TECNIACUSTICA MURCIA · 2014

45º CONGRESO ESPAÑOL DE ACÚSTICA
8º CONGRESO IBÉRICO DE ACÚSTICA
EUROPEAN SYMPOSIUM ON SMART CITIES AND ENVIRONMENTAL ACOUSTICS

ALGORITMO NLMS-OCF EN EL DOMINIO DE LA FRECUENCIA APLICADO A CONTROL ACTIVO DE RUIDO

REFERENCIA PACS: 43.50.Ki

Autores: Lorente, Jorge[1]; Ferrer, Miguel[2], De Diego, María[2], González, Alberto[2].

1 Instituto de Telecomunicaciones y Aplicaciones Multimedia, Universitat Politècnica de València, Camino de Vera s/n, jorlogi@iteam.upv.es
2 Departamento de comunicaciones, Universitat Politècnica de València, Camino de Vera s/n, {mferrer, mdediego, agonzal}@dcom.upv.es

ABSTRACT
The Normalized Least Mean Square (NLMS) algorithm is a widely used adaptive algorithm for Active Noise Control (ANC). One of the major drawbacks is its slow convergence. The speed of convergence can be increased by using the NLMS algorithm with orthogonal correlation factors (NLMS-OCF), but that also increases the computational complexity and limits the usage of the NLMS-OCF in massive real-time applications. In this paper we propose the NLMS-OCF algorithm in frequency domain for multichannel ANC systems. This implementation of the algorithm enables their parallel implementation using a Graphics Processing Unit (GPU) which would help to overcome the computational drawback in massive multichannel ANC systems.

RESUMEN
El algoritmo NLMS (*Normalized Least Mean Square*) es uno de los más usados en los sistemas de Control Activo de Ruido (CAR) aunque no proporciona una convergencia rápida. El algoritmo NLMS con factores de corrección ortogonal (NLMS-OCF) aumenta su velocidad de convergencia, pero también su coste computacional y ello limita su uso en aplicaciones en tiempo real. En este artículo proponemos el algoritmo NLMS-OCF en el dominio de la frecuencia aplicado a un sistema CAR multicanal. Esta implementación del algoritmo permite su implementación eficiente mediante la programación paralela en Unidades de Procesamiento Gráfico (GPU) y así se consigue paliar significativamente su desventaja computacional en sistemas multicanales masivos de CAR.

1. INTRODUCCIÓN

El Control Activo de Ruido (CAR) [1,2] se basa en el principio de interferencias destructivas entre una o varias señales de ruido llamadas ruido primario y una o varias señales secundarias generadas por fuentes secundarias controladas llamadas actuadores. El objetivo es cancelar, o al menos minimizar el ruido acústico. Para cancelar dicho ruido, el sistema CAR normalmente usa algoritmos adaptativos para generar las señales secundarias a partir de unas señales de referencia correladas con cada ruido primario. Con este objetivo, el ruido o señal indeseada se monitoriza en uno o varios puntos espaciales específicos mediante sensores que se denominan sensores de error. Por lo tanto, la cancelación se produce en el entorno de dichos puntos. Los sistemas multicanal tienen una complejidad computacional elevada, siendo esto su elevado coste computacional uno de sus principales cuellos de botella a la hora de su implementación práctica.

El algoritmo *Normalized Least Mean Squares* (NLMS) usando tanto el esquema de filtrado-x (FxNLMS) [3] como el de filtrado-x modificado (MFxNLMS) [4] y sus versiones multicanal [2], son las estrategias de filtrado adaptativo más usadas en aplicaciones de CAR, ya que son algoritmos fáciles de implementar y computacionalmente simples, pero la velocidad de convergencia es una de sus mayores desventajas. En cambio, el algoritmo de Proyección Afín (APA) [5] aumenta la rapidez de convergencia. Existen diversas variantes del APA, por ejemplo: el APA regularizado (R-APA) [6], el algoritmo de rango parcial (PRA) [7], el algoritmo de decorrelación (DA) [8] o el NLMS con factores de corrección ortogonal (NLMS-OCF) [9]. Estos algoritmos, se suelen agrupar en lo que se denomina la familia APA [10]. Su idea básica es que utilizan múltiples vectores de las señales de referencia para adaptar los filtros adaptativos, mientras que el MFxNLMS usa sólo un único vector de la señal de referencia. Tradicionalmente, los algoritmos de la familia APA se han implementado en el domino del tiempo trabajando muestra a muestra para sistemas monocanales y aplicaciones de identificación de canal o cancelación de ecos. En este artículo, se propone el algoritmo NLMS-OCF usando el esquema de filtrado-x modificado (MFxNLMS-OCF) para sistemas CAR multicanal. De entre los algoritmos de la familia APA, se utiliza el NLMS-OCF ya que su implementación es más sencilla que el APA, y además se puede considerar una generalización del APA que permite usar retardos mayores a la unidad entre los distintos vectores de entrada usados en la actualización de los filtros adaptativos [9].

