UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

Trabajo Fin de Grado

Grado en Tecnologías Específicas de Telecomunicación

Mención en Sistemas de Telecomunicación

Diseño de filtros digitales FIR mediante técnicas de computación evolutiva y estudio de su aplicación al procesado de señales biomédicas

AUTOR: DANIEL MAYOR TOMILLO
TUTORES: LUIS MIGUEL SAN JOSÉ REVUELTA
JUAN IGNACIO ARRIBAS SÁNCHEZ

VALLADOLID, OCTUBRE DE 2016

Índice

1. Resumen

El diseño de filtros digitales eficientes es una rama esencial del procesado de señales. Los filtros FIR son empleados en numerosas aplicaciones debido a su naturaleza de fase lineal y estabilidad frecuencial. Los métodos de diseño tradicionales sufren el problema del escaso control sobre la respuesta en frecuencia del filtro diseñado. Por esto, en este documento, se presenta una técnica de optimización novedosa, denominada Algoritmo de Polinización de Flores (FPA), junto con una novedosa función de aptitud múltiple, para la obtención del filtro FIR deseado. El algoritmo FPA se basa en el proceso de polinización de las flores. Dadas las especificaciones del filtro FIR, el algoritmo FPA obtiene un conjunto de coeficientes óptimos del filtro que mejor se aproxima a las especificaciones ideales. Los resultados obtenidos se han comparado con los métodos tradicionales de enventanado y algoritmo Parks-MacClellan (PM) y con otros métodos algorítmicos. Estos resultados numéricos muestran la superioridad del método de computación natural (FPA), junto con la función de aptitud múltiple en el diseño de filtros FIR paso bajo, paso alto, paso banda y elimina banda. Concretamente: Se consigue un mejor ajuste a las especificaciones del filtro deseado, una mayor atenuación de la banda eliminada y menor ancho de banda de transición a costa de aumentar ligeramente el rizado en la banda de paso.

2. Introducción

Los filtros son un componente clave en la mayoría de los sistemas electrónicos y de computación y una pieza fundamental en dispositivos que realizan un procesado de la señal. La finalidad del filtrado es extraer información sobre la señal de interés, ya sea eliminando el ruido, extrayendo componentes frecuenciales o separándola de otras señales indeseadas.

Para comenzar, vamos realizar una clasificación de los filtros en analógicos y digitales. Un filtro digital trabaja con señales discretas en el dominio del tiempo mientras que los filtros analógicos son empleados para el tratamiento de señales continuas en el tiempo.

En cuanto a los filtros digitales, se diferencian dos grandes grupos, los filtros FIR (*Finite Impulse Response)*, cuya respuesta al impulso es finita; y los filtros IIR, cuya respuesta al impulso es infinita. Los filtros FIR también son denominados MA (*Moving Average*) o de media en movimiento, y a los IIR se los nombra ARMA, pues están formados por la combinación de MA y AR (*Autorregresive*). El término autorregresivo, hace referencia a que los valores de salida actuales son calculados basados en valores anteriores de salida y por lo tanto, tienden, en cierta medida, a ella.

La principal característica que diferencia estos dos tipos de filtros es que, en los filtros FIR, la señal de salida solamente depende de valores pasados de la entrada; mientras que, en los filtros IIR, depende de valores pasados tanto de la salida como de la entrada, es decir, está realimentado. Cada tipo de filtro tiene sus propias ventajas e inconvenientes, por lo que ambos deberán ser considerados a la hora de diseñar un filtro digital.

Los filtros FIR también son denominados filtros digitales no recursivos debido a que no tienen realimentación. Su función de transferencia se asemeja a la ideal cuanto mayor es el orden del filtro, esto implica una mayor complejidad, una mayor carga computacional y un mayor tiempo necesario para el filtrado. No obstante, estos filtros digitales tienen muchas ventajas como la estabilidad y la posibilidad de obtener fase lineal generalizada, ya que, para ciertas señales, la característica de fase es una propiedad esencial para evitar perder información fundamental.

Por otro lado, los filtros IIR presentan ciertas ventajas sobre los filtros FIR. Frente a una misma respuesta al impulso, requieren un menor orden y por ello, un menor tiempo de cálculo y menor carga computacional. Sin embargo, el principal inconveniente de estos filtros es que, al estar realimentados y poseer polos finitos; se pueden producir inestabilidades en el sistema y desfases en la señal (no tienen fase lineal generalizada).

Los filtros que vamos a diseñar, inevitablemente necesitan ser causales, esto es, una muestra a la salida de nuestro sistema solamente depende de las muestras de entrada y salida en instantes de tiempo pasados, nunca futuros.

El diseño de filtros digitales es uno de los mayores problemas a los que nos enfrentamos en el procesado de señales. Para encontrar una solución óptima, hay que tener en cuenta diversos factores como las especificaciones del filtro, el criterio de diseño o el orden del filtro. Normalmente, estas características, entran en conflicto.

Desde un punto de vista matemático, el diseño de un filtro óptimo de acuerdo a un criterio especificado, puede verse como un problema de minimización que consiste en encontrar el filtro óptimo que mejor se adapte a nuestras necesidades [1]. Los problemas de optimización para el diseño de filtros son normalmente complejos, altamente no lineales y multimodales.

Una de las dificultades del diseño de filtros, planteado como un problema de optimización matemática, es la necesidad de obtener una función a la cual aplicar la minimización y que sirva para obtener una buena respuesta al impulso. A dicha función, se la denomina función de aptitud.

Los algoritmos naturalistas se han hecho muy populares en las últimas dos décadas debido al gran éxito que presentan en encontrar la solución óptima para problemas complejos de ingeniería e industria. La mayoría de estos algoritmos se basan en técnicas estocásticas que emplean un conjunto de soluciones aleatorias que se mejoran con cada iteración usando distintos mecanismos propios de cada algoritmo. El algoritmo se repite hasta que cierto criterio de parada se satisface. Este criterio para el diseño de filtros FIR será, para nuestro diseño, la minimización de la función de aptitud.

2.1. Motivación

Hay muchos métodos convencionales que permiten el diseño de filtros FIR como el método de enventanado, el método de muestreo frecuencial, el método de Parks and McClellan [31], etc. Pero estos métodos no ofrecen un control suficiente y preciso de varios parámetros como pueden ser las frecuencias de corte de la banda de paso y de la banda eliminada y el ancho de banda de transición. En los últimos años están surgiendo continuamente nuevos métodos para el diseño de filtros digitales. El método empleado con mayor frecuencia para el diseño de filtros FIR de fase lineal es la aproximación de Chebyshev basada en el algoritmo Remez desarrollado por Parks y McClellan en 1972 [31].

Los métodos de diseño de filtros convencionales están basados en técnicas de optimización clásica y tienen una mayor tendencia a quedarse estancados en mínimos locales debido, en parte, a que dependen de la solución inicial. La lenta tasa de convergencia de las técnicas de optimización convencionales limita en gran medida su empleo en el diseño de filtros. Por esto, surge la necesidad de superar las limitaciones de las técnicas clásicas de optimización mediante el desarrollo de métodos heurísticos de optimización inspirados en la naturaleza (conocidos como Computación Natural).

Los algoritmos detallados en este documento toman las ventajas de las técnicas de optimización naturalistas para el diseño de filtros FIR óptimos. El empleo de estos algoritmos produce una mejora de las características del filtro diseñado. Se consigue un menor rizado de la banda de paso y una mayor atenuación de la banda eliminada. También se obtiene un mayor control de las características frecuenciales que con las técnicas convencionales.

