$ con respecto a $\phi(k)$ para obtener

$$\beta_k[n] = |B(k)|\big[-w[n]\sin(\phi(k)) - r\sin(\frac{2\pi k}{N}-\phi(k))w[n-1]\big] + 2r\cos(\frac{2\pi k}{N})\beta_k[n-1] - r^2\beta_k[n-2] \tag{3.108}$$

Donde

$$\frac{\partial y[n]}{\partial \phi(k)} = \frac{\partial y_k[n]}{\partial \phi(k)} = \beta_k[n] \tag{3.109}$$

El error resultante es $e[n] = d[n] - y[n]$ y las ecuaciones de actualización de los coeficientes se vuelven

$$|B_{n+1}(k)| = |B_n(k)| + 2\mu e[n]\alpha_k[n] \tag{3.110a}$$

$$\phi_{n+1}(k) = \phi_n(k) + 2\mu e[n]\beta_k[n] \tag{3.110b}$$

Capítulo 4

Implementación en Matlab

4.1. Consideraciones Iniciales

En este capitulo mostraremos el trabajo desarrollado en MATLAB y SIMULINK para simular y probar el desempeño de los algoritmos descritos en el capítulo anterior. Se utilizara el software *MATLAB R2007a* para desarrollar filtros de Cancelación Activa del Ruido usando los algoritmos **LMS** y **RLS**.

Para poder comparar de una manera más efectiva el desempeño de los algoritmos mencionados se requiere establecer parámetros fijos en la simulación del filtrado. Para ello se usara como fuente primaria un tono senoidal con variación lineal de frecuencia, con ecuación:

$$x[n] = \sin(2\pi t^2) \tag{4.1}$$

La cual se puede observar en la figura 4.1, este será la señal que lleve la información deseada; por otra parte se usaran los propios algoritmos de MATLAB para generar una señal aleatoria que funcione como ruido blanco, la cual será la fuente secundaria de nuestros filtros.

Las condiciones iniciales para todos los casos serán con inicio de cero; esto es tanto la señal de entrada, como el vector de coeficientes tendrán sus valores iniciales en cero.

Los algoritmos serán simulados con un orden de $L = 6$ para las simulaciones en MATLAB y con $L = 24$ para los algoritmos en SIMULINK, y un valor de $\mu = 0.1$ para el caso del algoritmo *LMS* y $\lambda = 1$ para el algoritmo *RLS*.

4.2. Algoritmo LMS

El código del algoritmo LMS en MATLAB es el siguiente cuadro:

```
1 %Simulación del algoritmo LMS para Control Activo del Ruido
```

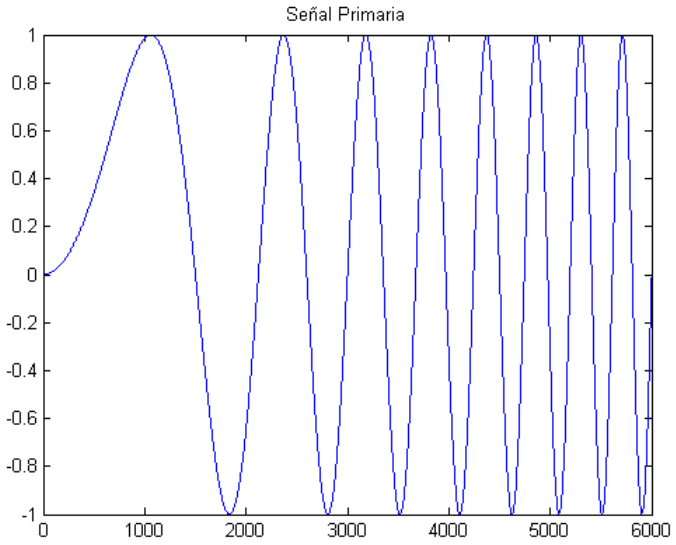


Figura 4.1: Gráfica de la señal primaria.

```
2  %
3  clear all;
4  close all;
5  clc;
6
7  %Establecemos el orden del sistema
8  refGain = 1;
9  worder = 6;
10 %La señal de referencia
11 N = 6000;
12 t=1:N;
13 signal = sin(2*pi.*t.*t/N/N*8);
14 wreal = randn(1,worder);
15 %Y el ruido a cancelar
16 anoise = randn(1,N);
17 ref = conv(anoise,wreal);
18 primary = signal + ref(1:length(signal));
19 fref = anoise*refGain;
20 w(1,:) = zeros(1,worder);
21 mu = .1;
22 %Condiciones iniciales = 0
23 frefpad = [zeros(1,worder -1) fref];
24 %se inicia el algoritmo
25 for n = 1:N;
26     %Movemos el contador para poder inciar en el orden correcto en caso de
27     %iniciar con cero
28     m = n + worder -1;
29     frefblock = frefpad(m-worder+1:1:m)';
30     refP(n) = w(n,:)*(frefblock);
```

```
31      output(n) = primary(n) - refP(n);
32      w(n+1,:) = w(n,:) + mu.*frefblock'.*output(n);
33  end;
34  %Se llama la función de graficación...
35  lms_graph
36
37  %Calculando el SNR
38  snr = 10*log10(sum(signal.^2) ./ sum(anoise.^2))
39  snr2 = 10*log10(sum(signal.^2) ./ sum(ref.^2))
```

Como podemos observar en el código mostrado, se siguió el método establecido en la sección 3.3. Esto es, primero se establece el orden del sistema y se generan las señales que serán usadas en el proceso, esto es la señal de prueba (4.1) y la señal de ruido; que en este caso es una señal de ruido blanco con un valor normalizado. Posteriormente se establecen las condiciones iniciales en cero para el vector de coeficientes y el valor de μ también es establecido (en 0.1 para este caso).

Una vez establecido lo anterior se procede a iniciar el algoritmo como se muestra en las ecuaciones (3.28), (3.29) y (3.30). El algoritmo termina en esta parte el código restante se usa para lograr las gráficas de resultado.

En la figura 4.2 se muestra la señal de entrada, la cual es la suma de la señal deseada más el ruido, la señal de salida o la señal filtrada y se gráfica el error cuadrático medio; todas graficadas con respecto al tiempo n

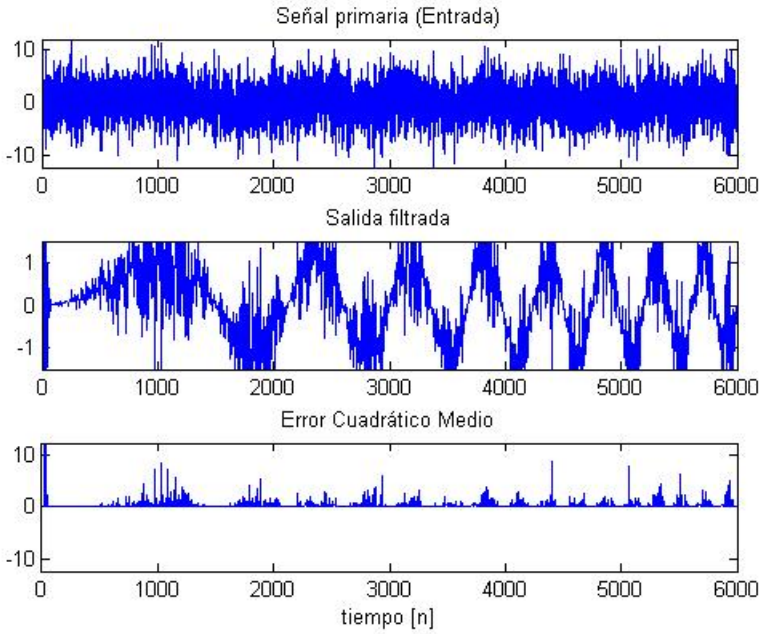


Figura 4.2: Resultado del algoritmo LMS (SNR=13.79 [db])

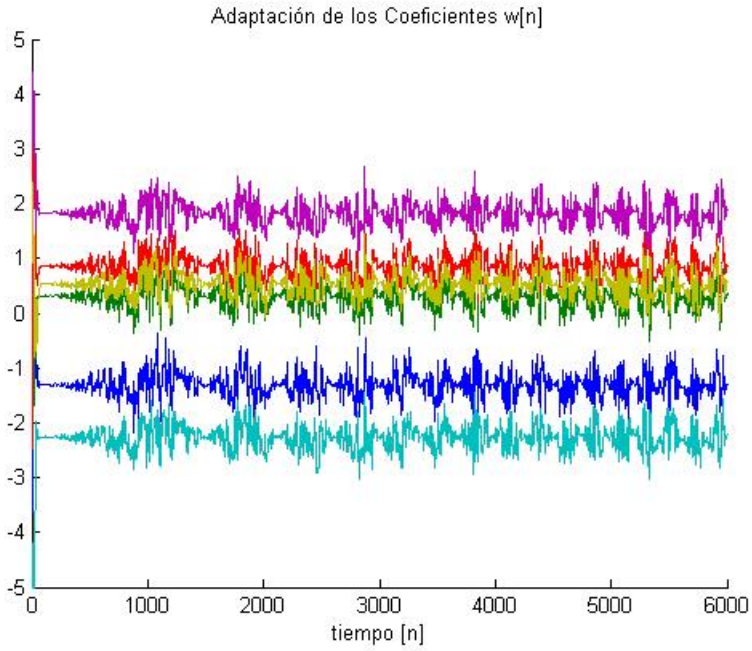


Figura 4.3: Proceso de adaptación de los coeficientes para el algoritmo LMS

Recordemos que la simulación se realizó con un sistema de orden $L = 6$, lo que significa que el sistema tendrá seis coeficientes, el valor de estos coeficientes es actualizado cada iteración con base al algoritmo LMS, la 'evolución' de estos seis coeficientes con respecto al tiempo n se muestra en la figura 4.3.

Haciendo una comparación de la señal de salida en la figura 4.2 y la señal original en la figura 4.1 observamos que el algoritmo LMS fue capaz de obtener la forma de onda de la señal original, sin embargo, esta todavía contiene mucho ruido lo cual puede ser observado facilmente en la gráfica del MSE.

4.2.1. Simulación en SIMULINK

Para la simulación del control activo del ruido con el algoritmo LMS usamos el modelo mostrado en la figura 4.4 el cual fue realizado en SIMULINK[19]. Observamos en el modelo que se uso el bloque "Chirp Generator" el cual genera una señal parecida a la usada en la prueba de anterior, esto es, una señal senoidal con variación lineal de frecuencia. Para este caso, la señal comienza con una frecuencia de $10Hz$ y alcanza un máximo de $30Hz$. Otro bloque que requiere de nuestra atención es el bloque del generador de ruido "Random Source" el cual se encarga de generar la señal contaminante, esta señal de ruido es pasada por un filtro paso bajas antes de ser sumada a la señal primaria; esto con el objetivo de asegurarnos que el contenido espectral de ruido se centre en las frecuencias bajas.

En la figura 4.5 se pueden observar las señales involucradas en el proceso, estas son (de

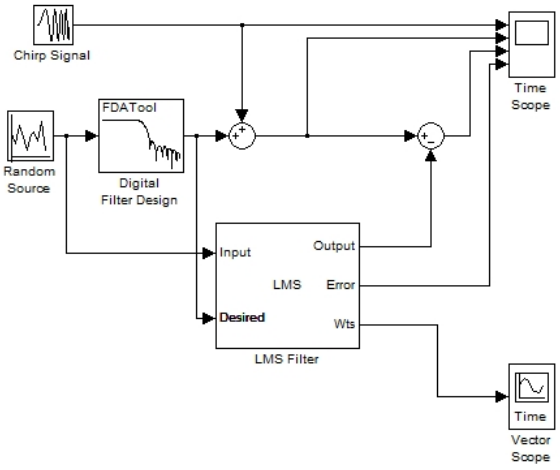


Figura 4.4: Modelo de un filtro con CAR y el algoritmo LMS en SIMULINK

arriba a abajo): la señal original, señal primaria (señal original más ruido), la señal de salida (filtrada) y la señal de error. Es importante notar que el algoritmo tiene un desempeño aceptable presentandose algunos errores cuando la señal original hace cambios de frecuencia.

Por último se graficó el vector de coeficientes $w[n]$ (figura 4.6 en el cual podemos observar el estado de los 24 coeficientes una vez que el sistema ya se estabilizó.

