Filtros adaptativos

José Antonio Morán Moreno
Joan Claudi Socoró Carrié

PID_00175663

UOC
Universitat Oberta de Catalunya

Índice

1. Introducción al filtrado adaptativo

En este módulo nos introduciremos en el fascinante mundo del filtrado adaptativo. La vida es dinámica, y la capacidad del hombre para adaptarse marca los principios de la supervivencia. En las aplicaciones de procesado de señal, los escenarios donde se aplican estos sistemas son también escenarios dinámicos que sufren cambios a lo largo del tiempo. No basta, pues, con desarrollar algoritmos óptimos desde el punto de vista frecuencial o estadístico, sino que resulta fundamental conseguir que dichos sistemas puedan ir adaptando su comportamiento a las necesidades específicas del sistema para cada momento.

El filtrado adaptativo determina la teoría de los sistemas de filtrado que son capaces de adaptar sus condiciones a los cambios de las características del entorno. Tal y como se ha visto en el filtrado de Wiener, la solución óptima del filtro pasa por la estimación de unos parámetros estadísticos del sistema. La estimación de estos parámetros de correlación se puede realizar de dos formas:

Ved también

Hemos visto el filtrado de Wiener en el apartado 5 del módulo "Filtrado lineal óptimo" de esta asignatura.

- **Procesado por bloques:** En este caso concreto, se determina una ventana de trabajo de *N* muestras donde se procede a realizar la estimación de los parámetros y a resolver la solución óptima del sistema para ese bloque. Una vez finalizado, si queremos que el sistema tenga capacidad de adaptación, debemos proceder al análisis del siguiente bloque de datos y al recálculo de la solución óptima. Estos métodos son apropiados cuando el número de datos que se procesan es finito y se cumple una cierta estacionariedad en las características de las señales durante la duración del bloque.

- **Procedimientos de filtrado adaptativo:** Los filtros adaptativos son soluciones apropiadas cuando no se dispone de bloque de datos para analizar y el sistema debe ir trabajando en tiempo real y especialmente cuando nos encontramos en entornos poco estacionarios. Si las características de las señales varían en el tiempo, la única posibilidad que nos queda es recurrir a métodos que dispongan de una capacidad de adaptación y seguimiento del entorno.

El número de aplicaciones donde el uso del filtrado adaptativo puede estar justificado es muy extenso, especialmente cuando nos encontramos trabajando en escenarios reales donde las características varían o evolucionan. Seguidamente se muestran algunos ejemplos de escenarios donde el filtrado adaptativo es la solución óptima:

- **Cancelación de ruido en aplicaciones de voz**: Tal y como se comentó en el módulo "Filtrado lineal óptimo", las señales de voz no son estacionarias. El patrón estadístico depende en gran medida del tipo de fonemas de la

locución, de modo que un sistema con la capacidad de adaptarse al patrón de voz será una excelente solución en este tipo de aplicaciones.

- **Cancelación de interferencias en antenas**: Los entornos de comunicación avanzados, como por ejemplo la telefonía móvil, supone en reto para el diseñador al encontrarnos con canales extremadamente variables y muy difíciles de modelar. El escenario de transmisión multicamino variante en el tiempo supone un reto en el diseño de los receptores. Los filtros adaptativos son excelentes soluciones para reducir las interferencias y ecualizar el canal en este tipo de entornos.

- **Ecualización adaptativa**: Incluso en el caso de los canales más simples, es bien sabido por el ingeniero de telecomunicación que la señal recibida experimenta distorsiones debido al canal de comunicación. Cuando se intenta aprovechar al máximo la capacidad del canal y llegar a los límites de comunicación marcados por Shannon, es necesario recurrir a sistemas avanzados de ecualización de canal. Este tipo de sistemas permitirán corregir en la medida de lo posible el efecto distorsionador del canal de comunicaciones y reconstruir de la forma más óptima posible la señal enviada.

- **Aplicaciones biomédicas**: El campo de la biomedicina ha experimentado grandes avances en los últimos años. El análisis de las señales biomédicas son un campo de gran interés para el diagnóstico y tratamiento de muchas patologías. El procesado óptimo es la base de los algoritmos sobre los que los equipos biomédicos pueden aislar señales y mejorar su calidad para obtener resultados óptimos en las pruebas.

- **Eliminación de interferencias en aplicaciones de imagen**: El procesado óptimo no siempre se realiza sobre variaciones en el dominio temporal, sino que puede ser muy útil también en aplicaciones en el dominio del espacio, como por ejemplo en el caso de imágenes digitales. La imagen distorsionada tomada sobre un objetivo en movimiento puede mejorarse al estimar el efecto del movimiento sobre la toma y ecualizándolo para obtener una imagen más nítida.

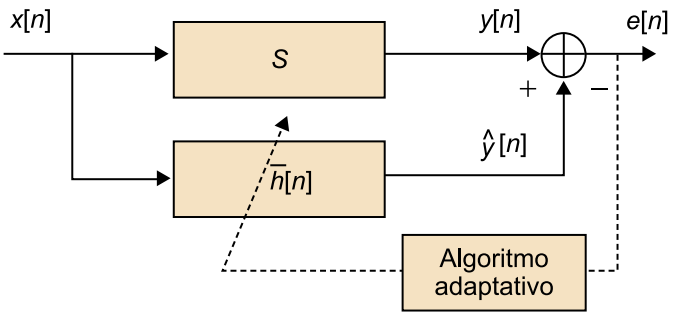
Como se ha podido observar, el ámbito del filtrado adaptativo y sus aplicaciones es un campo diverso y útil en diferentes disciplinas. El procesado de señales ha entrado tan de lleno en las aplicaciones cotidianas que en nuestras manos, cuando manejamos un Smartphone, tenemos un excelente ejemplo de pruebas de aplicaciones de procesado de señal adaptativas que se utilizan en un ámbito cada vez mayor.

2. Estructura general de un filtro adaptativo

En este apartado veremos a grandes rasgos algunos de los escenarios donde se aplican los filtros adaptativos. Será importante interiorizar estos escenarios para facilitar la posterior comprensión de los conceptos implicados en el desarrollo de este tipo de sistemas.

De forma genérica podemos decir que un filtro adaptativo puede representarse por un esquema como el de la figura 1.

Figura 1



Tenemos un sistema S que responde de una determinada forma a una señal de entrada $x[n]$ produciendo una salida $y[n]$. El objetivo del filtro adaptativo en este caso sería realizar una estimación de una respuesta impulsional lineal que se aproxime de forma óptima al comportamiento del sistema.

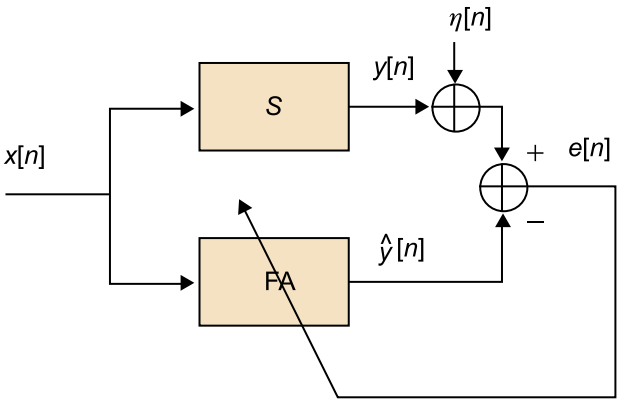
El sistema estará de entrada caracterizado por unos valores determinados antes de iniciarse el funcionamiento. En ausencia de más información se podría inicializar el sistema con una respuesta impulsional en forma de $d[n]$ o con todos sus coeficientes a cero. A medida que van entrando muestras al sistema, se procede a evaluar la señal de error $e[n]$, y está información, convenientemente utilizada, será la que alimentará el algoritmo adaptativo que muestra a muestra realizará una modificación de los coeficientes de la respuesta impulsional hasta conseguir estabilizarse en la solución óptima. Si el escenario es estacionario, una vez llegados a la solución óptima, el filtro tenderá a mantener sus coeficientes constantes. En cambio, en el caso de escenarios variantes en el tiempo, el filtro irá modificando sus coeficientes para seguir las variaciones del sistema.

A continuación veremos algunas aplicaciones con variaciones en la estructura del filtro con diferentes campos de aplicación.

2.1. Identificación de sistemas

El primer esquema (figura 2) corresponde al ejemplo base explicado anteriormente. Esta estructura será útil cuando tengamos la necesidad de modelar cualquier tipo de sistema. Si imaginamos por ejemplo que el sistema *S* es un motor, este esquema serviría para proporcionarnos un modelo lineal de su comportamiento. Este modelo podría utilizarse posteriormente para realizar un controlador óptimo para la velocidad de rotación del motor.

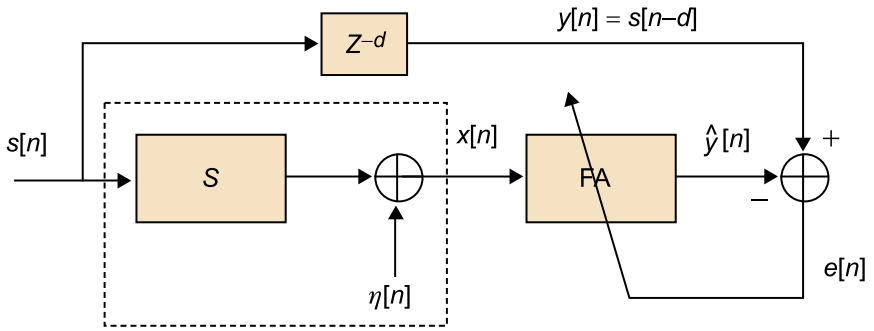
Figura 2



2.2. Ecualización de canal

La figura 3 corresponde al esquema de una aplicación de ecualización de canal. Todo sistema de transmisión de datos busca tener el canal óptimo en su transmisión. El proceso de ecualización de canal pretende deshacer en la medida de lo posible los efectos distorsionantes del canal en la transmisión. El canal se puede modelar de forma simple como un sistema *S* que distorsiona la señal de entrada, más una adicción de un ruido interferente, tal y como se observa en la figura 3. El filtro adaptativo, FA, intentará deshacer los efectos distorsionantes para conseguir obtener a la salida una señal igual a la que teníamos en la entrada.

Figura 3



En este ejemplo concreto es importante remarcar la necesidad del retardo de la señal de referencia del filtro que corresponde a la señal de entrada con un retardo de *d* muestras. Este retardo es necesario para compensar el retardo de

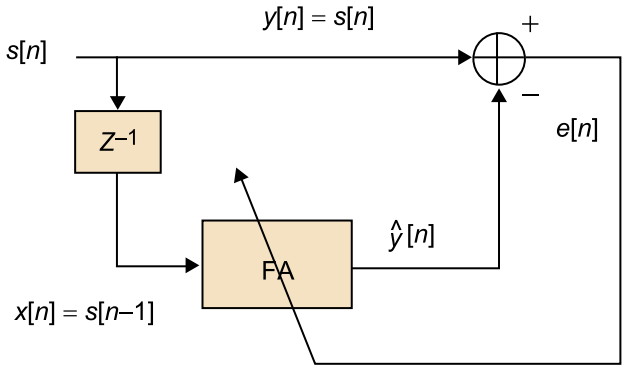
transmisión del canal, ya que, de no introducirlo, al filtro adaptativo le sería imposible reconstruir la señal de entrada, puesto que tendría que ser un sistema no causal.

La otra cuestión importante de este esquema está en ver que para realizar el entrenamiento es importante que se utilice una secuencia de datos conocida por el receptor. En los sistemas de ecualizado adaptativo, el protocolo incluye una etapa de *training* y otra de *tracking*. En la fase de *training* (o entrenamiento rápido) el sistema utiliza una secuencia conocida que le permita obtener una aproximación inicial de calidad del canal. Una vez finalizada esta fase se procede a una etapa de *tracking* donde el sistema simplemente realizar un seguimiento de las pequeñas variaciones que surjan a partir de ese momento.

2.3. Predicción lineal

Un sistema de predicción lineal es de utilidad en un gran número de aplicaciones. La codificación de voz, la evolución de los parámetros financieros o meteorológicos, los compresores de datos y otros, son campos de aplicación de los sistemas de predicción lineal. Para el hombre siempre ha sido importante anticiparse a los acontecimientos futuros, y los sistemas de predicción son sistemas que permiten predecir el valor futuro de una serie temporal.

Figura 4



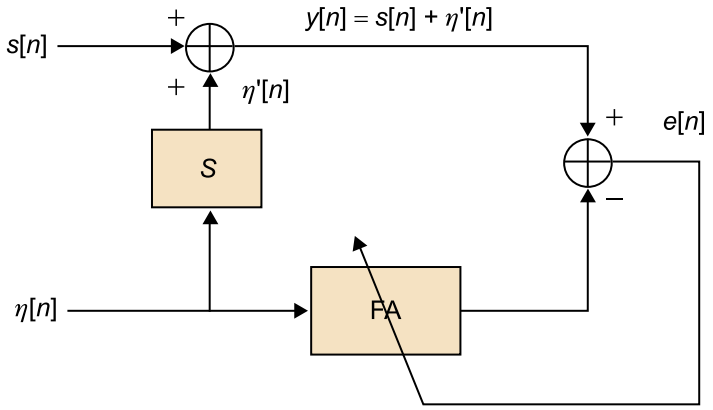
En el esquema de la figura 4 se observa cómo el F.A. se alimenta con una versión retardada de la señal objetivo y se entrena para obtener la señal objetivo como referencia. Si el entrenamiento es correcto, el sistema habrá sido capaz de determinar los coeficientes que minimizan el error de predicción con los datos de entrenamiento, de tal forma que esos coeficientes se podrán utilizar para realizar estimaciones de predicción en el futuro.

2.4. Cancelación de ruido

En esta aplicación (figura 5), el objetivo es eliminar la información que contamina la señal deseada $s[n]$. El ejemplo típico de una aplicación de este tipo puede ser la de la conversación en un helicóptero, donde la señal interferente del motor tiene una intensidad elevada que distorsiona sobremanera la voz deseada. El ruido del motor se transforma dentro del habitáculo y se suma a la

señal deseada. El sistema de entrenamiento utiliza el ruido del motor como señal de entrada y pretende obtener $y[n]$ a la salida. La señal $y[n]$ está compuesta por dos componentes, la voz del piloto y la componente de ruido. Dado que el filtro solo podrá establecer una predicción estadística sobre la componente de ruido, la señal de error después de la convergencia del sistema será la voz del piloto, $s[n]$.

Figura 5



En este caso también se ilustra otro de los aspectos importantes del filtrado óptimo. Que el filtro sea óptimo no implica que el error sea muy pequeño. En este caso, la señal de error es la señal deseada y tiene más potencia que el ruido, pero el diseño del sistema ha permitido cancelar únicamente la parte de señal correlada estadísticamente con el ruido, dejando la voz del piloto casi intacta al no existir correlación estadística entre la voz del piloto y el ruido.

Ved también

Podéis ver el filtrado óptimo en el módulo "Filtrado lineal óptimo" de esta asignatura.