2.2. Objetivos

Se van a enumerar los objetivos de esta memoria:

- Entender las características de los distintos tipos de filtros digitales: IIR y FIR.
- Analizar y comprender distintas técnicas para el diseño de filtros FIR, en concreto, los métodos algorítmicos.
- Clasificar las distintas técnicas empleadas para el diseño de filtros FIR, centrándose en la clasificación de las técnicas algorítmicas naturalistas.
- Realizar una revisión bibliográfica de las distintas técnicas naturalistas empleadas por diferentes autores para el diseño de filtros.
- Revisar el estado del arte de estas técnicas, es decir, plasmar y comparar los resultados numéricos obtenidos por distintos métodos naturalistas.
- Estudiar e implementar un método optimización naturalista que no se haya implementado anteriormente y supere las limitaciones de otros métodos ya existentes.
- Aplicar dicho método al diseño de filtros FIR y obtener los resultados numéricos a través de la herramienta *Matlab.*
- Comparar los resultados obtenidos con los de otros autores.
- Orientar futuras aplicaciones de los algoritmos evolutivos y de nube de partículas (swarm) para el diseño de filtros digitales.

2.3. Fases

Para el desarrollo e implementación de un nuevo algoritmo es necesario estudiar las distintas técnicas existentes para el diseño de filtros: las técnicas clásicas y otros métodos naturalistas empleados.

En primer lugar, para entender correctamente el diseño de filtros digitales FIR se ha realizado un estudio de los métodos tradicionales para el diseño de filtros digitales, entre los que destacan el método de enventanado y de muestreo en frecuencia (apartado 3.1).

Posteriormente, para comprender la resolución de un problema por medio de algoritmos evolutivos y naturalistas se ha efectuado una exhaustiva revisión bibliográfica de los métodos empleados por otros autores para el diseño de filtros FIR óptimos (apartado 3.3.3).

En tercer lugar, se ha realizado una revisión del estado del arte de un conjunto de algoritmos naturalistas empleados para el diseño de filtros y se ha elaborado la exposición de los resultados obtenidos por los diferentes autores (apartado 4). En dicha exposición se comparan los filtros obtenidos por distintas técnicas en cuanto a rizado de la banda de paso, atenuación de la banda eliminada y coste computacional.

A continuación, se ha realizado la búsqueda de un método de computación naturalista que tenga algunas ventajas frente a otros métodos como rápida convergencia y la evasión de los mínimos locales. El método elegido que no se ha empleado con anterioridad para el diseño de filtros FIR es el Algoritmo de Polinización de Flores o FPA (apartado 5). Lo hemos implementado en la herramienta Matlab.

Se han expuesto los resultados obtenidos por el método implementado y los hemos comparado con los resultados de otros métodos y autores (Apartado 6 y 7). También se han elaborado una serie de filtros para su uso en el preprocesado de señales biomédicas. En último lugar, se ha propuesto un futuro campo de trabajo en el cual seguir investigando la aplicación de los algoritmos naturalistas (Apartado 8).

2.4. Recursos

Los recursos que se han empleado son, en su mayoría, artículos publicados en reconocidas revistas científicas como pueden ser la *International Journal of Bio-inspired Computation* o *The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*. Las referencias de los artículos utilizados se encuentran detalladas en el apartado 9.

Para la fundamentación teórica del diseño de filtros digitales se ha empleado el libro de Oppenheim et al. denominado *Discrete-time signal processing* [36]. Para completar esta información se ha utilizado información de distintos artículos publicados en revistas internacionales.

Las señales biomédicas utilizadas se han adquirido de la base de datos pública *Physionet* [55].

La implementación del algoritmo elegido y la obtención de los resultados se ha llevado a cabo gracias a la herramienta matemática Matlab R2012a de *Mathworks*; ejecutada en un ordenador con 8GB de memoria RAM y un procesador de cuatro núcleos con frecuencia de reloj de 2,6 GHz.

2.5. Organización de la memoria

El resto del documento se organiza de la siguiente manera: En la sección 3, se expone la fundamentación teórica del diseño de filtros digitales, los problemas multimodales y los métodos de computación natural. En la sección 4 se exponen brevemente los resultados obtenidos mediante otras aproximaciones evolutivas empleadas para el diseño de filtros FIR. La sección 5 expone el nuevo método implementado para el diseño. La sección 6 describe los resultados simulados obtenidos empleando el nuevo método y la comparación con otras técnicas. La sección 7 muestra su aplicación en el filtrado de una señal biomédica. La sección 8muestra las conclusiones del documento y la sección 9 futuros campos de aplicación del método y futuras posibles investigaciones.

3. Fundamentación teórica

3.1. Diseño de Filtros Digitales

El diseño de un filtro digital, ya sea FIR o IIR, consiste en obtener los coeficientes de la respuesta al impulso del filtro **h**(n), de forma analítica o gráfica. Nos vamos a centrar en este trabajo en los filtros FIR.

La finalidad del diseño de filtros FIR es obtener la respuesta al impulso **h**(n) de un filtro FIR óptimo por medio de diferentes técnicas. El filtro también se puede caracterizar por su función de transferencia **H**(z):

$$\mathbf{H}(z) = h(0) + h(1)z^{-1} + \cdots + h(N-1)z^{-(N-1)} = \sum_{n=0}^{N-1} h(n)\, z^{-n} \qquad (1)$$

Donde **h**(n) es la respuesta al impulso, que se pude representar como una ecuación en diferencias compuesta por la señal de salida **y**(n) y por la señal de entrada **x**(n), de la siguiente manera:

$$y(n) = h(0)x(n) + h(1)x(n-1) + \cdots + h(N-1)x\big(n-(N-1)\big) \qquad (2)$$

Siendo N el número de coeficientes del filtro y, por tanto, N-1 el orden del filtro. Se ha utilizado este criterio (coeficientes del filtro de 0 a N-1) en el resto de la memoria para simplificar. Los valores de la respuesta al impulso **h**(n) son los que se van a determinar a través de los distintos algoritmos y técnicas de optimización. Dichos valores de **h**(n) son los que van a establecer qué tipo de filtro se está diseñando, filtro paso alto (HPF), filtro paso bajo (LPF), filtro paso banda (BPF) o filtro elimina banda (BSF).

La longitud de **h**(n) es N y, por tanto, el número de coeficientes **h**(n) también es N.

La respuesta en frecuencia de un filtro FIR se puede calcular como:

$$H(w_k) = \sum_{n=0}^{N-1} h(n)\, z^{-jw_k n} \qquad (3)$$

Donde $w_k = \frac{2\pi k}{N-1}$, $H(w_k)$ es el vector de la Transformada de Fourier Compleja. En este caso, $H(w_k)$ es la respuesta en frecuencia del filtro FIR **h**(n). La frecuencia discreta en el intervalo [0, 2π] se muestrea con N puntos.

Las especificaciones del Filtro suelen estar dadas en el dominio de la frecuencia como, por ejemplo, ganancia de la banda de paso, rizado de la banda de paso, frecuencias de corte, atenuación de la banda de corte, baja distorsión y el ancho de banda de transición, entre otros. Los métodos más comunes empleados para el diseño de filtros FIR son el **método de enventanado**, también conocido como el método de series de Fourier; el **método de muestreo**

en frecuencia y **métodos algorítmicos** que son los que se van a emplear y desarrollar en este documento.

Para el filtrado de ciertas señales, la característica de fase lineal es una propiedad esencial para evitar perder información fundamental. Un filtro FIR tiene la ventaja de que se pueden diseñar para tener fase lineal si su respuesta al impulso satisface las siguientes condiciones de simetría:

- Sistema no causal con respuesta al impulso conjugada simétrica (**h**(n)=**h***(-n)) tiene una Función de transferencia **H**(k) real.
- Sistema no causal con respuesta al impulso conjugada antisimétrica (**h**(n) = -**h***(-n)) tiene una Función de transferencia **H**(k) imaginaria pura.