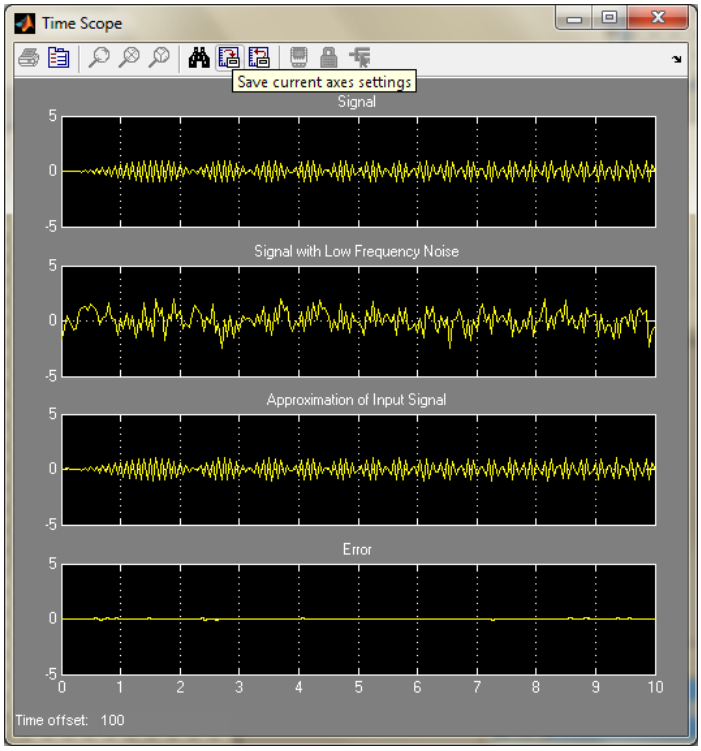


Figura 4.5: Gráficas de las señales de entrada (original y ruido) y salidas (señal filtrada y error) del modelo en SIMULINK

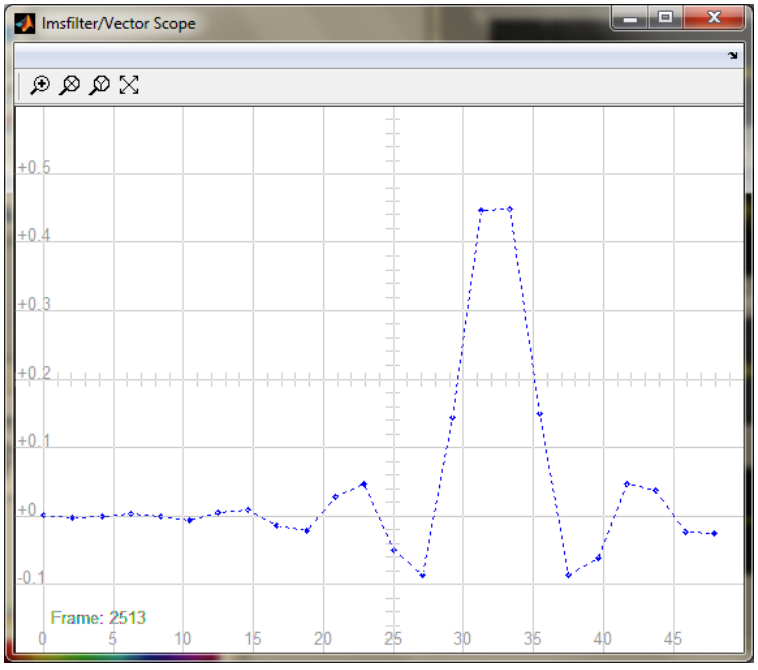


Figura 4.6: Vector de coeficientes del filtro con $L = 24$ coeficientes (Modelo en SIMULINK)

4.3. Algoritmo RLS

Para el código del algoritmo *RLS* se utilizó una estructura similar al *LMS*, como ya se mencionó al principio de este capítulo se utilizan los mismos parámetros; esto es, la misma señal original e igual orden del sistema ($L = 6$). El código se muestra a continuación.

```
%Simulación del algoritmo RLS para Control Activo del Ruido
%
clear all;
close all;
clc;

%Establecemos el orden del sistema
refGain = 1;
worder = 6;
%La señal de referencia
N = 6000;
t=1:N;
signal = sin(2*pi.*t.*t/N/N*8);
wreal = randn(1,worder);
%Y el ruido a cancelar
anoise = randn(1,N);
ref = conv(anoise,wreal);
primary = signal + ref(1:length(signal));
fref = anoise*refGain;
%El valor de lambda
lambda = 1;
%Condiciones iniciales = 0
w(1,:) = zeros(1,worder);
init = 100;
rinv = diag( ones(1,worder) * init );
frefpad = [zeros(1,worder -1) fref];
%se inicia el algoritmo
for n = 1:N;
    %Movemos el contador para poder inciar en el orden correcto en caso de
    %iniciar con cero
    m = n + worder -1;
    frefblock = frefpad(m-worder+1:1:m)';
    refP(n) = w(n,:)*(frefblock);
    output(n) = primary(n) - refP(n);
    k = lambda * rinv * frefblock / (1 + lambda*frefblock'*rinv*frefblock);
    w(n+1,:) = w(n,:) + k'*output(n);
    rinv = lambda*rinv - lambda*k*frefblock'*rinv;
end;
%Se llama la función de graficación...
rls_graph

%Calculando el SNR
snr = 10*log10(sum(signal.^2) ./ sum(anoise.^2))
snr2 = 10*log10(sum(signal.^2) ./ sum(ref.^2))
```

Podemos observar que este código comparte con el código del algoritmo LMS la inicialización de los parámetros de simulación, así como la nomenclatura de algunas señales. Esto con el objetivo de facilitar la comparación entre ambos códigos. El algoritmo para el cálculo del RLS fué tomado de las ecuaciones (3.69) a la (3.73).

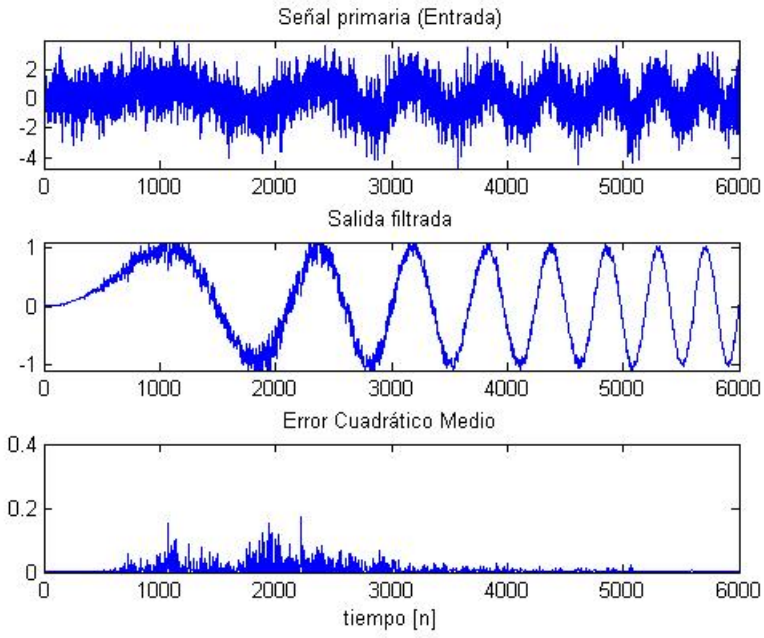


Figura 4.7: Resultado del algoritmo RLS (SNR=12.93 [db])

Una vez que se ejecutó el código observamos en la figura 4.7 los resultados. Aquí se grafican la señal de entrada (señal original más el ruido), la señal filtrada y el error cuadrático medio. Se nota inmediatamente la diferencia entre los dos algoritmos, al ofrecer el algoritmo *RLS* un mejor desempeño en el filtrado del ruido en la señal de salida, la cual a pesar de tener algo de ruido en ella es considerablemente menor que en el caso anterior (ver figura 4.2). Además de ofrecer un error cuadrático medio nulo.

Como en el algoritmo anterior la simulación es un sistema de orden $L = 6$, lo que significa que el sistema tendrá seis coeficientes, el valor de estos coeficientes es actualizado cada iteración con base al algoritmo RLS, la 'evolución' de estos seis coeficientes con respecto al tiempo n se muestra en la figura 4.8. Notamos que en este caso los coeficientes son más estables y convergen mucho más rápido que con el algoritmo LMS.

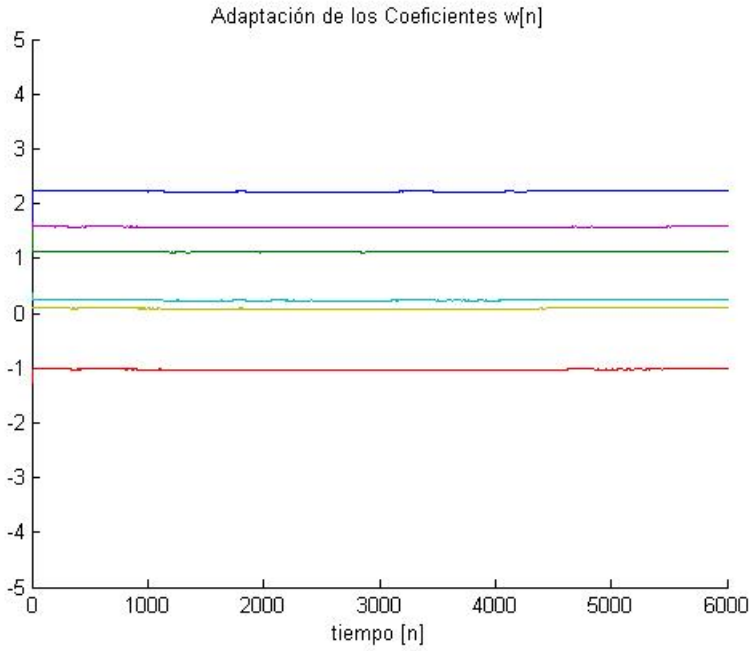


Figura 4.8: Proceso de adaptación de los coeficientes para el algoritmo RLS

4.3.1. Simulación en SIMULINK

Para la simulación del control activo del ruido con el algoritmo RLS usamos el modelo mostrado en la figura 4.9 el cual fue realizado en SIMULINK.

Al igual que con el código en MATLAB, la estructura de este modelo es similar al modelo para el algoritmo LMS, pudiendo destacar el uso de los bloques que generan la señal, el "Chirp Generator" que genera la señal senoidal con variación lineal de frecuencia. De esta manera la señal generada tiene una frecuencia inicial de $10Hz$ y alcanza una frecuencia máxima de $30Hz$. También conservamos el bloque del generador de ruido "Random Source" cuya señal es pasada por un filtro paso bajas antes de ser sumada a la señal primaria; esto con el objetivo de asegurarnos que el contenido espectral de ruido se centre en las frecuencias bajas.

En la figura 4.10 se pueden observar las señales involucradas en el proceso, estas son (de arriba a abajo): la señal original, señal primaria (señal original más ruido), la señal de salida (filtrada) y la señal de error. Notamos que a diferencia del modelo del algoritmo LMS, este modelo presenta mayor precisión al filtrar la señal primaria, con un error nulo, mayor velocidad de convergencia y mayor estabilidad en los coeficientes, los cuales pueden ser observados en la figura 4.10.

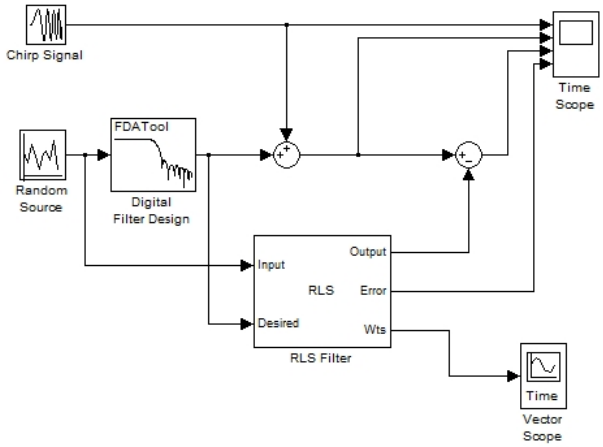


Figura 4.9: Modelo de un filtro con CAR y el algoritmo RLS en SIMULINK

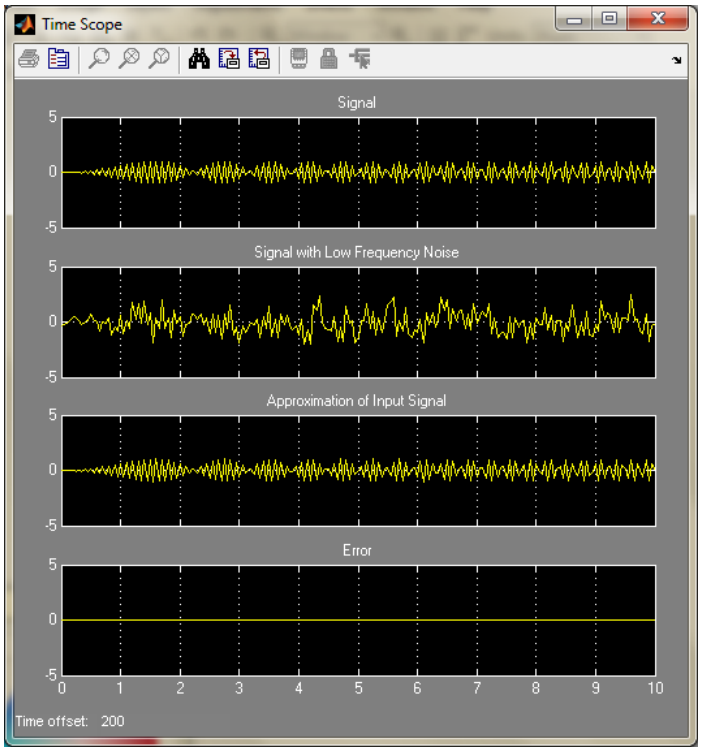


Figura 4.10: Gráficas de las señales de entrada (original y ruido) y salidas (señal filtrada y error) del modelo en SIMULINK