3. El filtro LMS

En el campo del filtrado adaptativo, el filtro LMS (*least mean square*) representa una de las estructuras de referencia en el análisis y comportamiento de los filtros adaptativos. Existen tres parámetros que controlan el comportamiento de un filtro adaptativo y que son las principales componentes que tiene que tener en cuenta el diseñador del sistema en el momento de escoger un tipo de filtrado u otro:

- **Velocidad de convergencia del filtro**: La velocidad de convergencia determina la capacidad con la que un filtro adaptativo es capaz de aproximarse a la solución óptima del sistema. Como veremos, las diferentes estructuras de filtros adaptativos presentan diferentes velocidades de convergencia.

- **Coste computacional del sistema**: Los algoritmos de cálculo de filtros adaptativos presentan un coste computacional diferente según el tipo de estructura con la que se trabaje. Este factor es fundamental en las aplicaciones de tiempo real, puesto que un sistema muy eficiente pero con un coste computacional muy elevado puede no resultar la solución óptima en una aplicación concreta. El diseñador debe evaluar la relación entre las prestaciones del sistema y el coste de su implementación, puesto que un coste computacional más elevado implicará el uso de un hardware de mayor coste que encarecerá el producto final. El diseñador debe evaluar estos aspectos para seleccionar la solución más apropiada.

- **Estabilidad del algoritmo**: Una vez el algoritmo converge a la solución óptima del filtro, debe ser capaz de mantenerse en esa solución y ser lo más robusto posible ante cambios en las señales. Las variaciones de las señales de entrada así como el hecho de trabajar con precisión finita en aplicaciones hardware hace que haya algoritmos que se puedan volver inestables simplemente por la falta de precisión numérica de un procesador. Evidentemente, este factor tiene un impacto cada vez inferior debido a la potencia de las estructuras de hardware actuales, pero en aplicaciones de bajo coste puede ser un factor importante a tener en cuenta.

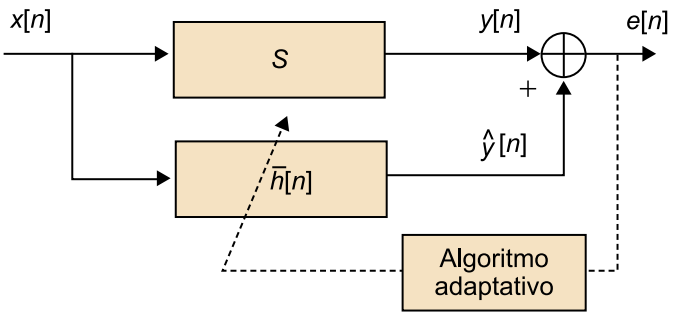
El filtro LMS por sus características es una solución que se adapta a un gran número de aplicaciones reales. Por una parte presenta un coste computacional extremadamente reducido, y el algoritmo presenta una buena estabilidad si se mantienen las condiciones de diseño apropiadas. El único inconveniente que se le puede encontrar a este algoritmo es su velocidad de convergencia, que según las características de las señales puede ralentizar el sistema en algunas aplicaciones.

3.1. Estructura del método LMS

El filtro LMS es la versión más sencilla de filtrado adaptativo que podemos encontrar en la bibliografía. Se trata de una solución iterativa del filtro óptimo de Wiener por el método del gradiente descendente (*steepest descent*) que permite desarrollar un algoritmo adaptativo de baja complejidad y gran utilidad en aplicaciones de procesado, siendo posiblemente una de las estructuras más utilizadas en aplicaciones reales.

Para la presentación del algoritmo recordaremos el esquema básico de filtrado adaptativo y la función de coste por minimizar en el caso del filtrado lineal óptimo o filtro de Wiener.

Figura 6



La idea del filtrado lineal óptimo es conseguir determinar el valor de los parámetros óptimos del filtro que minimicen la función de coste definida según:

$$J = E\left[\left(y[n] - \hat{y}[n]\right)^2\right] = E[e^2[n]]$$

3.2. El método del gradiente descendente

La función de coste definida en aplicaciones de filtrado lineal óptimo está definida según el error cuadrático medio. Conviene conocer con algo más de detalle las características de esta función de error, ya que ello nos aportará información relevante de cara al desarrollo del algoritmo LMS así como para comprender las posibilidades de aceleración del mismo que llevarán a estructuras que se explicarán en los siguientes subapartados.

Para el desarrollo de estas propiedades conviene repasar algunas operaciones vectoriales que nos permitirán justificar matemáticamente la función de error y comprender los aspectos más importantes de este algoritmo. No obstante, no se pretende abusar del desarrollo matemático sino simplemente aprender aquellos aspectos que realmente sean relevantes para la comprensión del mismo. Con el fin de simplificar la operatoria, y dado que ello no resta relevancia a los conceptos clave, realizaremos el desarrollo utilizando el caso de señales reales puesto que ayudará a simplificar el análisis.

Operando sobre la función de coste obtenemos que la función de error corresponde a la siguiente ecuación:

$$J = E\left[(y[n] - \hat{y}[n])^2\right] = E\left[\left(y[n] - \vec{h}^T\vec{x}\right)\left(y[n] - \vec{h}^T\vec{x}\right)^T\right]$$
$$= E\left[y^2[n]\right] - \vec{h}^T E\left[\vec{x}\,y[n]\right] - E\left[y[n]\vec{x}^T\right]\vec{h} + \vec{h}^T E\left[\vec{x}\vec{x}^T\right]\vec{h}$$

Los términos de esta ecuación se pueden relacionar con unos estadísticos que ya conocéis como son el vector de correlación cruzada y la matriz de autocorrelación:

Ved también

Podéis ver los diversos estadísticos que se tratan en el módulo "Filtrado lineal óptimo" de esta asignatura.

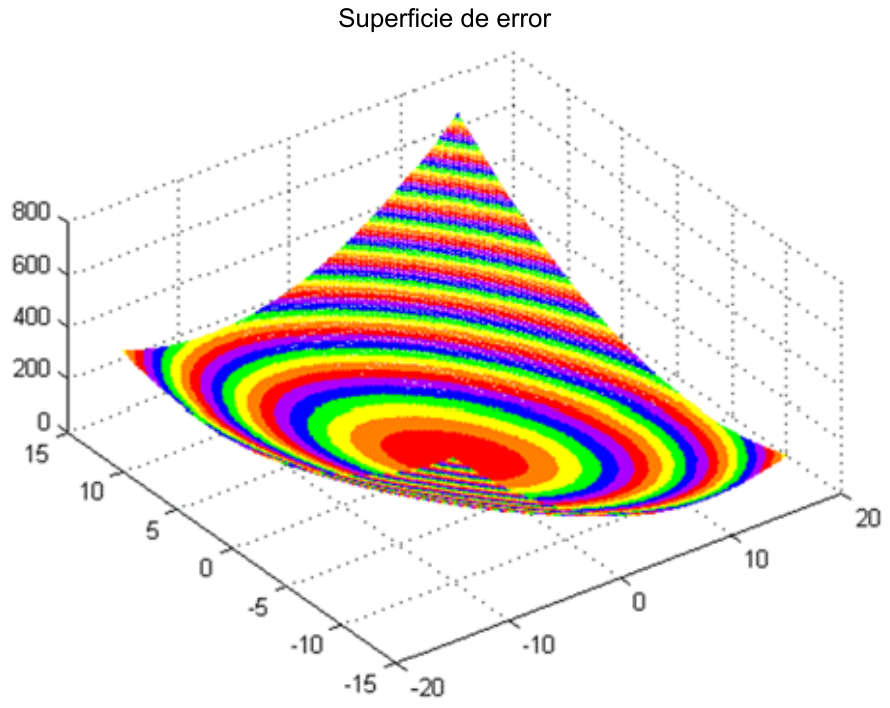
$$J = \sigma_y^2 - 2\vec{h}^T\vec{r}_{yx} + \vec{h}^T\bar{\bar{R}}_{xx}\vec{h}$$

Sabemos que la solución óptima de dicha ecuación corresponde a:

$$\vec{h}_{opt} = \bar{\bar{R}}_{xx}^{-1}\vec{r}_{yx}$$

Podemos descomponer la función de error J de modo que sea más fácil interpretar. Como bien sabemos se trata de una función cuadrática, de forma que esperamos que a medida que nos separemos de la solución óptima, el error aumentará de forma cuadrática, pero habrá direcciones donde el incremento será superior y otras donde será más suave.

Figura 7



La representación bidimensional de la función de error para un caso concreto ayudará a interpretar la morfología de la función y su impacto sobre las prestaciones de los filtros diseñados, ya que la adaptación de la solución dependerá de un recorrido a lo largo de esta función hasta encontrar el punto mínimo.

En la figura 7 observamos la forma de la función de error para un caso concreto bidimensional. La manipulación matemática nos permite reescribir la función de coste según:

$$J = J_{MIN} + (\vec{h} - \vec{h}_{opt})^T \bar{R}_{xx} (\vec{h} - \vec{h}_{opt})$$
$$J_{MIN} = \sigma_y^2 - \vec{r}_{yx}^T \bar{R}_{xx}^{-1} \vec{r}_{yx}$$

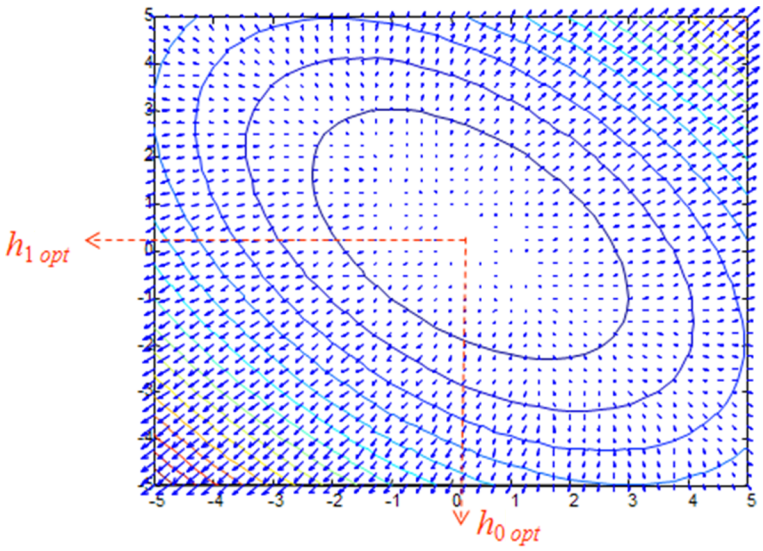
La interpretación de esta ecuación nos indica que, cuando la respuesta es la óptima, tenemos un mínimo de la función de valor J_{MIN} y que, a medida que nos separamos del valor óptimo, el crecimiento es cuadrático pero ponderado por la matriz de correlación. Sabemos que las matrices presentan ganancias diferentes en direcciones diferentes, cosa que corrobora que hay direcciones con un crecimiento mayor que otras.

La matriz de correlación es una matriz simétrica, y por propiedades de matrices sabemos que estas son diagonalizables, es decir, se pueden descomponer en direcciones propias con sus respectivos autovalores de la siguiente forma:

$$J = J_{MIN} + (\bar{h} - \bar{h}_{opt})^T \bar{Q} \quad \bar{\Lambda} \quad \bar{Q}^T (\bar{h} - \bar{h}_{opt})$$

Sabemos que la descomposición en autovectores y autovalores es como obtener las huellas dactilares de la matriz, dado que estas direcciones son relevantes en cuanto al comportamiento de la transformación. Las direcciones propias de la matriz de correlación son justamente los ejes de las elipses que se pueden observar en la función de coste *J*. La dirección que presenta el mayor autovalor será la que presenta un crecimiento mayor mientras que la que presenta el menor autovalor corresponde a la de menor pendiente.

Figura 8



Observamos en la figura 8 cómo el vector gradiente (que indica la dirección de máximo crecimiento de la curva) es cero en el valor óptimo, y va indicando la dirección de crecimiento siendo ortogonal a las curvas de nivel. Se observa que el vector gradiente presenta mayor módulo en el semieje menor de la elipse, indicando un crecimiento mayor de la función de coste y el módulo es más pequeño en la dirección del mayor semieje, indicando justamente lo contrario.

Teniendo en cuenta la representación de la función de error y los vectores gradientes, aplicando sencillamente el sentido común vemos que para encontrar el mínimo de la función nos bastaría con desplazarnos por la curva en sentido contrario al vector gradiente y realizar aproximaciones sucesivas hasta que el gradiente se hiciese cero. Esta idea es la que da origen al método de optimización del gradiente inverso o *steepest descent*.

$$J = \sigma_y^2 - 2\vec{h}^T \vec{r}_{yx} + \vec{h}^T \bar{\bar{R}}_{xx} \vec{h}$$
$$\nabla_{\vec{h}} J = \frac{\partial J}{\partial \vec{h}} = -2(\vec{r}_{yx} - \bar{\bar{R}}_{xx} \vec{h})$$

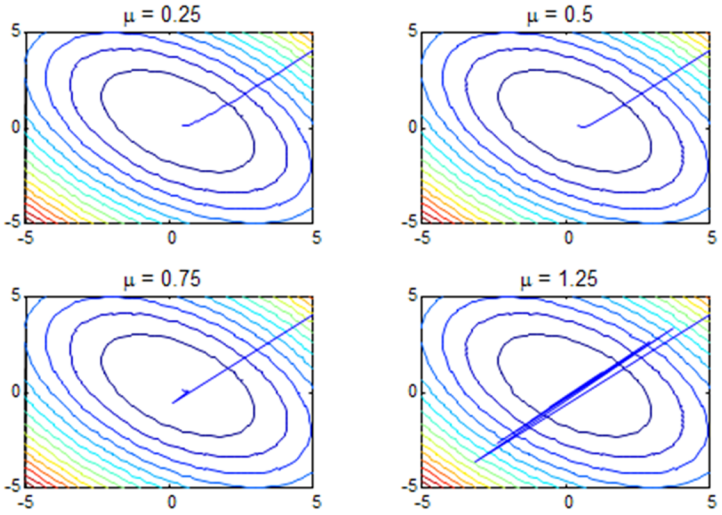
Una vez conocido el valor del gradiente de la curva, la ecuación del filtro adaptativo podría ser:

$$\vec{h}[n+1] = \vec{h}[n] - \mu \nabla_{\vec{h}} J = \vec{h}[n] + \mu(\vec{r}_{yx} - \bar{\bar{R}}_{xx} \vec{h}[n])$$

En la ecuación de adaptación observamos cómo el método del gradiente inverso modifica los coeficientes del filtro en la dirección del gradiente inverso multiplicando el gradiente por un factor de actualización conocido como paso de adaptación μ o *step-size*.

No hay mejor manera de entender el efecto del paso de cuantificación que observando el comportamiento del algoritmo para diferentes valores del coeficiente μ.

Figura 9



La figura 9 muestra cómo a medida que el algoritmo incrementa el valor del parámetro, la rapidez de convergencia del sistema aumenta, siendo necesarias menos iteraciones para alcanzar el mínimo del sistema. No obstante, en la última figura se observa que cuando el valor es demasiado elevado, se salta de un lado de la superficie a la siguiente y no cuesta imaginar que si el valor siguiese creciendo el sistema se volvería inestable y no conseguiría conseguir encontrar el mínimo de la curva.

Con esto podemos observar que el método del gradiente requiere una buena sintonización del parámetro de adaptación a fin de conseguir que el sistema funcione lo más rápido posible sin perder la estabilidad en su respuesta.