El uso de múltiples vectores de la señal de entrada para la actualización los filtros adaptativos supone una gran demanda computacional, especialmente en los casos multicanal. Esto limita seriamente la viabilidad de la implementación en prototipos CAR multicanales funcionando en tiempo real. Esta desventaja podría paliarse mediante la implementación en plataformas de computación paralela. Por ejemplo, una GPU (*Graphics Processing Unit*) podría acelerar el procesado mediante la paralelización de las operaciones. Para poder explotar los recursos de paralelización que ofrecen las GPUs en prototipos CAR multicanal, el algoritmo debe estar adaptado a sus requerimientos hardware, y por ello este artículo propone variar el algoritmo MFxNLMS-OCF al dominio de la frecuencia, con procesado de bloque y filtros particionados. A dicho algoritmo lo hemos denominado como *the Frequency Partitioned Block Modified Filtered-X Normalized Least Mean Square algorithm with Orthogonal Correlation Factors* (FPBMFxNLMS-OCF). Las razones que nos han llevado a su implementación son las siguientes: la implementación en el dominio de la frecuencia permite una implementación más rápida, las tarjetas de audio requieren trabajar con procesado de bloques en lugar de procesado muestra a muestra, y además trabajando con bloques de datos se explotan mejor los recursos paralelos de la GPU. Finalmente, teniendo en cuenta que los filtros adaptativos pueden ser más largos que el tamaño del bloque de datos, los filtros adaptativos deben ser particionados [11], y por lo tanto, la paralelización del algoritmo se mejora al ejecutar la adaptación de cada partición del filtro al mismo tiempo.

Este artículo se organiza del siguiente modo: la sección 2 describe los algoritmos; la sección 3 presenta los resultados y la sección 4 las conclusiones. La notación usada en este artículo es la siguiente: los símbolos con negrita se usan para vectores, los símbolos con letra minúscula para el dominio del tiempo, mientras que con letra mayúscula para el dominio de la frecuencia; el símbolo $||\cdot||^2$ denota la norma 2 de un vector, mientras que el símbolo $(\cdot)^*$ denota complejo conjugado. Por último, el símbolo $\circ$ denota el producto elemento a elemento entre dos vectores.

2. DESCRIPCIÓN DE LOS ALGORITMOS

Esta sección se centra en describir el algoritmo MFxNLMS-OCF para un sistema CAR multicanal. El algoritmo ha sido derivado del algoritmo NLMS-OCF publicado en [9] para una aplicación de identificación de canal. En los sistemas de identificación de canal, la señal de error se corresponde con el error resultante de estimar la denominada señal deseada que recogen los micrófonos. Por el contrario, en un sistema CAR, los micrófonos recogen la mezcla de las señales de ruido o indeseada y las señales secundarias usadas para cancelar el ruido.

Por lo tanto, las señales indeseadas (necesarias para calcular las señales de error [4]), no están disponibles. Esta es la razón por la que usamos del esquema de filtrado-x modificado, ya que nos permite estimar las señales indeseadas. Finalmente, con el objetivo de adecuar el algoritmo a una versión que permita una implementación paralela más eficiente, se describe el algoritmo FPBMFxNLMS-OCF, que es la versión en el dominio de la frecuencia, trabajando por bloques y con los filtros adaptativos particionados del algoritmo MFxNLMS-OCF. En las siguientes subsecciones, se describen los algoritmos aplicados a sistemas CAR multicanal (*I*:*J*:*K*), siendo I el número de señales (sensores) de referencia, J el número de señales (altavoces) secundarias y K el número de señales (sensores) de error. Además, se definen *L* y *M* como la longitud de los filtros adaptativos y de los filtros FIR que modelan los caminos acústicos (secundarios) estimados, que enlazan los altavoces con los sensores de error.