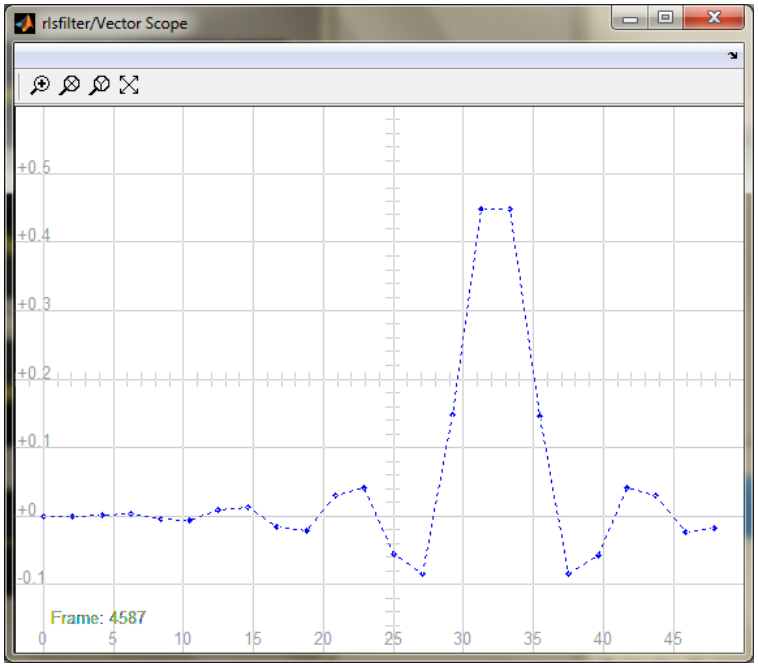


Figura 4.11: Vector de coeficientes del filtro con $L = 24$ coeficientes (Modelo en SIMULINK)

4.4. Conclusiones

De las simulaciones en MATLAB de los filtros adaptables con algoritmos LMS y RLS encontramos que se presentaron marcadas diferencias entre ambos, después de procesar la señal primaria encontramos que el algoritmo RLS se adapta mejor al ruido presente para lograr eliminar la mayor parte de este, esto es mucho más evidente al comparar las gráficas del error cuadrático medio en las figuras 4.2 y 4.7. Además la velocidad de adaptación del algoritmo RLS también son mucho mejores, comparando las figuras 4.3 y 4.8 podemos observar que el algoritmo tiene un mejor comportamiento en la adaptación de los coeficientes del filtro durante todo el proceso, al alcanzar los valores optimos ($w^o[n]$) mucho más rápido que el algortimo LMS y con menos variaciones durante todo el proceso.

Para el caso de los modelos en SIMULINK, con un orden de filtro mucho mayor ($L = 24$) al del caso en MATLAB (con $L = 6$), la diferencia de comportamiento estre ambos algoritmos es muy pequeña; al alcanzar el estado estable ambos algoritmos son capaces de minimizar el ruido de la señal primaria a niveles satisfactorios, presentando una ligera desventaja el algoritmo LMS el cual exhibe pequeños niveles de error cuando la señal original hace cambios de frecuencia lo cual es evidencia de la baja velocidad de adaptación característica del algoritmo.

En conclusión podemos inferir que el algoritmo LMS ofrece una opción simple para un proceso de adaptación, la calidad de la cancelación de las señales no deseadas es baja. De manera contraria, el algoritmo RLS ofrece mayor calidad en la atenuación de las señales no deseadas a cambio de mayor peso computacional en la adaptación de los coeficientes del filtro.

4.5. Control Activo del Ruido en el Dominio de la Frecuencia

Una vez que analizamos el comportamiento de los algoritmos LMS y RLS, simularemos el comportamiento de dos algoritmos que usan análisis en frecuencia para eliminar el ruido de la señal primaria, para estos casos se utilizó la transformada Discreta de Fourier y transformada Discreta de Cosenos para lograr el efecto deseado. Como en el caso del CAR en el dominio del tiempo, se realizaron 2 simulaciones para cada transformada, una utilizando código en MATLAB y la otra con un modelo en SIMULINK.

En la simulación de MATLAB, usamos la misma señal de entrada que en los experimentos de CAR en el dominio del tiempo; esto es, una señal *chirp* o tono senoidal con variación lineal en frecuencia, como es descrito en la ecuación 4.1 y que se muestra en la figura 4.1. Para las simulaciones en SIMULINK se utilizaron señales de audio previamente grabadas y digitalizadas, la señal primaría es una grabación de voz, mientras que la señal contaminante es la grabación de una trompeta, ambas señales en archivos WAV a 176 *kbps*.

Antes de realizar la simulación en SIMULINK, las señales de entrada son adaptadas para

poder ser usadas en el modelo; esto se realiza por medio de la instrucción *wavread*, la cual lee el contenido de un archivo WAV y coloca los valores en el espacio de trabajo de MATLAB. Las señales de entrada "voz.wav" y "trompeta.wav" fueron procesadas con un código especial el cual puede ser observado en el apéndice A.1.

Una vez que se tienen las señales de entrada preparadas, podemos proceder con la simulación en SIMULINK, los resultados pueden ser observados a continuación.

4.5.1. Transformada Coseno Discreta

Para la Transformada Coseno Discreta, el código utilizado de MATLAB es el siguiente:

```
clc; clear all;
D  = 16;                          % Retraso de las muestras
ntr= 1000;                        % Numero de iteraciones
s  = chirp(randn(1,ntr+D));       % Señal de entrada (chirp)
n  = 0.1*(randn(1,ntr+D));        % Señal de Ruido
r  = s+n;                         % Señal recibida
x  = r(1+D:ntr+D);                % Señal de entrada
d  = s(1:ntr);                    % Señal deseada
L  = 32;                          % Orden del filtro
mu = 0.01;                        % Coeficiente del filtro adaptable
ha = adaptfilt.tdafdct(L,mu);     % Cálculo del filtro
[y,e] = filter(ha,x,d);           % Se realiza el filtrado
%Graficando las señales de entrada y salida
figure
subplot(3,1,1)
plot(1:ntr,real(d))
title('Señal deseada')
subplot(3,1,2)
plot(1:ntr,real(y));
title('Señal de salida (recuperada)')
subplot(3,1,3)
plot(1:ntr,abs(e));
title('Error')
```

Al ejecutar el código anterior obtenemos como resultados la gráfica que se muestra en la figura 4.12.

Como podemos observar en la figura la gráfica de la parte superior muestra la señal de entrada al sistema, esto es la señal original más la señal ruidosa. En la gráfica de enmedio se observa la señal recuperada por el filtro adaptable notando que el filtro no recupera inmediatamente la señal, esto es más evidente al observar la ultima gráfica, marcada como la señal de error; en ella observamos como el error comienza con niveles muy elevados (casi 100 %) al inicio del proceso y sin embargo comienza a disminuir de manera pareja esta que el sistema se estabiliza.

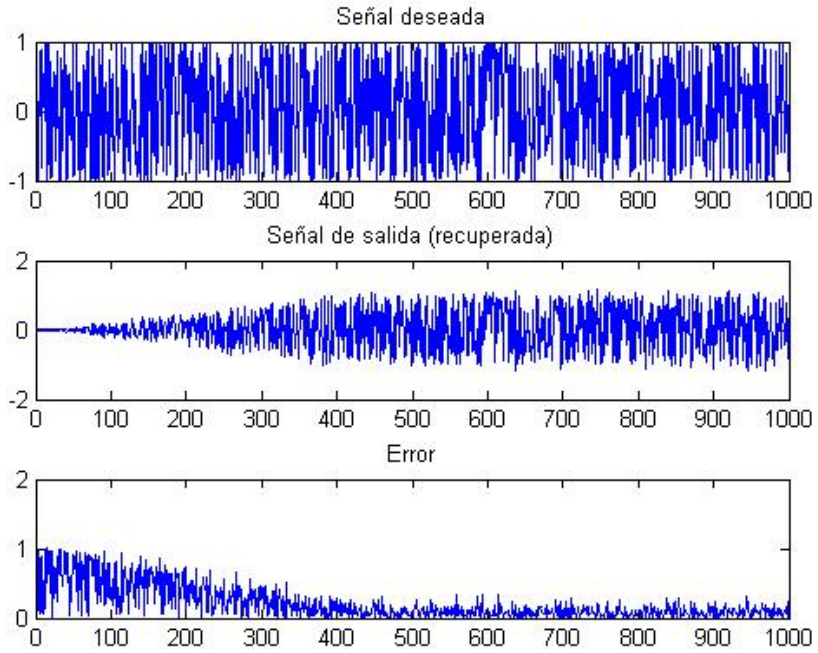


Figura 4.12: Gráficas de la Simulación en MATLAB

Una vez que observamos el comportamiento en MATLAB pasamos al modelo en SIMULINK utilizando la configuración mostrada en la figura 4.13. Las muestras de las señales primaria y de ruido son tomadas del espacio de trabajo de MATLAB y procesadas por simulink para formar a travez del operador suma la señal secundaria. Ambas señales son alimentadas a un buffer de 16 bits para posteriormente ser analizadas por el filtro por medio de la transformada coseno digital. A la salida de la señal de salida y de error se coloca un "Unbuffer" (de 16 bits) para poder realizar la visualización de las señales por medio de un osciloscopio.

En cada uno de los cuatro osciloscopios se grafica una señal diferente como se muestra en la figura 4.14 Podemos observar la señal original (fig.4.14a) seguida de la señal deseada (señal original más ruido) en la fig. 4.14b; para posteriormente observar la señal de salida y error (figs. 4.14c y 4.14d respectivamente).

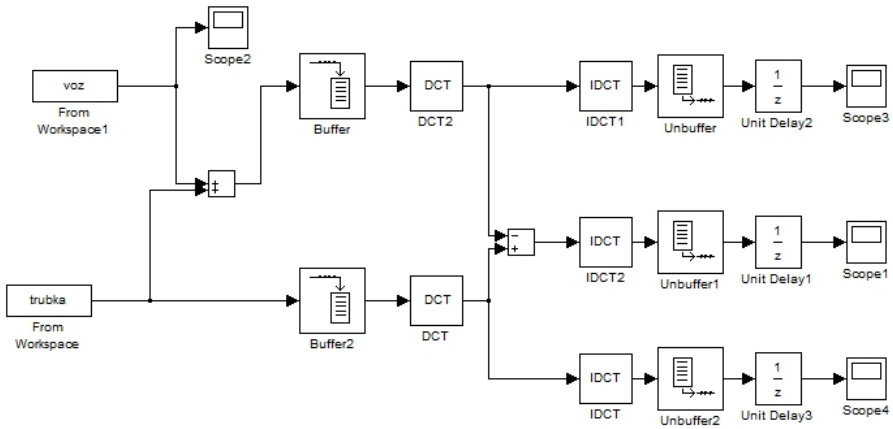
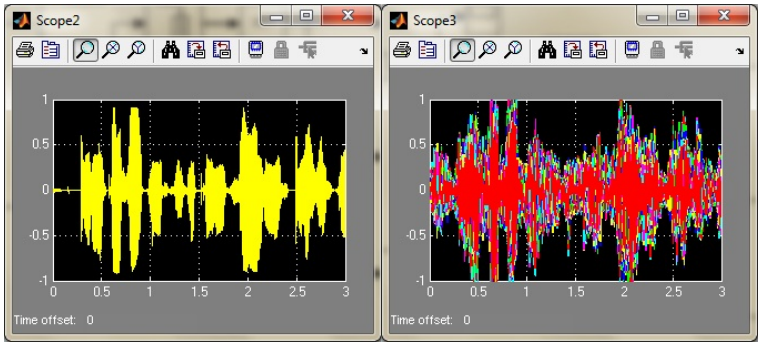
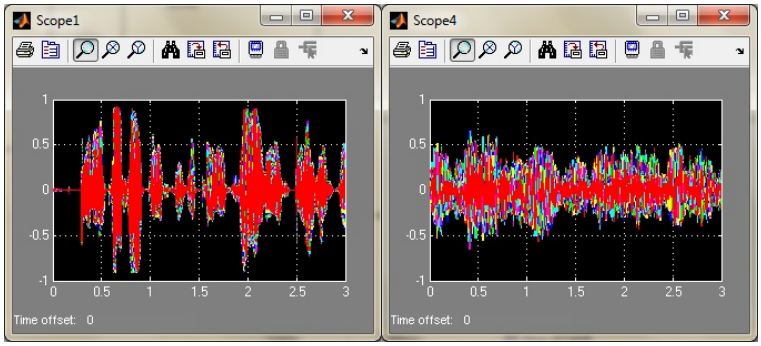


Figura 4.13: Modelo del filtro con análisis en frecuencia por medio de la Transformada Discreta de Cosenos.



(a) Señal Original **(b)** Señal Contaminada



(c) Señal de Salida **(d)** Señal de Error

Figura 4.14: Señales de salida del modelo anterior.

