8 Filtrado adaptivo en el domínio frecuencia

- Teniendo en cuenta que la **Transformada de Fourier** mapea señales en el domínio tiempo en el domínio frecuencia, es razonable llevar a cabo la adaptación en ese domínio.
- En ese caso discutiremos **algoritmos de filtrado adaptivo en el domínio frecuencia** (FDAF).
- Algunas de las razones principales para utilizar este tipo de algoritmos en el domínio frecuencia residen en los **algoritmos eficientes disponibles** para realizar el mapeo (implementación de convolución rápida) y las **propiedades de ortogonalización** asociadas a la transformación que permiten mayor velocidad de convergencia.
- Teniendo en cuenta algoritmos eficientes es posible considerar dos tipos de algoritmos FDAF:
  - **Implementación en bloques** de un filtro FIR, lo cual permite el uso eficiente de procesamiento en paralelo y en consecuencia resulta en mejoras de la velocidad computacional.
  - A partir de la **Transformada rápida de Fourier** (FFT), efectuando convolución rápida lo cual permite realizar la adaptación de los coeficientes del filtro en el domínio frecuencia en una forma sumamente eficiente.
- En relación a las propiedades de ortogonalización es posible considerar algoritmos subóptimos que en el caso ideal permiten tanto aumentar considerablemente la velocidad de convergencia como mejorar el condicionamiento numérico del algoritmo LMS.

8.1 Filtrado adaptivo en bloques

- En un **filtro adaptivo en bloques**, como mostrado, $u(n)$ se divide en bloques de L puntos por medio de un conversor serie-paralelo, y el bloque de datos de entrada así generado se aplica a un filtro FIR de longitud M de a un bloque por vez.
- Los coeficientes del filtro se mantienen fijos para cada bloque de datos de forma que la adaptación del filtro se realiza de bloque en bloque antes que de muestra en muestra como en el algoritmo LMS estandar. El índice k indicará entonces cada bloque y $\hat{\boldsymbol{w}}(k)$ el vector de coeficientes del filtro para el k-ésimo bloque, tal que

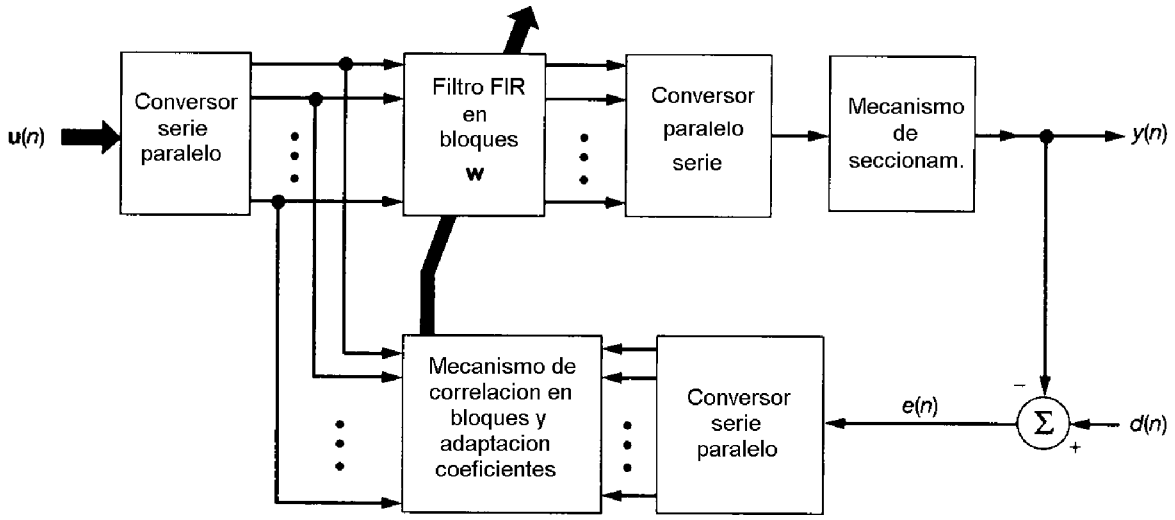

Figura 32: Filtro adaptivo en bloques.

$$\hat{\boldsymbol{w}}(k) = [\hat{w}_0(k), \hat{w}_1(k), \cdots, \hat{w}_{M-1}(k)]^T, \quad k = 0, 1, \cdots$$

- El índice n se reservará para cada muestra particular, lo que escrito en términos de bloques quedaría

$$n = kL + i, \quad i = 0, 1, \cdots, M-1, \quad k = 0, 1, \cdots$$

- Escribiendo el vector de entradas $\boldsymbol{u}(n)$ como

$$\boldsymbol{u}(n) = [u(n), u(n-1), \cdots, u(n-M+1)]^T$$

la salida del filtro $y(n)$ producida en el instante n será

$$y(n) = \hat{\boldsymbol{w}}^T(k)\boldsymbol{u}(n)$$

o en forma equivalente

$$\begin{aligned} y(kL+i) &= \hat{\boldsymbol{w}}^T(k)\boldsymbol{u}(kL+i) && (100) \\ &= \sum_{l=0}^{M-1} \hat{w}_l(k)u(kL+i-l), \quad i = 0, 1, \cdots, M && (101) \end{aligned}$$

- Es posible definir la respuesta deseada como $d(n) = d(kL+i)$, entonces la señal error será $e(kL+i) = d(kL+i) - y(kL+i)$, de forma que la señal de error varía a la velocidad de las muestras como en el algoritmo LMS estandar. La secuencia de error $e(n)$ se divide a su vez en bloques de L puntos en forma sincrónica con el final de la entrada al filtro adaptivo en bloques y se utiliza para el ajuste de los coeficientes del filtro, como mostrado en la figura 32.

- Para ilustrar la operación del filtro adaptivo en bloques, consideremos la longitud M y tamaño del bloque L igual a 3. La secuencia de salida calculada por el filtro en tres bloques consecutivos, $k-1, k$ y $k+1$ será

$$\text{bloque } (k-1) \quad \left\{ \begin{bmatrix} u(3k-3) & u(3k-4) & u(3k-5) \\ u(3k-2) & u(3k-3) & u(3k-4) \\ u(3k-1) & u(3k-2) & u(3k-3) \end{bmatrix} \begin{bmatrix} w_0(k-1) \\ w_1(k-1) \\ w_2(k-1) \end{bmatrix} = \begin{bmatrix} y(3k-3) \\ y(3k-2) \\ y(3k-1) \end{bmatrix} \right.$$

$$\text{bloque } k \quad \left\{ \begin{bmatrix} u(3k) & u(3k-1) & u(3k-2) \\ u(3k+1) & u(3k) & u(3k-1) \\ u(3k+2) & u(3k+1) & u(3k) \end{bmatrix} \begin{bmatrix} w_0(k) \\ w_1(k) \\ w_2(k) \end{bmatrix} = \begin{bmatrix} y(3k) \\ y(3k+1) \\ y(3k+2) \end{bmatrix} \right.$$

$$\text{bloque } (k+1) \quad \left\{ \begin{bmatrix} u(3k+3) & u(3k+2) & u(3k+1) \\ u(3k+4) & u(3k+3) & u(3k+2) \\ u(3k+5) & u(3k+4) & u(3k+3) \end{bmatrix} \begin{bmatrix} w_0(k+1) \\ w_1(k+1) \\ w_2(k+1) \end{bmatrix} = \begin{bmatrix} y(3k+3) \\ y(3k+4) \\ y(3k+5) \end{bmatrix} \right.$$

Algoritmo LMS en bloques.