2.1 El algoritmo NLMS con esquema de filtrado-x modificado y factores de corrección ortogonal (MFxNLMS-OCF)

El algoritmo NLMS-OCF fue introducido en [9] para un sistema de identificación de canal. Este algoritmo actualiza los filtros adaptativos utilizando múltiples vectores de entrada, mientras que el NLMS utiliza un solo vector. Se define *R* como el número de vectores sucesivos usados para adaptar los filtros adaptativos en el algoritmo NLMS-OCF y sus variantes. Además, el parámetro *D* define el número de iteraciones existentes entre los sucesivos vectores de entrada que se utilizan para la actualización de los filtros adaptativos. La mejora en la velocidad de convergencia es mayor si los sucesivos vectores de entrada utilizados son ortogonales. Si los vectores de entrada no son ortogonales, en [9] se propone ortogonalizar dichos vectores. En este artículo proponemos el algoritmo MFxNLMS-OCF a partir del MFxNLMS para sistemas CAR, de una forma similar a la que se utiliza en [9] para obtener el NLMS-OCF a partir del NLMS para sistemas de identificación de canal. Su versión multicanal se compone de los siguientes pasos:

1. Cálculo de las señales de salida.

La *j*-ésima señal de salida en el *q*-ésimo instante se calcula como

$$y_j(q) = \sum_{i=1}^{I} \mathbf{w}ij_q^T \mathbf{x}i_{L_q}, \tag{1}$$

donde $\mathbf{x}i_{L_q} = \left[xi(q) \quad xi(q-1) \quad xi(q-2) \quad \ldots \quad xi(q-L+1)\right]^T$ es el *i*-ésimo vector de señal de entrada que contien las últimas *L* muestras desde la *q*-ésima muestra de dicha señal.

2. Cálculo de la estimación de las señales indeseadas.

$$\hat{d}_k(q) = e_k(q) - \sum_{j=1}^{J} \mathbf{s}jk^T \mathbf{y}j_{M_q}, \tag{2}$$

donde $\hat{e}_k(q)$ es la *q*-ésima muestra de la *k*-ésima señal de error adquirida por el *k*-ésimo sensor de error, y $\mathbf{y}j_{M_q} = \left[yj(q) \quad yj(q-1) \quad yj(q-2) \quad \ldots \quad yj(q-M+1)\right]^T$ es el *j*-ésimo vector de señal de salida que contiene las últimas *M* muestras desde la *q*-ésima muestra. $\mathbf{s}jk$ es la estimación de tamaño M del camino secundario que une la j-ésima fuente secundaria con el k-ésimo sensor de error.

3. Actualización de los pesos de los filtros adaptativos

$$\mathbf{w}ij_{q+1} = \mathbf{w}ij_q - \sum_{k=1}^{K} \hat{\mu}_{ijk}^0 \mathbf{v}ijk_{Lq}^{\ 0} - \sum_{k=1}^{K} \hat{\mu}_{ijk}^1 \mathbf{v}ijk_{Lq}^{\ 1} - \ldots - \sum_{k=1}^{K} \hat{\mu}_{ijk}^R \mathbf{v}ijk_{Lq}^{\ R}, \tag{8}$$

donde $\mathbf{w}ij_q$ son los coeficientes del filtro adaptativo de tamaño L que filtra la *i*-ésima señal de referencia, obteniendo la señal generada por la *j*-ésima fuente secundaria en la muestra *q*. *R* es el número factores de corrección ortogonal, R+1 es el número de entrada usados para la

actualización de los pesos y $\mathbf{v}ijk_{L_q} = \left[v_{ijk}(q) \quad v_{ijk}(q-1) \quad \ldots \quad v_{ijk}(q-L+1)\right]^T$ es un vector con las últimas *L* muestras de la *i*-ésima señal de referencia filtrada por el *jk*-ésimo camino secundario, y se calcula como

$$v_{ijk}(q) = \mathbf{s}jk^T \mathbf{x}i_M, \tag{7}$$

Hay que tener en cuenta que $\mathbf{v}ijk_{L_q}^0, \mathbf{v}ijk_{L_q}^1, \ldots, \mathbf{v}ijk_{L_q}^R$ son ortogonales entre sí. Definimos $\mathbf{v}ijk_{L_q}^r$ como el vector obtenido a partir de $\mathbf{v}ijk_{L_{q-rD}}$ y que es ortogonal a $\mathbf{v}ijk_{L_{q-D}}, \mathbf{v}ijk_{L_{q-2D}}, \ldots, \mathbf{v}ijk_{L_{q-(r-1)D}}$. La ortogonalización de los vectores se puede realizar mediante la técnica de Gram-Smidth. Generalizando para un valor de *r*, la variable de paso $\hat{\mu}_{ijk}^r$ se calcula como

$$\hat{\mu}_{ijk}^r = \begin{cases} \mu \dfrac{\hat{e}_k^r(q)}{\left\|\mathbf{v}ijk_{L_q}^r\right\|^2}, & si \quad \left\|\mathbf{v}ijk_{L_q}^r\right\|^2 \neq 0 \\ 0, & otros \end{cases}. \tag{10}$$