Velocidad de convergencia del método del gradiente

Como se ha dicho, uno de los aspectos importantes de los algoritmos adaptativos es la velocidad de convergencia, es decir, la capacidad que tiene el algoritmo en llegar de forma rápida a la solución óptima. Algunas operaciones matemáticas sobre la ecuación de actualización de los coeficientes del filtro permiten reescribir la ecuación de forma que nos permita interpretar la velocidad de convergencia:

$$\vec{h}[n+1] = \vec{h}[n] + \mu(\vec{r}_{yx} - \bar{\bar{R}}_{xx}\vec{h}[n]) = (\bar{\bar{I}} - \mu\bar{\bar{R}}_{xx})\vec{h}[n] + \mu\vec{r}_{yx}$$

$$\vec{h}[n+1] = (\bar{\bar{I}} - \mu\bar{\bar{R}}_{xx})\vec{h}[n] + \mu\bar{\bar{R}}_{xx}\vec{h}_{opt} + \vec{h}_{opt} - \vec{h}_{opt}$$

$$(\vec{h}[n+1] - \vec{h}_{opt}) = (\bar{\bar{I}} - \mu\bar{\bar{R}}_{xx})(\vec{h}[n] - \vec{h}_{opt})$$

Finalmente, si resolvemos esta ecuación de forma iterativa desde la iniciación del sistema, nos quedaría de la siguiente forma:

$$\vec{h}[n] = \vec{h}_{opt} + (\bar{\bar{I}} - \mu\bar{\bar{R}}_{xx})^n(\vec{h}[0] - \vec{h}_{opt})$$

Se observa que en la iteración inicial, es decir, cuando n = 0 estaríamos en las condiciones iniciales y que cuando n tiende a infinito debería anularse el segundo término de la ecuación para que el algoritmo convergiese hacia h_{opt}. Vemos que la ecuación presenta una potencia de una matriz que para que se cumpla la estabilidad del algoritmo, esa potencia en el límite tiene que tender a cero:

$$\lim_{n\to\infty}(\bar{\bar{I}} - \mu\bar{\bar{R}}_{xx})^n = \bar{\bar{0}} \quad \Leftrightarrow \quad |\bar{\bar{I}} - \mu\bar{\bar{R}}_{xx}| < 1$$

Como estamos trabajando con el error cuadrático medio, podemos utilizar la norma euclidiana matricial, que determina que la norma de la matriz corresponde al módulo de su autovalor máximo. Considerando los diferentes autovalores de la matriz, el caso más crítico en cuanto a condiciones de estabilidad lo marcará el autovalor máximo cuando se cumpla la siguiente condición $1 - \mu\lambda_{max} = -1$ que marcaría el valor máximo de μ para garantizar la estabilidad en:

$$\mu \leq \frac{2}{\lambda_{\max}}$$

Evidentemente este valor estaría en el límite de la inestabilidad, así que no sería un valor apropiado por condiciones de diseño. En este caso sería mucho mejor que escogiésemos un valor más prudente como por ejemplo:

$$\mu \leq \frac{\beta}{\lambda_{\max}}$$

Siendo beta un parámetro de valor entre 0 y 1.

Bajo esta premisa de diseño podemos realizar una aproximación a la velocidad de convergencia del sistema así como a los parámetros de las funciones de entrada que influyen en este aspecto. Para las condiciones del diseño bajo consideración, se puede garantizar que:

$$(1 - \mu\lambda_k)^n \leq (1 - \mu\lambda_{\min})^n$$

De tal forma que la velocidad de convergencia del sistema se puede aproximar según el teorema de Taylor a una exponencial decreciente del tipo:

$$\left(1-\frac{\beta\lambda_{\min}}{\lambda_{\max}}\right)^{n} \cong e^{-\frac{\beta\lambda_{\min}}{\lambda_{\max}}n}$$

Ahora sí que estamos en condiciones de realizar una evaluación de los aspectos del sistema que influyen en la velocidad de convergencia del algoritmo.

Tal y como habíamos visto anteriormente, la constante μ es determinante en la velocidad y la deberemos adaptar para conseguir máximas prestaciones, pero existe otro aspecto importante que es la dispersión de autovalores. Una vez determinado el valor del parámetro beta, la dispersión en los autovalores de la matriz de correlación será un factor que provocará un enlentecimiento del comportamiento del algoritmo.

¿Cómo afectará la información de la señal de entrada a la dispersión de autovalores?

Esta pregunta será clave para poder determinar qué tipo de señales provocarán velocidades de convergencia rápidas y qué tipo de señales provocarán velocidades de convergencia lentas. En el caso de señales incorreladas la matriz de correlación será la identidad, y podemos deducir fácilmente que en este caso todos los autovalores tendrán el mismo valor. Si lo relacionamos con las curvas de nivel en dos dimensiones, las curvas de nivel serían circunferencias y la superficie tendría el mismo crecimiento en cualquiera de sus direcciones.

A medida que las señales de entrada empiezan a estar más correladas, las curvas de nivel tienden a formas elípticas cada vez más aplanadas. Veremos entonces que habrá una dirección con una pendiente mucho más pronunciada (la de mayor autovalor) y otra que será mucho más suave. El diseño del algoritmo lo deberemos realizar para garantizar la convergencia en la dirección de máxima pendiente, de ahí que el valor de μ esté limitado por el autovalor máximo, pero la velocidad de convergencia más lenta la tendremos cuando vengamos por la dirección de mínima pendiente.

Visto esto podemos intuir que cuando las señales son fuertemente correladas el algoritmo tenderá a presentar una velocidad de convergencia mucho más lenta. Esto adquiere sentido también si lo miramos en el dominio temporal, dado que cuando una señal está fuertemente correlada, cada muestra nueva aporta muy poca información que no estuviese contenida en las muestras anteriores, con lo que al algoritmo le costará más tiempo determinar en qué dirección debe modificar los coeficientes. En cambio, cuando la señal de entrada está incorrelada, cada muestra nueva aporta información que no estaba contenida en las muestras anteriores. Al haber este aporte de información nueva de forma mucho más clara, el algoritmo podrá evolucionar rápidamente hacia el mínimo de la función.

Más adelante veremos que trabajar con versiones incorreladas de las señales de entrada es la clave para el desarrollo de algoritmos adaptativos de gran velocidad de convergencia. La decorrelación clarifica la forma en la que se muestra la información e impide la confusión generada con vectores altamente correlados donde resulta complejo extraer la información nueva aportada por cada muestra.

3.3. El método LMS

Una vez analizada la estructura básica del método LMS, así como la conceptualización del método del gradiente, llega el momento de diseñar el algoritmo adaptativo del método LMS que permitirá su aplicación con señales reales.

Es importante remarcar que el método del gradiente descendente es útil para comprender los conceptos del LMS, pero que no se utiliza en la práctica puesto que si ya conocemos los valores de correlación cruzada y la matriz de autocorrelación podemos directamente resolver el sistema y obtener la solución óptima de forma directa. La gracia de un método adaptativo es que las modificaciones de los coeficientes del filtro se realicen mediante operaciones sencillas de los valores de entrada y sin la necesidad de estimar vectores y matrices de correlación.

El cálculo del vector gradiente se ha realizado en subapartados anteriores, de forma que no volveremos a repetir nuevamente la operatoria y sabemos que corresponde a la siguiente ecuación:

$$\nabla_{\vec{h}} J = \frac{\partial J}{\partial \vec{h}} = -2E[\vec{x}[n]e[n]] = -2E[y[n]\vec{x}[n]] + 2E[\vec{x}[n]\vec{x}[n]^T]\,\vec{h}[n]$$

Donde el primer término de la ecuación corresponde al vector de correlación cruzada entre la señal objetivo y las muestras de entrada y el segundo término a la matriz de autocorrelación.

El método LMS consiste en una idea muy sencilla que no es más que aproximar el cálculo del vector gradiente mediante un estimador que haga servir solo las muestras actuales.

$$\frac{\partial J}{\partial \vec{h}} = -2E[\vec{x}[n]e[n]] \approx -2\vec{x}[n]e[n]$$

Podemos pensar que esta aproximación es muy pobre para obtener un algoritmo con buenos resultados, pero se puede demostrar matemáticamente que el comportamiento estadístico del algoritmo es exactamente el mismo que el del *steepest descent* o gradiente descendente:

$$E\left[\vec{h}[n+1]\right] = E\left[\vec{h}[n] + \mu e[n]\vec{x}[n]\right]$$
$$= E\left[\vec{h}[n]\right] + \mu E\left[e[n]\vec{x}[n]\right]$$

La principal diferencia es que la evolución del algoritmo se realizará de forma más ruidosa, es decir, con más varianza que la que tendríamos en el método del gradiente, pero que finalmente los dos algoritmos nos llevarían a la misma solución. La varianza de la solución óptima se puede estimar y arroja el siguiente resultado:

$$\lim_{n\to\infty} E\left[\| \vec{h}[n] - \vec{h}_{opt} \|_2^2\right] = \mu J_{\min}$$

No entraremos en los detalles del cálculo de la misma, pero es importante que nos quedemos con la idea, obvia por otra parte, que cuanto mayor sea el valor de la constante de adaptación más varianza presentará la solución del sistema, es decir, más le costará al sistema estabilizarse en la solución correcta sin sufrir pequeñas variaciones alrededor de la misma.

Coeficiente de adaptación variable

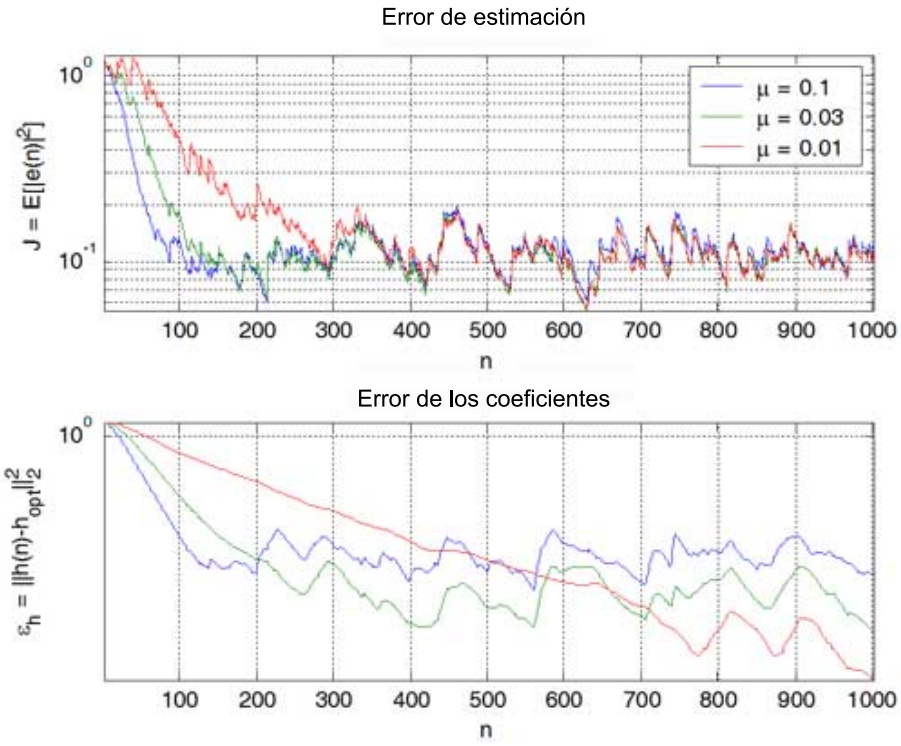
Como hemos observado, en las condiciones iniciales del algoritmo nos interesaría un coeficiente de adaptación lo más grande posible para conseguir una rápida aproximación hacia el mínimo de la función de error, pero en cambio, cuando nos encontramos en valores próximos al mínimo de la función, nos interesará una constante de adaptación variable que minimice la varianza de la estimación.

Con ello en mente podríamos establecer la siguiente estrategia:

- μ elevada en la fase inicial o fase de *training*.
- μ reducida en la fase de seguimiento o *tracking*.

El siguiente ejemplo ilustra el comportamiento de un filtro LMS en función de tres valores diferentes de la constante de adaptación μ.

Figura 10



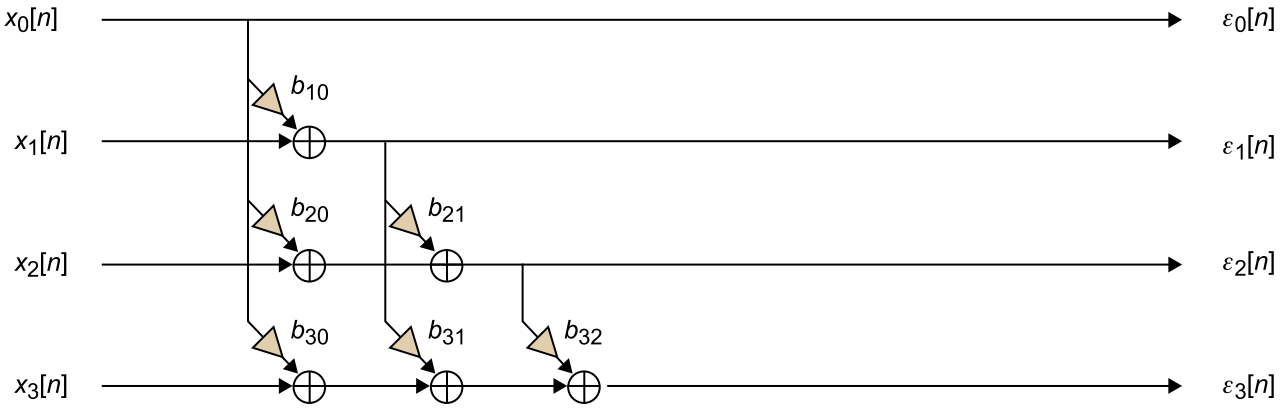
En la figura 10 podemos observar cómo un valor pequeño de la constante de adaptación μ produce una convergencia más lenta en el algoritmo tal y como podemos ver en la gráfica roja, pero la varianza del error de los coeficientes se hace más pequeña cuando alcanza la convergencia. Por su parte, cuando el valor de la constante de adaptación se hace mayor, como en la gráfica azul, la convergencia se da con mayor rapidez y se llega antes a los valores óptimos de los coeficientes del filtro pero la varianza de la estimación es superior. La gráfica ilustra lo comentado anteriormente con respecto al coeficiente de adaptación variable. En los primeros instantes de tiempo nos interesa un valor de convergencia elevado para acercarnos rápidamente a la zona del mínimo de la función, reducir el coeficiente y disponer de un sistema con capacidad para adaptarse a los cambios de entorno, pero que presente una varianza más baja en la estimación de los coeficientes del filtro, haciendo que el resultado presente unas prestaciones más estables y menos sensibles al efecto del ruido.

4. Gram-Schmidt

Las estructuras de Gram-Schmidt tienen la principal propiedad de conseguir generar un conjunto de *N* señales incorreladas a partir de *N* señales de entrada. Esta propiedad permite acelerar la velocidad de convergencia de las técnicas adaptativas, dado que si se dispone de señales incorreladas las direcciones de avance son mucho más precisas que las que se pueden conseguir con señales que tienen ciertas dependencias lineales.

A continuación se muestra el esquema de un procesador de Gram-Schmidt de dimensión $N = 4$.