- Del desarrollo presentado para el algoritmo LMS de secciones anteriores, la forma de adaptación aplicada al vector de coeficientes de una iteración para otra (suponiendo datos reales) es:

$$\begin{pmatrix} \text{Ajuste del} \\ \text{vector de} \\ \text{coeficientes} \end{pmatrix} = \begin{pmatrix} \text{Factor de} \\ \text{convergencia} \end{pmatrix} \begin{pmatrix} \text{Vector de} \\ \text{entradas} \end{pmatrix} (\text{señal de error})$$

- Teniendo en cuenta que en el algoritmo LMS en bloques la señal error puede variar a la velocidad de las muestras de entrada se concluye que cada bloque de entrada tiene diferentes valores de la señal error para usar en el proceso de adaptación. En consecuencia, para el k-ésimo bloque es posible sumar el producto $\boldsymbol{u}(kL+i)e(kL+i)$ sobre todos los valores de i y definir así la ecuación de adaptación para los coeficientes del algoritmo LMS en bloques operando sobre datos reales como

$$\begin{aligned} \hat{\boldsymbol{w}}(k+1) &= \hat{\boldsymbol{w}}(k) + \mu \sum_{i=0}^{L-1} \boldsymbol{u}(kM+i)e(kM+i) \\ &= \hat{\boldsymbol{w}}(k) + \mu \boldsymbol{\phi}(k) \end{aligned} \quad (102)$$

donde μ es el factor de convergencia y $\boldsymbol{\phi}(k) = \boldsymbol{u}(kM+i)e(kM+i)$. El j-ésimo elemento del vector $\boldsymbol{\phi}(k)$ está definido por

$$\varphi_j(k) = \sum_{i=0}^{L-1} u(kM+i-j)e(kM+i), \qquad j = 0, 1, \cdots, M \quad (103)$$

- Una característica distintiva del algoritmo LMS en bloques es que se incorpora una **estimación promedio** del vector gradiente, dada por

$$\hat{\boldsymbol{\nabla}}(k) = -\frac{2}{L}\sum_{i=0}^{L-1} \boldsymbol{u}(kL+i)e(kL+i) \tag{104}$$

donde el factor 2 se incluye para ser consistente con el vector de gradiente utilizado en las secciones anteriores, y el factor $1/L$ se incluye para que $\hat{\boldsymbol{\nabla}}(k)$ sea un promedio temporal no sesgado. De esta forma es posible reformular el algoritmo LMS en bloques como

$$\hat{\boldsymbol{w}}(k+1) \;=\; \hat{\boldsymbol{w}}(k) - \frac{1}{2}\mu_B \hat{\boldsymbol{\nabla}}(k)$$

donde μ_B puede interpretarse como el factor de convergencia **efectivo** para el algoritmo LMS en bloques: $L\mu$.

Propiedades de convergencia del algoritmo LMS en bloques

- El algoritmo LMS en bloques tiene propiedades similares al algoritmo LMS estandar en el sentido que los dos minimizan la función del error medio cuadrático $J = \frac{1}{2}E[e^2(n)]$, donde E es el operador esperanza estadística.
- La diferencia fundamental entre esos dos algoritmos reside en las estimaciones del vector gradiente utilizada y en las respectivas implementaciones.
- Comparando la estimación del algoritmo LMS en bloques con la del algoritmo LMS convencional, vemos que el primero utiliza una estimación más precisa del vector gradiente debido a el promedio en tiempo, tal que la precisión mejora en la medida que aumenta el tamaño del bloque.
- Sin embargo, esta mejora no implica mayor velocidad de convergencia como se podrá concluir al estudiar las propiedades de convergencia del algoritmo LMS en bloques.
- El análisis de convergencia es similar al realizado en las secciones anteriores para el algoritmo LMS convencional. Existe solo una pequeña modificación a ser considerada, o sea, la sumatoria de ciertas esperanzas sobre el índice $i = 0, 1, \cdots, L-1$, el cual está relacionado con la muestra de tiempo n por $n = kL + i$. Una síntesis de esas propiedades es la siguiente

  1. **Condición de convergencia.**

$$\lim_{k \to \infty} E\left[\hat{\boldsymbol{w}}(k)\right] = \boldsymbol{R}^{-1}\boldsymbol{p} = \boldsymbol{w}_o$$

La condición que debe satisfacer el factor de convergencia μ para que el algoritmo LMS en bloques converja en media es $0 < \mu < \frac{2}{L\lambda_{max}}$, donde L es el tamaño del bloque y λ_{max} es el mayor autovalor de $\boldsymbol{R}$.

2. **Desajuste.** En base a las definiciones de exceso de error medio cuadrático $J_{ex}(k)$ y el error cuadrático medio mínimo J_{min} de las secciones anteriores, es posible notar que para que $J_{ex}(k)$ calculado para el algoritmo LMS en bloques converja a una constante $J_{ex}(\infty) < J_{min}$ cuando el número de iteraciones k tienda a infinito, el factor de convergencia μ debe satisfacer la condición más restrictiva

$$0 < \mu < \frac{2}{L \sum_{i=1}^{M} \lambda_i} \tag{105}$$

tal que el desajuste correspondiente es

$$\mathcal{M} = \frac{\mu}{2} \sum_{i=1}^{M} \lambda_i$$

- Comparando los resultados descriptos del algoritmo LMS en bloques con los correspondientes al algorimo LMS estandar es posible observar que
  - El valor de convergencia en media del vector de coeficientes y el desajuste del algoritmo LMS en bloques son idénticos a los del algoritmo LMS estandar.
  - La condición de convergencia en media del algoritmo LMS en bloques es más restrictiva que la correspondiente del algoritmo LMS estandar. En particular, la cota más estrecha sobre el factor de convergencia μ puede causar que este algoritmo converja más lentamente que el estandar, en especial cuando la dispersión de autovalores de $\boldsymbol{R}$ es grande.

Elección del tamaño del bloque

- Un aspecto importante que necesita ser considerado en el diseño de un filtro adaptivo en bloques es como elegir el tamaño del bloque. De (102) observamos que la operación del algoritmo LMS en bloques está bien definida siempre que L sea un entero mayor que la unidad. Independientemente de esto, lo usual es elegir $L = M$, la longitud del filtro.
- Alguna justificación para esta elección es la siguiente:
  - Cuando $L > M$, se realizan operaciones redundantes en el proceso debido a que la estimación del vector gradiente (calculado sobre L puntos) utiliza más información que el filtro mismo.
  - Cuando $L < M$, algunos de los coeficientes del filtro no son utilizados debido a que la secuencia de entrada no es suficientemente larga para alimentar el filtro completo.

8.2 Algoritmo LMS en bloques eficiente

- Dada una aplicación de procesamiento adaptivo de señales para la cual el algoritmo LMS en bloques es una solución satisfactoria, la cuestión clave a considerar es como implementarlo en forma computacionalmente eficiente. La complejidad computacional del algoritmo LMS en bloques está relacionada con
  - La ecuación (101), que define una **convolución lineal** de la secuencia de entrada y los coeficientes del filtro.
  - La ecuación (103), que define una **correlación lineal** entre la secuencia de entrada y el error.
- El algoritmo de **transformada rápida de Fourier** provee una poderosa herramienta para efectuar **convolución rápida** y **correlación rápida**. Estas observaciones apuntan a un método en el domínio frecuencia para implementación eficiente del algoritmo LMS en bloques.
- Especificamente, antes que efectuar la adaptación en el domínio tiempo como descripto anteriormente, la adaptación de los parámetros del filtro se realiza en el domínio frecuencia usando el algoritmo de FFT.
- La convolución rápida puede efectuarse usando el método **overlap-save** o, alternativamente, el método **overlap-add**. Sin embargo, para implementar el algoritmo LMS este segundo método resulta en mayor complejidad que el primero. De acuerdo a resultados experimentales, la mayor eficiencia computacional obtenida con el primero de los métodos se logra para un solapamiento del 50 %, por lo que se utilizará ese porcentaje en lo que sigue.