El error estimado se calcula del siguiente modo

$$\hat{e}_k^r(q) = \hat{d}_k(q - rD) + \sum_{i=1}^{I}\sum_{j=1}^{J} \mathbf{w}ij_{q+1}^{r\,T} \mathbf{v}ijk_{L_{q-rD}}, \tag{11}$$

Por último, la actualización de los pesos para el vector de entrada *r*, se calcula como

$$\mathbf{w}ij_{q+1}^r = \mathbf{w}ij_q - \sum_{k=1}^{K} \hat{\mu}_{ijk}^0 \mathbf{v}ijk_{L_q}^0 - \sum_{k=1}^{K} \hat{\mu}_{ijk}^1 \mathbf{v}ijk_{L_q}^1 - \ldots - \sum_{k=1}^{K} \hat{\mu}_{ijk}^{r-1} \mathbf{v}ijk_{L_q}^{r-1}. \tag{12}$$

Cuanto mayor sea el parámetro R, mayor será la velocidad de convergencia, aunque la elección de *R* debe depender de la capacidad computacional disponible, ya que al aumentar *R*, se aumenta mucho la complejidad del algoritmo. Por otra parte, existe un valor de *R* donde la velocidad de convergencia se satura, y por lo tanto no tiene sentido utilizar valores de *R* más grandes. En cuanto al parámetro *D*, se obtiene mayor velocidad de convergencia para valores grandes de *D*, especialmente si el valor de *R* es pequeño. Esto es debido a que cuanto mayor es *D*, más separados están los sucesivos vectores de entrada que se usan en la actualización de los filtros adaptativos, y por tanto las señales están más incorreladas.

2.2 Algoritmo NLMS de bloque, particionado, en el dominio de la frecuencia, usando el esquema de filtrado-x modificado y factores de corrección ortogonal (FPBMFxNLMS-OCF)

Este algoritmo se obtiene a partir del MFxNLMS-OCF. Las muestras se procesan en bloques de tamaño *B*. Si *L* y *M* son mayores que *B*, tanto los filtros adaptativos como los que modelan los caminos secundarios deben ser particionados en *F* y *P* particiones respectivamente [4]. De ese modo, el algoritmo trabaja simultáneamente con todas las particiones de tamaño *B*. En la descripción del algoritmo, el subíndice *n* y los superíndices *f* y *p* denotan la iteración y la partición respectivamente. Los siguientes pasos describen el algoritmo FPBMFxNLMS-OCF:

1. <u>Cálculo de las señales de salida,</u>

$$\mathbf{Y}j_N = \sum_{i=1}^{I}\sum_{f=1}^{F} \mathbf{w}ij_n^f \circ \mathbf{x}i_{n-f+1}, \tag{13}$$

donde $\mathbf{x}i_n = \mathrm{FFT}\left[\mathbf{x}i_{B_{n-1}} \quad \mathbf{x}i_{B_{n-1}}\right]$, y $\mathbf{x}i_{B_n} = \left[xi(Bn) \quad xi(Bn-1) \quad \ldots \quad xi(Bn-B+1)\right]^T$. $\mathbf{W}ij_n^f$ es la FFT de tamaño 2B de la *f*-ésima partición del filtro adaptativo $\mathbf{w}ij$ en la iteración *n*-ésima. Las señales de salida $\mathbf{y}j_{B_n}$ son las últimas B muestras de la IFFT del vector $\mathbf{Y}j_N$.

2. Cálculo de la estimación de las señales indeseadas,

$$\hat{\mathbf{D}}k_n = \mathbf{E}k_n - \mathbf{YF}k_n, \tag{14}$$

donde $\mathbf{E}k_n$ es la k-ésima señal de micrófono en el dominio de la frecuencia, $\mathbf{E}k_n = FFT\left[\mathbf{0}_B \quad \mathbf{e}k_{B_n}\right]$. $\mathbf{e}k_{B_n}$ son los *B* muestras adquiridas por el *k*-ésimo micrófono en la *n*-ésima iteración. El vector $\mathbf{YF}k_n$ se define como

$$\mathbf{YF}k_n = \sum_{j=1}^{J}\sum_{p=1}^{P}\mathbf{Y}j_{n-p+1} \circ \mathbf{S}jk^p, \tag{15}$$

donde $\mathbf{S}ijk^p$ es la FFT de tamaño 2B de la *p*-ésima partición del camino acústico **s**jk.