Por tanto, las fases pueden ser 0 ó $\frac{\pi}{2}$. Si queremos que las secuencias sean realizables, habrá que retardar dichas secuencias un número de muestras adecuadas, haciendo que estas sean causales. Dependiendo de si el número de coeficientes del filtro es par o impar y del tipo de simetría, existen cuatro tipos posibles de filtros FIR como se muestra en la tabla 1.

Tipo:	Número de Coeficientes (N)	Simetría
I	Impar	Simétrico $\mathbf{h}(\mathrm{n}) = \mathbf{h}(\mathrm{N}-1-\mathrm{n});\ \ \mathrm{n} = 0,1,\dots,\mathrm{N}-1$
II	Par	Simétrico $\mathbf{h}(\mathrm{n}) = \mathbf{h}(\mathrm{N}-1-\mathrm{n});\ \ \mathrm{n} = 0,1,\dots,\mathrm{N}-1$
III	Impar	Antisimétrico $\mathbf{h}(\mathrm{n}) = -\mathbf{h}(\mathrm{N}-1-\mathrm{n});\ \ \mathrm{n} = 0,1,\dots,\mathrm{N}-1$
IV	Par	Antisimétrico $\mathbf{h}(\mathrm{n}) = -\mathbf{h}(\mathrm{N}-1-\mathrm{n});\ \ \mathrm{n} = 0,1,\dots,\mathrm{N}-1$

Tabla 1. Tipos de Filtros FIR según el número de coeficientes y la simetría de los mismos.

3.1.1. Diseño de Filtros FIR empleando ventanas

El método de enventanado se basa en truncar la respuesta al impulso infinita de un filtro ideal. El procedimiento utilizado es el siguiente:

- Se obtiene la respuesta al impulso del filtro ideal que deseamos diseñar $\mathbf{h}_i(\mathrm{n})$. Según sea un filtro paso bajo, paso alto, elimina banda o paso banda, la respuesta al impulso variará. La Ec. (4) muestra la respuesta al impulso de un filtro paso bajo como ejemplo.

$$\mathbf{h}_i(\mathrm{n}) = \begin{cases} \frac{\mathrm{w_c}}{\pi}\ \frac{\sin(\mathrm{w_c n})}{\mathrm{w_c n}} \\ \frac{\mathrm{w_c}}{\pi} \end{cases} \qquad (4)$$

- Se trunca o *enventana* la respuesta al impulso a través de una ventana **w**(n); así, la respuesta al impulso del filtro FIR enventanado será **h**(n) = $\mathbf{h}_i$(n)·**w**(n), donde **w**(n) es la respuesta al impulso de la ventana *y* $\mathbf{h}_i$(n) la respuesta del filtro ideal. La respuesta de la ventana debe ser de la forma:

$$\mathbf{w}(\mathrm{n}) = \begin{cases} \text{Función simétrica en el intervalo: } -\frac{\mathrm{N}-1}{2} \le \mathrm{n} \le \frac{\mathrm{N}-1}{2} \\ 0 \text{ en el resto} \end{cases} \qquad (5)$$

Siendo N la longitud de la ventana.

- Se desplaza la respuesta al impulso enventanada h(n) un número adecuado de muestras $(\frac{N-1}{2})$ para hacerla causal.

Como el producto en el dominio de tiempo discreto equivale a una convolución en el dominio de la frecuencia, podemos estudiar qué efecto tiene este enventanado sobre la respuesta al frecuencial del filtro ideal. En la Fig. 1 se pueden observar los efectos producidos por el enventanado del filtro ideal ($H_d(e^{jw})$) con la ventana $W(e^{jw})$. Estos efectos se pueden sintetizar en tres: Rizados de la banda de paso y de la banda eliminada, ancho de banda de transición y atenuación mínima de la banda de corte.

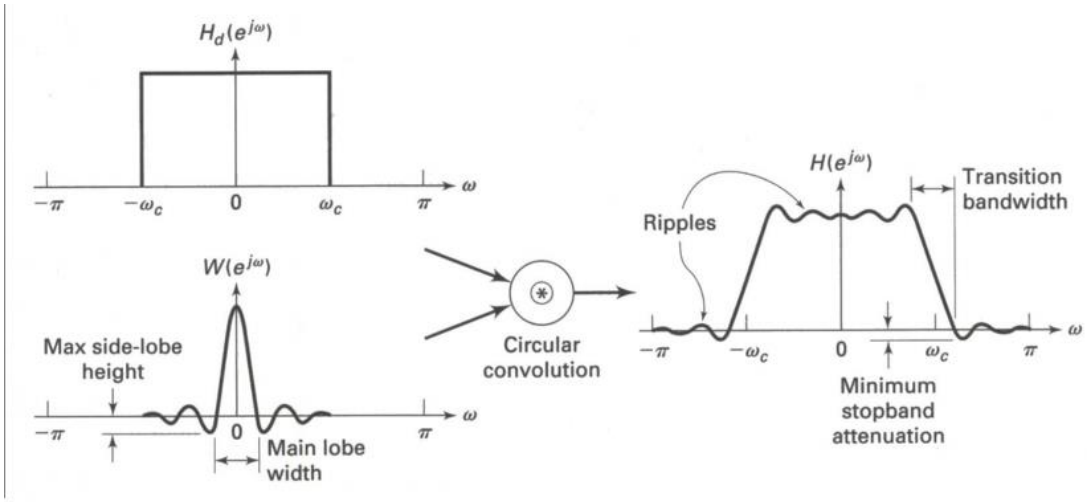

Figura 1. Efectos de la ventana sobre la respuesta en frecuencia del filtro ideal $|H_d(e^{jw})|$ [36].

Si nos centramos en la ventana más sencilla, la ventana rectangular, la desplazamos para que sea causal y realizamos su transformada de Fourier, obtenemos la siguiente respuesta en frecuencia representada en la Fig. 2.

$$\mathbf{w}(n) = \begin{cases} 1 & 0 \leq n \leq N-1 \\ 0 & n \geq N \end{cases} \quad (6)$$

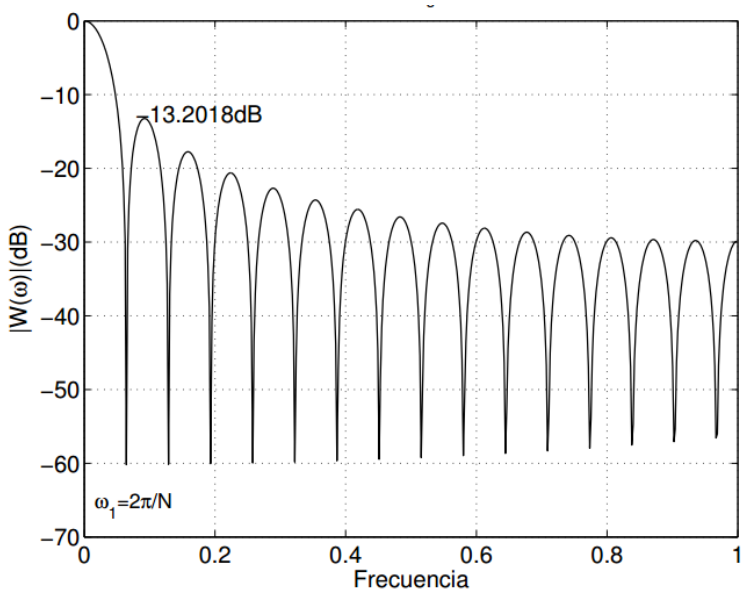

Figura 2. Respuesta en frecuencia en decibelios de una ventana rectangular de longitud N=31.