4.5.2. Transformada Discreta de Fourier

Posteriormente pasamos a la simulación usando la Transformada Discreta de Fourier utilizando e siguiente código en la simulación en MATLAB

```
clc; clear all;
D  = 16;                          % Retraso de las muestras
ntr= 1000;                        % Numero de iteraciones
s  = chirp(randn(1,ntr+D));       % Señal de entrada (chirp)
n  = 0.1*(randn(1,ntr+D));        % Señal de Ruido
r  = s+n;                         % Señal recibida
x  = r(1+D:ntr+D);                % Señal de entrada
d  = s(1:ntr);                    % Señal deseada
L  = 32;                          % Orden del filtro
mu = 0.01;                        % Coeficiente del filtro adaptable
ha = adaptfilt.tdafdft(L,mu);
[y,e] = filter(ha,x,d);
%Graficando las señales de entrada y salida
figure
subplot(3,1,1)
plot(1:ntr,real(d))
title('Señal deseada')
subplot(3,1,2)
plot(1:ntr,real(y));
title('Señal de salida (recuperada)')
subplot(3,1,3)
plot(1:ntr,abs(e));
title('Error')
```

Despues de ejecutar el código obtenemos los resultados en la forma de la gráfica 4.15

Al igual que en el caso de la Transformada Coseno Discreta en la primera gráfica encontramos la señal deseada (original más ruido), debajo de ella se grafica la señal de salida del filtro que corresponde a la señal recuperada, notamos que el filtro con Transformada Discreta de Fourier actúa más rápido que el filtro con Transformada Coseno Discreta este se aprecia mejor al comparar las señales de error que genera cada simulación donde observamos que el filtro de CAR con TDF alcanza mucho más rápido la estabilidad.

Para el modelo en SIMULINK utilizamos una configuración similar al caso anterior, utilizando como señales de entrada la información proveniente del espacio de trabajo de MATLAB, creando la señal deseada previamente a la acción de filtrado. La configuración usada se muestra en la figura 4.16.

En cada uno de los cuatro osciloscopios se grafica una señal diferente como se muestra en la figura 4.17

Podemos observar la señal original (fig.4.17a) seguida de la señal deseada (señal original más ruido) en la fig. 4.17b; para posteriormente observar la señal de salida y error (figs. 4.17c y 4.17d respectivamente).

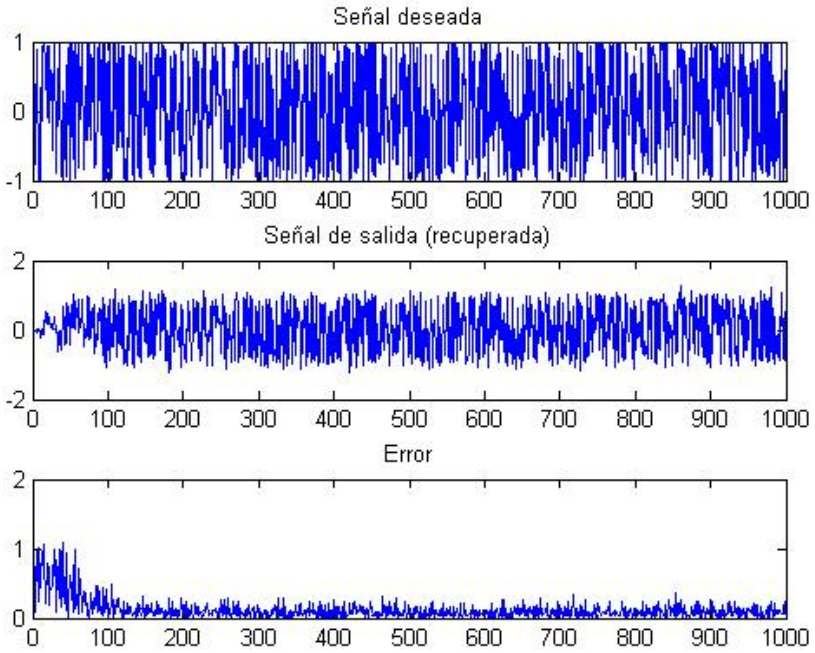


Figura 4.15: Gráficas de la Simulación en MATLAB

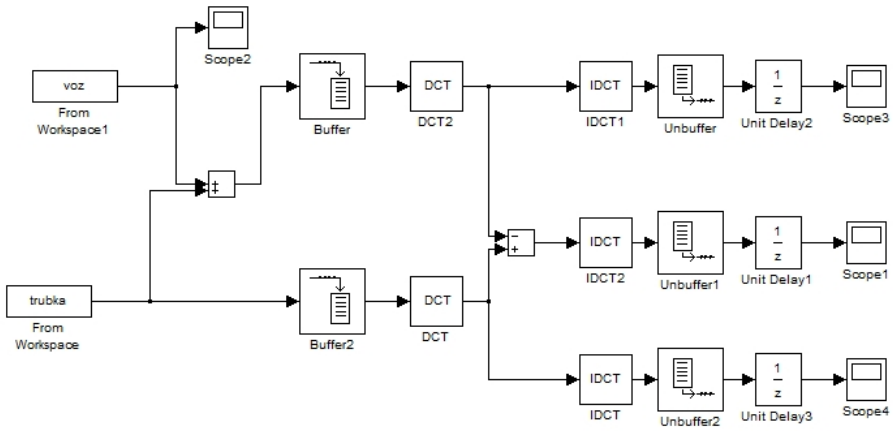
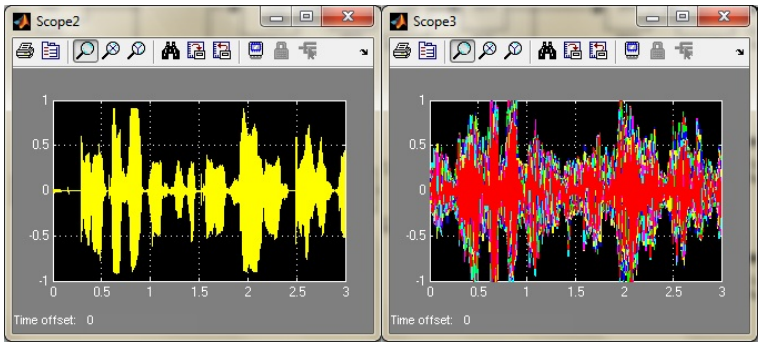
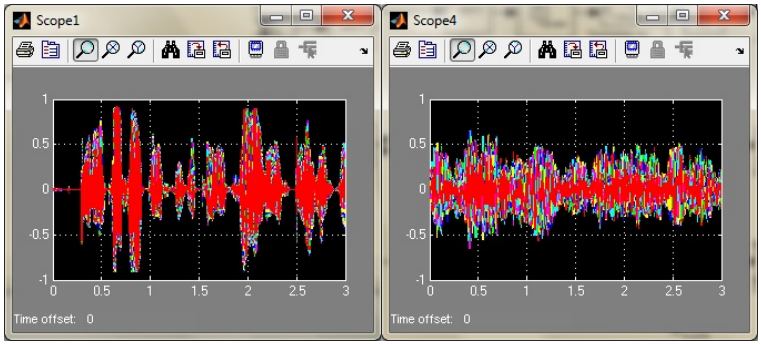


Figura 4.16: Modelo del filtro con análisis en frecuencia por medio de la Transformada Discreta de Fourier.



(a) Señal Original **(b)** Señal Contaminada



(c) Señal de Salida **(d)** Señal de Error

Figura 4.17: Señales de salida del modelo anterior.

Capítulo 5

Control Activo del Ruido en el DSP TMS320C25

5.1. Introducción

El microprocesador TMS320C25 es la segunda iteración de la segunda generación de la familia de Procesadores Digitales (DSP) TMS320 de *Texas Instruments*. Desarrollados con tecnología CMOS, son capaces de manejar operaciones con punto flotante de 32 bits y fabricado en un encapsulado tipo PGA (*Pin Grid Array*) de 68 pins.

Los procesadores digitales de señales son empleados para una amplia gama de aplicaciones, desde sistemas de comunicación y control hasta procesamiento de voz e imágenes. Los DSP's de propósito general son empleados principalmente en aplicaciones de comunicaciones (sistemas de comunicación celular). Los DSP's con aplicaciones específicas son los más usados en productos dirigidos al consumidor final; como lo son teléfonos celulares, fax/modems, radios, impresoras, auxiliares de audición, reproductores MP3, televisiones de alta definición (HDTV), cámaras digitales, etc. Estos procesadores se han vuelto de común elección debido a su menor costo y a que pueden manejar diferentes tareas con solo reprogramarlos.

Los procesadores DSP's son empleados principalmente en aplicaciones en tiempo real, en donde el procesamiento debe seguir el paso de los eventos externos, siendo en varias ocasiones señales analógicas.

La arquitectura del TMS320C25; que usa un reloj de 100 ns, una longitud de palabra de 32 bits para datos y programa y un bus de direcciones de 24 bits, le permite ejecutar instrucciones a una tasa de 12.5 MIPS[1].

Una característica importante de este DSP es la capacidad del procesador de realizar (ciertas) operaciones en paralelo, esto es, la multiplicación en paralelo y la carga al acumulador, todo esto en un solo ciclo. Además, el procesador posee un conjunto de registros para

[1]MIPS=Millones de instrucciones por segundo.

llevar el control de las operaciones y un registro de 64 palabras de longitud empleado por la memoria de programa. Están presentes también, 2 unidades aritméticas que trabajan en paralelo, cada una con un registro auxiliar asociado.

Los bloques de memoria interna son de acceso dual ya que pueden accesar dos operandos en un ciclo y existe un canal destinado para acceso directo a la memoria (DMA) que soporta operaciones de entrada/salida concurrentes, disminuyendo la carga de operaciones del CPU.

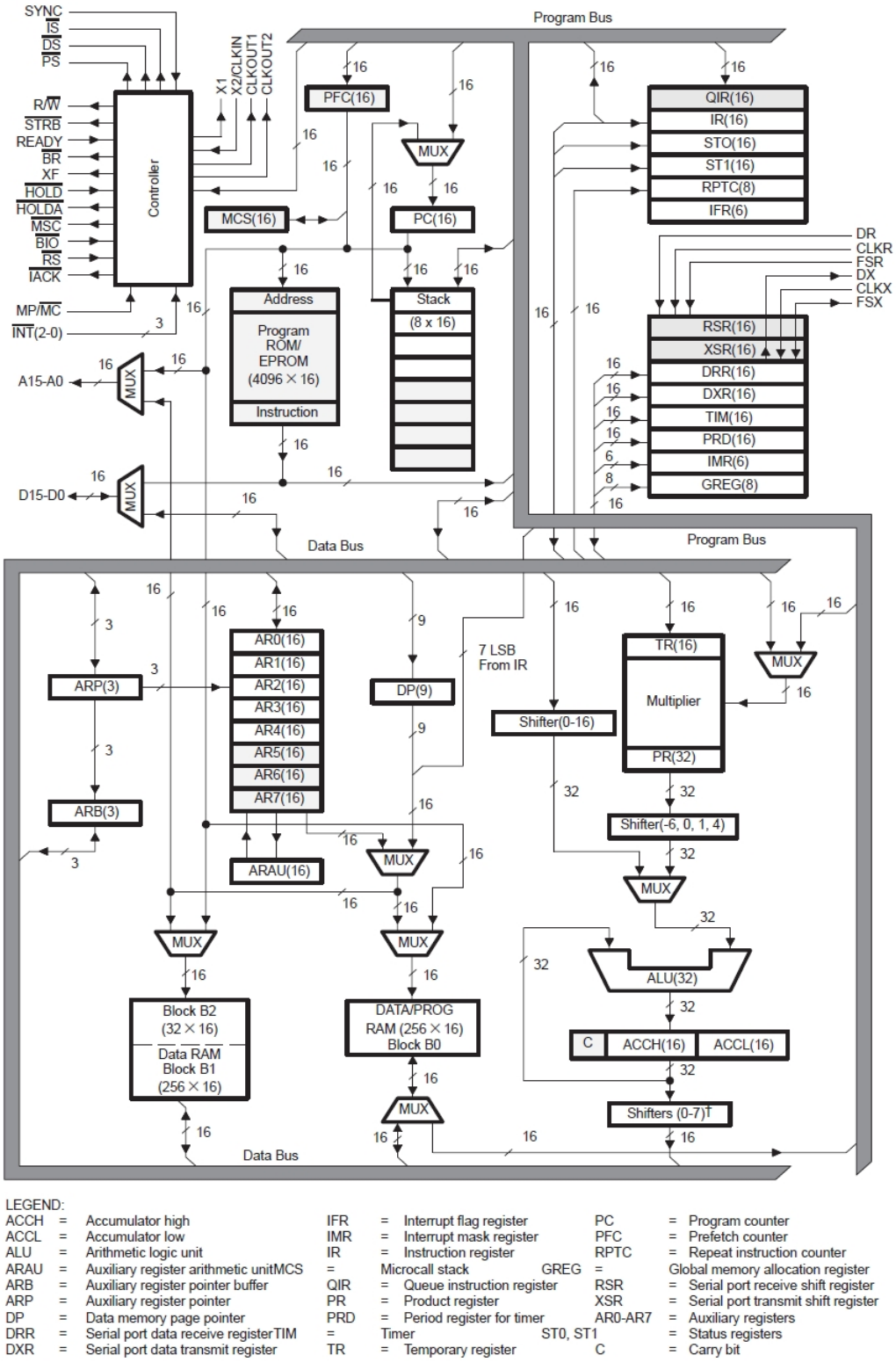


Figura 5.1: Arquitectura del DSP TMS320C25

5.2. Arquitectura del DSP TMS320C25

La familia del procesadores TMS320C2x utiliza una arquitectura Hardvard que le otorga flexibilidad y velocidad. En esta arquitectura, la transferencia entre espacios de programa y datos incrementa le flexibilidad del DSP. Esta modificación permite que los coeficientes sean almacenados en la memoria del programa para ser leídos por la memoria RAM, eliminando la necesidad de una memoria ROM dedicada a los coeficientes. Esto permite también que las instrucciones y subrutinas basadas en los valores calculados estén disponibles en cualquier momento.