3. Actualización de los pesos de los filtros adaptativos.

La actualización de los coeficientes de cada partición del *ij*-ésimo filtro adaptativo se calcula en el dominio de la frecuencia teniendo en cuenta los *R* vectores de referencia del siguiente modo

$$\mathbf{W}ij_{n+1} = \mathbf{W}ij_n - \sum_{k=1}^{K}\mathrm{FFT}\left[\boldsymbol{\varphi}ijk_0^f \quad \mathbf{0}_B\right] - \sum_{k=1}^{K}\mathrm{FFT}\left[\boldsymbol{\varphi}ijk_1^f \quad \mathbf{0}_B\right] - \ldots - \sum_{k=1}^{K}\mathrm{FFT}\left[\boldsymbol{\varphi}ijk_R^f \quad \mathbf{0}_B\right] \tag{16}$$

R es el número de vectores de entrada usados en la actualización. Los *R* vectores de entrada utilizados en la actualización son las señales de entrada filtradas con el correspondiente camino secundario, $\mathbf{V}ijk_n$, de las últimas *R·D* iteraciones. El vector $\mathbf{V}ijk_n$ se define como

$$\mathbf{V}ijk_n = \sum_{p=1}^{P}\mathbf{S}jk^p \circ \mathbf{X}i_{n-p+1}. \tag{17}$$

Generalizando para un vector de entrada *r*, el vector $\boldsymbol{\varphi}ijk_r^f$ corresponde a las *B* primeras muestras de la IFFT de tamaño *2B* de la partición *f* de la variable de paso $\tilde{\boldsymbol{\mu}}ijk_r^f$

$$\begin{gathered}\left[\boldsymbol{\varphi}ijk_r^f \quad \overline{\boldsymbol{\varphi}}ijk_r^f\right] = IFFT\left\{\tilde{\boldsymbol{\mu}}ijk_r^f\right\}, \\ f = n, n-1, \ldots, n-F, \text{ and } r = 1,2,\ldots,R.\end{gathered} \tag{18}$$

Definimos $\mathbf{V}ijk_{n-f+1}^r$ como el vector obtenido a partir de $\mathbf{V}ijk_{n-f+1-(rF)D}$ y que es ortogonal a $\mathbf{V}ijk_{n-f+1}, \mathbf{V}ijk_{n-f+1-(F)D}, \mathbf{V}ijk_{n-f+1-(2F)D}, \ldots, \mathbf{V}ijk_{n-f+1-((r-1)F)D}$. Definimos $\tilde{\boldsymbol{\mu}}ijk_r^f$ como

$$\tilde{\boldsymbol{\mu}}ijk_r^f = \begin{cases}\mu\dfrac{\hat{\mathbf{E}}k_n^r \circ \left(\mathbf{V}ijk_{n-f+1}^r\right)^*}{\left\|\mathbf{V}ijk_{n-f+1}^r\right\|^2}, & \text{if } \left\|\mathbf{V}ijk_{n-f+1}^r\right\|^2 \neq 0 \\ \mathbf{0}_{2B}, & \text{otherwise.}\end{cases}. \tag{19}$$

Las señales de error estimadas $\hat{\mathbf{E}}k_n^r$ se calculan como

$$\hat{\mathbf{E}}k_n^r = \hat{\mathbf{D}}k_{n-rD} + \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{f=1}^{F}\mathbf{V}ijk_{n-f+1-(rF)D} \circ \mathbf{W}ij_n^f, \tag{20}$$

Debido a que este algoritmo trabaja con bloques de muestras, incluso cuando *D=1*, los sucesivos vectores de entrada usados para la actualización están separados por al menos B muestras. Además cuando la señal de entrada $\mathbf{x}i$ es ruido blanco aleatorio, cada bloque de muestras $\mathbf{x}i_{B_n}$ es ortogonal a los bloques de las iteraciones anteriores. Esto significa que debido a la naturaleza de la señal de entrada $\mathbf{V}ijk_{n-f+1-rF}$ es ortogonal a $\mathbf{V}ijk_{n-f+1}, \mathbf{V}ijk_{n-f+1-(F)D}, \mathbf{V}ijk_{n-f+1-(2F)D}, \ldots, \mathbf{V}ijk_{n-f+1-((r-1)F)D}$, y por lo tanto no es necesario hacer el proceso de ortogonalización, usando $\mathbf{V}ijk_{n-f+1}^r = \mathbf{V}ijk_{n-f+1-rF}$. Éste será el caso de

estudio en este artículo. En la Tabla. 2 se encuentra un resumen de las instrucciones que realiza el algoritmo FPBMFxNLMS-OCF en cada iteración.

Tabla 2. Resumen de las instrucciones de los algoritmos MFxLMS-OCF y FPBMFxLMS-OCF.