La anchura del lóbulo principal en esta ventana es $\frac{4\pi}{N}$, por tanto, cuando N crece, el lóbulo principal se estrecha. Dado que la respuesta en frecuencia del filtro diseñado será igual a la convolución en el dominio de la frecuencia de la respuesta en frecuencia del filtro ideal y de la ventana, ésta tendrá un papel determinante en las características del filtro diseñado. El efecto del enventanado es doble:

- Por un lado, la anchura del lóbulo principal está relacionada con la aparición del ancho de banda de transición del filtro. Cuanto mayor sea el lóbulo principal, mayor será la banda de transición del filtro. Sin embargo, como hemos dicho antes, se puede reducir el ancho del lóbulo principal aumentando la longitud de la ventana N, lo que disminuiría el ancho de banda de transición. El coste de aumentar N es un aumento de la carga computacional.
- Por otro lado, la presencia de los lóbulos secundarios o laterales lleva a la aparición de un rizado en la banda de paso y en la banda de corte. La diferencia entre el lóbulo principal y los lóbulos secundarios está relacionada con la atenuación de la banda de corte. Esto depende únicamente del tipo de ventana utilizada.

Son numerosas las funciones empleadas para enventanar la respuesta al impulso ideal. Dependiendo de las características del problema, se puede decidir entre las diferentes opciones de enventanado. Por ejemplo, si dada una longitud del filtro fija, se necesita reducir al máximo la banda de transición, utilizaremos una ventana rectangular; si, por el contrario, se necesita atenuar al máximo la banda de corte, utilizaremos una ventana de Blackman. En la tabla 2 se puede observar la anchura de la banda de transición que se obtiene con cada ventana, así como la atenuación mínima de la banda de corte.

VENTANA	ANCHURA DE LA BANDA DE TRANSICIÓN DEL FILTRO DISEÑADO	ATENUACIÓN MÁXIMA DE LA BANDA DE CORTE DEL FILTRO DISEÑADO (DB)
RECTANGULAR	$\frac{1.8\pi}{N}$	-21
BARLETT (TRIANGULAR)	$\frac{6.1\pi}{N}$	-25
HANNING	$\frac{6.2\pi}{N}$	-44
HAMMING	$\frac{6.6\pi}{N}$	-53
BLACKMAN	$\frac{11\pi}{N}$	-74

Tabla 2. Características de las distintas Ventanas y sus parámetros de ancho de banda de transición y atenuación de banda de corte del filtro diseñado.

Se puede observar que las ventanas con mejor ancho de banda de transición son las que tienen una menor atenuación de la banda de corte y viceversa. Debido a esto, existe un compromiso en el que también entra en juego la longitud de la ventana.

3.1.2. Diseño de Filtros FIR empleando muestreo en frecuencia

Para realizar el diseño de filtros FIR empleando el método de muestreo en frecuencia, es necesario establecer la respuesta en frecuencia del filtro ideal $H_i(w)$, que variará en función del filtro que se desee diseñar. Posteriormente, se muestrea dicha respuesta en frecuencia con N muestras o puntos uniformemente distribuidos por todo el espectro digital, obteniéndose así **H**(k).

$$\mathbf{H}(k) = H_i(w)\big|_{w=\frac{2\pi k}{N}} \quad (7)$$

Se puede obtener la respuesta al impulso del filtro diseñado **h**(n) a partir de la transformada inversa de **H**(k). Con este método de diseño, obtenemos un filtro con respuesta en frecuencia que pasa por los puntos muestreados; sin embargo, no podemos controlar el resto de valores de la respuesta. De esta manera, si representamos la respuesta en frecuencia del filtro diseñado, después de realizar el proceso de muestreo y de transformada inversa, observamos que aparece un rizado entre las muestras, como ocurre en la Fig. 3, pero los valores de las muestras coinciden con los originales del filtro ideal.

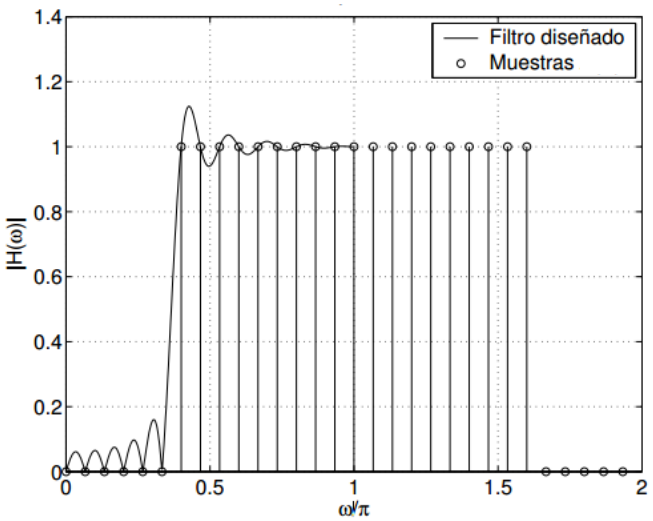

*Figura 3. Respuesta en frecuencia [0,2π] del filtro diseñado (paso alto) y de las muestras de **H**(k)*

Método de diseño:

- Dada la respuesta en frecuencia del filtro ideal, se elige la longitud deseada y se toman N muestras equiespaciadas en el intervalo [0,2π].
- Se utiliza la transformada inversa de Fourier para determinar **h**(n).

Características del filtro diseñado:

- La diferencia entre la respuesta en frecuencia del filtro ideal y la del filtro diseñado es cero en las frecuencias muestreadas k.
- El error de aproximación en el resto de frecuencias depende de la respuesta ideal tomada como referencia. Transiciones bruscas en dicha respuesta implican mayores errores.
- El error es mayor en los límites de las bandas y menor dentro de ellas.

Como se observa en la Fig. 3, el problema que presenta este diseño es la aparición del rizado y las sobreoscilaciones en los puntos con mayor discontinuidad sin poder tener ningún control sobre los mismos. Una forma de evitar estos efectos es ampliar la zona de transición del filtro ideal, evitando una caída abrupta y, por consiguiente, obtener una disminución del rizado. Este efecto se puede apreciar en la Fig. 4, donde la transición entre la banda de paso y la banda de corte no es abrupta, sino que posee muestras intermedias que, aunque amplían el ancho de banda de transición, disminuyen considerablemente el rizado.

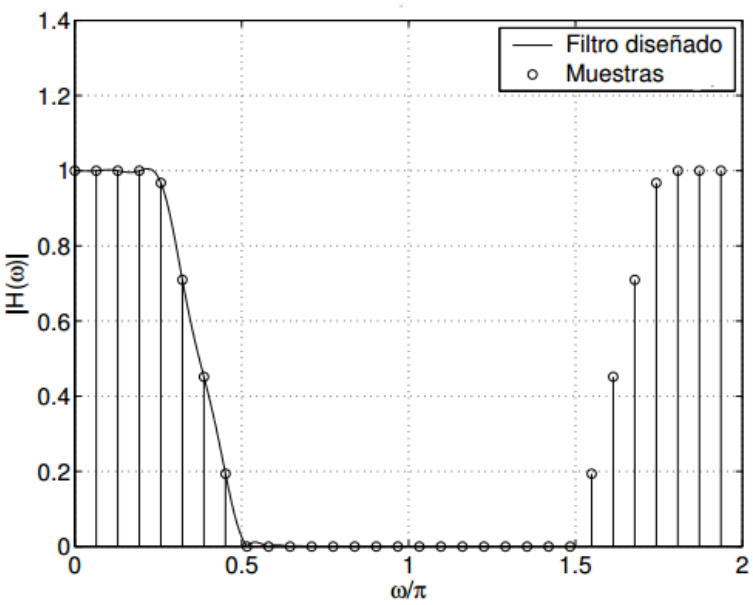

Figura 4. Respuesta en frecuencia [0,2π] del filtro diseñado (paso bajo) y de las muestras de H(k) con banda de transición de 0.25π a 0.5π

3.1.3. Diseño de Filtros FIR empleando métodos algorítmicos

Desde un punto de vista matemático, el diseño de un filtro óptimo puede verse como un problema de minimización que consiste en encontrar el filtro que mejor se adapte a nuestras necesidades [1]. Los problemas de optimización para el diseño de filtros son normalmente complejos, altamente no lineales y multimodales.