Figura 11. Procesador de Gram-Schmidt de dimensión igual a $N = 4$



Como se puede apreciar en la figura 11, en conjunto de señales de salida, $\varepsilon_i[n]$, se genera a partir de sencillas combinaciones lineales de las señales de entrada, $x_i[n]$. El objetivo de la estructura es conseguir eliminar las dependencias lineales entre las variables de entrada, utilizando los valores apropiados de las constantes utilizadas en el proceso de combinación, b_{ij}. Este proceso se realiza de forma acumulativa, lo cual se consigue siguiendo un proceso iterativo: se empieza por eliminar la dependencia lineal entre las dos primeras variables de entrada, $x_0[n]$ y $x_1[n]$, obteniendo las salidas $\varepsilon_0[n]$ y $\varepsilon_1[n]$. A continuación, se elimina la dependencia lineal entre estas dos señales, ya descorrelacionadas, y la tercera señal de entrada, $x_2[n]$, obteniendo una tercera señal dentro de lo que ya denominamos el conjunto de señales descorrelacionadas, $\{\varepsilon_0[n], \varepsilon_1[n], \varepsilon_2[n]\}$. Se sigue eliminando la dependencia lineal entre este conjunto de señales descorrelacionadas y la tercera señal de entrada, $x_3[n]$, y así sucesivamente.

4.1. Análisis de la solución óptima para *N* = 2 señales

Empecemos realizando el análisis para la segunda variable de salida, para ver el **criterio de diseño de los coeficientes** b_{ij} del procesador de Gram-Schmidt. Dicho criterio se basa sencillamente en la minimización de la potencia de las señales de salida, es decir, para el caso de la señal de salida $\varepsilon_1[n]$ de:

$$E[|\varepsilon_1[n]|^2] = E[|x_1[n] + b_{10}x_0[n]|^2]$$

Si deseamos minimizar la potencia de la señal $\varepsilon_1[n]$ procederemos de la siguiente forma, suponiendo señales reales:

$$E[|\varepsilon_1[n]|^2]|_{MIN} \qquad \Leftrightarrow \qquad \frac{\partial E[|\varepsilon_1[n]|^2]}{\partial b_{10}} = 0$$

Dado que estamos aplicando un criterio basado en la minimización de un error cuadrático sobre un operador que es lineal, podemos llegar a la solución óptima resolviendo la anterior derivada:

$$\frac{\partial E[|\varepsilon_1[n]|^2]}{\partial b_{10}} = \frac{\partial E[(x_1[n]+b_{10}x_0[n])^2]}{\partial b_{10}} = 2E[x_0[n](x_1[n]+b_{10}x_0[n])] =$$
$$= 2E[x_0[n]x_1[n]] + 2b_{10}E[x_0[n]x_0[n]] = 0$$

Con lo cual la solución final es la siguiente:

$$b_{10} = -\frac{E[x_0[n]x_1[n]]}{E[x_0[n]x_0[n]]} = -\frac{E[\varepsilon_0[n]x_1[n]]}{E[\varepsilon_0[n]\varepsilon_0[n]]}$$

Como podemos observar, el coeficiente b_{10} es un coeficiente de correlación cruzada entre las señales $x_0[n]$ y $x_1[n]$ convenientemente normalizado según la potencia de la señal $x_0[n]$ y cambiado de signo. Es decir, que para estimar la parte de señal de $x_1[n]$ que es independiente, o más bien dicho, no está correlacionada con la señal $x_0[n]$, debemos sustraer de $x_1[n]$ la proporción de $x_0[n]$ justa que está relacionada con esta relación de dependencia lineal entre ambas señales, y normalizada respecto de la potencia de $x_0[n]$ para independizarla de su energía.

Principio de ortogonalidad

La solución óptima está directamente relacionada con lo que se denomina el principio de ortogonalidad, de forma que minimizar la potencia de la señal de salida $\varepsilon_1[n]$ es equivalente a conseguir que esta señal sea ortogonal a la señal

anterior $\varepsilon_0[n]$, que es justamente el propósito del procesador de Gram-Schmidt. Esto se puede ver justamente en el desarrollo de la solución óptima anterior. Si rescribimos el gradiente respecto del coeficiente b_{10} de la forma siguiente:

$$\frac{\partial E[|\varepsilon_1[n]|^2]}{\partial b_{10}} = 2E[x_0[n](x_1[n] + b_{10}x_0[n])] = 2E[x_0[n]\varepsilon_1[n]] = 0$$

$$\Rightarrow \qquad E[\varepsilon_0[n]\varepsilon_1[n]] = 0 \qquad \equiv \qquad \varepsilon_0[n] \perp \varepsilon_1[n]$$

Es decir que $\varepsilon_0[n]$ y $\varepsilon_1[n]$ son señales incorrelacionadas, o lo que es lo mismo, son linealmente independientes.

4.2. Análisis de la solución óptima para *N* = 3 señales

Dado que se trata de un proceso incremental, el análisis para $N = 3$ señales incluye la solución para las $N = 2$ primeras variables. Analicemos a continuación el caso de la tercera señal de salida. Como se ha comentado en el subapartado anterior, el criterio se basa en la minimización de la potencia de la señal de salida. En este caso, y recordando el esquema de la figura 11:

$$E[|\varepsilon_2[n]|^2] = E[|x_2[n] + b_{20}\varepsilon_0[n] + b_{21}\varepsilon_1[n]|^2]$$

Para minimizar la anterior función, primero hemos de derivar la anterior función respecto a cada uno de los coeficientes de la combinación lineal que genera la tercera señal de salida:

$$\begin{aligned}\frac{\partial E[|\varepsilon_2[n]|^2]}{\partial b_{20}} &= \frac{\partial E[(x_2[n]+b_{20}\varepsilon_0[n]+b_{21}\varepsilon_1[n])^2]}{\partial b_{20}} = \\ &= 2E[\varepsilon_0[n](x_2[n] + b_{20}\varepsilon_0[n] + b_{21}\varepsilon_1[n])] = \\ &= 2E[\varepsilon_0[n]x_2[n]] + 2b_{20}E[\varepsilon_0[n]\varepsilon_0[n]] + 2b_{21}\underbrace{E[\varepsilon_0[n]\varepsilon_1[n]]}_{=0} = \\ &= 2E[\varepsilon_0[n]x_2[n]] + 2b_{20}E[\varepsilon_0[n]\varepsilon_0[n]]\end{aligned}$$

$$\begin{aligned}\frac{\partial E[|\varepsilon_2[n]|^2]}{\partial b_{21}} &= \frac{\partial E[(x_2[n]+b_{20}\varepsilon_0[n]+b_{21}\varepsilon_1[n])^2]}{\partial b_{21}} = \\ &= 2E[\varepsilon_1[n](x_2[n] + b_{20}\varepsilon_0[n] + b_{21}\varepsilon_1[n])] = \\ &= 2E[\varepsilon_1[n]x_2[n]] + 2b_{20}\underbrace{E[\varepsilon_1[n]\varepsilon_0[n]]}_{=0} + 2b_{21}E[\varepsilon_1[n]\varepsilon_1[n]] = \\ &= 2E[\varepsilon_1[n]x_2[n]] + 2b_{21}E[\varepsilon_1[n]\varepsilon_1[n]]\end{aligned}$$

Donde se ha aplicado el hecho de que las señales de salida $\varepsilon_0[n]$ y $\varepsilon_1[n]$ son señales decorreladas, lo que conlleva a una simplificación evidente de ambos gradientes.

Con lo que la solución del proceso de minimización es la siguiente:

$$E[|\varepsilon_2[n]|^2]\,|_{MIN} \qquad \Leftrightarrow \qquad \frac{\partial E[|\varepsilon_2[n]|^2]}{\partial b_{20}} = \frac{\partial E[|\varepsilon_2[n]|^2]}{\partial b_{21}} = 0$$

$$b_{20} = -\frac{E[\varepsilon_0[n]x_2[n]]}{E[\varepsilon_0[n]\varepsilon_0[n]]} \qquad\qquad b_{21} = -\frac{E[\varepsilon_1[n]x_2[n]]}{E[\varepsilon_1[n]\varepsilon_1[n]]}$$

Comprobemos la ortogonalidad de la señal $\varepsilon_2[n]$ con las dos señales anteriores $\varepsilon_0[n]$ y $\varepsilon_1[n]$, empezando por la primera de ellas:

$$\begin{aligned} E[\varepsilon_2[n]\varepsilon_0[n]] &= E[(x_2[n] + b_{20}\varepsilon_0[n] + b_{21}\varepsilon_1[n])\varepsilon_0[n]] = \\ &= E[x_2[n]\varepsilon_0[n]] + b_{20}E[\varepsilon_0[[n]\varepsilon_0[n]] + b_{21}\underbrace{E[\varepsilon_0[n]\varepsilon_1[n]]}_{=0} = \\ &= E[x_2[n]\varepsilon_0[n]] + (-\tfrac{E[\varepsilon_0[n]x_2[n]]}{E[\varepsilon_0[n]\varepsilon_0[n]]})E[\varepsilon_0[n]\varepsilon_0[n]] = 0 \end{aligned}$$

Es fácil realizar el mismo procedimiento para verificar que, efectivamente, tanto $\varepsilon_0[n]$ como $\varepsilon_1[n]$ son señales ortogonales a $\varepsilon_2[n]$, además de serlo entre ellas (ya verificado en el anterior subapartado).

4.3. Solución general óptima

Como se puede apreciar, si se prosigue con el análisis de más variables de salida (por ejemplo, para $N = 4$), siguiendo exactamente el mismo desarrollo para la señal de salida i-ésima:

$$\begin{aligned} \frac{\partial E[|\varepsilon_i[n]|^2]}{\partial b_{ik}} &= \frac{\partial E[(x_i[n] + \sum_{j=0}^{i-1} b_{ij}\varepsilon_j[n])^2]}{\partial b_{ik}} = 2E[\varepsilon_k[n]\varepsilon_i[n]] = \\ &= 2E[\varepsilon_k[n](x_i[n] + \sum_{j=0}^{i-1} b_{ij}\varepsilon_j[n])] \\ &= 2E[\varepsilon_k[n]x_i[n]] + 2\sum_{j=0}^{i-1} b_{ij}\underbrace{E[\varepsilon_k[n]\varepsilon_j[n]]}_{=0\ j\neq k} = \\ &= 2E[\varepsilon_k[n]x_i[n]] + 2b_{ik}E[\varepsilon_k[n]\varepsilon_k[n]] = 0 \end{aligned}$$

Con lo que las ecuaciones de diseño óptimo para los coeficientes de la estructura darán a lugar la solución general siguiente:

$$b_{ik}|_{opt} = -\frac{E[\varepsilon_k[n]x_i[n]]}{E[\varepsilon_k[n]\varepsilon_k[n]]} = -\frac{E[\varepsilon_k[n]x_i[n]]}{E[\varepsilon_k^2[n]]}$$

$$\varepsilon_i[n] = x_i[n] + \sum_{j=0}^{i-1} b_{ij}\varepsilon_j[n]$$

De la solución óptima se deduce que en general se cumplirá la **condición de ortogonalidad** para todas las variables de salida, escogidas de dos en dos:

$$E[\varepsilon_k[n]\varepsilon_i[n]] = \frac{1}{2}\frac{\partial E[\varepsilon_i^2[n]]}{\partial b_{ik}} = 0 \qquad \forall\, k \neq i$$

4.4. Solución adaptativa

4.4.1. Algoritmo del gradiente descendente y estudio de su convergencia

A continuación se deriva la solución adaptativa basada en el algoritmo del gradiente descendiente para la estructura de Gram-Schmidt, y a continuación se deriva la condición de convergencia que se debe cumplir para el parámetro de adaptación μ del algoritmo.

En primer lugar, si aplicamos este algoritmo para la adaptación del coeficiente genérico:

$$b_{ik}[n] = \quad b_{ik}[n-1] - \mu\frac{\partial E[\varepsilon_i^2[n]]}{\partial b_{ik}} = b_{ik}[n-1] - \mu E[\varepsilon_k[n]\varepsilon_i[n]] \qquad (1)$$

donde se ha omitido el valor constante 2 que multiplica al valor del gradiente, por poder ser absorbido en el propio factor de convergencia μ.

Para asegurar la convergencia hacia la solución óptima debemos primero analizar con más detenimiento qué valores del parámetro de adaptación μ permitirán dicha convergencia. Para ello, primero se debe expresar la ecuación de adaptación de la siguiente forma:

$$\begin{aligned} b_{ik}[n] = b_{ik}[n-1] - \mu E[\varepsilon_k[n]\varepsilon_i[n]] \quad = \\ = b_{ik}[n-1] - \mu(E[\varepsilon_k[n]x_i[n]] + b_{ik}[n-1]E[\varepsilon_k[n]\varepsilon_k[n]]) = \\ = b_{ik}[n-1]\,(1 - \mu E[\varepsilon_k^2[n]]) - \mu E[\varepsilon_k[n]x_i[n]] \end{aligned}$$

Restando a ambos lados de la igualdad la solución óptima para el coeficiente b_{ik}:

$$\begin{aligned} b_{ik}[n] - b_{ik}\big|_{opt} = b_{ik}[n-1](1 - \mu E[\varepsilon_k^2[n]]) - \mu E[\varepsilon_k[n]x_i[n]] - b_{ik}\big|_{opt} = \\ = b_{ik}[n-1](1 - \mu E[\varepsilon_k^2[n]]) - \mu E[\varepsilon_k[n]x_i[n]] - (-\frac{E[\varepsilon_k[n]x_i[n]]}{E[\varepsilon_k^2[n]]}) = \\ = b_{ik}[n-1](1 - \mu E[\varepsilon_k^2[n]]) - (-\frac{E[\varepsilon_k[n]x_i[n]]}{E[\varepsilon_k^2[n]]})\,(1 - \mu E[\varepsilon_k^2[n]]) = \\ = (b_{ik}[n-1] - b_{ik}\big|_{opt})\,(1 - \mu E[\varepsilon_k^2[n]]) \end{aligned}$$

Partiendo de una solución inicial $b_{ik}[0]$ (para $n=0$), podemos expresar, a partir de la anterior ecuación recursiva, la solución para un instante cualquiera en función de la solución inicial como sigue:

$$b_{ik}[n]-b_{ik}\big|_{opt}=(b_{ik}[0]-b_{ik}\big|_{opt})\,(1-\mu E[\varepsilon_k^2[n]])^n$$

Es decir, la solución después de la n-esima iteración depende exponencialmente del factor de convergencia μ así como de la energía de la salida $E\left[\varepsilon_k^2[n]\right]$. Como se puede apreciar en la ecuación anterior, únicamente si $\left|1-\mu E\left[\varepsilon_k^2[n]\right]\right|<1$ entonces el algoritmo convergerá hacia la solución óptima, dado que la potencia n-ésima de este factor tenderá hacia cero:

$$\lim_{n\to+\infty}\left(b_{ik}[n]-b_{ik}\big|_{opt}\right)=\lim_{n\to+\infty}\left(b_{ik}[0]-b_{ik}\big|_{opt}\right)\underbrace{\left(1-\mu E\left[\varepsilon_k^2[n]\right]\right)^n}_{-1<r<+1}=0$$

$$\Rightarrow\qquad \lim_{n\to+\infty} b_{ik}[n]=b_{ik}\big|_{opt}$$

De lo anterior se deduce que los valores del parámetro de adaptación μ que garantizarán la convergencia del algoritmo del gradiente hacia la solución óptima serán los siguientes:

$$-1<1-\mu E[\varepsilon_k^2[n]]<+1\qquad\Rightarrow\qquad 0<\mu<\frac{2}{E[\varepsilon_k^2[n]]}\qquad(2)$$

Una conclusión muy interesante en este punto es que el procesador de Gram-Schmidt adaptativo permite asegurar, gracias a sus propiedades de decorrelación entre las variables que se obtienen, una convergencia eficiente hacia la solución óptima si se dispone de una estimación de la energía de las variables de salida del procesador. Como ya se vio en la solución LMS para un estimador del tipo FIR, cuanto mayor es el valor del parámetro μ mayor es la velocidad de convergencia, siempre y cuando no se supere el umbral que llevaría hacia la inestabilidad del algoritmo. En este caso, este umbral queda definido de forma inversamente proporcional a la energía de salida, como se puede ver en la ecuación (2).