La capacidad de los dispositivos TMS320C2x para obtener un mejor desempeño en aplicaciones de DSP se logra a través de instrucciones que multiplican/almacenan en un solo ciclo con opción de mover los datos, además de 8 registros auxiliares con una unidad aritmética dedicada y alta velocidad en la entrada/salida (I/O) de datos para procesamiento de señales con uso intensivo de captura de datos. La arquitectura del TMS320C25 esta optimizada para realizar funciones de filtrado en estructuras FIR[15].

5.2.1. Acumulador y ALU de 32 bits

El C25 cuenta con una Unidad de Aritmética Lógica (ALU) y acumulador que realiza un amplio rango de funciones lógicas y aritméticas en un solo ciclo de máquina. El ALU ejecuta una variedad de instrucciones las cuales proveen las siguientes capacidades:

- Dirigir hacia una dirección especificada por el acumulador
- Normalizar números de punto fijo contenidos en el acumulador
- Probar un bit específico de una palabra contenida en memoria

Una entrada al ALU siempre proviene del acumulador, mientras la otra entrada puede provenir del registro de producto (Product Register - PR) del multiplicador o la entrada de datos que han sido recogidos de la memoria RAM a través del bus de datos. Después de que el ALU ha realizado las operaciones lógicas o aritméticas, el resultado es guardado en el acumulador. El acumulador de 32 bits es divididos en dos segmentos de 16 bits para almacenamiento de datos.

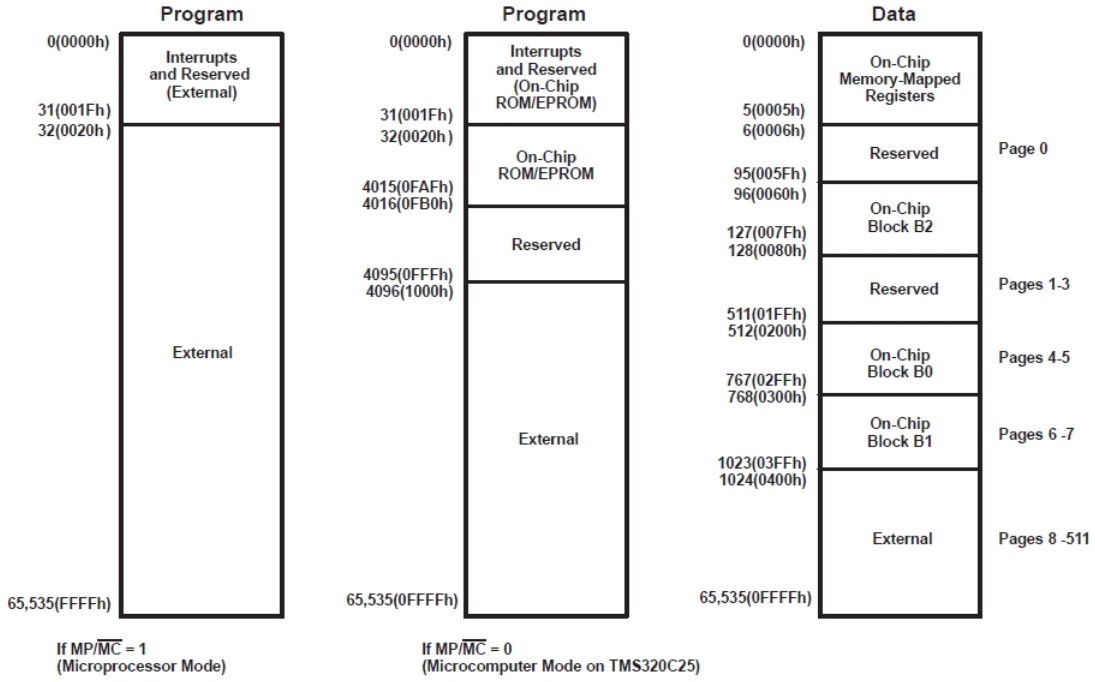
5.2.2. Multiplicador en paralelo de 16 $\times$ 16

El multiplicador es capaz de calcular productos de 32 bits (con o sin signo) en un solo ciclo de máquina. El multiplicador tiene los siguientes registros asociados:

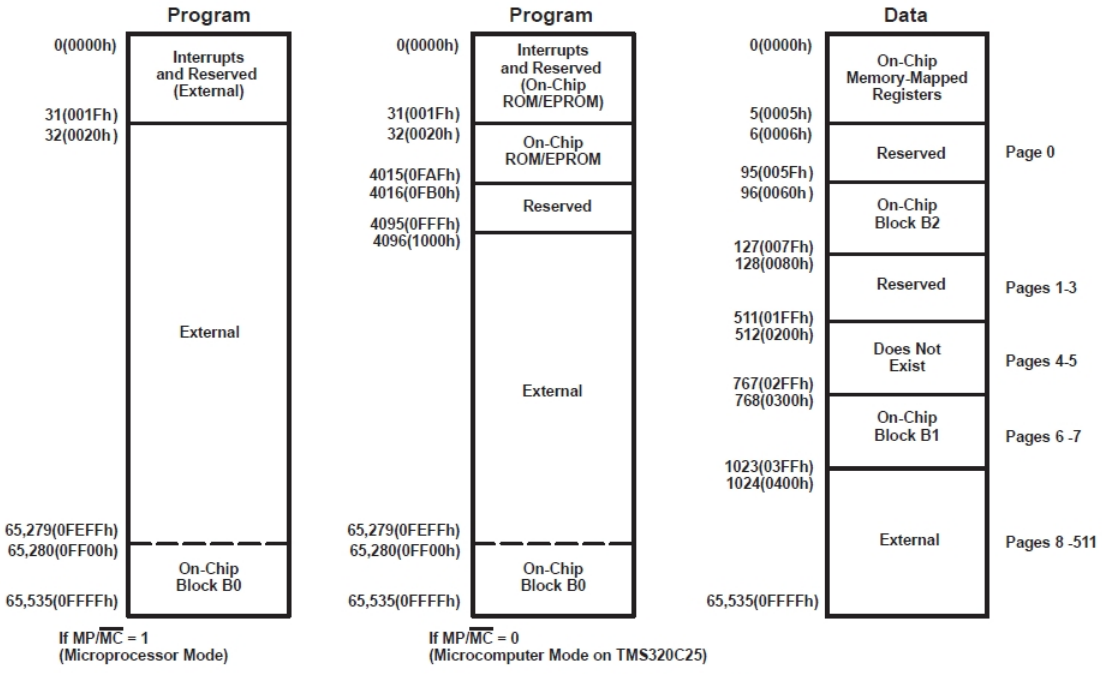
- Un registro temporal (Temporary Register - TR) de 16 bits, que almacena uno de los operandos del multiplicador.

- Un registro de producto (Product Register - PR) que almacena el resultado de la operación

Dentro del conjunto de operaciones incluido en el C25 se encuentran instrucciones de multiplicación/acumulación en un solo ciclo que permiten que ambos operandos sean procesados de manera simultanea. Los datos de estas operaciones pueden residir en cualquier lugar de la memoria interna o externa y pueden ser transferidos al multiplicador cada ciclo por medio de los *buses* de programa o datos.



(a) Memory Maps After a CNFD Instruction



(b) Memory Maps After a CNFP Instruction

Figura 5.2: Mapas de memoria para el TMS320C25.

5.2.3. Timer

El C25 provee de un temporizador de 16 bits asignado a la memoria para el control de operaciones. El registro del temporizador del chip (TIM) realiza un conteo regresivo que es continuamente actualizado por CLKOUT1 en el TMS320C25. Una señal de interrupción de temporizador (timer interrupt - TINT) es generada cada vez que el temporizador llega a cero. El timer se habilita con el valor indicado en el registro de periodo (period register - PRD) en el siguiente ciclo después de que alcanza el valor de cero de manera que las interrupciones pueden ser programadas para ocurrir a ciertos intervalos regulares cada $PRD + 1$ ciclos.

5.2.4. Memoria

El TMS320C25 provee de un total de 544 palabras de 16 bits de almacenamiento en RAM, divididos en tres bloques distintos (B0, B1, B2). De las 544 palabras, 288 de ellas (pertenecientes a los bloques B1 y B2) son asignadas a memoria de datos, las 256 palabras restantes (el bloque B0) pueden ser programadas como memoria de datos o programa. La memoria de datos de 544 palabras permite que el C25 maneje un arreglo de datos de 512 palabras (256 palabras si la memoria RAM del chip se utiliza para memoria de programa), dejando 32 localidades para almacenaje intermedio. Cuando se usa el bloque B0 como memoria de programa las instrucciones pueden ser descargadas desde una memoria externa a la memoria RAM interna y de ahí ser ejecutadas.

El TMS320C25 provee de tres direcciones separadas para memoria de programa, memoria de datos y datos de entrada/salida (I/O). La memoria interna esta asignada, ya sea dentro de los 64K palabras de memoria de datos o en el espacio de memoria de programa; dependiendo de la configuración de memoria, como se muestra en la figura 5.2. Las instrucciones CNFD (configurar bloque B0 como memoria de datos) y CNFP (configurar el bloque B0 como memoria de programa) permiten configuración dinámica de los mapas de memoria a través de software. Sin importar la configuración el usuario puede ejecutar programas desde la memoria externa de programa.

El TMS320C25 tiene seis registros que están asignados en el espacio de la memoria de datos: un registro de datos recibidos en el puerto serial, un registro de transmisión de datos en el puerto serial, registro temporizador, registro de periodo, registro de mascara de interrupción (interrupt mask register) y un registro de asignación de memoria global.

5.2.5. Interrupciones y Subrutinas

El TMS320C25 tiene tres interruptores (INT2-INT0) disponibles para que dispositivos externos interrumpan el procesador. Interrupciones internas son generadas por el puerto serial (RINT y XINT), el temporizador (TINT) y por instrucción de software (TRAP). Las interrupciones son priorizadas, *reset* (RS) tiene la más alta prioridad y el transmisor del puerto serial (XINT) la más baja prioridad. Un mecanismo de seguridad interno previene

que instrucciones multi-ciclo sean interrumpidas. Si una interrupción ocurre durante una instrucción multi-ciclo, la interrupción no es procesada hasta que la instrucción multi-ciclo es completada.

5.2.6. Interfaces Externas

El TMS320C25 soporta una amplia variedad de dispositivos externos para realizar comunicación con ellos. Un puerto serial *full-duplex* permite la comunicación con dispositivos externos como CODECS, convertidores A/D seriales y otros sistemas. Las señales de las interfaces son compatibles con codecs y otros dispositivos con un mínimo de hardware externo. Este puerto serial también puede ser usado para comunicación entre procesadores para aplicaciones multiprocesador.

El puerto serial tiene asignado dos registros en memoria: el registro de transmisión de datos (DXR) y el registro de recepción de datos (DDR). Ambos registros operan ya sea en modo de byte o modo de palabra de 16 bits, y pueden ser accesados de la misma manera que cualquier otra locación en la memoria de datos. Cada registro tiene un reloj externo, un pulso de sincronización de trama y sus registros asociados.

5.2.7. Conjunto de Instrucciones

Con un total de 131 instrucciones, el microprocesador TMS320C25 provee instrucciones tanto de propósito general como aquellas especialmente diseñadas para el cálculo intensivo.

Para un máximo desempeño, la siguiente instrucción es pre-cargada mientras la instrucción actual es ejecutada. Debido a que el mismo bus de datos es usado para comunicar datos/programa externo o espacio de entrada/salida, el número de ciclos puede variar dependiendo de donde sea pre-cargada la siguiente instrucción (de memoria de programa interna o externa). Un mejor desempeño se logra manteniendo los datos de programa en la memoria interna o en memoria externa de alta velocidad.

5.2.8. Modos de direccionamiento

El conjunto de instrucciones del TMS320C25 provee de tres modos de direccionamiento: directo, indirecto e inmediato.

Los modos de direccionamiento directo e indirecto pueden ser usados para accesar la memoria de datos. En modo directo, siete bits de la instrucción son concatenados con nueve bits del apuntador a la página de la memoria de datos para formar los 16 bits de la dirección de datos. En modo indirecto la memoria se accede a través de los registros auxiliares. En modo inmediato, los datos son basados en una porción de las palabras de instrucción.

En direccionamiento directo, la palabra de instrucción contiene los últimos siete bits de la

dirección de la memoria de datos. Este campo es concatenado con nueve bits del apuntador pagina de memoria de datos para formar la dirección completa de 16 bits. En este modo, la memoria contiene un total de 512 paginas, cada una conteniendo 128 palabras.

Existen ocho registros auxiliares (AR0-AR7) que hacen posible el direccionamiento indirecto. Para seleccionar un registro auxiliar en específico, el apuntador del registro auxiliar (Auxiliary Register Pointer - ARP) es cargado con un valor entre 0 y 7 para AR0 o AR7 respectivamente.