Una de las dificultades del diseño de filtros, planteado como un problema de optimización matemática, es la necesidad de obtener una función a la cual aplicar la minimización y que sirva para obtener una buena respuesta al impulso (función de aptitud).

En los métodos algorítmicos empleados en este trabajo, los coeficientes de **h**(n) se representan por distintos elementos dependiendo cual sea el algoritmo: por partículas en el PSO (*Particle Swarm Optimization*), cromosomas en los GAs (*Genetic Algorithms*) o nidos en el caso del CSA (*Cuckoo Search Algorithm*). En cada iteración de los algoritmos, dichas partículas, cromosomas, nidos… son actualizados. Los valores de aptitud, también denominados valores de error de aptitud, se recalculan con los nuevos coeficientes (nueva solución).

Para el diseño de filtros empleando algoritmos, la mejor o mejores soluciones de la iteración actual son elegidas y almacenadas para la siguiente iteración hasta que el criterio de parada es alcanzado. Posteriormente, se prosiguen las iteraciones hasta que se alcanza el criterio de parada, es decir, se alcanza el número máximo de iteraciones o se consigue obtener el valor de error de aptitud mínimo establecido.

Una de las ventajas que tiene el diseño de filtros FIR de fase lineal es que son simétricos y por tanto sus coeficientes también. Solamente la mitad de los coeficientes son empleados para las potenciales soluciones de los algoritmos naturalistas que se van actualizando en cada iteración. Posteriormente, son duplicados para formar la otra mitad de los coeficientes debido a la naturaleza simétrica de los filtros FIR. Así se reduce la dimensión del problema a la mitad.

Se van a definir dos respuestas al impulso que permitirán obtener posteriormente la función de aptitud.

$$\mathbf{H}_d(w) = [H_d(w_1), H_d(w_2), H_d(w_3), \dots, H_d(w_{N-1})]^T \quad (8)$$

$$\mathbf{H}_i(w) = [H_i(w_1), H_i(w_2), H_i(w_3), \dots, H_i(w_{N-1})]^T \quad (9)$$

Donde $\mathbf{H}_d$, representa la respuesta en frecuencia del filtro diseñado y $\mathbf{H}_i$, representa la respuesta en frecuencia del filtro ideal, que variará según estemos diseñando un filtro LPF, BPF, HPF o BSF.

Para un filtro Paso Bajo (LPF):

$$\mathbf{H}_i(w) = \begin{cases} 1 & \text{si} \quad 0 \le w \le w_c; \\ 0 & \text{resto} \end{cases} \quad (10)$$

Para un filtro Paso Alto (HPF):

$$\mathbf{H}_i(w) = \begin{cases} 0 & \text{si} \quad 0 \le w \le w_c; \\ 1 & \text{resto} \end{cases} \quad (11)$$

Para un filtro Paso Banda (BPF):

$$\mathbf{H}_i(w) = \begin{cases} 1 & \text{si} \quad w_{pl} \le w \le w_{ph}; \\ 0 & \text{resto} \end{cases} \quad (12)$$

Para un filtro Elimina Banda (BSF):

$$\mathbf{H}_i(w) = \begin{cases} 0 & \text{si} \quad w_{pl} \le w \le w_{ph}; \\ 1 & \text{resto} \end{cases} \quad (13)$$

Donde w_c es la frecuencia de corte de los filtros Paso Bajo y Paso Alto; w_{pl} y w_{ph} son las frecuencias límite de banda inferior y superior, respectivamente, de los filtros Paso Banda y Elimina Banda.

En cualquier optimización llevada a cabo por algoritmos evolutivos, es necesario fijar una función de error o de aptitud adecuada, ya que juega un papel importante en dichos métodos. Diferentes funciones de error conllevan distintos resultados para un mismo problema, por lo tanto, la elección de una función de error adecuada es muy importante para el buen funcionamiento de los algoritmos naturalistas.

Una función de error muy empleada en los algoritmos evolutivos es la propuesta por Parks y McClellan [31], que utilizaron para el desarrollo de su algoritmo PM para diseño de filtros FIR en 1972.

$$\mathbf{E}(w) = \mathbf{G}(w)\,[\,\mathbf{H}_d(w) - \mathbf{H}_i(w)\,] \quad (14)$$

$$\mathbf{G}(w) = \begin{cases} \frac{1}{k} & ; \quad 0 \le w \le w_p \\ k & ; \quad w_s \le w \le \pi \end{cases} \quad (15)$$

Donde G(w) es la función de pesos o de ponderación y w_p y w_s son las frecuencias límites de la banda de paso y de la banda de corte, respectivamente. Dicha función es la encargada de

proporcionar distinta importancia (ponderación) a las bandas de frecuencia a la hora de calcular los errores aproximados. Un posible valor de la función G(w) es el dado por la Ec. (15), donde k se establece como el ratio $k = \delta_p/\delta_s$. La mayor desventaja del algoritmo PM es que el ratio k está prefijado; δ_p es el rizado máximo de la banda de paso y δ_s es el rizado máximo de la banda de corte.

Otras funciones de aptitud son empleadas por los autores para el diseño de filtros FIR, como las empleadas por Karaboga y Cetinkayal en [32]:

$$\text{Error} = \max\left\{\sum[|\mathbf{H}_\text{d}(\text{w})| - |\mathbf{H}_\text{i}(\text{w})|]\right\} \quad (16)$$

Y por Najjarzadeh y Ayatollahi en [33]:

$$\text{Error} = \left\{\sum[|\mathbf{H}_\text{d}(\text{w})| - |\mathbf{H}_\text{i}(\text{w})|]^2\right\}^{\frac{1}{2}} \quad (17)$$

Con el fin de mejorar la flexibilidad de la función de error de aptitud, se busca que los niveles de rizado δ_p y δ_s sean especificados individualmente, sin depender el uno del otro. Esto es posible conseguirlo mediante la siguiente función de aptitud, desarrollada por Ababneh y Bataineh en 2008 [34]:

$$\text{J}_1 = \max_{\text{w}\le\text{w}_\text{p}}(\left|\mathbf{E}(\text{w}) - \delta_\text{p}\right|) + \max_{\text{w}\ge\text{w}_\text{s}}(|\mathbf{E}(\text{w}) - \delta_\text{s}|) \quad (18)$$

Donde δ_p y δ_s son los rizados máximos de la banda de paso y la banda de corte respectivamente; y w_p y w_s son las frecuencias normalizadas de la banda de paso y de la banda de corte, respectivamente. En [25] y [29], se introduce, por parte de los autores, una novedosa función de error de aptitud con el fin de alcanzar una mayor atenuación de la banda de corte y tener un control aproximado de la banda de transición.

Empleando la siguiente aproximación, se ha demostrado que los resultados presentan una considerable mejora sobre los obtenidos con la función de error de aptitud PM y otras técnicas de optimización:

$$\text{J}_2 = \sum \text{abs}[\text{abs}(|\mathbf{H}_\text{d}(\text{w})| - 1) - \delta_\text{p}] + \sum \text{abs}(|\mathbf{H}_\text{d}(\text{w})| - \delta_\text{s}) \quad (19)$$

El primer término de J_2 incluye una porción de la banda de transición, mientras que el segundo término incluye el resto de la banda de transición. Las porciones de la banda de transición dependen de las frecuencias de paso y de corte elegidas. La función de error de aptitud J_2 no considera solamente los errores máximos de cada banda, sino que implica la suma de todos los errores absolutos de toda la banda de frecuencia. Esto conlleva una mayor atenuación de la banda de corte y menor rizado, así como una reducción del ancho de banda de transición. Cada algoritmo trata de minimizar la función de error de aptitud elegida, lo que optimiza el comportamiento del filtro diseñado.