Ved también

Hemos visto la solución LMS para un estimador del tipo FIR en el apartado 3 de este módulo didáctico.

4.4.2. Algoritmo LMS

Igual que en el resto de técnicas adaptativas, el procesador de Gram-Schmidt puede ser adaptado a una versión LMS que no requiera de la estimación de valores promedios (o esperados) para su entrenamiento. De hecho, la versión LMS parte de la solución basada en el algoritmo del gradiente descendiente, estudiada en el apartado anterior, y la adapta para poder usar el valor de las variables instantáneas de que se dispone en cada momento.

En primer lugar, veamos cómo se obtiene el valor de la variable de adaptación. Siguiendo el desarrollo del apartado anterior vimos que el valor de μ debe estar acotado según el valor de la energía de la salida, para cada rama del procesador. Por otro lado, recordemos que valores elevados de μ permitirán obtener convergencias más rápidas que valores menores, siempre que no se supere el límite que lleva a la inestabilidad del método, o lo que es lo mismo, a no converger hacia la solución óptima. Con todo ello, y recordando que el objetivo del método LMS es evitar el uso de valores esperados en las ecuaciones de adaptación, empezamos por establecer el método de obtención de la energía de la salida $\varepsilon_k[n]$ por medio de un sencillo promediado o filtrado paso-bajo con un filtro IIR de orden 1, también denominado filtro de suavizado o de *smoothing*:

$$E_k[n] = \lambda E_k[n-1] + (1-\lambda)\varepsilon_k^2[n] \qquad (3)$$

donde λ es el denominado **parámetro de *smoothing***, definido en el intervalo $0 < \lambda < 1$.

Análisis del filtro de *smoothing*

Si analizamos la función de transferencia de este filtro, $H_{lp}(z)$, que tiene como entrada el cuadrado de la salida, $w_k[n] = \varepsilon_k^2[n]$, y como salida la variable de energía $E_k[n]$:

$$H_{lp}(z) = \frac{W_k(z)}{E_k(z)} = \frac{1-\lambda}{1-\lambda z^{-1}}$$

Sigamos analizando su respuesta frecuencial y concretamente la ganancia o su respuesta en módulo:

$$H_{lp}(e^{j\omega}) = \frac{W_k(e^{j\omega})}{E_k(e^{j\omega})} = \frac{(1-\lambda)}{1-\lambda e^{-j\omega}}$$

$$|H_{lp}(e^{j\omega})| = \left|\frac{1-\lambda}{1-\lambda e^{-j\omega}}\right| = \frac{1-\lambda}{\sqrt{(1-\lambda\cos(\omega))^2 + \lambda^2\sin^2(\omega)}} = \frac{1-\lambda}{\sqrt{1+\lambda^2-2\lambda\cos(\omega)}}$$

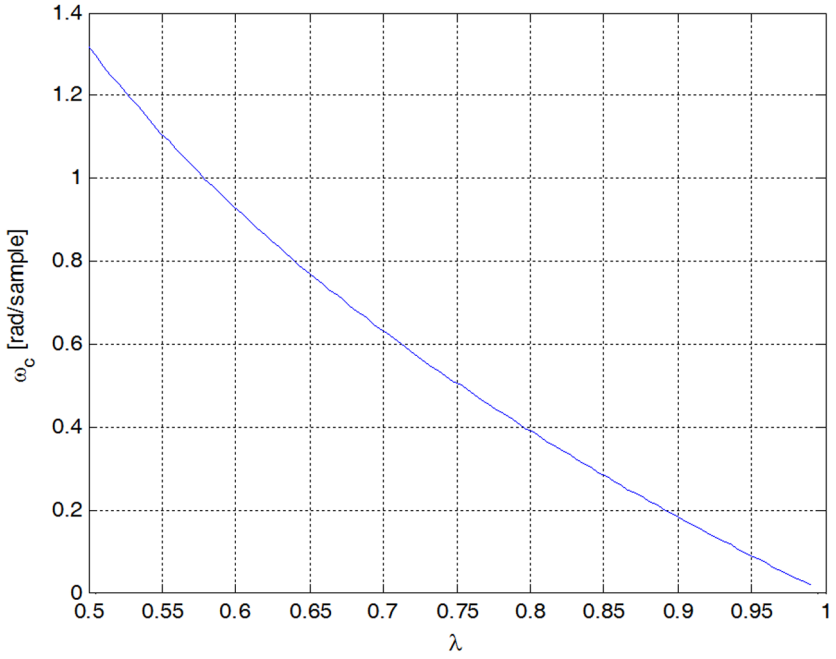
Podemos observar cómo dicho filtro tiene ganancia unitaria para la pulsación $\omega = 0$, y que a medida que ω crece hasta el valor de la frecuencia de Nyquist ($\omega = \pi$), el módulo irá decreciendo de forma progresiva hasta su valor mínimo. Para hallar su frecuencia de corte, ω_c, donde el módulo ha decrecido hasta la mitad de la máxima ganancia, igualamos dicha ganancia en la expresión del módulo de la función de transferencia:

$$\left|H_{lp}(e^{j\omega})\right|_{\omega=\omega_c} = \frac{1}{2} = \frac{1-\lambda}{\sqrt{1+\lambda^2-2\lambda\cos(\omega_c)}} \quad \Rightarrow \quad \cos(\omega_c) = \frac{1+\lambda^2-4(1-\lambda)^2}{2\lambda}$$

$$\Rightarrow \quad \omega_c = \operatorname{acos}\left(\frac{-3+8\lambda-3\lambda^2}{2\lambda}\right)$$

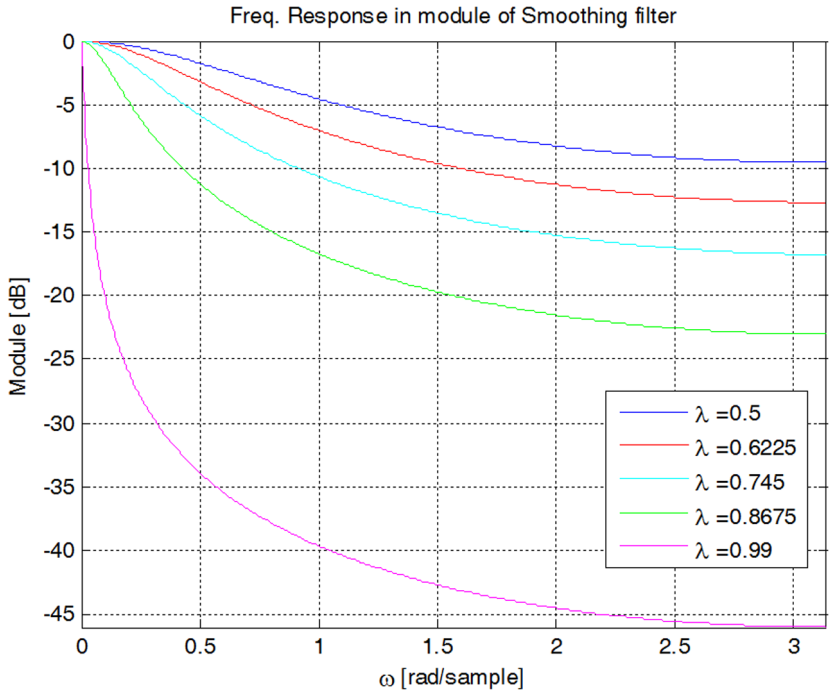
En la figura 12 se puede apreciar la dependencia entre la frecuencia de corte ω_c del filtro paso-bajo y el parámetro de *smoothing* λ. Queda claro que valores del parámetro de *smoothing* más cercanos a 1 provocan un mayor promediado de la señal, es decir, se filtra con un filtro paso-bajo con frecuencia de corte menor. En cambio, para valores de menores λ el promediado es más sutil, o el filtro deja de seleccionar frecuencias bajas para dejar pasar frecuencias medias o más altas. Así, el parámetro de control del filtro λ permite modificar fácilmente la forma en que queremos promediar o suavizar la señal dada.

Figura 12. Relación entre el parámetro de *smoothing* λ y la frecuencia de corte ω_c del filtro de *smoothing*



En la figura 13 se representa el módulo del filtro de *smoothing* para diferentes valores del parámetro de *smoothing* λ. Como se puede apreciar, la ganancia del filtro en todos los casos es de 0 dB (1 en valores lineales de ganancia), mientras que la caída del filtro es más abrupta cuanto más cercano a 1 es el valor del parámetro de *smoothing*. Este parámetro no debe fijarse al valor de 1 si se quiere evitar que el filtro se haga inestable para frecuencias de entrada igual a 0, en cuyo caso la respuesta del filtro sería creciente en el tiempo, de modo que la estimación de la energía crecería de forma indefinida.

Figura 13. Módulo de la respuesta frecuencial del filtro de *smoothing* para cinco valores del parámetro de *smoothing* λ



Volviendo al uso de este tipo de filtro para el cómputo de la energía de la señal, si retomamos la ecuación temporal (3) del filtro aplicado sobre el cuadrado de la señal, podemos ver que en el fondo estamos produciendo un promediado del cuadrado, es decir, estamos de algún modo acumulando la señal. Esta operación sirve como sustituta del operador "valor esperado" que aparecía en la ecuación (2), de forma que podemos establecer un método para ajustar la constante de adaptación μ del algoritmo, de forma que se asegure la convergencia del método y se acelere a la vez la velocidad de convergencia.

Pasemos ahora a ver la ecuación de adaptación de los coeficientes de la estructura de Gram-Schmidt, según la perspectiva del LMS. Si retomamos la ecuación (1) de adaptación de la versión del algoritmo del gradiente, la versión LMS de dicha ecuación es la siguiente:

$$b_{ik}[n] = b_{ik}[n-1] - \mu E[\varepsilon_k[n]\varepsilon_i[n]] \approx b_{ik}[n-1] - \mu\varepsilon_k[n]\varepsilon_i[n]$$

Como vemos, se sustituye el operador valor esperado por una sencilla estimación instantánea de la señal implicada, de forma que se delega el proceso de promediado a la propia estrategia de iteración del método. Dado que el método consiste en actualizar la versión anterior de cada coeficiente, $b_{ik}[n-1]$, añadiendo un nuevo término, $-\mu\varepsilon_k[n]\varepsilon_i[n]$, este proceso acumulativo ya supone en sí un proceso de promediado, que acabará por determinar una dirección correcta hacia la convergencia del método.

5. RLS

El método RLS (del inglés, *recursive least squares*, 'mínimos cuadrados recursivos') realiza la adaptación de los coeficientes del filtro FIR de orden N (N + 1 coeficientes) mediante la minimización de la suma de un error cuadrático acumulado según una función de ponderación exponencial decreciente hacia el pasado.

A diferencia del método LMS, el algoritmo RLS parte del planteamiento de una **función de coste determinista**, es decir, en la que no interviene ningún valor esperado. Esta función se define como un error cuadrático medio ponderado con un factor de memoria que permite que el filtro se adapte mejor al tipo de variaciones temporales del sistema que se intenta compensar o estimar. A medida que se reciben las muestras de la señal incidente, se calcula la solución óptima del filtro de forma recursiva, dando lugar al método RLS.

El algoritmo RLS presenta **gran velocidad de convergencia** con respecto al método LMS incluso cuando la matriz de correlación de la señal de entrada tiene autovalores muy dispersos y presenta excelentes características en entornos variantes con el tiempo. El objetivo del método RLS es encontrar los coeficientes óptimos de tal forma que la señal de salida del filtro sea lo más parecida posible al de referencia según el criterio de los mínimos cuadrados ponderados según un **factor de olvido** denominado λ.

5.1. Solución óptima

Definiremos el vector $\vec{x}[n]$ como un vector columna de $\mathbb{R}^{N+1}$ que contiene la información del filtro en el instante n, es decir:

$$\vec{x}[n] = \begin{bmatrix} x[n] \\ x[n-1] \\ \vdots \\ x[n-N] \end{bmatrix}$$

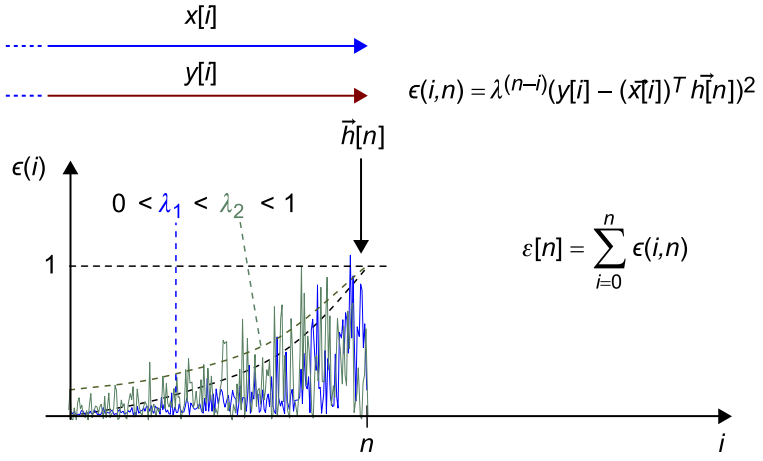
Los coeficientes del filtro se calcularán minimizando la siguiente expresión de un error determinista:

$$\varepsilon[n] = \sum_{i=0}^{n} \lambda^{(n-i)} \left(y[i] - (\vec{x}[i])^T \vec{h}[n] \right)^2 \qquad (4)$$

donde $\vec{h}[n]$ es el vector de coeficientes del filtro en el instante n, y λ es el factor de olvido, con valores entre $0 < \lambda < 1$.

$\varepsilon[n]$ es el denominado **error instantáneo del filtro** (en el instante n), y se trata de una norma definida positiva por lo que no puede dar lugar a valores negativos. Además, es una función cuadrática de los coeficientes del filtro, $h_k[n]$ siendo k el índice de coeficiente y n el instante temporal en el que se mide, lo cual permite abordar la solución mediante sistemas lineales de ecuaciones. Esta nomenclatura, a diferencia de la usada en la función de coste del método LMS, deja explícito que los coeficientes del filtro varían temporalmente, al ser el algoritmo RLS un método adaptativo. Así pues, tal y como se lee de la ecuación (4), el error instantáneo del filtro depende del valor de los coeficientes en cada instante, pero se obtiene a partir de una suma cuadrática de términos que está ponderada por un término que decrece con el tiempo pasado. Este término de ponderación depende del **factor de olvido** λ.