- De acuerdo a este método, los M coeficientes del filtro se rellenan con igual número de ceros y se utiliza una FFT de N puntos, donde $N = 2M$. Si $\hat{\boldsymbol{W}}(k)$ $(N \times 1)$ es el vector de coeficientes de la FFT de $\hat{\boldsymbol{w}}(k)$ rellenado, o sea

$$\hat{\boldsymbol{W}}(k) = FFT \begin{bmatrix} \hat{\boldsymbol{w}}(k) \\ \mathbf{0} \end{bmatrix}$$

donde $FFT[\,]$ es la transformada rápida de Fourier. Notar que $\hat{\boldsymbol{W}}(k)$ es dos veces más largo que $\hat{\boldsymbol{w}}(k)$. En consecuencia, si $\boldsymbol{U}(k)$ es la matriz diagonal de $N \times N$ obtenida de los datos de entrada como sigue:

$$\boldsymbol{U}(k) = diag\{FFT[\underbrace{u(kM-M),\cdots,u(kM-1)}_{\text{bloque } (k-1)\text{-ésimo}},\underbrace{u(kM),\cdots,u(kM+M-1)}_{\text{bloque } k\text{-ésimo}}]\}$$

- Aplicando el método overlap-save a la convolución lineal de (101) se obtiene el vector

$$\begin{aligned} \boldsymbol{y}^T(k) &= [y(kM), y(kM+1), \cdots, y(kM+M-1)] \\ &= \text{últimos M elementos de la } IFFT[\boldsymbol{U}(k)\hat{\boldsymbol{W}}(k)] \end{aligned} \quad (106)$$

donde $IFFT[\,]$ es la transformada rápida de Fourier inversa. Se usan solo los últimos M elementos de la ecuación anterior porque los primeros M deben asociarse a una convolución circular.

- Considerando luego la correlación lineal de la ecuación (103), para el k-ésimo bloque definimos el vector de respuesta deseada

$$\boldsymbol{d}(k) = [d(kM), d(kM+1), \cdots, d(kM+M-1)]^T$$

y el vector de error correspondiente

$$\begin{aligned} \boldsymbol{e}(k) &= [e(kM), e(kM+1), \cdots, e(kM+M-1)]^T \\ &= \boldsymbol{d}(k) - \boldsymbol{y}(k) \end{aligned}$$

- Notando que en la implementación de la convolución lineal descripta en (106) se descartan los primeros M elementos de la salida, es posible transformar el vector de error $\boldsymbol{e}(k)$ al domínio frecuencia como sigue

$$\boldsymbol{E}(k) = FFT \begin{bmatrix} \mathbf{0} \\ \boldsymbol{e}(k) \end{bmatrix}$$

- Luego, teniendo en cuenta que una correlación lineal es basicamente una forma **revertida** de convolución lineal, aplicando el método overlap-save a la correlación lineal de (103) se obtiene

$$\boldsymbol{\phi}(k) = \text{primeros M elementos de la } IFFT\left[\boldsymbol{U}(k)\boldsymbol{E}(k)\right] \quad (107)$$

- Notar también que mientras en el caso de convolución lineal de (106) se descartan los primeros M elementos, en la ecuación anterior se descartan los M últimos.

- Finalmente, considerando la ecuación (102) para la adaptación del vector de coeficientes del filtro y notando que en la definición de $\hat{\boldsymbol{W}}(k)$, $\hat{\boldsymbol{w}}(k)$ es seguido de M ceros, es posible transformar (102) al domínio frecuencia como sigue

$$\hat{\boldsymbol{W}}(k+1) = \hat{\boldsymbol{W}}(k) + \mu FFT \begin{bmatrix} \boldsymbol{\phi}(k) \\ \mathbf{0} \end{bmatrix} \quad (108)$$

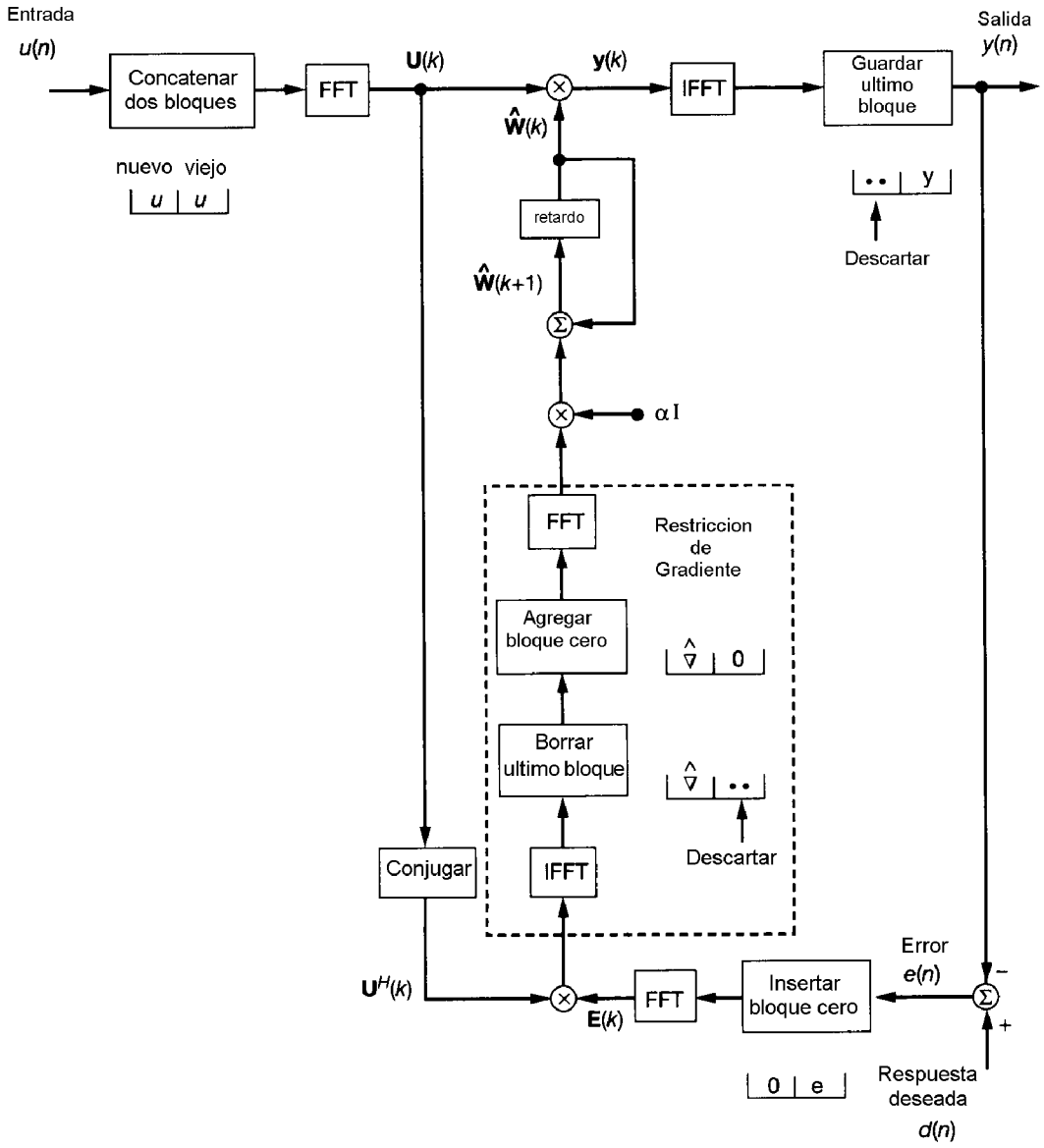

Figura 33: FDAF con overlap-save.

- La figura 33 muestra un representación de este algoritmo. Este algoritmo es la representación precisa del algoritmo LMS en bloques en el domínio frecuencia. Como tal, las propiedades de convergencia son idénticas a las discutidas para ese algoritmo.

Complejidad computacional

- La complejidad del algoritmo LMS en bloques eficiente se comparará con el algoritmo LMS estandar. La comparación se basa en el número total de multiplicaciones involucradas en cada una de esas implementaciones para un tamaño de bloque M.