Algoritmo FPBMFxNLMS-OCF

Repite los siguientes pasos cada nueva iteración:

(1) $\mathbf{Y}j_N = \sum_{i=1}^{I}\sum_{f=1}^{F} \mathbf{w}ij_n^f \circ \mathbf{x}i_{n-f+1}$

(2) $\hat{\mathbf{D}}k_n = \mathbf{E}k_n - \mathbf{YF}k_n$

(3) $\mathbf{YF}k_n = \sum_{j=1}^{J}\sum_{p=1}^{P} \mathbf{Y}j_{n-p+1} \circ \mathbf{S}jk^p$

(4) $\mathbf{V}ijk_n = \sum_{p=1}^{P} \mathbf{S}jk^p \circ \mathbf{X}i_{n-p+1}$

(5) $\hat{\mathbf{E}}k_n = \hat{\mathbf{D}}k_n + \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{f=1}^{F} \mathbf{V}ijk_{n-f+1} \circ \mathbf{W}ij_n^f$

(6) $\tilde{\boldsymbol{\mu}}_{ijk} = \mu \dfrac{\hat{\mathbf{E}}k_n \circ \left(\mathbf{V}ijk_{n-f+1}\right)^*}{\left\|\mathbf{V}ijk_{n-f+1}\right\|^2}$

(7) $\left[\boldsymbol{\varphi}ijk^f \quad \overline{\boldsymbol{\varphi}}ijk^f\right] = IFFT\left\{\tilde{\boldsymbol{\mu}}ijk^f\right\}$
$f = n, n-1, \ldots, n-F$

(8) $\mathbf{W}ij_{n+1}^{1\,f} = \mathbf{W}ij_n^{\,f} - \sum_{k=1}^{K} \mathrm{FFT}\left[\boldsymbol{\varphi}ijk_0^f \quad \mathbf{0}_B\right]$

Bucle '*FOR*', para cada 1 ≤ r ≤ R, ejecuta los siguientes pasos

(9) Calcula $\mathbf{V}ijk_{n-f+1}^r$ como el vector obtenido a partir de $\mathbf{V}ijk_{n-f+1-(rF)D}$ y que es ortogonal a $\mathbf{V}ijk_{n-f+1}$, $\mathbf{V}ijk_{n-f+1-(F)D}$, $\mathbf{V}ijk_{n-f+1-(2F)D}$, …, $\mathbf{V}ijk_{n-f+1-((r-1)F)D}$.

(10) $\hat{\mathbf{E}}k_n^r = \hat{\mathbf{D}}k_{n-rD} + \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{f=1}^{F} \mathbf{V}ijk_{n-f+1-(rF)D} \circ \mathbf{W}ij_n^f$

(11) $\tilde{\boldsymbol{\mu}}ijk_r^f = \begin{cases} \mu \dfrac{\hat{\mathbf{E}}k_n^r \circ \left(\mathbf{V}ijk_{n-f+1}^r\right)^*}{\left\|\mathbf{V}ijk_{n-f+1}^r\right\|^2}, & \text{si} \left\|\mathbf{V}ijk_{n-f+1}^r\right\|^2 \neq 0 \\ \mathbf{0}_{2B}, & \text{otros.} \end{cases}$

(12) $\left[\boldsymbol{\varphi}ijk_r^f \quad \overline{\boldsymbol{\varphi}}ijk_r^f\right] = IFFT\left\{\tilde{\boldsymbol{\mu}}ijk_r^f\right\}$
$f = n, n-1, \ldots, n-F,$ and $r = 1, 2, \ldots, R.$

(13) $\mathbf{W}ij_{n+1}^{r+1\,f} = \mathbf{W}ij_{n+1}^{r\;\;f} - \sum_{k=1}^{K} \mathrm{FFT}\left[\boldsymbol{\varphi}ijk_r^f \quad \mathbf{0}_B\right]$

Fin bucle '*FOR*'

(15) $\mathbf{W}ij_{n+1}^{\;\;f} = \mathbf{W}ij_{n+1}^{R-1\,f}$

3. RESULTADOS

Se ha analizado la velocidad de convergencia y el coste computacional del algoritmo FPBMFxNLMS-OCF. Para el análisis de la velocidad de convergencia, las medidas se han realizado en un prototipo CAR montado en una sala acústica. La velocidad de convergencia se ha analizado para distintos valores de *R*, comparándolo con el algoritmo FPBMFxNLMS (*R=0*).