Figura 14. Representación gráfica del cómputo del error cuadrático medio ponderado de la función de coste RLS



Como se puede apreciar en la figura 14, el error instantáneo, $\epsilon(i, n)$, en instantes anteriores o igual al presente, $i \leq n$, se calcula con la respuesta impulsional actual del filtro, $\vec{h}[n]$, y la entrada y la salida en el tiempo i. De este modo, la respuesta del filtro FIR $\vec{h}[n]$ tendrá en cuenta un error acumulado pero ponderado según un factor exponencialmente decreciente en el tiempo. Esta ponderación viene marcada por el término $\lambda^{(n-i)}$, que toma el valor 1 en $i = n$, y toma valores menores a menudo que i se hace menor que n, o lo que es lo mismo, se consideran tiempos más pasados al actual. Como se puede ver también en la misma figura, para valores del factor de olvido más cercanos a la unidad, se tendrá en cuenta el error acumulado durante más tiempo pasado, mientras que para valores más cercanos a cero, el error acumulado recogerá únicamente el pasado más reciente.

Como se puede apreciar en la ecuación de error RLS (4), el error acumulado empieza a contabilizarse en $i = 0$, instante que se supone, por convenio, el instante inicial en el que se dispone de información de las señales tanto de entrada $x[n]$ como de la señal deseada $y[n]$. Derivando e igualando a cero la expresión anterior, se llega al siguiente resultado:

$$\vec{h}[n] = \mathbf{R}_{xx}^{-1}[n]\vec{r}_{yx}[n] = \underbrace{\left(\sum_{i=0}^{n} \lambda^{(n-i)}\vec{x}[i](\vec{x}[i])^T\right)}_{\mathbf{R}_{xx}[n]}{}^{-1} \underbrace{\left(\sum_{i=0}^{n} \lambda^{(n-i)}y[i]\vec{x}[i]\right)}_{\vec{r}_{yx}[n]} \qquad (5)$$

donde $\mathbf{R}_{xx}[n]$ es una matriz de autocorrelación de la señal de entrada $x[n]$ y $\vec{r}_{yx}[n]$ es un vector de correlación cruzada entre la señal deseada $y[n]$ y la señal de entrada $x[n]$. A diferencia de la solución del filtro FIR óptimo, en este caso ambos términos no son estocásticos, pues estos se expresan a partir de una ecuación determinista, o lo que es lo mismo, expresada como un promedio acumulado de valores pasados y presentes.

Ved también

Hemos visto la solución del filtro FIR óptimo en el apartado 3.2. de este módulo didáctico.

Ejercicio

Deducir la solución óptima del filtro RLS, es decir, aquella solución del vector de coeficientes $\vec{h}[n]$ que minimiza el error cuadrático $\varepsilon[n]$ (ecuación (5)).

Solución

El tratamiento matemático para derivar la solución óptima del filtro FIR-RLS lo haremos en base al principio de ortogonalidad aplicado sobre el producto escalar siguiente:

$$< a[i], b[i] > = \sum_{i=0}^{n} \lambda^{(n-i)}a[i]b[i]$$

El error a minimizar lo podemos reescribir como una norma o producto escalar:

$$\varepsilon[n] = \sum_{i=0}^{n} \lambda^{(n-i)}\left(y[i] - (\vec{x}[i])^T\vec{h}[n]\right)^2 = \sum_{i=0}^{n} \lambda^{(n-i)}e2[i] = < e[i], e[i] >$$

Siendo:

$$e[i] = y[i] - (\vec{x}[i])^T\vec{h}[n]$$

Dado que el error instantáneo $e[i]$ es el error entre la señal deseada $y[i]$ y una combinación lineal, según los coeficientes $h_k[n]$ del vector $\vec{h}[n]$, de las señales $x[i-k]$ para $0 \le k \le N$, el valor mínimo de la norma de este error se conseguirá, según el **teorema de la proyección ortogonal**, cuando el error $e[i]$ conseguido sea ortogonal a las señales $x[i-k]$ para $0 \le k \le N$, es decir:

$$< e[i], x[i-k] > = 0 \qquad 0 \le k \le N$$

Desarrollemos, pues, este resultado:

$$< e[i], x[i-k] > = \sum_{i=0}^{n} \lambda^{(n-i)} x[i-k] e[i] =$$

$$= \sum_{i=0}^{n} \lambda^{(n-i)} x[i-k] \left(y[i] - (\vec{x}[i])^T \vec{h}[n] \right) =$$

$$= \sum_{i=0}^{n} \lambda^{(n-i)} y[i] x[i-k] - \sum_{i=0}^{n} \lambda^{(n-i)} x[i-k] (\vec{x}[i])^T \vec{h}[n] =$$

$$= \sum_{i=0}^{n} \lambda^{(n-i)} y[i] x[i-k] - \left(\sum_{i=0}^{n} \lambda^{(n-i)} x[i-k] (\vec{x}[i])^T \right) \vec{h}[n] = 0$$

Si iteramos la anterior ecuación para los distintos valores de k dentro del margen establecido $k \in [0,N]$, podemos reescribir las $N+1$ ecuaciones de forma compacta con una única ecuación matricial, que es la siguiente:

$$\sum_{i=0}^{n} \lambda^{(n-i)} y[i] \vec{x}[i] - \left(\sum_{i=0}^{n} \lambda^{(n-i)} \vec{x}[i] (\vec{x}[i])^T \right) \vec{h}[n] = 0$$

donde, como se puede apreciar, se ha sustituido los valores de la señal de entrada por el vector de señal $\vec{x}[i]$.

Si multiplicamos la ecuación anterior por la matriz inversa de la matriz que multiplica por la izquierda al vector de coeficientes se llega a la solución del filtro RLS (ecuación (5)).

La ecuación (5) se debe leer de la siguiente forma: el primer término o sumatorio es un vector, puesto que se trata de una suma de factores escalares que ponderan valores de una señal ($y[i]$) por vectores de señal de entrada ($\vec{x}[i]$). En cambio, el segundo término es una matriz, formada por una suma de matrices, y cada una de estas matrices es el resultado del producto de dos vectores de señal, un vector columna $\vec{x}[i]$ por un vector fila $(\vec{x}[i])^T$, y el factor asociado a la ponderación exponencial decreciente que es función del factor de olvido.

Una observación importante a remarcar a esta altura de la explicación es que la matriz de autocorrelación de la señal de entrada, $\mathbf{R}_{xx}[n]$, es de dimensiones $(N+1) \times (N+1)$, por lo que **su inversión en cada tiempo de muestra requiere de un coste computacional de orden** $O((N+1)^3)$, por ejemplo, usando un algoritmo como la eliminación de Gauss-Jordan. Esta complejidad computacional puede ser excesiva, por lo que en el siguiente apartado se analiza la solución adaptativa del método RLS, la cual permite reducir el orden de complejidad de cálculo de la solución óptima.

5.2. Solución adaptativa

Para llegar a la solución adaptativa del filtro RLS, es necesario primero estudiar la identidad de Woodbury o también llamada **lema de inversión de matrices**. Este teorema nos permite simplificar la inversión de la matriz de autocorrelación para llegar a una versión adaptativa o incremental, que repercutirá en un algoritmo de complejidad computacional mucho menor.

Lema de inversión de matrices

Sean las matrices $\mathbf{A} \in \mathbb{C}^{N\times N}$ y $\mathbf{R} \in \mathbb{C}^{M\times M}$ no singulares[1], y $\mathbf{X} \in \mathbb{C}^{N\times M}$ y $\mathbf{Y} \in \mathbb{C}^{M\times N}$. Sea la matriz $\mathbf{B} \in \mathbb{C}^{N\times N}$ definida por la ecuación siguiente:

$$\mathbf{B} = \mathbf{A} + \mathbf{XRY}$$

(1)Una matriz (cuadrada) no singular es una matriz invertible, pues se cumple que su determinante es diferente a 0, o que su rango (o número de vectores fila o columna linealmente independientes) es igual a su dimensión.

Entonces, **B** es también una matriz no singular, cuya inversa se puede expresar como:

$$\mathbf{B}^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{X}\left(\mathbf{R}^{-1} + \mathbf{YA}^{-1}\mathbf{X}\right)^{-1}\mathbf{YA}^{-1}$$

Este teorema puede demostrarse de forma directa verificando que la matriz **B** por su inversa da como resultado la matriz identidad, aspecto que os proponemos como ejercicio.

Solución adaptativa para la matriz inversa de correlación

El lema de inversión de matrices permite conseguir un cálculo recursivo de la matriz de autocorrelación $\mathbf{R}_{xx}^{-1}[n]$. Primero conviene desglosar la matriz como suma de dos términos, uno asociado al pasado y otro al presente:

$$\mathbf{R}_{xx}[n] = \sum_{i=0}^{n} \lambda^{(n-i)} \vec{x}[i](\vec{x}[i])^T = \underbrace{\lambda \sum_{i=0}^{n-1} \lambda^{(n-1-i)} \vec{x}[i](\vec{x}[i])^T}_{\text{Pasado}} + \underbrace{\vec{x}[n](\vec{x}[n])^T}_{\text{Presente}}$$

Como se puede apreciar, el término asociado al pasado es una versión anterior de dicha matriz, pues:

$$\mathbf{R}_{xx}[n] = \lambda \mathbf{R}_{xx}[n-1] + \vec{x}[n] \times 1 \times (\vec{x}[n])^T$$

donde 1 es una matriz identidad de $\mathbb{C}^{1\times 1}$, coincidiendo con las dimensiones de número de columnas de $\vec{x}[n]$ y de filas de $(\vec{x}[n])^T$, o sea, el escalar 1.

Si observamos detenidamente la ecuación anterior, podemos identificar los términos asociados a la matriz **B** del lema de inversión de matrices, concretamente:

$$\mathbf{A} = \lambda \mathbf{R}_{xx}[n-1] \qquad \mathbf{X} = \vec{x}[n] \qquad \mathbf{R} = 1 \qquad \mathbf{Y} = (\vec{x}[n])^T$$

Con ello, podemos expresar la matriz inversa de la matriz de autocorrelación de forma recursiva aplicando el lema de inversión:

$$\mathbf{R}_{xx}^{-1}[n]=\left(\frac{1}{\lambda}\right)\mathbf{R}_{xx}^{-1}[n-1]-\left(\frac{1}{\lambda}\right)\mathbf{R}_{xx}^{-1}[n-1]\vec{x}[n]\times$$
$$\times\left(1^{-1}+(\vec{x}[n])^T\left(\frac{1}{\lambda}\right)\mathbf{R}_{xx}^{-1}[n-1]\vec{x}[n]\right)^{-1}(\vec{x}[n])^T\left(\frac{1}{\lambda}\right)\mathbf{R}_{xx}^{-1}[n-1]=$$
$$=\left(\frac{1}{\lambda}\right)\mathbf{R}_{xx}^{-1}[n-1]-\left(\frac{1}{\lambda}\right)\mathbf{R}_{xx}^{-1}[n-1]\frac{\vec{x}[n](\vec{x}[n])^T\mathbf{R}_{xx}^{-1}[n-1]}{\lambda+(\vec{x}[n])^T\mathbf{R}_{xx}^{-1}[n-1]\vec{x}[n]}=$$
$$=\left(\frac{1}{\lambda}\right)\left(\mathbf{R}_{xx}^{-1}[n-1]-\mathbf{K}[n](\vec{x}[n])^T\mathbf{R}_{xx}^{-1}[n-1]\right)$$

$$\mathbf{K}[n]=\frac{\mathbf{R}_{xx}^{-1}[n-1]\vec{x}[n]}{\lambda+(\vec{x}[n])^T\mathbf{R}_{xx}^{-1}[n-1]\vec{x}[n]}$$

donde el vector columna $\mathbf{K}[n]\in\mathbb{C}^{1\times(N+1)}$ es denominado **ganancia de Kalman**.

Observemos cómo, en este caso, al ser el término entre paréntesis asociado a $(\mathbf{R}^{-1}+\mathbf{Y}\mathbf{A}^{-1}\mathbf{X})$ de dimensiones 1×1, su inversión es directamente el mismo término como cociente multiplicando al resto. Observar que esta ecuación permite reducir la complejidad computacional de la inversión, que originalmente era de orden $O((N+1)^3)$, a otro menor $O((N+1)^2)$ asociado a los productos matriciales que aparecen, siempre que se disponga de la matriz de correlación inversa de la iteración anterior, $\mathbf{R}_{xx}^{-1}[n-1]$.

Teorema

La ganancia de Kalman se puede expresar como la matriz inversa de correlación multiplicada por el vector de señal:

$$\mathbf{K}[n]=\mathbf{R}_{xx}^{-1}[n]\vec{x}[n]$$

Demostración

Partiendo de la ecuación adaptativa anterior, en la que la matriz inversa de correlación se expresa de forma recursiva a partir de su valor anterior, desarrollamos y obtenemos:

$$\mathbf{R}_{xx}^{-1}[n]\vec{x}[n]=\left(\frac{1}{\lambda}\right)\left(\mathbf{R}_{xx}^{-1}[n-1]-\mathbf{K}[n](\vec{x}[n])^T\mathbf{R}_{xx}^{-1}[n-1]\right)\vec{x}[n]=$$
$$=\left(\frac{1}{\lambda}\right)\left(\mathbf{R}_{xx}^{-1}[n-1]\vec{x}[n]-\mathbf{K}[n](\vec{x}[n])^T\mathbf{R}_{xx}^{-1}[n-1]\vec{x}[n]\right)=$$
$$=\left(\frac{1}{\lambda}\right)\left(\mathbf{R}_{xx}^{-1}[n-1]\vec{x}[n]-(\mathbf{R}_{xx}^{-1}[n-1]\vec{x}[n]-\lambda\mathbf{K}[n])\right)=\mathbf{K}[n]$$

Donde en la tercera igualdad se ha substituido el término $\mathbf{K}[n](\vec{x}[n])^T\mathbf{R}_{xx}^{-1}[n-1]\vec{x}[n]$ por el valor que se obtiene despejando de la ecuación que define la ganancia de Kalman.