- Consideramos primero el algoritmo LMS estandar con M coeficientes operando sobre datos reales: M multiplicaciones para la salida y otras M para los coeficientes, o sea $2M$ multiplicaciones por iteración. Para un bloque de M muestras de salida, el número total de multiplicaciones es $2M^2$.

- Considerando ahora el algoritmo LMS en bloques, cada FFT de N puntos (y IFFT) requiere aproximadamente $n \log_2 N$ multiplicaciones reales, donde $N = 2M$. Existen 5 transformaciones por lo que serán $5N \log_2 N$ multiplicaciones y vector de salida en el domínio frecuencia requiere $4N$ multiplicaciones, lo mismo que las correlaciones cruzadas asociadas a la estimación del vector gradiente. Entonces, el número total de multiplicaciones de este algoritmo es

$$\begin{aligned} 5N \log_2 N + 8N &= 10M \log_2(2M) + 16M \\ &= 10M \log_2 M + 26M \end{aligned}$$

$$\begin{aligned} \text{Relación de complejidad} &= \frac{10M \log_2 M + 26M}{2M^2} \\ &= \frac{5 \log_2 M + 13}{M} \end{aligned}$$

Por ejemplo, para $M = 1024$, el algoritmo en bloques es 16 veces más rápido que el algoritmo estandar en términos computacionales.

Mejora en la velocidad de convergencia

- A pesar de que el algoritmo LMS en bloques, del cual el del domínino frecuencia es obtenido, utiliza una estimación más precisa del vector gradiente que el algoritmo LMS estandar, los coeficientes deben adaptarse en incrementos suficientemente pequeños para asegurar la estabilidad del algoritmo. Esto está relacionado con μ que asegura un desajuste menor que 100 % y permite garantizar la convergencia en media cuadrática. La cuestión de la estabilidad del algoritmo es particularmente seria cuando se opera en un ambiente para el cual los autovalores de la matriz correlación son muy dispersos.

- Es posible mejorar la velocidad de convergencia de este algoritmo en base a las siguientes observaciones

  - Los coeficientes se adaptan independientemente, lo cual significa que cada uno se asocia con un modo del proceso de adaptación. Dado que los modos son facilmente accesibles, las velocidades de convergencia individuales pueden variarse directamente. Así, mientras en el algoritmo LMS estandar cada coeficiente es responsable por una mezcla de modos, en el algoritmo eficiente es responsable por un único modo y su velocidad de convergencia entonces puede optimizarse.
  - Suponiendo entradas ESA, el tiempo de convergencia para el i-ésimo modo es inversamente proporcional a $\mu\lambda_i$, donde λ_i es el i-ésimo autovalor de $\boldsymbol{R}$; donde λ_i es una medida de la potencia promedio de entrada al i-ésimo bin de frecuencia.

    De acuerdo a esto, es posible hacer que los modos del proceso de adaptación converjan esencialmente a la misma velocidad asignando a cada coeficiente un factor de convergencia, como

$$\mu_i = \frac{\alpha}{P_i} \quad i = 0, 1, \cdots, M\ -1$$

donde α es una constante y P_i es una estimación de la potencia promedio en el i-ésimo bin. Bajo esta condición, los coeficientes convergen con la misma constante de tiempo, definida por

$$\tau\ = \frac{2M}{\alpha}$$

Cuando el ambiente no es estacionario, es posible utilizar la siguiente recursión para obtener una estimación de la potencia promedio

$$P_i(k) = \gamma P_i(k-1) + (1-\gamma)|U_i(k)|^2, \quad i = 0, 1, \cdots, 2M\ -1$$

donde $U_i(k)$ es la entrada al i-ésimo coeficiente del algoritmo LMS en bloques eficiente en el instante k y γ es una constante $(0 < \gamma < 1)$, que determina la memoria del proceso de estimación anterior. De esta forma, dada la estimación $P_i(k)$ de la potencia promedio en el i-ésimo bin, el factor de convergencia μ se reemplaza por una matriz diagonal de $(M \times M)$, dada por

$$\boldsymbol{\mu}(k) = \alpha \boldsymbol{D}(k)$$

donde $\boldsymbol{D}(k) = diag[P_0^{-1}(k), P_1^{-1}(k), \cdots, P_{M-1}^{-1}(k)]$.

- La tabla siguiente resume el algoritmo LMS en bloques eficiente.

Inicialización

$\hat{\boldsymbol{W}}(0) = \mathbf{0}$ $(2M \times 1)$

$P_i(0) = \delta_i, \quad i = 0, \cdots, 2M-1$

Para cada bloque nuevo de k muestras, calcular

$\boldsymbol{U}(k) = diag\{FFT[u(kM-M), \cdots, u(kM-1), u(kM), \cdots, u(kM+M-1)]^T\}$

$\boldsymbol{y}(k) =$ últimos M elementos de la $IFFT[\boldsymbol{U}(k)\hat{\boldsymbol{W}}(k)]$

$\boldsymbol{e}(k) = \boldsymbol{d}(k) - \boldsymbol{y}(k)$

$$\boldsymbol{E}(k) = FFT \begin{bmatrix} \mathbf{0} \\ \boldsymbol{e}(k) \end{bmatrix}$$

$P_i(k) = \gamma P_i(k-1) + (1-\gamma)|U_i(k)|^2, \quad i = 0, 1, \cdots, 2M-1$

$\boldsymbol{D}(k) = diag[P_0^{-1}(k), P_1^{-1}(k), \cdots, P_{2M-1}^{-1}(k)]$

$\boldsymbol{\phi}(k) =$ primeros M elementos de la $IFFT[\boldsymbol{U}(k)\boldsymbol{E}(k)]$

$$\hat{\boldsymbol{W}}(k+1) = \hat{\boldsymbol{W}}(k) + \alpha FFT \begin{bmatrix} \boldsymbol{\phi}(k) \\ \mathbf{0} \end{bmatrix}$$

8.3 Filtrado adaptivo en el domínio frecuencia

- El algoritmo LMS en bloques eficiente puede interpretarse como un tipo de filtrado adaptivo en el domínio frecuencia con restricciones. Especificamente, dos de las cinco FFT involucradas en su operación son necesarias para imponer una **restricción en el domínio tiempo** con el propósito de efectuar una correlación lineal correspondiente a $\boldsymbol{\phi}(k)$. La restricción en el domínio tiempo consiste en las siguientes operaciones
  - Los últimos M elementos de FFT inversa de $\boldsymbol{U}^H(k)\boldsymbol{E}(k)$ se descartan.
  - Reemplazar los elementos descartados por un bloque de M ceros antes de reaplicar la FFT, como descripto en la ecuación (108).
- La combinación de las cuatro operaciones descriptas se encuentra en el rectángulo a trazos de la figura 33. Esta combinación se denomina **restricción de gradiente**, teniendo en cuenta que está asociada a la estimación del vector gradiente. Notar que esta es en realidad una restricción en el domínio tiempo. Basicamente, garantiza que los $2M$ coeficientes en el domínio frecuencia correspondan a solo M coeficientes en el domínio tiempo. Esta es la razón por la que se incluye un bloque cero en la restricción de gradiente de la figura 33.
- En el **filtro adaptivo en el domínio frecuencia sin restricciones** esta restricción se elimina por completo. El resultado es una implementación simple que involucra solo tres FFT. Así, la combinación de las ecuaciones (107) y (108) en el algoritmo LMS en bloques eficiente se reemplaza ahora por

$$\hat{\boldsymbol{W}}(k+1) = \hat{\boldsymbol{W}}(k) + \mu \boldsymbol{U}^H(k)\boldsymbol{E}(k)$$

- Sin embargo, es importante notar que la estimación del vector gradiente calculado de esta forma ya no corresponde a una correlación lineal como la de la ecuación (104), sino que corresponde a una convolución circular.