3.2. Problemas multimodales

Los problemas Multimodales son denominados de esta forma porque, además de la solución óptima, existen múltiples soluciones sub-óptimas en las que el método de diseño puede quedarse "atrapado" y no conseguir alcanzar la solución óptima global. Los métodos de optimización clásicos son generalmente rápidos y efectivos y se ha demostrado que funcionan razonablemente bien para el diseño de filtros digitales. Algunos métodos clásicos empleados para obtener el filtro óptimo son: el procedimiento de búsqueda de los valores óptimos de **h**(n) para la banda de transición desarrollado por Gold et al. en [37] y una técnica iterativa desarrollada por Hofstetter el al. en [38] con la finalidad de obtener un rizado igual en la banda de paso y la banda de corte.

El método clásico de optimización más conocido es el algoritmo PM [31], desarrollado por Parks y McClellan (1972). En su trabajo, proponen además del algoritmo de optimización, la función de aptitud (14) expuesta en el apartado anterior y empleada por multitud de autores como referencia para diseñar el filtro óptimo mediante métodos algorítmicos. Otro método de optimización empleado por algunos autores es el de mínimos cuadrados ponderados o WLS, empleado por Y. C. Lim et al. para el diseño de filtros digitales [39].

Estas técnicas convencionales, aunque son muy buenas para detectar mínimos locales, no están diseñados para descartar soluciones sub-óptimas (mínimos locales) en favor de mejores soluciones. Sintetizando, tienden a encontrar mínimos locales en el entorno del punto inicial.

Para ilustrar estos problemas multimodales y entender bien los conceptos de óptimos o mínimos locales y globales, se propone aplicar la optimización a un ejemplo de función matemática:

Encontrar el máximo de la función bidimensional:

$$z = f(x,y) = 3 * (1-x)\text{^}2 * \exp(-(x\text{^}2) - (y+1)\text{^}2) - 10 * (x/5 - x\text{^}3 - y\text{^}5) * \exp(-x\text{^}2 - y\text{^}2) - 1/3 * \exp(-(x+1)\text{^}2 - y\text{^}2)$$

Programamos la representación de esta función con la herramienta matemática Matlab con el siguiente código:

```
x1 = linspace(-4, 4, 70); % (1×70)
y1 = linspace(-4, 4,100); % (1×100)
[x, y] = meshgrid(x1, y1); % (100×70)

z=3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) - 10.*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) - 1/3.*exp(-(x+1).^2 - y.^2);

surf(x, y, z)
```

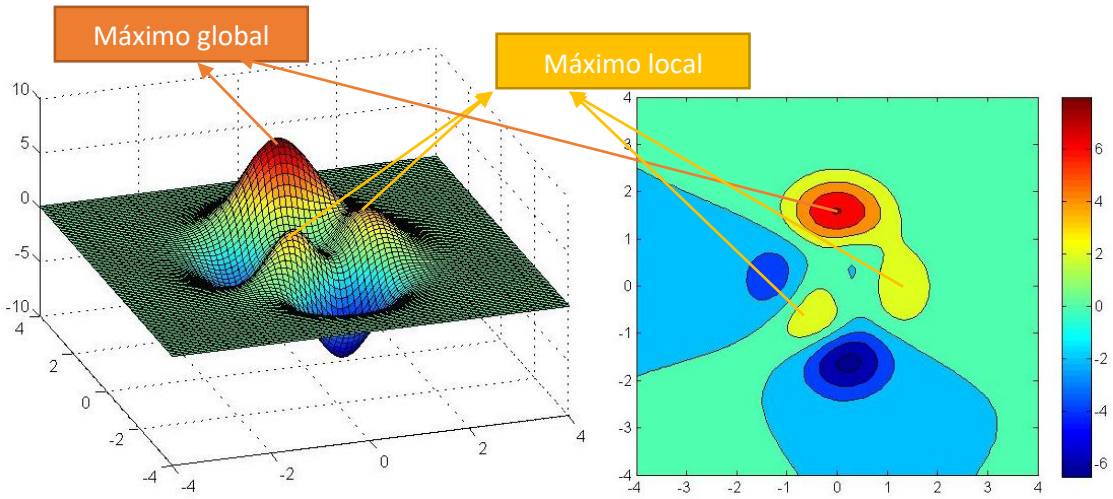

Figura 5.Representación tridimensional de la función con los máximos destacados.

Como se puede observar en la representación tridimensional de la función, esta tiene 3 máximos (zonas con un color más rojizo). El valor mayor, es decir, el que tiene una mayor altura en el gráfico de la izquierda y el color más rojo intenso en la imagen de la derecha, es al que se ha denominado como máximo global, siendo los otros dos, máximos locales.

Si empleásemos un método tradicional para este problema de optimización (encontrar un máximo de la función) ocurrirían las situaciones que se describen a continuación:

1. Las búsquedas se focalizan en un único punto (dependiendo del punto inicial) y, por tanto, se pierde diversidad. Esto quiere decir que, del espacio de búsqueda posible (toda la gráfica), los métodos convencionales se centran en una zona reducida cercana al punto inicial, perdiendo la capacidad de encontrar más óptimos y estancándose en este caso en un máximo local.

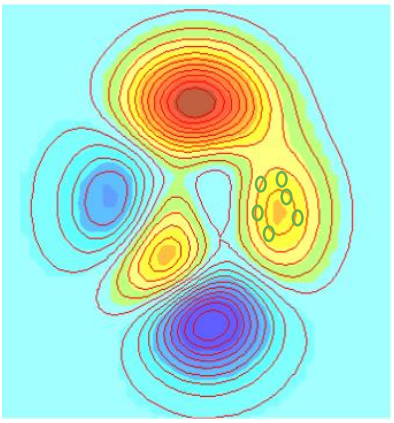

Figura 6.Representación de la función tridimensional con la búsqueda centrada en un sub-óptimo.

2. Converge prematuramente, esto implica que el óptimo alcanzado puede no ser un óptimo global, sino uno local, limitando la solución a nuestro problema.

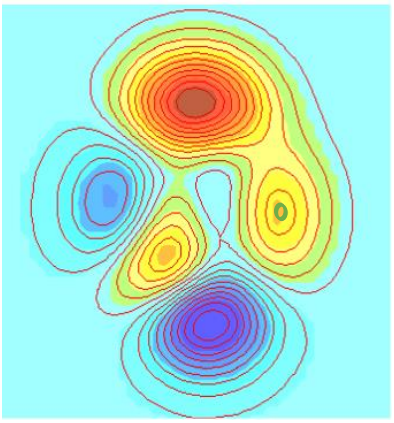

Figura 7.Representación del sub-óptimo alcanzado con el método tradicional.

3. No se obtienen todos los óptimos (máximos de la función), sino que se obtiene solamente una solución que puede no ser el óptimo global al problema planteado.

De esto se deduce que los resultados obtenidos por métodos tradicionales, aunque son buenas soluciones para nuestro diseño, no siempre obtiene soluciones óptimas, haciendo difícil el diseño de filtros. No son métodos eficientes. Por esta razón, actualmente, se están realizando muchas investigaciones y desarrollando numerosos algoritmos con el fin de desarrollar técnicas de diseño, más eficientes, que superen las limitaciones de los métodos convencionales.

3.3. Métodos de Computación Natural

Las técnicas que se van a detallar en esta sección son diversos métodos de búsqueda de soluciones en espacios multimodales (permiten la búsqueda simultánea en diferentes áreas) para el diseño de filtros digitales FIR, pertenecientes al grupo de computación natural o métodos naturalistas. Se denominan así porque extraen ideas de la naturaleza para desarrollar diferentes algoritmos de optimización. Estas técnicas o métodos han sido desarrollados y aplicados por numerosos autores con el fin de obtener un método de resolución a los problemas de optimización en un tiempo relativamente pequeño y con una baja carga computacional.