Solución adaptativa del método RLS

Llegado al punto de expresar la matriz de autocorrelación de forma recursiva, a partir de su versión temporal anterior, podemos llegar a la solución adaptativa del filtro RLS de forma sencilla. En primer lugar, expresamos el vector de correlación cruzada de forma adaptativa, tal y como se hizo con la matriz de autocorrelación, desglosándolo en los términos asociados al pasado y el término asociado al presente, esto es:

$$\vec{r}_{yx}[n]=\sum_{i=0}^{n}\lambda^{(n-i)}y[i]\vec{x}[i]=\sum_{i=0}^{n-1}\lambda^{(n-i)}y[i]\vec{x}[i]+y[n]\vec{x}[n]=$$
$$=\lambda\underbrace{\sum_{i=0}^{n-1}\lambda^{(n-1-i)}y[i]\vec{x}[i]}_{\vec{r}_{yx}[n-1]}+y[n]\vec{x}[n]=\lambda\vec{r}_{yx}[n-1]+y[n]\vec{x}[n]$$

A continuación volvemos a expresar la solución óptima del filtro RLS y empezamos por sustituir el vector de correlación cruzada por su versión adaptativa:

$$\vec{h}[n]=\mathbf{R}_{xx}^{-1}[n]\vec{r}_{yx}[n]=\mathbf{R}_{xx}^{-1}[n]\left(\lambda\vec{r}_{yx}[n-1]+y[n]\vec{x}[n]\right)=\lambda\mathbf{R}_{xx}^{-1}[n]\vec{r}_{yx}[n-1]+\mathbf{R}_{xx}^{-1}[n]y[n]\vec{x}[n]$$

Entonces, sustituimos la matriz inversa de correlación por su versión adaptativa en el primer término de la anterior ecuación:

$$\vec{h}[n]=\lambda(\tfrac{1}{\lambda})\ \ (\mathbf{R}_{xx}^{-1}[n-1]-\mathbf{K}[n](\vec{x}[n])^T\mathbf{R}_{xx}^{-1}[n-1])\vec{r}_{yx}[n-1]+\mathbf{R}_{xx}^{-1}[n]y[n]\vec{x}[n]=$$
$$=\mathbf{R}_{xx}^{-1}[n-1]\vec{r}_{yx}[n-1]-\mathbf{K}[n](\vec{x}[n])^T\mathbf{R}_{xx}^{-1}[n-1]\vec{r}_{yx}[n-1]+\mathbf{R}_{xx}^{-1}[n]\vec{x}[n]y[n]$$

Podemos identificar en la ecuación anterior dos productos que se corresponden con la definición de la solución óptima del filtro RLS para el instante de tiempo $n-1$, es decir, $\vec{h}[n-1]=\mathbf{R}_{xx}^{-1}[n-1]\vec{r}_{yx}[n-1]$, por lo que:

$$\vec{h}[n]=\vec{h}[n-1]-\mathbf{K}[n](\vec{x}[n])^T\vec{h}[n-1]+\underbrace{\mathbf{R}_{xx}^{-1}[n]\vec{x}[n]}_{\mathbf{K}[n]}y[n]=$$
$$=\vec{h}[n-1]+\mathbf{K}[n]\underbrace{(y[n]-(\vec{x}[n])^T\vec{h}[n-1])}_{e[n]}=\vec{h}[n-1]+\mathbf{K}[n]e[n]$$

En este desarrollo se ha utilizado el teorema anterior para llegar a la solución final.

Como se puede observar en la ecuación de adaptación del filtro RLS, la respuesta impulsional se adapta a partir del error de salida que tiene en cuenta la respuesta del filtro en el tiempo anterior, $e[n]=y[n]-(\vec{x}[n])^T\vec{h}[n-1]$, y la ganancia de Kalman. Llegado a este punto, conviene realizar algunas sustituciones, teniendo en cuenta la ecuación que define la ganancia de Kalman, para ver ciertas similitudes con la ecuación de adaptación del filtro LMS).

$$\vec{h}[n]=\vec{h}[n-1]+\mu[n]\mathbf{R}_{xx}^{-1}[n-1]\vec{x}[n]e[n]$$

Ved también

Hemos visto la ecuación de adaptación del filtro LMS en el apartado 3 de este módulo didáctico.

donde $\mu[n] = \frac{1}{\lambda + (\vec{x}[n])^T \mathbf{R}_{xx}^{-1}[n-1]\vec{x}[n]}$ es una constante de adaptación variable.

Interpretación

A continuación, deducida la expresión iterativa para la adaptación del filtro FIR según el criterio de los mínimos cuadrados recursivos, podemos realizar una interpretación de los términos que en esta aparecen, contrastándolos con los que aparecen en la ecuación de adaptación del filtro adaptativo que hace uso del criterio LMS.

Para empezar, vemos que si comparamos estas dos versiones, llaman la atención dos aspectos muy concretos: la existencia de un parámetro de adaptación μ adaptativo para la solución RLS en lugar de ser un parámetro constante, como lo es para la versión LMS; y la existencia de una matriz de decorrelación de los datos de entrada, $\mathbf{R}_{xx}^{-1}[n-1]$, para la versión RLS, no existiendo en cambio para el filtro LMS.

A continuación, podemos concretar qué nos aportan ambas diferencias:

- **Matriz de decorrelación de la señal de entrada**: esta matriz multiplica por la izquierda el vector de muestras de señal de entrada, $\vec{x}[n]$, por lo que su misión es justamente conseguir decorrelar los datos de entrada y conseguir, por lo tanto, un vector $\vec{w}[n] = \mathbf{R}_{xx}^{-1}[n-1]\vec{x}[n]$ de $N+1$ variables decorreladas entre ellas. Este proceso permite que la convergencia de la solución RLS se enfoque de forma más precisa en el sentido del mínimo error cuadrático medio definido al principio de todo, evitando así las convergencias más lentas típicas de la solución LMS, especialmente cuando el vector de señal de entrada está formado por variables muy correladas entre ellas. Recordemos, además, que la versión adaptativa del filtro RLS se ha deducido sin realizar ninguna aproximación, a diferencia del algoritmo LMS, el cual sí que es una versión aproximada del algoritmo del gradiente. Por ello, el filtro RLS consigue en cada iteración minimizar el error definido de forma instantánea y sin hacer uso de valores esperados.

- **Parámetro de adaptación μ inteligente**: A diferencia de la versión LMS, el filtro RLS adaptativo dispone de un parámetro de aprendizaje inteligente, que crece o decrece en función de la similitud de los nuevos datos proporcionados con respecto a los datos procesados hasta el momento. Cuando los datos que procesa el filtro, que están incluidos dentro de las posiciones del vector de señal $\vec{x}[n]$, son datos similares a los obtenidos hasta el momento, el término $(\vec{x}[n])^T \mathbf{R}_{xx}^{-1}[n-1]\vec{x}[n]$ dará lugar a un valor elevado, lo que hará que el parámetro de adaptación tenga un valor pequeño, y permitirá aproximarse mejor a un error cada vez menor. Esto sucederá, por ejemplo, durante los periodos de tiempo en los que el sistema que se está siguiendo (en el caso de la aplicación del filtrado adaptativo, el problema

de la estimación de sistemas lineales variantes en el tiempo) no tenga variaciones temporales significativas, consiguiendo por lo tanto una buena aproximación al error mínimo. No obstante, cuando el sistema varía de forma abrupta, es necesario aumentar el valor del parámetro de adaptación para seguir mejor al sistema, sacrificando capacidad para aproximarse de forma precisa al error mínimo. Esto último significa que el término $(\vec{x}[n])^T \mathbf{R}_{xx}^{-1}[n-1]\vec{x}[n]$ dará valores pequeños, pues las últimas muestras de señal observadas por el filtro estarán poco correladas con las observadas hasta entonces, por lo que el parámetro μ dará valores mayores y se consigue así la capacidad de seguir grandes cambios en los coeficientes del sistema.

Resumen del algoritmo RLS adaptativo

A continuación se resumen las ecuaciones del filtro RLS adaptativo, donde la matriz inversa de correlación de la señal de entrada, $\mathbf{R}_{xx}^{-1}$, ha sido renombrada a **P**:

1) Inicialización

$$\mathbf{P}[-1] = \delta \mathbf{I} \qquad (\delta = 1/\hat{P}_x)$$

$$\vec{h}[-1] = \vec{x}[-1] = [0,0,\ldots,0]^T$$

2) Para $n \geq 0$:

$$\mathbf{K}[n] = \frac{\mathbf{P}[n-1]\vec{x}[n]}{\lambda + (\vec{x}[n])^T \mathbf{P}[n-1]\vec{x}[n]}$$

$$\mu[n] = \frac{1}{\lambda + (\vec{x}[n])^T \mathbf{P}[n-1]\vec{x}[n]}$$

$$\vec{h}[n] = \vec{h}[n-1] + \mu[n]\mathbf{P}[n-1]\vec{x}[n]e[n]$$

$$\mathbf{P}[n] = \left(\frac{1}{\lambda}\right)\left(\mathbf{P}[n-1] - \mathbf{K}[n](\vec{x}[n])^T \mathbf{P}[n-1]\right)$$

La inicialización del filtro RLS implica la puesta a cero de las condiciones iniciales, tanto para el vector de muestras de señal como para el vector asociado a la respuesta impulsional del filtro FIR. Además, la matriz inversa de correlación se puede inicializar a una matriz diagonal multiplicada por el factor de inicialización δ, el cual se puede corresponder con el inverso de la estimación de la potencia de señal de entrada ($1/\hat{P}_x$). Este factor tampoco es crítico, y generalmente hay un buen rango de valores que se puede usar como valor de inicialización.

En cuanto a las ecuaciones de funcionamiento del filtro RLS, podemos ver en primer lugar el cálculo del vector ganancia de Kalman, $\mathbf{K}[n]$, las ecuaciones de cálculo del parámetro de adaptación $\mu[n]$ así como del vector de coeficientes del filtro adaptativo $\vec{h}[n]$, y finalmente, la ecuación recursiva asociada al cálculo de la matriz inversa de correlación de la señal de entrada, $\mathbf{P}[n]$, que se usará en la siguiente iteración del filtro.

Actividades

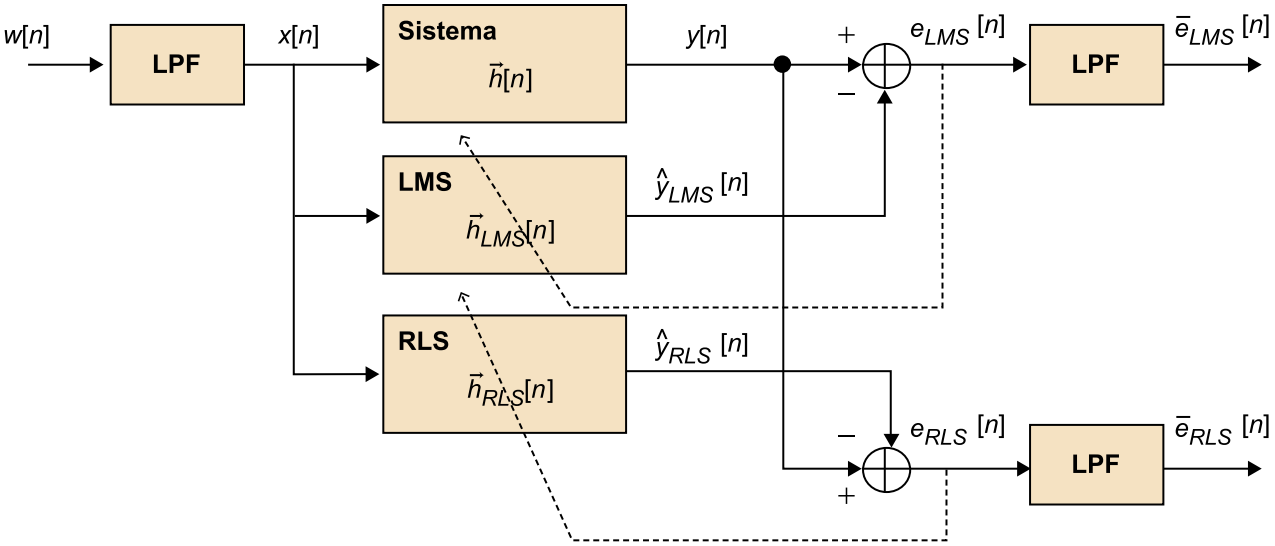
Programar en Matlab® los algoritmos LMS y RLS aplicados a un problema de identificación de sistemas lineales y variantes en el tiempo (tipo FIR). Estudiar, a través de la simulación, la velocidad de convergencia de ambos algoritmos y su dependencia con respecto a las características de correlación de la señal de entrada. Para ello, comparar tanto el error de salida $e[n] = y[n] - \hat{y}[n]$ de ambos algoritmos como también el error de estimación de la respuesta impulsional del sistema, definido como:

$$e_h[n] = \| \vec{h}[n] - \hat{h}[n] \|_2^2$$

siendo $\vec{h}[n]$ el vector de la respuesta impulsional del sistema, y $\hat{h}[n]$ el vector de la respuesta impulsional del estimador, ambos sistemas de tipo FIR.

El esquema es el que se dibuja en la figura 15. Como se puede apreciar, la señal de entrada al sistema, $x[n]$, se generará a partir de una señal de ruido blanco y gaussiano de variancia 1 y media 0 ($w[n]$) filtrado a través de un filtro paso bajo (LPF) de *smoothing*, como el analizado en el subapartado 4.4.2. Este filtro permitirá introducir cierta correlación en la señal de entrada al sistema, así como también los filtros adaptativos que realizan la tarea de estimación o identificación, dado que estos están situados en paralelo junto con el sistema y tienen la misma señal de entrada. El factor λ_x del filtro LPF de la señal de entrada permitirá generar una señal decorrelada (para $\lambda_x = 0$, tenemos un filtro paso-todo) o bien una señal muy correlada ($\lambda_x \to 1$, tenemos un filtro paso-bajo con frecuencia de corte muy pequeña).

Figura 15. Diagrama de bloques del ejercicio de simulación



Los errores de salida de los filtros adaptativos también se promediarán con filtros de *smoothing*, con el fin de poder apreciar mejor las diferencias entre el comportamiento de ambos filtros.

Del mismo modo, el sistema a simular será un sistema FIR (cuya longitud en número de coeficientes se fijará igual al número de coeficientes de los estimadores LMS y RLS) variante con el tiempo.

La variabilidad del sistema se implementará a partir de otro filtro de *smoothing*, pero que trabajará con todo el vector de muestras, el cual se alimentará a partir de un vector aleatorio con ruido blanco gaussiano. Para generar las muestras de entrada al filtro, cada muestra del vector estará totalmente decorrelada de las otras. La ecuación de adaptación del modelo de sistema variante será el siguiente:

$$\vec{h}[n] = \lambda_s \vec{h}[n-1] + (1-\lambda_s)\vec{q}[n]$$

donde $\vec{q}[n]$ es un vector con muestras de ruido blanco Gaussiano (incorreladas entre ellas) de variancia 1 y media 0.

De este modo, la constante λ_s permitirá controlar la velocidad de variación del sistema, siendo esta elevada si $\lambda_s \to 0$ (en este caso, $\vec{h}[n] = \vec{q}[n]$, y cada componente del vector $\vec{h}[n]$ será to-

talmente independiente de sus anteriores valores), mientras que el sistema variará más lentamente si $\lambda_s \to 1$ (el ruido blanco gaussiano filtrado con un filtro paso-bajo de frecuencia de corte baja tendrá una evolución más lenta).

La función será un *script* en la que se definan las siguientes variables de simulación:

- Parámetros generales de simulación
  - Número de coeficientes del sistema (*N*)
  - Número de muestras de señal a simular (*M*)
  - Factor de correlación de la señal de entrada (*lambda_x*)
  - Número de muestras entre visualizaciones del error (*L*)
  - Factor de promediado para el error de los filtros (*lambda_E*)

- Parámetros del sistema variante con el tiempo
  - Factor de memoria del sistema (*lambda_S*)

- Parámetros del algoritmo LMS
  - Constante de adaptación (*mu_LMS*)

- Parámetros del filtro RLS
  - Constante de inicialización de la matriz **P** (*delta_RLS*)
  - Factor de memoria (*lambda_RLS*)

El parámetro *L* permitirá modificar el número de muestras entre visualizaciones de los errores, tanto de salida como de estimación. La función *script* mostrará los resultados mediante una figura en la que se dibujará:

- La autocorrelación de las señales $w[n]$ y $x[n]$, con un retraso de ±20 muestras.
- Los errores de salida de ambos filtros promediados con el filtro de *smoothing* ($\bar{e}_{LMS}[n]$ y $\bar{e}_{RLS}[n]$).
- Los errores de estimación de ambos filtros promediados con el filtro de *smoothing* ($\bar{e}_h^{LMS}[n]$ y $\bar{e}_h^{RLS}[n]$).
- La respuesta impulsional original así como la estimada por ambos filtros ($\vec{h}[n]$, $\vec{h}_{LMS}[n]$, $\vec{h}_{RLS}[n]$).