- En consecuencia, es posible concluir que el filtro adaptivo en el domínio frecuencia sin restricciones de la ecuación anterior no tiene el mismo comportamiento que el algoritmo LMS en bloques eficiente y por lo tanto no converge a la solución de Wiener cuando el número de bloques procesados es suficientemente grande. Otro aspecto a notar es que apesar que la velocidad de convergencia del filtro adaptivo en el domínio frecuencia sin restricciones puede incrementarse con factores de convergencia variantes en el tiempo, la mejora se pierde por el incremento en el desajuste. Además, este algoritmo requiere dos veces más iteraciones que el algoritmo con restricciones para producir el mismo nivel de desajuste.

8.4 Algoritmo adaptivo con preprocesamiento

- Se ha discutido como las técnicas en el domínio frecuencia mejoran la eficiencia computacional del algoritmo LMS cuando la aplicación de interés requiere una memoria muy grande. Ahora discutiremos otro aspecto importante de filtrado adaptivo, o sea, el de mejorar la velocidad de convergencia del algoritmo LMS. Esta mejora se obtiene sin embargo a expensas de un incremento en la complejidad computacional.

- Para motivar la discusión consideremos un vector de entrada $\boldsymbol{u}(n)$ caracterizado por la matriz correlación $\boldsymbol{R}$. El **algoritmo de filtrado adaptivo ortogonalizador** para un ambiente estacionario en sentido amplio se describe por

$$\hat{\boldsymbol{w}}(n+1) = \hat{\boldsymbol{w}}(n) + \alpha \boldsymbol{R}^{-1}\boldsymbol{u}(n)e(n) \qquad (109)$$

donde la constante α $(0 < \alpha < 1)$ puede elegirse $\alpha = \frac{1}{2M}$, donde M es la longitud del filtro. Una propiedad importante del algoritmo de filtrado ortogonalizador es que teoricamente puede garantizar una velocidad de convergencia constante independientemente de la estadística de la entrada.

- Para probar esta propiedad definimos el vector de error de los coeficientes del filtro $\boldsymbol{\epsilon}(n) = \hat{\boldsymbol{w}}(n) - \boldsymbol{w}_o$, donde el vector $\boldsymbol{w}_o$ es, como usualmente, la solución de Wiener. Usando esta definición es posible escribir el algoritmo anterior como

$$\boldsymbol{\epsilon}(n+1) = (\boldsymbol{I} - \alpha \boldsymbol{R}^{-1}\boldsymbol{u}(n)\boldsymbol{u}^T(n))\boldsymbol{\epsilon}(n) + \alpha \boldsymbol{R}^{-1}\boldsymbol{u}(n)e_o(n)$$

donde $e_o(n)$ es el valor óptimo del error producida por la solución de Wiener. Aplicando la esperanza estadística a ambos lados de

la anterior, e invocando la suposición de independencia (o sea, $\hat{\boldsymbol{w}}(n)$ es independiente de $\boldsymbol{u}(n)$) se obtiene

$$E[\boldsymbol{\epsilon}(n+1)] = (\boldsymbol{I} - \alpha \boldsymbol{R}^{-1} E[\boldsymbol{u}(n)\boldsymbol{u}^T(n)])E[\boldsymbol{\epsilon}(n)] + \alpha \boldsymbol{R}^{-1}\boldsymbol{u}(n)e_o(n)$$

- Luego es posible concluir que
  - Por definición $E[\boldsymbol{u}(n)\boldsymbol{u}^T(n)] = \boldsymbol{R}$.
  - Por el principio de ortogonalidad $E[\boldsymbol{u}(n)e_o(n)] = \boldsymbol{0}$.
- En consecuencia, es posible simplicar (110) como sigue

$$E[\boldsymbol{\epsilon}(n+1)] = (\boldsymbol{I} - \alpha \boldsymbol{R}^{-1}\boldsymbol{R})E[\boldsymbol{\epsilon}(n)] = (1-\alpha)E[\boldsymbol{\epsilon}(n)]$$

- Esta ecuación representa una ecuación a diferencias de primer orden cuya solución es

$$E[\boldsymbol{\epsilon}(n)] = (1-\alpha)^n E[\boldsymbol{\epsilon}(0)] \qquad (110)$$

donde $\boldsymbol{\epsilon}(0)$ es el valor inicial del vector de error de coeficientes. En consecuencia, con $0 < \alpha < 1$, se tiene que

$$\lim_{n \to \infty} E[\boldsymbol{\epsilon}(n)] = \boldsymbol{0} \qquad \lim_{n \to \infty} E[\hat{\boldsymbol{w}}(n)] = \boldsymbol{w}_o$$

- Lo más importante que puede notarse en (110) es que la velocidad de convergencia es completamente independiente de la estadística de entrada.

Ejemplo. Para ilustrar las propiedades de convergencia del algoritmo de filtrado adaptivo ortogonalizador consideremos el caso de una entrada ruido blanco cuya matriz correlación es

$$\boldsymbol{R} = \sigma^2 \boldsymbol{I}$$

donde σ^2 es la varianza del ruido. Para esa entrada el algoritmo de adaptación será

$$\hat{\boldsymbol{w}}(n+1) = \hat{\boldsymbol{w}}(n) + \frac{1}{2M\sigma^2}\boldsymbol{u}(n)e(n)$$

que coincide con el algoritmo LMS estandar para $\mu = \frac{1}{2M\sigma^2}$. En otras palabras, para el caso especial de una secuencia de entrada ruido blanco caracterizada por una dispersión de autovalores unitária, el algoritmo LMS se comporta de la misma forma que el algoritmo de filtrado adaptivo ortogonalizador.

Filtro adaptivo de dos etapas

- Este último ejemplo sugiere que es posible mecanizar un filtro adaptivo ortogonalizador para un contexto arbitrario procediendo en dos etapas:

  1. $\boldsymbol{u}(n)$ es transformado en un vector correspondiente de variables no correlacionadas.
  2. El vector transformado puede utilizarse como la entrada a un algoritmo LMS.

- De las secciones introductorias es posible recordar que teoricamente el primer objetivo puede realizarse usando la **transformada de Karhunen - Loève** (KLT). Especificamente, dado

un vector $\boldsymbol{u}(n)$ de media cero obtenido de un contexto estacionario en sentido amplio, el vector de salida de la KLT está definido por

$$\nu_i(n) = \boldsymbol{q}_i^T \boldsymbol{u}(n), \quad i = 0, 1, \cdots, M-1$$

donde $\boldsymbol{q}_i$ es el autovector asociado con el i-ésimo autovalor λ_i correspondiente a la matriz $\boldsymbol{R}$ del vector de entrada $\boldsymbol{u}(n)$. Las salidas individuales de la KLT tienen media cero y son variables no correlacionadas o sea

$$E[\nu_i(n)\nu_j(n)] = \begin{cases} \lambda_i, & j = i \\ 0, & j \neq i \end{cases}$$

- De esta manera es posible expresar la matriz correlación de $\boldsymbol{v}(n)$ como

$$\begin{aligned} \boldsymbol{\Lambda} &= E[\boldsymbol{\nu}(n)\boldsymbol{\nu}^T(n)] \\ &= diag[\lambda_0, \lambda_1, \cdots, \lambda_{M-1}] \end{aligned}$$

cuya inversa es

$$\boldsymbol{\Lambda}^{-1} = diag[\lambda_0^{-1}, \lambda_1^{-1}, \cdots, \lambda_{M-1}^{-1}]$$

- Considerando ahora el algoritmo de filtrado adaptivo ortogonalizador de la ecuación (109) con el vector transformado $\boldsymbol{v}(n)$ y su matriz correlación inversa $\boldsymbol{\Lambda}^{-1}$ usado en lugar de $\boldsymbol{u}(n)$ y $\boldsymbol{R}^{-1}$ respectivamente. Bajo esas circunstancias, la ecuación (109) toma la forma

$$\hat{\boldsymbol{w}}(n+1) = \hat{\boldsymbol{w}}(n) + \alpha \boldsymbol{\Lambda}^{-1} \boldsymbol{v}(n) e(n)$$

donde el i-ésimo elemento puede escribirse como

$$\hat{w}_i(n+1) = \hat{w}_i(n) + \frac{\alpha}{\lambda_i} \nu_i(n) e(n), \qquad i = 0, 1, \cdots, M-1$$