3.1 Análisis de la velocidad de Convergencia

En esta sección se evalúa la velocidad de convergencia del algoritmo FPBMFxNLMS-OCF para diferentes valores del parámetro *R*. Para ello, se utiliza un sistema multicanal CAR con configuración 1:2:2 (I:J:K) con 1 fuente de ruido, 2 fuentes secundarias y 2 micrófonos. Se ha usado ruido blanco aleatorio como señal de ruido a cancelar. Además, se usan los siguientes valores para los parámetros: B=2048, L=M=4096, R=0 i R=4. Las curvas de aprendizaje del algoritmo que se usan para analizar la velocidad de convergencia son las siguientes:

$$A[n] = 10\log_{10}\left(\frac{P_e[n]}{P_d[n]}\right), \qquad (21)$$

con

$$P_e[n] = P_e[n-1] + (1-\alpha)p_e[n], \text{ siendo } p_e[n] = \sum_{k=1}^{K} e_k^2[n]$$
$$P_d[n] = P_d[n-1] + (1-\alpha)p_d[n], \text{ siendo } p_d[n] = \sum_{k=1}^{K} d_k^2[n] \quad (22)$$

La figura. 1 muestra que cuando el valor de R aumenta, el algoritmo FPBMFxNLMS-OCF ofrece mayor velocidad de convergencia. Sin embargo, se puede ver que existe un valor de R para el que la mejora en velocidad de convergencia se satura. Para este caso en particular, la saturación ocurre cuando se usan 12 o más vectores de entrada en la actualización de los filtros (R≥12). En cuanto a la mejora en el transitorio, se puede apreciar que cuando R=12, el algoritmo alcanza los 20 dB de atenuación con 0.8×10^6 muestras procesadas, mientras que cuando R=0, el algoritmo sólo necesita 0.1×10^6 muestras. Esto significa que la convergencia es 8 veces más rápida. Otra aspecto a reseñar es que, también se obtienen buenos resultados aunque se usen valores de R pequeños. Por ejemplo cuando R=1, el tiempo de convergencia es aproximadamente la mitad que con R=0. Por otra parte, también se puede apreciar que el estado estacionario es independiente del valor de R.

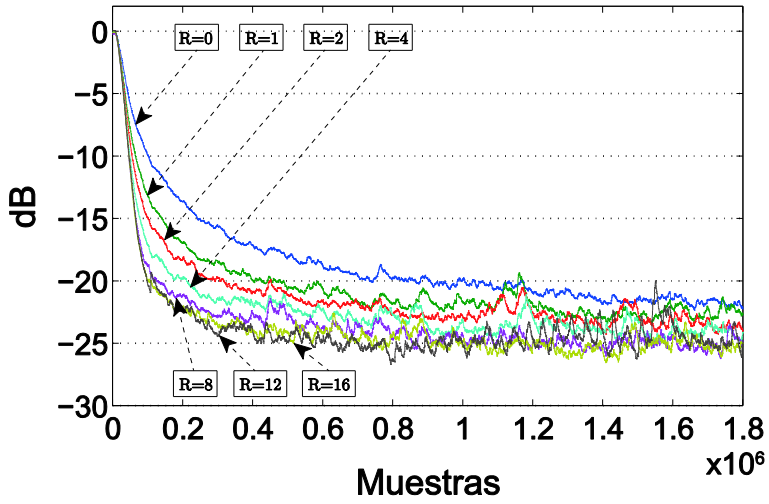

Figura 1.Curvas de aprendizaje del algoritmo FPBMFxNLMS-OCF para diferentes valores de *R,* y ruido de banda ancha como señal de referencia.

3.2 Complejidad Computacional

Como ya se ha comentado, la principal desventaja de este algoritmo es su coste computacional, especialmente en sistemas multicanal. En esta sección, se analiza el coste computacional del algoritmo para diferentes configuraciones I:J:K y variando el valor de R. Hay que tener en cuenta que cuando R=0, el algoritmo FPBMFxNLMS-OCF se convierte en el FPBMFxNLMS. El coste computacional en términos de multiplicaciones, sumas y número de FFTs se muetra en la Tabla. 3. El coste computacional se ha indicado en referencia a L. Para ello, se han usado los siguientes valores: M=L, B=L/2 (los filtros se dividen en 2 particiones). En primer lugar, la Tabla.3 muestra como para un valor fijo de R, la complejidad computacional aumenta significativamente cuando aumentan los canales. Como ejemplo, si nos fijamos en los valores cuando R=0 (algoritmo FPBMFxNLMS), si el sistema ANC pasa del 1:1:1 (monocanal) al 1:4:4 (16 canales), hay 12 veces más multiplicaciones, 13 veces más sumas y 3.4 veces más FFTS. Más o menos lo mismo ocurre para valores de R>0. Por ejemplo, cuando R=16, las multiplicaciones aumentan 13 veces, las sumas 14 veces y las FFTs casi 4 veces. Por otra parte, la tabla también muestra que al aumentar el número de factores ortogonales (R>0), el algoritmo FPBMFxNLMS aumenta considerablemente su complejidad. Esto es debido a que el proceso de actualización de los filtros se repite *R* veces. Por tanto, visto el notable aumento de complejidad computacional con el aumento del número de canales y el valor de *R*, se puede

concluir que el coste computacional es un cuello de botella a la hora de implementar dichos algoritmos en sistemas ANC multicanal que funcionen en tiempo real.