5.3. Implementación del algoritmo LMS

Aprovechando las ventajas que ofrecen el conjunto de instrucciones del TMS320C25, el desarrollo de un filtro de Control Activo del Ruido con algoritmo LMS es simplificado usando funciones tales como MPYA, MPYS o ZARL. Notamos en la ecuación (3.27) del algoritmo LMS, que el factor $2\mu e[k]$ es un valor constante que puede ser calculado una sola vez y almacenado en una variable en el registro T. De este modo, el cálculo se vuelve una instrucción de multiplicación/acumulación y redondeo. En este caso el bloque B1 contendrá el vector de datos y el bloque B0 los coeficientes del filtro. El código, en el archivo **adap.asm**, se muestra a continuación:

```
1  **********************************
2  * Filtro adaptivo, algoritmo LMS  *
3  *    Junio de 1996                *
4  **********************************
5  *
6           AORG >0000
7  RESET    B    INIT
8  *
9           AORG >0020
10 *
11 * INICIALIZACION DEL MICROCONTROLADOR
12 *
13 ONE      EQU  >0060
14 BETA     EQU  >0061
15 ERR      EQU  >0062
16 ERRF     EQU  >0063
17 YN       EQU  >0064
18 XN       EQU  >0065
19 DN       EQU  >0066
20 ORDER    EQU  10
21 *
22 FRSTAP   EQU  >0300
23 LASTAP   EQU  768+ORDER
24 *
25 COEFFP   EQU  >FF00
26 COEFFD   EQU  >0200
```

```
*
*************************
* Inicializacion        *
*************************
INIT    SOVM              ; Trabaja en saturacion
        LDPK 0            ; Trabaja con bandera zero
        LACK 1            ; Carga variable ONE con un bit para redondeo
        SACL ONE          ; a la variable ONE
        ZAC               ; A CERO el registro ACC y las variables:
        SACL YN           ; YN = 0
        SACL BETA         ; BETA = 0
        SACL ERR          ; ERR = 0
        SACL ERRF         ; ERRF = 0
        SACL DN           ; DN = 0
        LARP AR4          ; Habilita registro AR4
        LRLK AR4,>61      ; La direccion  >61 de B2 a la AR4
        BLKP MU2,*        ; 4CCC=0.5833 a variable MU2
        LRLK AR4,LASTAP   ; 768+10=778 al registro AR4
        RPTK ORDER-1      ; Se repite la instrucción 10 veces
        IN   *-,PA1       ; <PA1>--> 778,777,...,768 Memoria B1
        RPTK ORDER-2      ; Se repite la que sigue 9 veces
        IN   DN,PA2       ; <PA2>--> DN
        IN   DN,PA2       ; <PA2>--> DN
        LRLK AR4,512+ORDER ; 512+10=522--> AR4
        RPTK ORDER-1      ; Se repite la instruccion 10 veces
        SACL *-           ; Limpia bloque B0
*************************
* Realiza filtro FIR    *
*************************
CICLO   CNFP              ; B0 como Memoria de programa
        MPYK 0            ; 0-->PR
        LAC  ONE,14       ; En ACC se carga el bit de redondeo
        LARP AR3          ; Se habilita registro auxiliar AR3
        LRLK AR3,LASTAP   ; 778--> AR3
FIR     RPTK ORDER-1      ; La instruccion que sigue se repite 10 veces
        MACD COEFFP,*-    ;
        CNFD              ; B0 como la memoria de datos
        APAC              ; Se suma <PR>+<ACC>-->ACC
        SACH YN,1         ; El resultado --> YN: salida del filtro
        OUT  YN,PA5
        NEG               ; <-ACC>-->ACC
        ADD  DN,15        ; <DN-ACC>-->ACC
        SACH ERR,1        ; <ACC>-->ERR
        OUT  ERR,PA3      ; <ERR>--> PA3  El error al registro PA3
*************************
*  ALGORITMO LMS        *
*************************
        LT   ERR          ; <ERR>--> TR
        MPY  BETA         ; <ERR*BETA> --> PR
        PAC               ; <ERR*BETA> --> ACC
        ADD  ONE,14       ; REDONDEA EL RESULTADO
        SACH ERRF,1       ; <ERROR(i)*BETA> --> ERRF(i)
```

```
79
80          MAR  *+              ;
81          IN   XN,PA1          ; <PA1>—>XN Actualiza el vector X1(n+1)
82          LAC  XN              ; <XN>—>ACC
83          SACL *               ; <ACC>—>AR4
84
85          LARK AR1,ORDER-1     ; 9-->AR1
86          LRLK AR2,COEFFD      ; <0200>-->AR2
87          LRLK AR3,LASTAP+1    ; LASTAP+1= contiene el vector X1(n)
88                               ; 769-->AR3
89          LT   ERRF            ; <ERRF>—>TR
90          MPY  *-,AR2          ;
91          SPM  1               ; 1--> al registro PM
92
93  ******************************
94  * Actualiza los coeficientes *
95  *   W(n+1)=W(n)+2Mue(n)X1(n) *
96  ******************************
97
98  ADAPT   ZALR *,AR3           ; <W19>—>ACCH, 0-->ACCL AR3 Activo
99          MPYA *-,AR2          ; <X19>*<ERRF>—>PR, <W18>, AR2 Activo
100         SACH *+,0,AR1        ; <ACCH>—>AR2, <X18>
101         BANZ ADAPT,*-,AR2    ; Salto a ADAPT si Wn no es cero
102
103         CALL NUEVO           ; Se llama subrutina NUEVO
104         B    CICLO           ; Salto a etiqueta CICLO
105 NUEVO   IN   DN,PA2          ; Nueva muestra <PA2>—>DN
106         SPM  0               ; 0-->PM sin corrimiento
107         RET                  ; Regresar a CICLO
108
109 ***************************
110 *    Valor de 2*Mu        *
111 ***************************
112 MU2     DATA >4CCC           ; La constante MU2
113 *
```

En la parte de inicialización de variables se observa que el vector de coeficientes (bloque B0) se inicializa con ceros y el vector de datos (bloque B1) se inicializa con $\{x[1], x[2], \cdots, x[L]\}$, repitiendo L veces la instrucción IN con L el orden del filtro ($L = 20$ en este caso).

Enseguida se realiza el filtrado FIR, el valor de la salida del filtro se resta a la señal de referencia; la cual se obtiene del puerto PA2, para así generar la señal de error y enviarla al puerto de salida PA3. La constante BETA contiene el valor de 2μ de manera que al multiplicar por el error se obtiene $2\mu e[k]$ y se procede a almacenar esta contante en el registro T.

Finalmente, se actualizan los coeficientes del filtro conforme la ecuación (3.27), donde el último elemento se calcula como.

$$w[n+1] = w[n] + BETA * x[n] \tag{5.1}$$

La instrucción MPYA realiza la multiplicación y almacena el resultado en el registro P, además de que en la misma instrucción almacena el producto previo $i-1$.

En general, ruido por redondeo ocurre después de cada multiplicación. Sin embargo, el TMS320C25 tiene un multiplicador de 16×16 y un acumulador de 32 bits, por lo que no existe redondeo durante la suma de los productos. Todas la multiplicaciones se presentan con precisión completa y el redondeo ocurre después de la multiplicación. Como los coeficientes del filtro adaptivo son almacenados en la memoria RAM, el orden máximo del filtro transversal en esta implementación puede ser 256.

5.4. Consideraciones de implementación

Las estructuras y algoritmos descritos en capítulos anteriores fueron derivados usando como base la aritmética de precisión infinita. Al momento de implementar estas estructuras y algoritmos en un DSP que maneja números enteros de punto fijos, existe una limitación en la precisión de estos filtros, debido al hecho de que el DSP trabaja con un número finito de bits. Por lo tanto, el diseño debe de ser cuidadoso para reducir los efectos de la longitud de palabra finita. En general estos efectos son: cuantización de las señales de entrada, redondeo en las operaciones aritméticas, limitaciones del rango dinámico y cuantización de los coeficientes del filtro. Estos efectos pueden causar desviaciones del diseño original o causar ruido a la salida del filtro.

5.4.1. Limitaciones en el rango dinámico

Como se mencionó en la sección 3.3, el filtro transversal más usado junto con el algoritmo LMS se especifíca con las siguientes equaciones:

$$y[n] = \sum_{i=0}^{N-1} w_i[n]x[n-1] \tag{5.2}$$

y

$$w_i[n] = w_i[n] + u * e[n] * x[n-1] \quad , \; para\, i = 0, 1, \cdots, N-1 \tag{5.3}$$

Donde $x[n-1]$ es la secuencia de entrada y $w_i[n]$ los coeficientes del filtro.

Si la secuencia de entrada y los coeficientes del filtro están apropiadamente normalizados de modo que sus valores estén entre 1 y -1 usando el formato Q15, no aparece error en la suma. Sin embargo, la suma de dos números puede ser mayor a 1; esto se le conoce como *overflow*. El TMS320C25 provee de 4 maneras de manejar esta situación.

- Modo "Overflow" (aritmética de saturación)
- Ramificaciones de las condiciones de sobresaturación.

- Desplazamiento a la derecha del Registro de Producto
- Desplazamiento a la derecha del acumulador

Una técnica para evitar la probabilidad de error es el escalamiento, esto es, no permitir que la magnitud de los nodos del filtro adaptivo sobrepasen |1|. La condición es:

$$x_{max} < 1/\sum_{i=0}^{N-1} |w_i[n]| \tag{5.4}$$

Donde x_{max} denota el valor máximo absoluto de la entrada. Usando desplazamiento a la derecha, el C25 puede implementar el escalamiento para evitar sobre-saturación en la ecuación (5.2). Al colocar los bits de PM en el registro de estado ST1 a 11 usando la instrucción SPM o LST1, la salida de registro P es desplazada 6 bits a la derecha. Esto permite hasta 128 acumulaciones sin la posibilidad de sobre-saturación. La instrucción SFR también permite desplazar un bit a la derecha cuando existe la posibilidad de sobre-saturación.

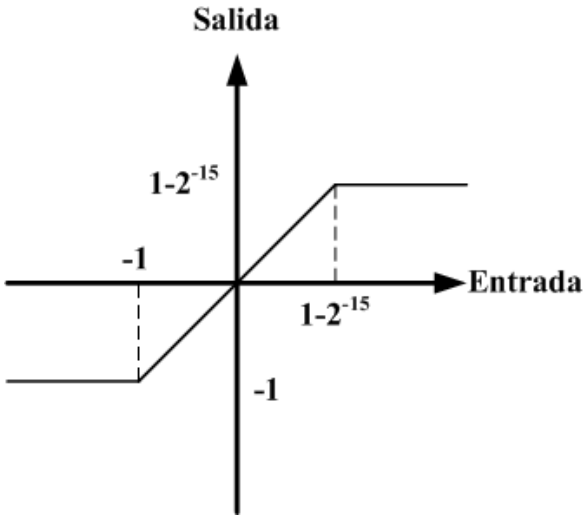


Figura 5.3: Aritmética de Saturación

Otra forma efectiva de prevenir overflow en la ecuación (5.2) es usar aritmética de saturación. Como se ilustra en la figura 5.3, si el resultado de una suma resulta en sobre-saturación, la salida se reduce al valor máximo. Si la aritmética de saturación se usa, los valores de la entrada serán mayores que el límite superior dado en la ecuación (5.4). La aritmética de saturación es controlada en el C25 por medio del bit OVM en el registro de estado ST0 y puede ser controlado con las instrucciones SOVM (set overflow mode), ROVM (reset overflow mode) y LST (load status register).

5.4.2. Errores de precisión finita

El TMS320C25 tiene un procesador de punto fijo de 16 bits. Cada muestra de datos es representada por un número fraccional que usa 15 bits de magnitud y un bit de signo. El intervalo de cuantización:

$$\delta = 2^{-b} \tag{5.5}$$

Con $b = 15$, donde los números son cuantizados en incrementos de δ. Los productos de

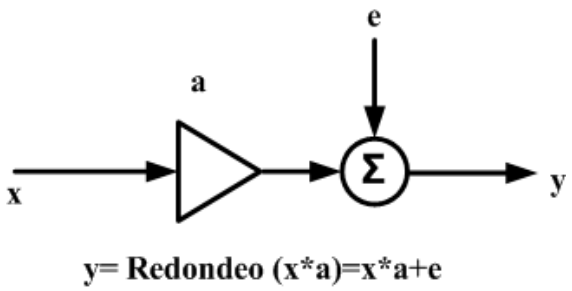


Figura 5.4: Modelo de Ruido por Redondeo de Punto Fijo

las multiplicaciones de datos por los coeficientes dentro del filtro deben de ser redondeados o truncados para poder ser almacenados en la memoria o los registros de CPU. Como se muestra en la figura 5.4, el error de redondeo puede ser modelado como ruido añadido al filtro por cada operación de redondeo. Este ruido blanco tiene una distribución uniforme sobre el intervalo de cuantización y para el redondeo

$$-1/2\delta < e \leq 1/2\delta \tag{5.6}$$

y

$$\delta_e^2 = (1/12)\delta^2 \tag{5.7}$$

Donde δ_e^2 es la variancia del ruido blanco.