- Esta ecuación puede relacionarse directamente con la forma normalizada del algoritmo LMS. La normalización aqui implica que a cada coeficiente se le asigna su propio factor de convergencia que está relacionado con el correspondiente autovalor de la matriz correlación de la entrada.
- La KLT es una transformación dependiente de la señal y su implementación la hacen impráctica para aplicaciones en tiempo real.
- Afortunadamente, la **transformada discreta coseno** (DCT) provee un conjunto de vectores base predeterminado que representa una buena aproximación a la KLT.
- Además, para un proceso estacionario de Markov de primer orden, con media cero, que es lo suficientemente general para el estudio en procesamiento de señales, la DCT es asintoticamente equivalente a la KLT. Obviamente mientras la KLT depende de la señal, la DCT es independiente de la señal y puede entonces implementarse en una forma computacionalmente eficiente.
- La figura 34 muestra el esquema que consiste en dos etapas, en la primera se implementa una **DCT desplazable** y en la segunda una versión **normalizada** del algoritmo LMS.

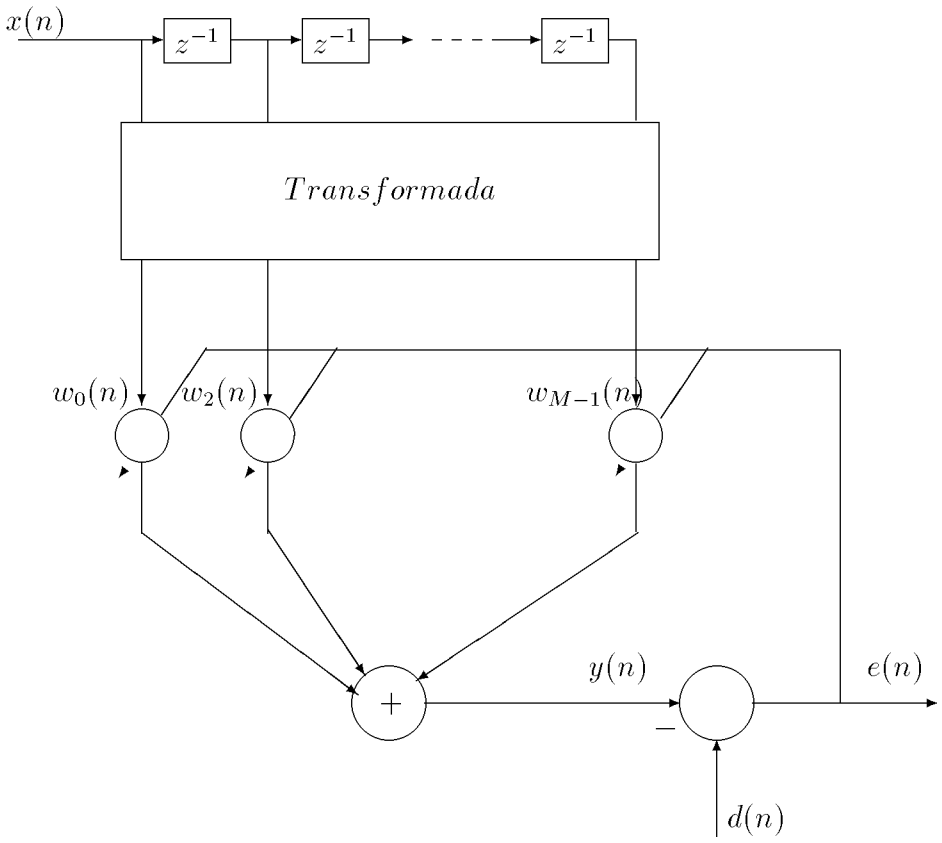

Figura 34: Filtro adaptivo ortogonalizador

DCT desplazable

- La DCT para esta aplicación utiliza una ventana desplazable donde los cálculos se efectúan por cada nueva muestra de entrada.
- Esto a su vez habilita al algoritmo LMS a operar a la velocidad de los datos de entrada como en la forma convencional. De esta forma, a diferencia del algoritmo LMS en bloques eficiente, el algoritmo de filtrado adaptivo en el domínio frecuencia discutido aqui no es un algoritmo en bloques y por lo tanto no es computacionalmente eficiente.
- Teniendo en cuenta que la transformada discreta de Fourier de una secuencia par resulta en la transformada discreta coseno, es posible desarrollar un algoritmo eficiente para calcular la DCT desplazable.
- Consideremos una secuencia de M muestras $u(n)$, $u(n-1)$, $\cdots$, $u(n-M+1)$. Construimos luego una secuencia extendida $a(n)$, simétrica alrededor del punto $(n-M+1/2)$ como sigue

$$a(i) = \begin{cases} u(i) & i = n, n-1, \cdots, n-M+1 \\ u(-i+2n-2M+1), & i = n-M, n-M-1, \cdots, n-2M+1 \end{cases}$$

y definimos $W_{2m} = e^{-j(2\pi)(2M)}$. El elemento m-ésimo de la DFT de $2M$ puntos de la secuencia extendida $a(i)$ está definida por

$$A_m(n) = \sum_{i=n-2M+1}^{n} a(i) W_{2M}^{m(n-i)}$$

de forma que usando estas dos últimas ecuaciones es posible escribir

$$\begin{aligned} A_m(n) &= \sum_{i=n-M+1}^{n} a(i) W_{2M}^{m(n-i)} + \sum_{i=n-2M+1}^{n-M} a(i) W_{2M}^{m(n-i)} \\ &= \sum_{i=n-M+1}^{n} u(i) W_{2M}^{m(n-i)} + \sum_{i=n-2M+1}^{n-M} u(-i+2n-2M+1) W_{2M}^{m(n-i)} \\ &= \sum_{i=n-M+1}^{n} u(i) W_{2M}^{m(n-i)} + \sum_{i=n-M+1}^{n} u(i) W_{2M}^{m(i-n+2M-1)} \end{aligned}$$

- Extrayendo el término $W_{2M}^{m(M-1/2)}$ y combinando las dos sumatorias se obtiene

$$\begin{aligned} A_m(n) &= W_{2M}^{m(M-1/2)} \sum_{i=n-M+1}^{n} u(i) \left(W_{2M}^{-m(i-n+M-1/2)} + W_{2M}^{m(i-n+M-1/2)} \right) \\ &= 2(-1)^m W_{2M}^{-m/2} \sum_{i=n-M+1}^{n} u(i) \cos\left(\frac{m(i-n+M-1/2)\pi}{M} \right) \end{aligned}$$

- Excepto por un factor de escala, la sumatoria anterior es la DCT de la secuencia $u(n)$. Especificamente se tiene que

$$C_m(n) = k_m \sum_{i=n-M+1}^{n} u(i) \cos\left(\frac{m(i-n+M-1/2)\pi}{M} \right)$$

donde la constante k_m está definida por

$$k_m = \begin{cases} 1/\sqrt{2}, & m = 0 \\ 1, & \text{para todo otro } m \end{cases}$$

- De esta forma, la DCT de la secuencia $u(n)$ está relacionada con la DFT de la secuencia extendida $a(n)$ como sigue

$$C_m(n) = \frac{1}{2} k_m (-1)^m W_{2M}^{m/2} A_m(n), \quad m = 0, 1, \cdots, M \qquad (111)$$

- La DFT de la secuencia extendida $a(n)$ puede expresarse como dos DFT complementarias como sigue

$$\begin{aligned} A_m(n) &= \sum_{i=n-M+1}^{n} u(i) W_{2M}^{m(n-i)} + \sum_{i=n-M+1}^{n} u(i) W_{2M}^{m(i-n+2M-1)} \\ &= A_m^{(1)}(n) + A_m^{(2)}(n) \end{aligned} \tag{112}$$