La computación natural surge como una alternativa a la computación clásica en la búsqueda de nuevos paradigmas que puedan proporcionar una solución efectiva a las limitaciones de los modelos convencionales. Actualmente, dentro del concepto de Computación Natural se engloba un conjunto de modelos que tienen como característica común la simulación del modo en que la naturaleza actúa/opera. Es decir, estudia la forma en que las diversas leyes de la naturaleza producen modificaciones en determinados sistemas (desde cómo interactúan las neuronas de nuestro cerebro, hasta la interacción entre grupos de seres vivos, pasando el proceso de evolución Darwiniana).

En este trabajo, se va a hacer una clasificación de distintas técnicas, basadas en los modelos de computación natural, para el diseño de filtros FIR. Podemos catalogarlas en tres grandes grupos:

- **Métodos evolutivos,** también se denominan computación evolutiva ya que extraen ideas de la biología evolutiva con el fin de desarrollar mecanismos de búsqueda y optimización. Son algoritmos no determinísticos, que a menudo presentan, implícitamente, una estructura paralela (múltiples agentes), y son adaptativos (utilizan realimentación con el entorno para modificar el modelo y los parámetros).

- **Métodos bioinspirados "de grupo".** Están basados o inspirados en la naturaleza, permitiendo el desarrollo de nuevas herramientas computacionales para la resolución de problemas. Modelan, de forma aproximada, un fenómeno existente en el entorno, un comportamiento de un conjunto de seres vivos o cualquier otro sistema o proceso presente en la naturaleza, constituyendo así una metáfora biológica para resolver problemas.
- **Métodos neuronales**, también denominados como redes de neuronas artificiales (RNA) o en inglés como *ANN*. Son un modelo de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso biológico. Se trata de un sistema de interconexión de neuronas que colaboran entre sí para producir un estímulo de salida. En inteligencia artificial es frecuente referirse a ellas como redes de neuronas o redes neuronales.
- **Métodos híbridos y otras aproximaciones de computación natural**. Los métodos híbridos combinan diferentes modelos o métodos de computación (pueden ser métodos naturales o tradicionales) con el fin de aprovechar las ventajas de cada modelo para dar una solución conjunta. De esta forma, se trata de reducir las desventajas de los métodos por separado y realizar un modelo único utilizando técnicas conjuntas. Algunos de los algoritmos revisados no se pueden englobar en ninguna de las categorías anteriores, por lo que se incluirán en esta sección.

3.3.1. Métodos de Computación Evolutiva

Como se ha descrito anteriormente, estos métodos se basan en la simulación de comportamientos evolutivos presentes en la naturaleza. Dentro de este grupo podemos incluir los algoritmos genéticos (GAs) y algoritmos de evolución diferencial (DE).

3.3.1.1. **Algoritmos Genéticos (GAs)**

Los Algoritmos Genéticos (GAs), desarrollados por Holland en 1975 [20], son métodos adaptativos que pueden usarse para resolver un problema complejo de búsqueda y optimización, y así ofrecer una solución al problema planteado. Se pueden definir como una serie de procedimientos de búsqueda basados en la genética y la selección natural.
A pesar de que estos algoritmos requieren una gran carga computacional, ofrecen unas características únicas frente a los algoritmos clásicos.
Vamos a dar algunas de las definiciones de algoritmos genéticos propuestas por diferentes autores:

- Los GAs son métodos de búsqueda de naturaleza estocástica que pueden ser utilizados para encontrar una solución óptima a la función de evolución de un problema de optimización [19].
- Holland, el desarrollador de estos algoritmos, los definió como unos programas de ordenador que se mimetizan con el proceso de evolución natural de los seres vivos [20].

En 1980, De Jong extendió los algoritmos genéticos con métodos híbridos con el fin de alcanzar una optimización global [21]. Mientras que Goldberg, en 1989, detalló un modelo matemático de implementación de estos algoritmos en [9].

La optimización clásica y los métodos de búsqueda difieren de los algoritmos genéticos en varios aspectos. En lugar de focalizarse en una solución única, los GAs operan en paralelo con un grupo de posibles soluciones. Estos manejan una **población** de individuos de cada **generación** (iteración) donde cada individuo, denominado **cromosoma**, representa una solución candidata

al problema. Dentro de la población, los individuos tratan de sobrevivir para reproducirse y así recombinar su material genético para producir nuevos individuos (**descendencia**). El material genético de los cromosomas es modelado por estructuras de datos binarios de longitud finita.

Como ocurre en la naturaleza, la **selección** provee el mecanismo para que las mejores soluciones sobrevivan. Cada solución candidata es evaluada de acuerdo con la función de **aptitud** que refleja como de buena es dicha solución en comparación con el resto. El proceso de recombinación genética es simulado mediante un mecanismo de **cruce** que realiza intercambios de datos entre cromosomas. Además, nuevo material genético es introducido a partir del proceso de **mutación**, que causa alteraciones aleatorias de las estructuras de datos. La frecuencia con la que ocurren estos fenómenos es controlada mediante ciertas probabilidades preestablecidas. La selección, el cruce y la mutación constituyen la base del ciclo del algoritmo genético que es repetido hasta que algún criterio predeterminado es alcanzado. A través de este proceso, mejores individuos son producidos en cada generación.

Los fundamentos de los GA se pueden resumir en 4 pasos esenciales:

1) Crear una población inicial de soluciones aleatorias (cromosomas).
2) Evaluar los cromosomas con una función de aptitud usando un criterio para obtener la solución deseada y crear una élite de cromosomas seleccionando un número de ellos, los que mejor satisfacen la función de aptitud.
3) Si los cromosomas con mejor aptitud satisfacen totalmente los requerimientos del problema, esos cromosomas serán la solución y se paran las iteraciones. Si no es así se continúa al paso 4.
4) Aplicamos el algoritmo de cruce entre los pares de cromosomas con mejor aptitud para generar más cromosomas. Posteriormente, se les aplicará el proceso de mutación aleatoria. Volvemos al paso 2.

A continuación, en la Fig. 8, se muestra una representación esquemática de este algoritmo genético, donde se pueden apreciar los elementos empleados en su implementación.

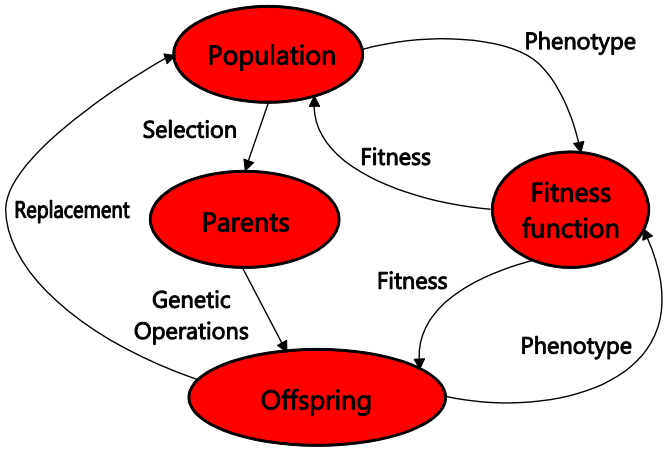

Figura 8. Representación conceptual del proceso de optimización mediante un GA.

3.3.1.2. Algoritmo de Evolución Diferencial

El método de Evolución Diferencial (DE) es un algoritmo de optimización basado en poblaciones. Fue desarrollado por Storn y Price en 1997 [40]. Está fundamentado en los principios de los algoritmos genéticos (GA) pero con las operaciones de cruce y mutación que trabajan con vectores continuamente evaluados.

