# 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.

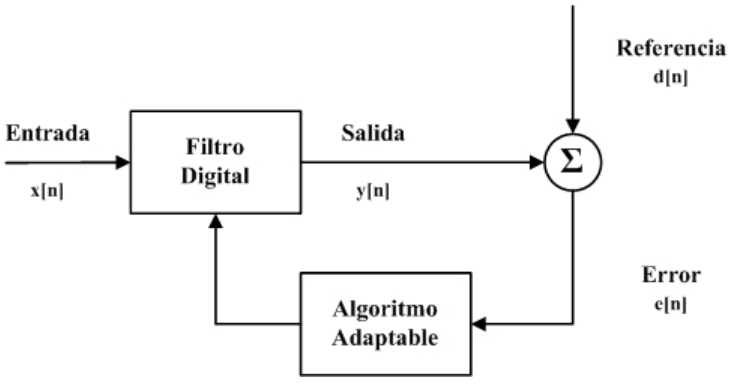


**Figura 3.1:** Filtro adaptable alimentado a priori *(feedforward)*

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.

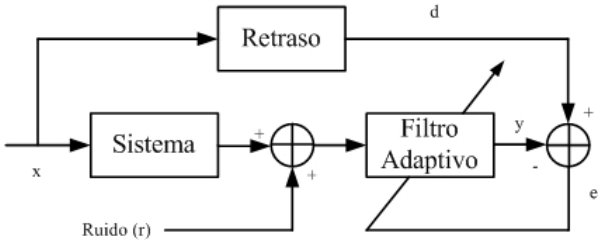


**Figura 3.2:** Cancelación de ruido a la salida

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.

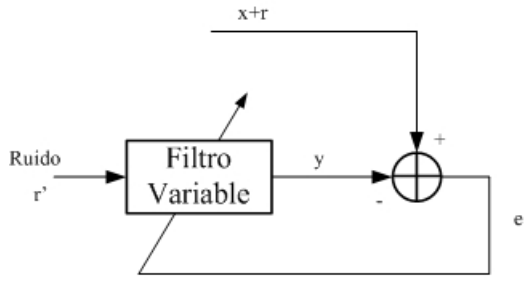


**Figura 3.3:** Cancelación de ruido a la entrada

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.

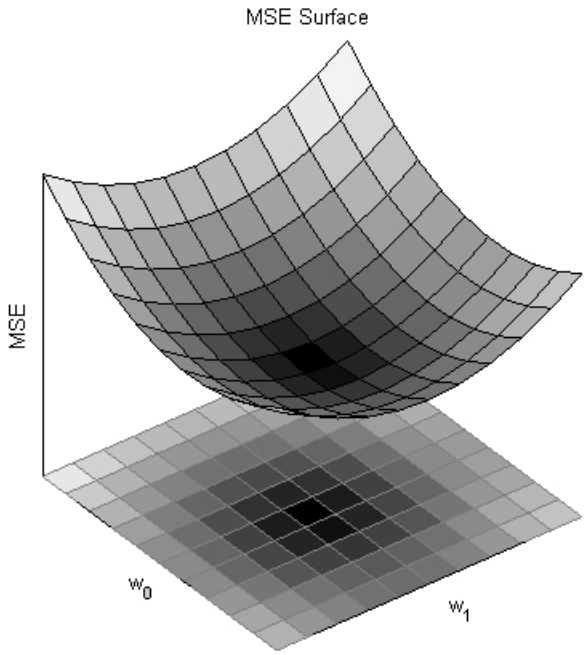


**Figura 3.4:** Superficie de desempeño tridimensional para el caso $L = 2$

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 $\xi_{min}$, 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 $\mu$ 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 $\mu$, 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:

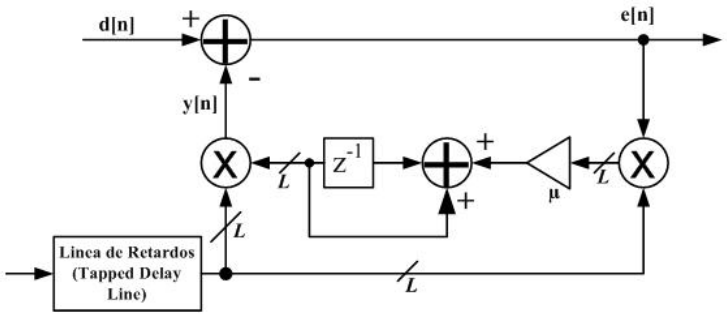


**Figura 3.5:** Diagrama de bloques de el algoritmo LMS

El algoritmo puede ser resumido como:

1. Se escogen los parámetros y las condiciones iniciales: $L$, $\mu$ y $w[0]$. Donde $L$ es el orden del filtro, $\mu$ 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 $\mu$:

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

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**

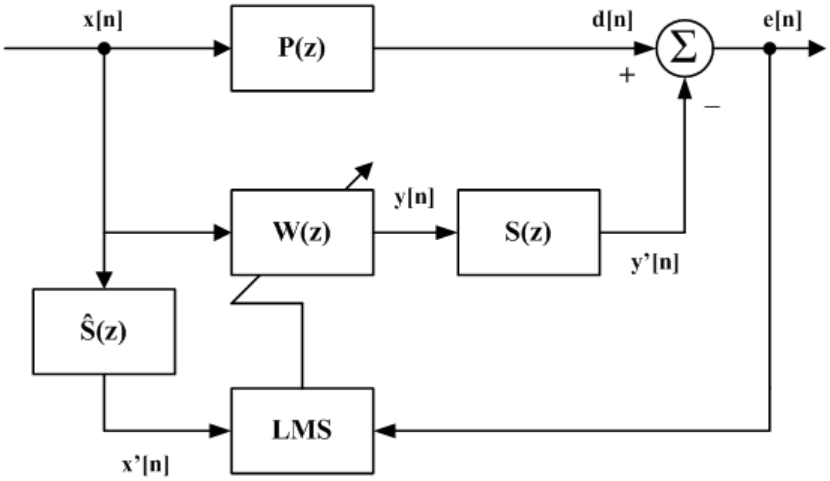


**Figura 3.6:** Diagrama de bloques de el algoritmo FXLMS

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.

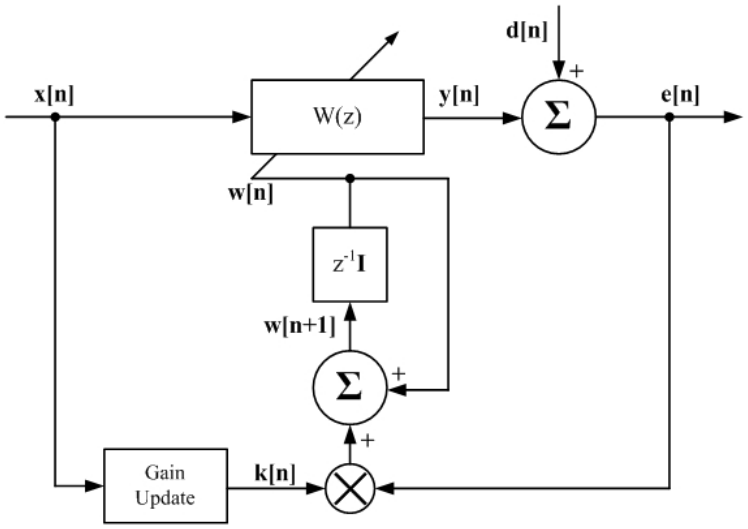


**Figura 3.7:** Diagrama de bloques de el algoritmo RLS.

### 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.

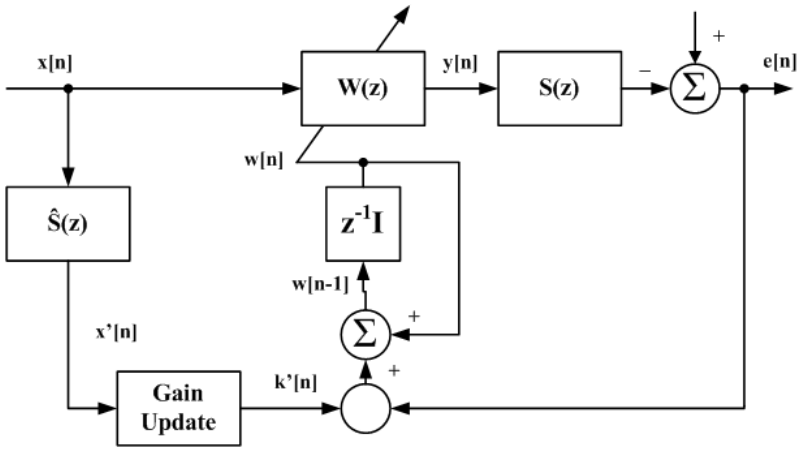


**Figura 3.8:** Diagrama de bloques de el algoritmo **FXRLS**.

## 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

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

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