- Considerando la primera DFT $A_m^{(1)}(n)$ y separando la muestra $u(n)$ se tiene que

$$A_m^{(1)}(n) = u(n) + \sum_{i=n-M+1}^{n-1} u(i) W_{2M}^{m(n-i)} \tag{113}$$

pero

$$\begin{aligned} A_m^{(1)}(n-1) &= \sum_{i=n-M}^{n-1} u(i) W_{2M}^{m(n-1-i)} \\ &= (-1)^m W_{2M}^{-m} u(n-M) + W_{2M}^{-m} \sum_{i=n-M+1}^{n} u(i) W_{2M}^{m(n-i)} \end{aligned}$$

de forma que multiplicando la anterior por W_{2M}^{m} y sustrayendo el resultado de (113) se tiene que

$$A_m^{(1)}(n) = W_{2M}^{m} A_m^{(1)}(n-1) + u(n) - (-1)^m u(n-M), \quad m = 0, 1, \cdots, M-1 \tag{114}$$

- Considerando ahora el cálculo recursivo de la segunda DFT $A_m^{(2)}$ definida en la ecuación (112) y separando la muestra $u(n)$ (teniendo en cuenta que $W_{2M}^{2mM} = 1$), es posible expresar esta DFT como

$$A_m^{(2)}(n) = W_{2M}^{-m} u(n) + W_{2M}^{-m} \sum_{i=n-M+1}^{n-1} u(i) W_{2M}^{m(i-n)}$$

- Evaluando $A_m^{(2)}(n-1)$ se obtiene

$$\begin{aligned} A_m^{(2)}(n-1) &= \sum_{i=n-M}^{n-1} u(i) W_{2M}^{m(n-i)} \\ &= W_{2M}^{mM} u(n-M) + \sum_{i=n-M+1}^{n-1} u(i) W_{2M}^{m(n-i)} \\ &= (-1)^m u(n-M) + \sum_{i=n-M+1}^{n-1} u(i) W_{2M}^{m(n-i)} \end{aligned} \quad (115)$$

tal que multiplicando este resultado por W_{2M}^{-m} y luego sustrayendo el resultado de (115) se obtiene

$$A_m^{(2)}(n) = W_{2M}^{-m} A_m^{(2)}(n-1) + W_{2M}^{-m}(u(n) - (-1)^m u(n-M)), \quad m = 0, 1, \cdots, M-1 \quad (116)$$

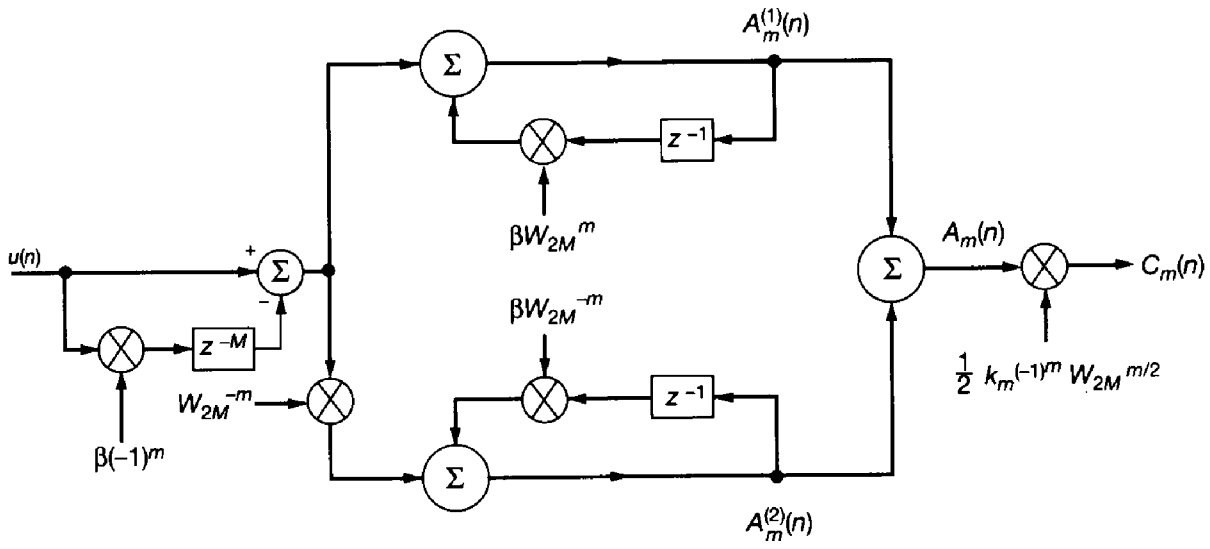

Figura 35: Cálculo indirecto de la DCT desplazable.

- La construcción de la figura 35 ha sido simplificada notando que
  - Las operaciones que involucran $u(n)$ y $u(n-M)$ son comunes a las DFT $A_m^{(1)}(m)$ y $A_m^{(2)}(m)$, por ello la terminación común de la figura 35.
  - z^{-M} en el camino directo y z^{-1} dentro de los lazos de realimentación en la figura 35 se multiplican por un nuevo parámetro β.
- El sistema discreto de la figura 35 se denomina **filtro de muestreo en frecuencia**. Exibe una forma de simetría estructural inherente de la simetría matemática asociadaa la definición de la transformada discreta coseno.
- La función transferencia del filtro mostrado en la figura 35 desde la entrada $u(n)$ a la DCT de salida $C_m(n)$, está dada por (con $\beta = 1$)

$$H_m(z) = \frac{1}{2}k_m\left(e^{\left(-j\frac{m\pi}{2M}\right)}\frac{(-1)^m - z^{-M}}{1 - e^{\left(j\frac{m\pi}{M}\right)}z^{-1}} + e^{\left(j\frac{m\pi}{2M}\right)}\frac{(-1)^m - z^{-M}}{1 - e^{\left(-j\frac{m\pi}{M}\right)}z^{-1}}\right)$$

- El numerador común de esta ecuación$((-1)^m - z^{-M})$ representa un conjunto de ceros uniformemente espaciados sobre el círculo unitario del plano z, dados por

$$z_m = e^{\left(j\frac{m\pi}{M}\right)}, \quad m = 0, \pm 1, \cdots, \pm(M-1)$$

- El denominador del primer término de la derecha tiene un polo simple en $e^{\left(j\frac{m\pi}{M}\right)}$, mientras que el denominador del segundo término de la derecha tiene un polo simple en $e^{\left(-j\frac{m\pi}{M}\right)}$. De acuerdo a esto, estos polos se cancelan con un cero particular del término del numerador. El resultado neto es que el filtro de la figura 35

es equivalente a dos bancos de filtros todo cero de banda estrecha operando en paralelo. Cada banco de filtros corresponde a M bins de la DCT.

- Con $\beta = 1$, los filtros de muestreo en frecuencia son marginalmente estables debido a que para cada bin de la DCT los polos de los dos caminos de realimentación de la figura 35 caen exactamente sobre el círculo unitário, y los errores de redondeo (aunque pequeños pueden dar lugar a inestabilidad. Este problema puede reducirse desplazando los ceros del camino directo y los polos del camino de realimentación hacia adentro del círculo unitário, por ello la inclusión del parámetro β $(0 < \beta < 1)$.