En general el ruido de redondeo ocurre despues de cada multiplicación. Sin embargo, el TMS320C25 tiene un acumulador de precisión completa, esto es, un multiplicador de 16×16 con un acumulador de 32 bits, por lo que no hay redondeo cuando se implementa una serie de multilpicaciones y sumas como en la ecuación (5.2). El redondeo se aplica cuando el resultado es almacenado en memoria $y[n]$, de modo que solo una fuente de ruido este presente en cualquier operación.

5.5. Simulación

Para probar el código desarrollado necesitamos usar el simulador *SIM25.EXE*, pero para poder hacerlo debemos de compilar primero el programa. Para esto se usa el programa *XAS25.EXE* el cual, al ser ejecutado, genera dos archivos: **adap.lst** y **adap.mpo**, que son

archivos auxiliares conteniendo las enlaces a las librerías necesarias y el el programa compilado en lenguaje de máquina.

Para probar el funcionamiento del programa desarrollado, el TMS320C25 deberá de filtrar un tono senoidal contaminado por otro tono de diferente frecuencia. El primer tono o señal primaria es una señal senoidal de 200 Hz de frecuencia; el segundo tono o señal contaminante, será una señal cosenoidal con 60 Hz de frecuencia.

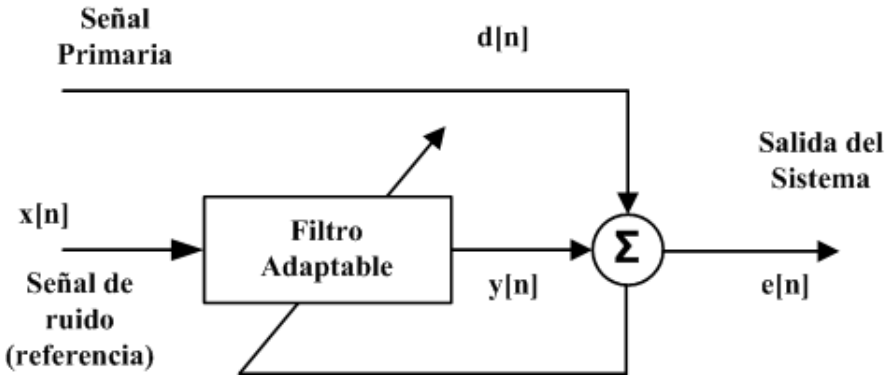


Figura 5.5: Esquema de cancelación de ruido usado en el Programa.

Para lograr el objetivo, usamos el esquema básico de cancelación del ruido (mostrado en la figura 5.5) monocanal. Donde se usan tienen las dos señales de entrada. La señal senoidal de 200 Hz ($d[n]$) es la señal primaria o de datos, la cual está contaminada con la señal de ruido, misma señal que es usada como señal de referencia. Entonces tenemos la señal de datos:

$$d[n] = s[n] + A_o \cos(\omega_o n + \phi_o) \tag{5.8}$$

Y la señal de referencia

$$x[n] = A \cos(\omega_0) \tag{5.9}$$

Una representación gráfica de ambas señales puede ser observada en las figuras 5.6 y 5.7. Después de ejecutar y dejar el filtro adaptivo por 5000 iteraciones en el SIM25 obtenemos de salida las señales mostradas en las figuras 5.8 y 5.9.

A la salida del filtro adaptivo con algoritmo LMS obtenemos la señal recuperada, observamos que contiene algunos errores, sobre todo al principio de la secuencia, esto es evidencia del filtro adaptando sus coeficientes para poder eliminar la señal no deseada.

También a la salida del filtro adaptivo obtenemos la señal de error del proceso, observamos que el error disminuye conforme el tiempo de procesamiento aumenta, de acuerdo con lo ya establecido en la teoría del algoritmo LMS.

La estructura y algoritmo usados probaron ser adecuadas para recuperar la señal de información eliminando la señal contaminante. El TMS320C25 tiene la ventaja de poder ejecutar en tiempo real el código y filtrar las señales de entrada de manera inmediata para poder recuperar la señal de información.

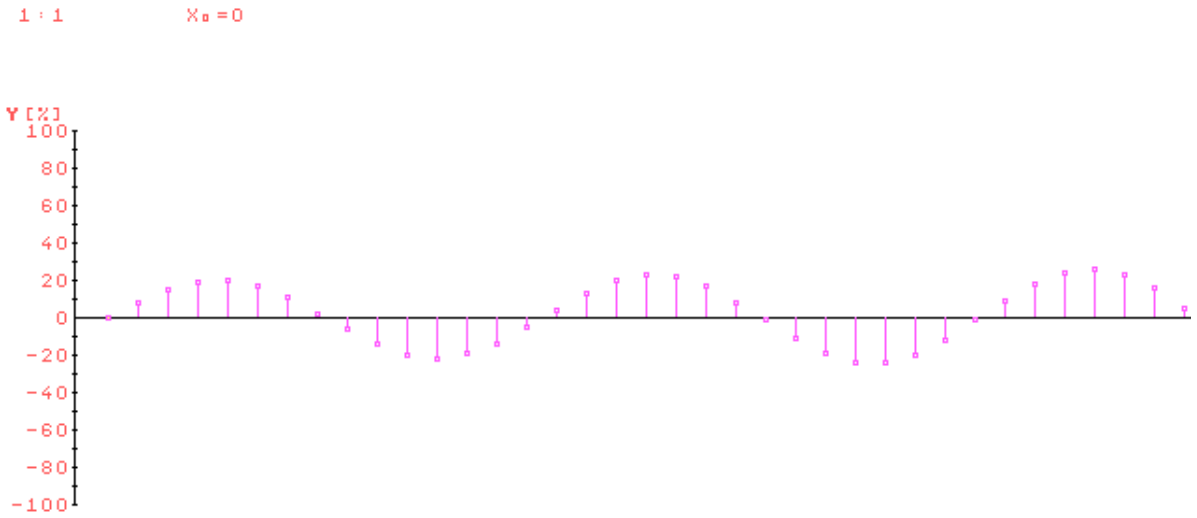


Figura 5.6: Señal de primaria.

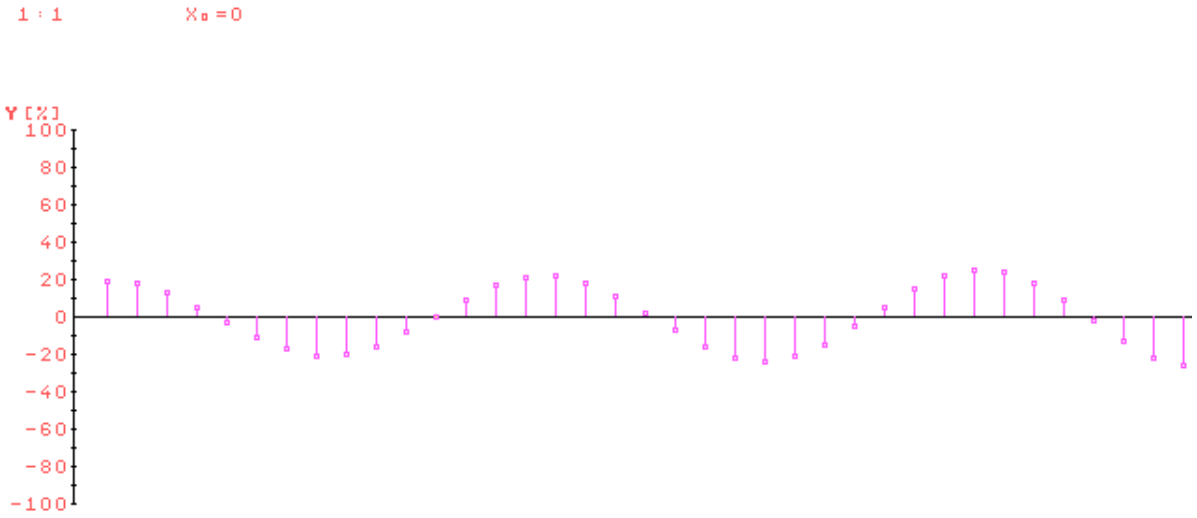


Figura 5.7: Señal de ruido.

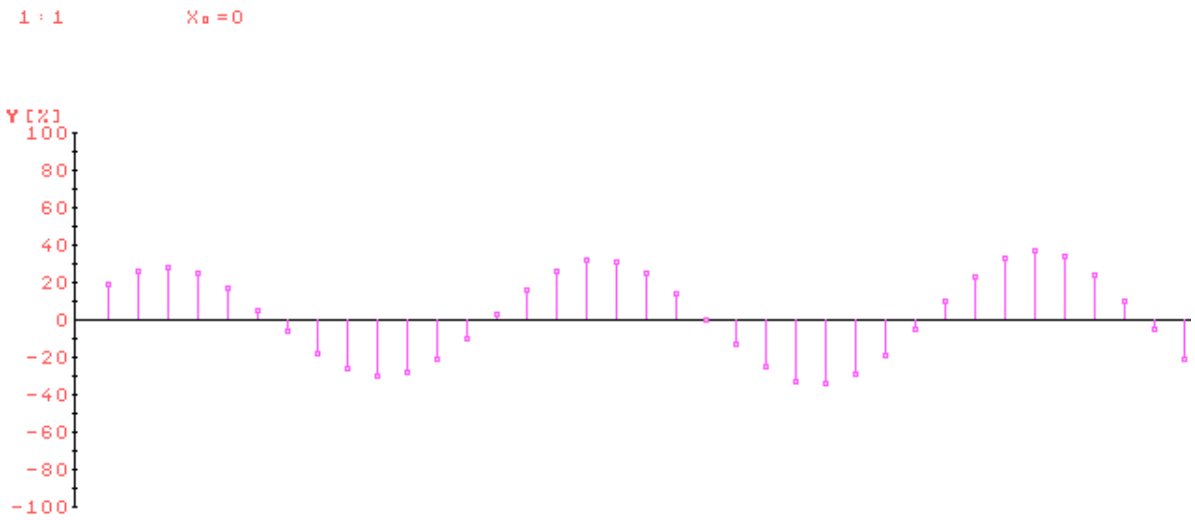


Figura 5.8: Señal de salida (señal recuperada).

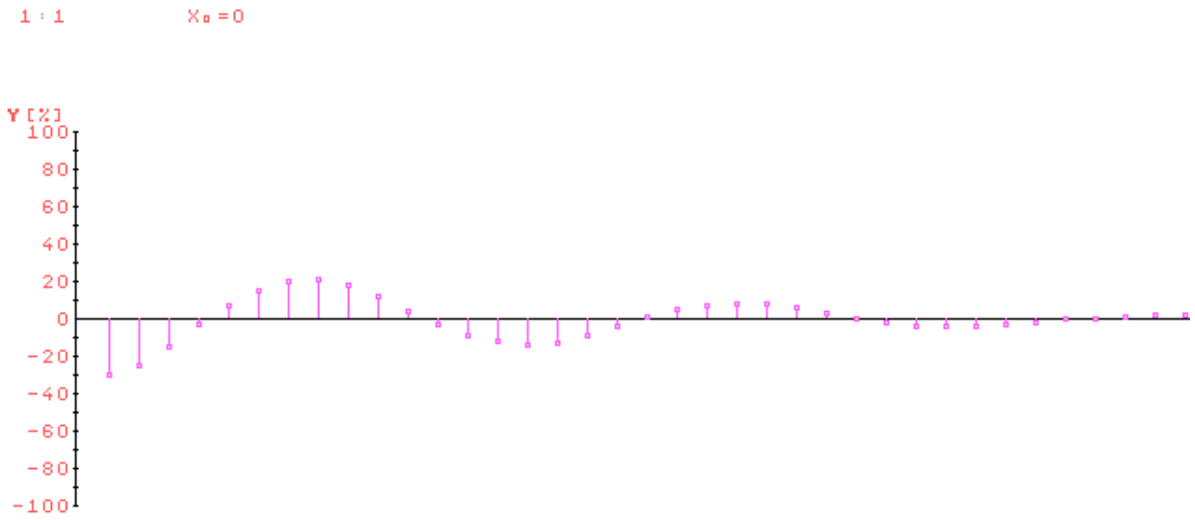


Figura 5.9: Señal de error.

Capítulo 6

Conclusión

El control activo del ruido es un campo muy complejo que se ha beneficiado mucho gracias a recientes avances tecnológicos (en software y hardware) y matemáticos; con nuevos y más completos algoritmos que permiten un mejor desempeño de los filtros adaptables.

En este proyecto se implementaron 4 algoritmos diferentes de control activo del ruido, haciendo simulaciones en MATLAB y SIMULINK, y uno en el DSP TMS320C25. Se analizaron las características de cada uno de los algoritmos y se mostraron los resultados después de realizar la cancelación.

El primer algoritmo implementado es el LMS, el algoritmo más sencillo de todos los estudiados en esta tesis. Usando una estructura básica en configuración de identificación de sistemas, el filtro adaptable con el algoritmo LMS estima, apoyándose en la teoría del Método de pasos descendentes, la trayectoria del ruido para poder eliminar este de la fuente primaria y obtener la señal de información.