Comparar los errores de salida y de estimación de ambos sistemas para un orden del sistema $N = 5$, escogiendo un número total de muestras de $M = 10.000$, un número de muestras entre visualizaciones del error de $L = 100$, y un factor de promediado de los errores de los filtros de *lambda_E* = 0,99. Realizar el estudio de los errores de salida tanto para un sistema invariante con el tiempo (*lambda_S* = 1) como para un sistema variante (con *lambda_S* = 0,999), así como para una señal de entrada decorrelada (*lambda_x* = 0) como para una señal de entrada correlada (*lambda_x* = 0,95). En cada escenario, escoger las constantes de adaptación de los filtros (μ del filtro LMS, y λ del filtro RLS) que garanticen una rápida convergencia así como la estabilidad del algoritmo.

Nota

En este ejercicio práctico se ha definido *N* como el número de coeficientes del filtro FIR, para mayor simplicidad, aunque en los desarrollos teóricos previos se ha considerado que el número de coeficientes del filtro era $N + 1$.

Solucionario

A continuación se muestra el código del *script* Matlab® que realiza la simulación que se pide en el ejercicio. Primero se muestra la parte del código en la que se definen las constantes o parámetros de simulación, donde se puede apreciar que se simula una situación en la que la señal de entrada al sistema y a los filtros está correlada (*lambda_x* = 0,95).

La inicialización incluye:

- La de los vectores de respuesta impulsional, de *N* muestras, tanto de los filtros como del sistema (*h_LMS*, *h_RLS*, *h*), siendo la de los sistemas a un vector columna de ceros, mientras que la del sistema a un vector columna aleatorio (utilizando la función *randn*).
- La matriz **P** del filtro *RLS* (utilizando la función eye que define una matriz identidad).
- Vectores de muestras para la visualización de las diferentes señales de *M* o de *N* puntos (*nv*, *nh*).
- Vector de *N* muestras consecutivas de la señal de entrada, *xv*, inicializado a ceros, que se utilizará tanto para generar la salida del sistema como la de ambos filtros adaptativos.
- Vectores de M muestras, asociados a los errores de salida y de estimación de ambos filtros adaptativos (*e_LMS*, *e_RLS*, *eh_LMS*, *eh_RLS*), inicializados a vectores con ceros.

```
%% Parámetros generales de la simulación
%-------------------------------------------------------------------------
N = 5;                  % Número de coeficientes del sistema
M = 10000;              % Número de muestras a simular
Lambda_x = 0.95;         % Factor de correlación de la señal de entrada
L = 100;                % Número de muestras entre visualizaciones del error
Lambda_E = 0.99;        % Factor de promediado para el error de los filtros

%% Sistema variante con el tiempo
%-------------------------------------------------------------------------
Lambda_S = 0.999;       % Factor de memoria del sistema


%% Parámetros del filtro LMS
%-------------------------------------------------------------------------
mu_LMS = 0.1;               % Constante de adaptación

%% Parámetros del Filtro RLS
%-------------------------------------------------------------------------
Lambda_RLS = 0.8;       % Factor de memoria
delta_RLS = 1;          % Constante de inicialización de la matriz P

%% Inicialización
%-------------------------------------------------------------------------
h_LMS = zeros(N,1);
h_RLS = zeros(N,1);
P = delta_RLS*eye(N);
h = randn(N,1);
nv = 1:M;
nh = 1:N;
xv = zeros(N,1);
e_LMS = zeros(M,1);
e_RLS = zeros(M,1);
eh_LMS = zeros(M,1);
eh_RLS = zeros(M,1);
minh = min(h);

%% Generación de la señal de entrada
w = randn(M,1);

% Filtrado paso-bajo de la señal (correlación)
x = filter(1-Lambda_x,[1 -Lambda_x], w);

% Cálculo correlaciones original y señal filtrada
Rww = xcorr(w,20,'unbiased');
Rxx = xcorr(x,20,'unbiased');
maxh = max(x);

%% Visualización resultados
figure(1);clf
subplot(221);plot(-20:20,Rww/max(Rww),-20:20,Rxx/max(Rxx));axis
tight;grid;title('Autocorrelación');
legend('Original','Entrada x[n]')
```

La segunda parte del código mostrada es la que realiza el bucle general de la simulación. En la visualización de los resultados se ha empleado la función *semilogy*, que permite usar un eje de abscisas logarítmico, más apropiado para mostrar los errores tanto de salida como de estimación.

Para cada una de las 4 situaciones simuladas (entrada correlada o incorrelada, y sistema invariante o variante) se han escogido los valores de las constantes de adaptación (*mu_LMS* y *lambda_RLS*) que permiten obtener errores de estimación menores. En el caso del algoritmo RLS, la constante *lambda_RLS* se define entre 0 y 1, mientras que en el caso del algoritmo LMS se puede escoger una *mu_LMS* mayor que uno, para acelerar aún más la convergencia. No obstante, un valor demasiado elevado puede producir inestabilidad del algoritmo, aspecto que debe evitarse.

```
%% Bucle de simulación general
for n = 1:M
    % Sistema
    %-------------------------------------------------------------------
    xv = [x(n);xv(1:(N-1))];
    y = transpose(xv)*h;
    h = Lambda_S*h + (1-Lambda_S)*randn(N,1);

    % Filtro LMS
    %-------------------------------------------------------------------
    y_LMS = transpose(xv)*h_LMS;
    ei_LMS = (y - y_LMS);
    e_LMS(n+1) = e_LMS(n)*Lambda_E + (1-Lambda_E)*ei_LMS;
    h_LMS = h_LMS + mu_LMS*xv*ei_LMS;

    eih_LMS = norm(h - h_LMS)^2;
    eh_LMS(n+1) = eh_LMS(n)*Lambda_E + (1-Lambda_E)*eih_LMS;

    % Filtro RLS
    %-------------------------------------------------------------------
    mu_RLS = 1/(Lambda_RLS + transpose(xv)*P*xv);
    K = (P*xv)*mu_RLS;
    y_RLS = transpose(xv)*h_RLS;
    ei_RLS = (y - y_RLS);
    e_RLS(n+1) = e_RLS(n)*Lambda_E + (1-Lambda_E)*ei_RLS;
    h_RLS = h_RLS + mu_RLS*P*xv*ei_RLS;
    P = (1/Lambda_RLS)*(P - K*transpose(xv)*P);

    eih_RLS = norm(h - h_RLS)^2;
    eh_RLS(n+1) = eh_RLS(n)*Lambda_E + (1-Lambda_E)*eih_RLS;

    %% Visualización resultados
    if (max(h)>maxh)
        maxh = max(h);
    end
    if (min(n)<minh)
        minh = min(h);
    end
    if (mod(n,L) == 0)
        figure(1);
        subplot(222);semilogy(nv,abs(e_LMS),nv,abs(e_RLS));
        axis tight;grid;title('Error Salida')
        legend('LMS','RLS');
        subplot(223);plot(nh,h,nh,h_LMS,nh,h_RLS);
        axis tight;grid;title('Respuesta Impulsional')
        %axis([1 N minh*0.7 maxh*0.7])
        legend('Sistema','LMS','RLS');
        subplot(224);semilogy(nv,abs(eh_LMS),nv,abs(eh_RLS));
        axis tight;grid;title('Error Estimación')
        legend('LMS','RLS');
    end
end
```

En la tabla siguiente se muestran los valores de los parámetros que permiten obtener las figuras de resultados que se muestran a continuación.

	Lambda_x	Lambda_S	mu_LMS	Lambda_RLS
Sistema invariante y entrada incorrelada	0	1	0,2	0,9
Sistema invariante y entrada correlada	0,95	1	2	0,8
Sistema variante y entrada incorrelada	0	0,999	0,1	0,7
Sistema variante y entrada correlada	0,95	0,999	0,1	0,7

En todos los casos se han escogido los factores de adaptación (*mu_LMS* y *lambda_RLS*), que conseguían una convergencia rápida sin llegar a divergir. A continuación, en las figuras 16, 17, 18 y 19 se muestran los resultados obtenidos para cada una de las 4 simulaciones realiza-

das. Como se puede apreciar, en todos los casos el algoritmo RLS consigue errores de salida así como de estimación de la respuesta impulsional menores, aunque esta diferencia se acentúa especialmente cuando la señal de entrada está correlada. Recordemos que el algoritmo RLS es una técnica que mejora la convergencia al aplicar una decorrelación de los datos de entrada de forma dinámica, y que su constante de adaptación inteligente permite adaptarse mejor a las condiciones de variación de un sistema dinámico.

Figura 16. Resultados de la simulación con un sistema invariante y entrada incorrelada

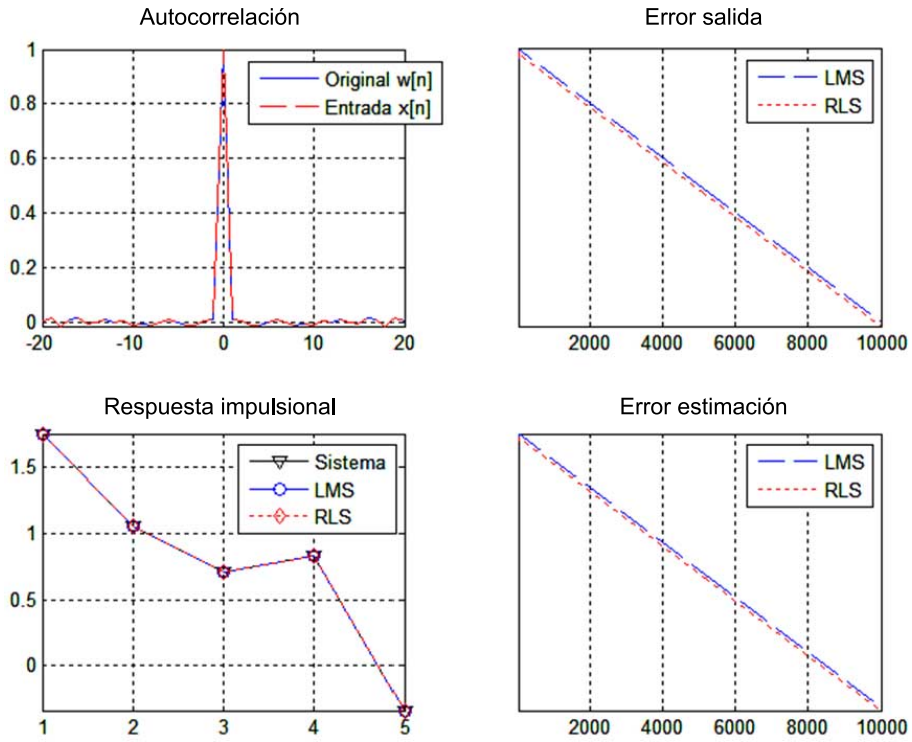


Figura 17. Resultados de la simulación con un sistema invariante y entrada correlada

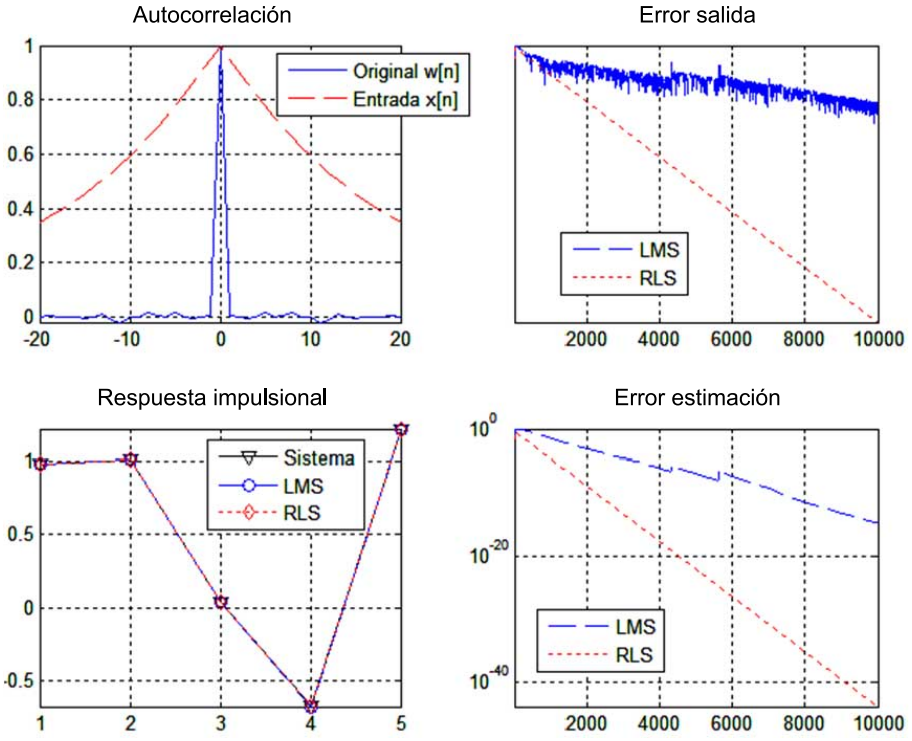


Figura 18. Resultados de la simulación con un sistema variante y entrada incorrelada

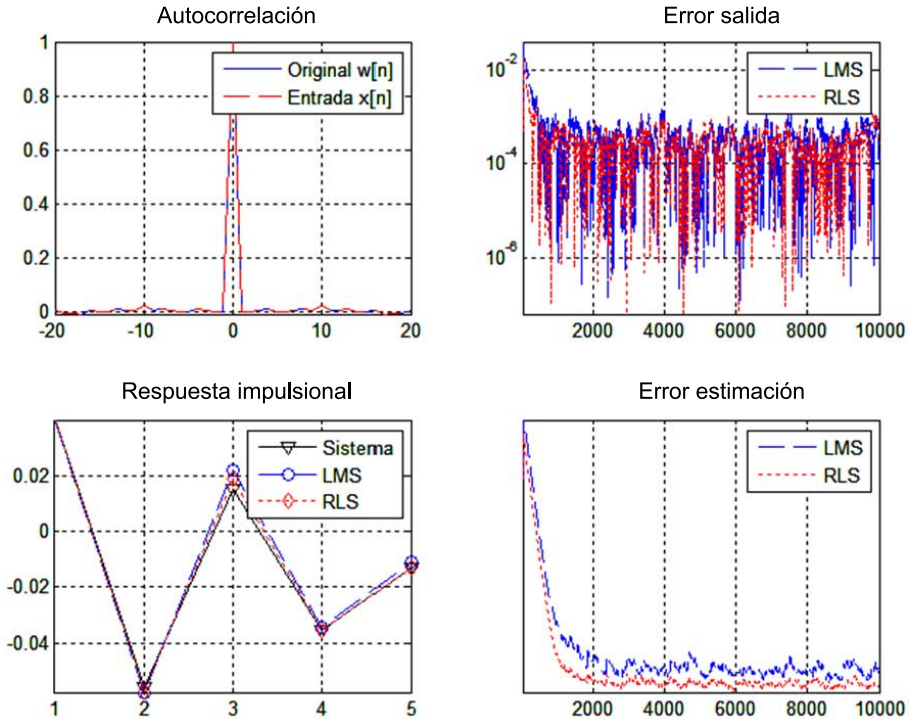


Figura 19. Resultados de la simulación con un sistema variante y entrada correlada