Estimación de autovalores

- El único aspecto que debe aún ser considerado en el diseño del algoritmo LMS-DCT es como estimar los autovalores de la matriz correlación $\boldsymbol{R}$, que define los factores de convergencia. Suponiendo que el proceso estacionario asociado a la entrada es **ergódico**, es posible estimar $\boldsymbol{R}$ con

$$\hat{\boldsymbol{R}}(n) = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{u}(i)\boldsymbol{u}^T(i)$$

conocida como **matriz autocorrelación muestral**. Los coeficientes de la DCT proveen una aproximación a la matriz $\boldsymbol{Q}$ $(M \times M)$ cuyas columnas representan los autovectores asociados con los autovalores de $\boldsymbol{R}$. Si $\hat{\boldsymbol{Q}}$ es esta aproximación, entonces el vector de salidas producida por la DCT $C_0(n), C_1(n), \cdots, C_{M-1}(n)$ puede expresarse como

$$\begin{aligned} \hat{\boldsymbol{v}}(n) &= [C_0(n), C_1(n), \cdots, C_{M-1}(n)]^T \\ &= \hat{\boldsymbol{Q}}\,\boldsymbol{u}(n) \end{aligned}$$

- Además, la aproximación de la DCT a una transformación ortogonal puede escribirse como

$$\begin{aligned}\hat{\boldsymbol{\Lambda}}(n) &= \hat{\boldsymbol{Q}}\,\hat{\boldsymbol{R}}\,\hat{\boldsymbol{Q}}^T \\ &= \frac{1}{n}\sum_{i=1}^{n}\hat{\boldsymbol{Q}}\,\boldsymbol{u}(i)\boldsymbol{u}^T(i)\hat{\boldsymbol{Q}}^T = \frac{1}{n}\sum_{i=1}^{n}\hat{\boldsymbol{v}}(i)\hat{\boldsymbol{v}}^T(i)\end{aligned}$$

o en forma escalar

$$\hat{\lambda}_m(n) = \frac{1}{n}\sum_{i=1}^{n} C_m^2(i), \quad m = 0, 1, \cdots, M-1$$

- Esta ecuación puede reescribirse en forma recursiva teniendo en cuenta que

$$\begin{aligned}\hat{\lambda}_m(n) &= \frac{1}{n}C_m^2(n) + \frac{1}{n}\sum_{i=1}^{n-1} C_m^2(i) \\ &= \frac{1}{n}C_m^2(n) + \frac{n-1}{n}\frac{1}{n-1}\sum_{i=1}^{n-1} C_m^2(i)\end{aligned}$$

y como $\hat{\lambda}_m(n-1) = \frac{1}{n-1}\sum_{i=1}^{n-1} C_m^2(i)$, se tendrá que

$$\hat{\lambda}_m(n) = \hat{\lambda}_m(n-1) + \frac{1}{n}(C_m^2(i) - \hat{\lambda}_m(n-1))$$

- Esta ecuación se aplica al contexto de un proceso ESA. En el caso de filtrado adaptivo en un contexto no estacionario es posible usar el parámetro γ $(0 < \gamma < 1)$ lo que modifica la ecuación anterior a

$$\hat{\lambda}_m(n) = \gamma\hat{\lambda}_m(n-1) + \frac{1}{n}(C_m^2(i) - \gamma\hat{\lambda}_m(n-1)), \quad m = 0, 1, \cdots, M-1$$

Resúmen del algoritmo LMS-DCT

Inicialización

Para $m = 0, 1, \cdots, M-1$ hacer

$$A_m^{(1)}(0) = A_m^{(2)}(0) = 0$$
$$\hat{\lambda}_m(0) = 0$$
$$\hat{w}_m(0) = 0$$
$$k_m = \begin{cases} 1/\sqrt{2}, & m = 0 \\ 1, & \text{de otra forma} \end{cases}$$

Selección de parámetros

$$\alpha = \frac{1}{2M}$$
$$\beta = 0.99$$
$$0 < \gamma < 1$$

DCT desplazable

Para $m = 0, 1, \cdots, M-1$ y $n = 1, 2, \cdots,$ hacer

$$A_m^{(1)}(n) = \beta W_{2M}^m A_m^{(1)}(n-1) + u(n) - \beta(-1)^m u(n-M)$$
$$A_m^{(2)}(n) = \beta W_{2M}^m A_m^{(2)}(n-1) + W_{2M}^{-m}(u(n) - \beta(-1)^m u(n-M))$$
$$A_m(n) = A_m^{(1)}(n) + A_m^{(2)}(n)$$
$$C_m(n) = \tfrac{1}{2} k_m (-1)^m W_{2M}^{m/2} A_m(n),$$

donde $W_{2M} = \exp\left(-j\frac{2\pi}{2M}\right)$

Algoritmo LMS

$$y(n) = \sum_{m=0}^{M-1} C_m(n)\hat{w}_m(n)$$
$$e(n) = d(n) - y(n)$$
$$\hat{\lambda}_m(n) = \hat{\lambda}_m(n-1) + \tfrac{1}{n}(C_m^2(i) - \hat{\lambda}_m(n-1))$$
$$\hat{w}_m(n+1) = \hat{w}_m(n) + \frac{\alpha}{\hat{\lambda}_m(n)} C_m e(n)$$

8.5 Clasificación de algoritmos

En base a los algoritmo discutidos en esta sección y en la previa y adelantando un poco el material a ser discutido posteriormente, clasificaremos los algoritmos como mostrado en la siguiente tabla

Clase	Adaptación por muestras	Adaptación por bloques
Gradiente estocástico	LMS	LMS en bloques
Con preprocesamiento	LMS-DCT; GAL	SOBAF
Cuadrados mínimos	RLS	LS en bloques

donde se han identificado dos clases de algoritmos de filtrado adaptivo:

- algoritmos de gradiente estocástico.
- algoritmos de cuadrados mínimos exactos.
- Los algoritmos de gradiente estocástico incluyen a los algoritmos con preprocesamiento como subclase. En cada caso, existen dos formas básicas de adaptar los parámetros del algoritmo:
  - en base a muestras (secuencialmente).
  - en base a bloques de datos.
- El algoritmo LMS estandar (con adaptación en base a muestras) y el algoritmo LMS en bloques (con adaptación por bloques) son algoritmos de tipo gradiente estocástico. El uso de la **FFT para convolución** provee un método eficiente para implementar el algoritmo LMS en bloques. Esta es una de las formas en las cuales la transformada de Fourier puede utilizarse para implementar filtrado adaptivo lineal.
- Otra forma en la cual la transformada de Fourier juega un papel útil es para **ortogonalización**, como ejemplificado en el algoritmo LMS-DCT. Especificamente, la transformada discreta

coseno provee un método en el domínio frecuencia para aproximar un conjunto ortogonal de muestras. El algoritmo LMS-DCT utiliza una adaptación por muestras. En contraste, en el **filtro adaptivo en bloques ortogonalizador** (SOBAF) la adaptación es por bloques. Además, el algoritmo LMS en bloques eficiente puede interpretarse como una forma de aproximación al SOBAF; la secuencia P_i es una estimación de la densidad de potencia espectral del vector de entrada $\boldsymbol{u}(n)$.

- En el algoritmo LMS-DCT la ortogonalización de la entrada se aproxima en el domínio frecuencia a través de una descomposición en autovalores. Por otro lado, en el algoritmo **lattice adaptivo de gradiente** (GAL), la ortogonalización de los datos de entrada se realiza en el domínio tiempo a través de la factorización de Cholesky. A pesar que estos algoritmos aproximan una ortogonalización de los datos de entrada en diferentes formas, son similares en el sentido de imponer una matriz de estructura Toeplitz (implícita ó explícita) sobre la estimación de $\boldsymbol{R}$. Colocado de otra manera, las discusiones asociadas a ambos algoritmos tienen sus raíces en la teoría de procesos estocásticos estacionarios. La suposición Toeplitz también se aplica al algoritmo SOBAF.

- Los algoritmos de filtrado adaptivo lineal asociados a la familia de **cuadrados mínimos**, ejemplificada por el algoritmo de **cuadrados mínimos recursivo** (RLS) con adaptación por muestras y el algoritmo de **cuadrados mínimos en bloques** se obtiene una ortogonalización **exacta** de los datos de entrada en el domínio tiempo. En otras palabras, no se realiza ninguna aproximación en la obtención de esos algoritmos, de donde puede concluirse la rápida velocidad de convergencia y otras importantes propiedades que caracterizan a estos algoritmos.