Tabla 3. Número total de multiplicaciones, sumas y FFTs por cada iteración del algoritmo FPBMFxLMS-OCF para diferentes I:J:K configuraciones, variando el valor de *R* y con L=M.

	configuración I:J:K								
	1:1:1			1:2:2			1:4:4		
	R=0	R=4	R=16	R=0	R=4	R=16	R=0	R=4	R=16
Multiplicaciones	13*L*	33*L*	93*L*	44*L*	112*L*	316*L*	160*L*	420*L*	1200*L*
Sumas	10.5*L*	34.5*L*	106.5*L*	37*L*	125*L*	389*L*	140*L*	476*L*	1484*L*
FFTs	5	13	37	9	25	73	17	49	145

3. CONCLUSIONES

Este artículo presenta el algoritmo MFxNLMS-OCF para una aplicación CAR multicanal. El algoritmo ha sido derivado del algoritmo NLMS-OCF, anteriormente propuesto para una aplicación de identificación de canal. Como muestran los resultados, el algoritmo propuesto mejora sustancialmente la velocidad de convergencia del algoritmo MFxNLMS (algoritmo MFxNLMS-OCF con R=0), muy comúnmente implementado en sistemas CAR. LA principal diferencia entre ambos algoritmos es que el MFxNLMS-OCF actualiza los filtros adaptativos a partir de múltiples vectores de señal de referencia, mientras que el MFxNLMS sólo utiliza un vector de entrada en la actualización. Por ello, el algoritmo propuesto tiene la desventaja de ser más costoso computacionalmente. Esta desventaja computacional se acrecienta en sistemas multicanal. Por lo tanto, como su principal desventaja es el coste computacional, se ha variado el algoritmo a una versión que permita su implementación en arquitecturas paralelas (como por ejemplo la GPU) que aceleren su ejecución y permitan su implementación en prototipos CAR que funcionen en tiempo real. Para ello, el algoritmo se ha adaptado a su versión en el dominio de la frecuencia, trabajando por bloques y con los filtros adaptativos particionados, dando como resultado el algoritmo FPBMFxNLMS-OCF.

4. REFERENCIAS

[1] S. J. Elliot, P. A. Nelson, Active noise control, IEEE Signal Processing Magazine 10 (4) (1994) pp. 12-–35.
[2] S. J. Elliott, I. M. Stothers, P. A. Nelson, A multiple error LMS algorithm and its application to the active control of sound and vibration, IEEE Transactions on Acoustics, Speech and Signal Processing 35 (10) (1987) 1423–1434.
[3] B. Widrow, S. D. Stearns, Adaptive signal processing, Prentice-Hall. New York, 1985.
[4] E. Bjarnason, Active noise cancellation using a modified form of the filtered-x LMS algorithm, 6th European Signal Processing Conference, Vol. 2, 1992, pp. 1053–1056.
[5] K. Ozeki, T. Umeda, An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties, Electronics Communications 67-A (1994) 19–27.
[6] S. L. Gay, J. Benesty, Acoustic signal processing for telecommunication, Springer, 2000.
[7] S. G. Kratzer, D. R. Morgan, The partial-rank algorithm for adaptive beamforming, in: 29th Annual Technical Symposium, International Society for Optics and Photonics, 1986, pp. 9–14.
[8] M. Rupp, A family of adaptive filter algorithms with decorrelating properties, Signal Processing, IEEE Transactions on 46 (3) (1998) 771– 775.
[9] S. G. Sankaran, A. Beex, Normalized LMS algorithm with orthogonal correction factors, in: Signals, Systems & Computers, 1997. Conference Record of the Thirty-First Asilomar Conference on, Vol. 2, IEEE, 1997, pp. 1670–1673.
[10] H.-C. Shin, A. H. Sayed, Mean-square performance of a family of affine projection algorithms, Signal Processing, IEEE Transactions on 52 (1) (2004) 90–102.
[11] J. Páez Borrallo, M. Garcia Otero, On the implementation of a partitioned block frequency domain adaptive filter (PBFDAF) for long acoustic echo cancellation, Signal Processing 27 (3) (1992) 301–315.