Posteriormente pasamos a probar el algoritmo RLS, conservando la misma estructura del filtro, solo cambiamos el algoritmo de la parte adaptiva. Encontramos que el filtro adaptable con algoritmo RLS tiene mayor velocidad de convergencia, logrando eliminar el ruido de la señal primaria mucho más rápido que el algoritmo LMS; esto es en congruencia con la teoría detrás de estos dos algoritmos. Mientras que el algoritmo LMS se enfoca en simplicidad y estabilidad, sacrifica velocidad de convergencia; a diferencia del algoritmo RLS, el cual aprovecha los mínimos cuadrados recursivos para acelerar la velocidad de convergencia, con el impacto en complejidad de cálculos y estabilidad que esto conlleva.

El segundo grupo de algoritmos probados corresponden a los algoritmos de Cancelación del Ruido en el dominio de la frecuencia. A diferencia de los algoritmos anteriores que trabajan con las muestras de las señales primarias y secundarias en el dominio del tiempo, estos algoritmos utilizan las versiones digitales de la transformada de Fourier ó transformada Cosenos para analizar las señales involucradas y poder realizar el filtrado de la señal contaminante. Al trabajar estos algoritmos en el dominio de la frecuencia, es necesario realizar la transformada correspondiente a la entrada y salida del filtro, lo que impone una gran carga

computacional. Sin embargo, el desempeño de los filtros los convierte en una valiosa herramienta cuando la complejidad de cálculos no es la principal limitante, ya que la velocidad de convergencia y estabilidad de ambos algoritmos es formidable; teniendo como mejor opción el algoritmo de Control Activo del Ruido con Transformada Discreta de Fourier que ofrece la mayor velocidad de convergencia debido al uso de la Transformada Rápida de Fourier.

Por último se implementó el algoritmo LMS en el DSP Texas Instruments TMS320C25, demostrando que es posible implementar un esquema monocanal de Control Activo del Ruido de manera simple y sencilla, en este caso aprovechando las cualidades del algoritmo LMS y maximizando las propiedades del C25 permitiendo que el filtro adaptable implementado trabaje de manera rápida y efectiva filtrando el ruido contaminante de la señal primaria recuperando de esta la señal de datos.

Los esquemas y estructuras implementados corresponden a las metas y objetivos planteados en el trabajo. Además de implementar un filtro de Cancelación Activa del Ruido en un DSP, dando pauta a que algoritmos más complejos sean implementados en DSP más complejos y con mayores recursos.

6.1. Trabajo Futuro

Como ya se mencionó, se analizaron los algoritmos en simulación, por lo que es necesario implementar estos algoritmos en en un procesador de señales digitales a modo que se compruebe el comportamiento de estos en tiempo real.

Gracias al desarrollo de la tecnología, cada día existen DSP's y procesadores más poderosos, con mayor capacidad de cálculo y recursos, menor consumo de energía; que además ofrecen mejores herramientas para su programación y manipulación; estos procesadores son cada día más accesibles, al ser incluidos en dispositivos de uso común como teléfonos celulares inteligentes, computadores personales y otros dispositivos.

Esto abre la posibilidad de implementar esquemas de Control Activo del Ruido en la mayoría de estos dispositivos, tomando ventaja de las propiedades de esta nueva generación de procesadores y los sistemas operativos que utilizan. Para de esta forma lograr reducir el ruido que es inherente a todos los sistemas de telecomunicaciones.

Apéndice A

Código Auxiliar

En esta sección se encuentra el código de programas que fueron utilizados en la creación de los programas de Control Activo del Ruido para MATLAB, SIMULINK y el TMS320C25 (código en ensamblador).

A.1. Código Auxiliar de MATLAB

El siguiente código se usó para general la señal primaria usada en la simulaciones de MATLAB para control activo del ruido en el dominio del tiempo (capítulo 4).

```
N = 6000;
t=1:N;
signal = sin(2*pi.*t.*t/N/N*8);
plot(signal)
title('Señal Primaria')
```

El siguiente código fue usado para procesar los archivos de audio que contienen las señales a ser usadas en los modelos de SIMULINK para el Control Activo del Ruido en el Dominio de la Frecuencia (sección 4.5).

```
NomArch='voz.wav';       % Nombre del Archivo a leer
fs=22050;                % Frecuencia de muestreo
ts=1/fs;
tsamp=3.5;               % Tiempo de muestreo
N=tsamp*fs+1;            % Numero de muestras
 % Se lee el archivo WAV
Samp=wavread(NomArch,N);
t=0:ts:(ts*(size(Samp,1)-1));
t=t';                    % Se guardan los datos en la variable t
 % La variable 'VOZ' será usada en el modelo en SIMULINK
```

```
11 voz(:,1)=t;
12 voz(:,2)=Samp;
```

A.2. Código Auxiliar Ensamblador

Código de un generador de señales senoidales:

```
1  ********************************
2  * Generador de Senal Senoidal *
3  ********************************
4  *
5           AORG    >0000
6  RESET    B       INIT
7  *
8           AORG    >0020
9  *
10 * Inicio del MicroControlador
11 *
12 INIT     SOVM
13      LDPK    0           ;Flag Cero
14      ZAC             ;Anula el acumulador
15      LARP    AR2         ;Actualiza el Registro Auxiliar AR2
16      LRLK    AR2,>0060       ;Inicializa el bloque B2
17      RPTK    31          ;Repite la instruccion que sigue 6 veces
18      SACL    *+          ;Anula el bloque B2
19      LRLK    AR2,>0060       ;Inicia el bloque B2 direccion >0060
20      RPTK    2           ;Repetir 5 veces
21      BLKP    COEF,*+         ;Cargar los coeficientes en el bloque B2
22      LACK    1
23      SACL    ONE
24 *
25 * Declaracion de las Variables
26 *
27 B1   EQU >0060           ;\
28 A1   EQU >0061           ; > Coeficientes del Filtro
29 A2   EQU >0062           ;/
30 YN   EQU >0063           ;Senal de salida
31 XN   EQU >0064           ;\
32 N1   EQU >0065           ; \
33 N2   EQU >0066           ;  > Variables del Filtro
34 N3   EQU >0067           ; /
35 N4   EQU >0068           ;/
36 ONE  EQU >0069
37 *
38 * Programa en ensamblador
39 *
40 OTRO     IN  XN,PA1          ;Inicio, entrada
41      LT  XN          ;<N>-->ACC
```

```
42      MPY B1          ;<ACC>—>YN
43      LTA N4          ;<YN>—>PA2
44      MPY A1          ;0—->ACC
45      MPYA    A1          ;<XN>—>TR
46      APAC                ;<A1*XN>—>PR
47      ADD N2,15           ;<N>—>TR  <A1*XN>—>ACC
48      SACH    N3,1            ;<N*B1>—>PR
49      ZAC             ;<A1*XN+B1*N>—>ACC
50      LAC ONE,14          ;<A1*XN+B1*N+N2>—>ACC
51      MPY A2          ;<A1*XN+B1*N+N2>—>N1
52      APAC                ;0—->ACC
53      SACH    N1,1            ;<B2*N>—>PR
54      OUT N4,PA2          ;<N1>—>TR  <N1>—>N  <B2*N>—>ACC
55      LTD N3          ;<N3>—>N2
56      LTD N1
57      ZAC             ;0—->ACC
58      LAC ONE,14
59      MPYK    0           ;0—->PR
60      B   OTRO            ;Salto a la etiqueta OTRO
61  *
62  * Constantes del filtro
63  *
64  COEF    DATA    >4BC3,>678D,>8000
65          END
```

Código de un generador de señales senoidales y cosenoidales:

```
1   ********************************
2   * Generador de senos y cosenos *
3   ********************************
4   *
5           AORG    >0000
6   RESET   B       INIT
7   *
8           AORG    >0020
9   *
10  * Inicio del MicroControlador
11  *
12  INIT    SOVM
13      LDPK    0               ;Flag Cero
14      ZAC                     ;Pone en CERO el valor del acumulador
15      LARP    AR2             ;Habilita el Registro Auxiliar AR2
16      LRLK    AR2,>0060       ;Inicializa el bloque B2
17      RPTK    5               ;Repite la instrucción que sigue 6 veces
18      SACL    *+              ;Anula el bloque B2
19      LRLK    AR2,>0060       ;Inicia el bloque B2 dirección >0060
20      RPTK    2               ;Repetir 5 veces
21      BLKP    COEF,*+         ;Cargar los coeficientes en el bloque B2
22      LACK    1
23      SACL    ONE
```

```
24      ZAC
25  *
26  * Declaracion de las Variables
27  *
28  B    EQU >0060
29  A    EQU >0061
30  XN   EQU >0062       ;Senal de salida
31  ONE  EQU >0063       ;\
32  YN   EQU >0064       ; \
33  N    EQU >0065       ;  > Variables del Filtro
34  N1   EQU >0066       ; /
35  N2   EQU >0067       ;/
36  N3   EQU >0068
37  SUMA     EQU >0069
38  *
39  * Programa en ensamblador
40  *
41      B       ET2
42  ET1 ZAC
43      SACH    XN
44  ET2 LT      N3
45      MPY     B
46      LTA     N1
47      NEG
48      MPY     A
49      APAC
50      ADD     XN,15
51      SACH    N,1
52      OUT     N,PA3
53      ZAC
54      LAC     ONE,14
55      MPY     B
56      LTA     N3
57      MPY     A
58      APAC
59      SACH    N2,1
60      LTD     N
61      LTD     N2
62      OUT     N2,PA2
63      ZAC
64      B   ET1
65  *
66  * Constantes del filtro
67  *
68  COEF    DATA    13804,30041,6553
69          END
```

Bibliografía

[1] G. Coutu and M. Dignan. Adaptive discrete cosine transform for feedback active noise control. In *Signals, Systems and Computers, 1995. 1995 Conference Record of the Twenty-Ninth Asilomar Conference on*, volume 1, pages 459–463 vol.1, Oct-1 Nov 1995.

[2] Paulo S. R. Diniz. *Adaptive Filtering: Algorithms and Practical Implementation*. Springer Science, third edition edition, 2008.

[3] S.J. Elliott. Down with noise [active noise control]. *Spectrum, IEEE*, 36(6):54–61, Jun 1999.

[4] S.J. Elliott and P.A. Nelson. Active noise control. *Signal Processing Magazine, IEEE*, 10(4):12–35, Oct 1993.

[5] W.S. Gan and S.M. Kuo. An integrated audio and active noise control headset. *Consumer Electronics, IEEE Transactions on*, 48(2):242–247, May 2002.

[6] J.H. Husoy and M.S.E. Abadi. A comparative study of some simplified rls-type algorithms. In *Control, Communications and Signal Processing, 2004. First International Symposium on*, pages 705–708, 2004.

[7] Texas Instruments. TMS320 Second-generation Digital Signal Processors, 1990.

[8] S.M. Kuo and D.R. Morgan. Active noise control: a tutorial review. *Proceedings of the IEEE*, 87(6):943–973, Jun 1999.

[9] R.R. Leitch and M.O. Tokhi. Active noise control systems. *Physical Science, Measurement and Instrumentation, Management and Education, Reviews, IEE Proceedings A*, 134(6):525–546, June 1987.

[10] Paul Lueg. Process of silencing sound oscillations. *U.S. Patent No. 2043416*, 1936.

[11] Sanjit K. Mitra. *Digital Signal Processing. A Computer Based Approach*. McGraw-Hill, 3rd edition, 2005.

[12] A.H.A. Moustafa, N.W. Messiha, A. El-Malawany, M. El-Messiry, and M. Shafik. Classical active noise control technique. In *Radio Science Conference, 1998. NRSC '98. Proceedings of the Fifteenth National*, pages E3/1–E313, Feb 1998.

[13] Alexander D. Poularikas and Zayed M. Ramadan. *Adaptive Filtering Primer with MATLAB*. CRC/Taylor & Francis, 2006.

[14] Bohumil Pšenička. *Procesamiento digital de señales. Filtros Digitales*. Facultad de Ingeniería. UNAM, 1994.

[15] Aplication Report. Design of active noise control systems with the TMS320 family. Technical report, Texas Instruments, 1996.

[16] Sen M. Kuo y Dennis R. Morgan. *Active Noise Control Systems. Algorithms and DSP Implementations*. John Wiley & Sons, 1996.

[17] Raymond H. Knowg y Edward W. Johnston. A variable step sized LMS algorithm. *IEEE Transactions on Signal Processing*, 40(07), July 1992.

[18] Bohumil Pšenička y Mauricio Ortega. *Aplicaciones de los microprocesadores TMS320CXX*. UNAM. Facultad de Ingeniería, 2000.

[19] Ondračka J. y Oravec R. y Kadlec Jiří y Cocherová E. Simulation of RLS and LMS algorithms for adaptive noise cancellation in matlab. *Sborník příspěvků 8. ročníku konference MATLAB 2000*, pages 301–305, 2000.

[20] Alan V. Oppenheim y Ronald W. Shaffer y John R. Buck. *Discrete-Time Signal Processing*. Prentice Hall, 1999.

[21] Bohumil Pšenička y Salvador Landeros Ayala y Milan Karpf. *Practicas de laboratorio con Microprocesadores TMS320C30*. UNAM. Facultad de Ingeniería, 2002.

[22] Bernard Widrow y Samuel D. Stearns. *Adaptive Signal Processing*. Prentice-Hall, 1985.

[23] John G. Proakis y Vinay K. Ingle. *Digital Signal Processing Using Matlab V4*. PWS Publishing Company, 1997.