La idea principal de este algoritmo es generar **vectores de parámetros de pruebas** y añadir la diferencia escalada entre dos vectores de población a un tercero. Como otros algoritmos evolutivos, los objetivos del DE son evolucionar a una población de Np individuos con vectores de parámetros de dimensión D. Cada individuo es una solución candidata al problema.

Los vectores de población tienen la forma:

$$\mathbf{x}_{i,G} = (x_{1,i,G}, x_{2,i,G}, \dots\dots\dots\dots, x_{D,i,G}), \quad \text{donde } i = 1, 2, 3, \dots, Np \qquad (20)$$

G indica la generación en la que se encuentra el algoritmo. Se puede hacer una descripción de la secuencia de pasos que sigue el Algoritmo de Evolución Diferencial:

1) **Inicialización.** Inicializar los Np individuos aleatoriamente (con valores límite de [-2,2]). Cada individuo consta de ((N+1) / 2) coeficientes de **h**(n) para desarrollar un filtro de orden N simétrico. Se inicializa también el contador de generación G=*1*.
2) **Mutación.** Se genera un vector mutado $\mathbf{v}_{i,G} = (v_{1,i,G}, v_{2,i,G}, \dots, v_{D,i,G})$ correspondiente a cada individuo $\mathbf{x}_{i,G}$ a partir de la siguiente fórmula. El mejor valor de escalado F determinado para este problema es 0.5-0.8.

$$\mathbf{v}_{i,G} = \mathbf{x}_{i,G} + F\cdot(\mathbf{x}_{best,G} - \mathbf{x}_{i,G}) + F\cdot(\mathbf{x}_{r1',G} - \mathbf{x}_{r2',G}) \qquad (21)$$

   $\mathbf{x}_{best,G}$ es el vector del individuo de la generación *G* que posee mejor función de aptitud, mientras que $\mathbf{x}_{r1',G}$ y $\mathbf{x}_{r2',G}$ son dos individuos seleccionados aleatoriamente de la población.

3) **Cruce.** Consiste en la generación de un vector de ensayo $\mathbf{u}_{i,G}$ ($\mathbf{u}_{i,G} = (u_{1,i,G}, u_{2,i,G}, \dots, u_{D,i,G})$) para cada vector de individuo $\mathbf{x}_{i,G}$. Se obtiene el vector de ensayo según el siguiente criterio:
   Para i=1 hasta Np, se obtiene el valor r_i = rand (0,1). Un valor aleatorio uniformemente distribuido de [0,1]. Se establece $\mathbf{u}_{i,G}$ como:

$$\mathbf{u}_{i,G} = \begin{cases} \mathbf{v}_{i,G}\ ; & \text{si } r_i \le CR \\ \mathbf{x}_{i,G}\ ; & \text{resto} \end{cases} \qquad (22)$$

CR es un parámetro que se establece al comienzo que se denomina probabilidad de cruce.

4) **Selección.** Se recorren todos los vectores de la población y los vectores de ensayo obtenidos. Si la función de aptitud del vector de ensayo $\mathbf{u}_{i,G}$ es mejor que la función de aptitud del vector de población $\mathbf{x}_{i,G}$, se establece $\mathbf{u}_{i,G}$ como vector de población para la siguiente generación ($\mathbf{x}_{i,G+1} = \mathbf{u}_{i,G}$). Si no es mejor, se mantiene el vector para la siguiente generación $\mathbf{x}_{i,G+1} = \mathbf{x}_{i,G}$.
5) Se incrementa el contador de generación G= G+1. El ciclo de generación se repite desde el paso 2 hasta que el contador de generación llega al número máximo de ciclos de generación establecido. Finalmente, los coeficientes óptimos del filtro de orden N se obtienen duplicando los ((N+1) / 2) coeficientes del individuo con mejor función de aptitud para obtener la respuesta al impulso óptima del filtro.

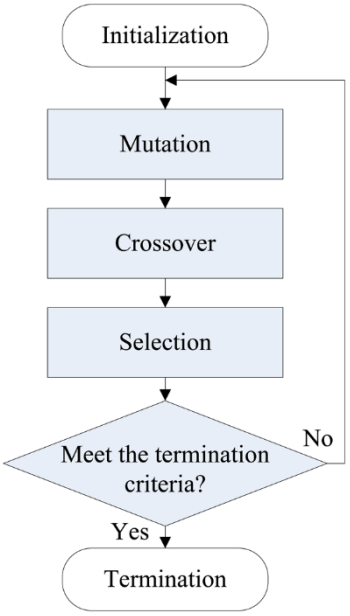

Figura 9. Diagrama de flujo de los pasos a seguir para la implementación del algoritmo de evolución diferencial.

3.3.1.3. Algoritmo de Evolución Diferencial basado en Genes Reservados

Una variante de este algoritmo es presentada por Qingshan Zhao et al. en 2010 [24], el algoritmo de evolución diferencial basado en los genes reservados. Se basa en reservar genes de un individuo seleccionado para aumentar la diversidad de la población. La aptitud media de la generación anterior se compara con la aptitud media de la nueva generación para determinar la convergencia. Si dicha aptitud media es igual para ambas generaciones y además la aptitud del mejor individuo es igual también para las dos generaciones, se emplean estos genes reservados para "escapar" del óptimo local.

A los pasos descritos en el algoritmo de evolución diferencial clásico, se debe añadir un paso intermedio entre el 4 y el 5. De esta manera, se deberá examinar la convergencia del algoritmo para evitar quedar atrapado en un óptimo local. El paso es el siguiente:

Si el mejor individuo de la generación actual es el mismo que el de la generación anterior, se compara el valor aptitud media de la generación actual con dicho valor de la generación previa. Se determina la convergencia del algoritmo empleando estos dos criterios. Si la aptitud media de la generación previa es igual a la de la generación actual, se establece la convergencia del algoritmo y para salir de ella, se lleva a cabo la evolución empleando los vectores de los individuos mejor y peor adaptados. Se crea un nuevo individuo combinando los dos anteriores, llamado $\mathbf{v}_1$. Se reemplaza un individuo de la población por $\mathbf{v}_1$ y se repite el proceso de evolución N_p veces.

3.3.1.4. Algoritmo de Recocido Simulado (*Simulated Annealing*)

El Recocido Simulado (SA) es una técnica de optimización global basada en movimientos aleatorios. El principio físico que simulan es el recocido (A*leanning*) de los metales en las fundiciones. Consiste en un calentamiento del sistema siendo optimizado a una temperatura muy elevada y posteriormente ir disminuyendo su temperatura (estando la sustancia el suficiente tiempo con esa temperatura para alcanzar el equilibrio térmico) en pasos muy pequeños hasta que el sistema se enfría y ya no se producen más cambios. Su característica más

destacable es la de evitar los mínimos locales aceptando soluciones menos óptimas (que la actual) con una probabilidad establecida.

El algoritmo SA, en primer lugar, genera una solución vecina a la solución actual, después decide si aceptar esta nueva solución según el valor de su función objetivo y con una probabilidad establecida. Finalmente se templa la temperatura de acuerdo al programa de recocido.

Se puede resumir el Algoritmo *Simmulated Aleanning* básico para encontrar el mínimo de una función f(x) en 4 pasos:

1) **Inicialización.** En primer lugar, se elige el vector de pasos (Éste es utilizado para especificar el máximo paso o modificación que se puede aplicar a la solución en cada iteración. Este vector es actualizado cada vez que la temperatura disminuye con el fin de restringir las zonas de búsqueda de posibles soluciones). Se obtienen las especificaciones deseadas del filtro en frecuencia, se elige una solución inicial $\mathbf{x}_0$ y se computa el valor inicial del parámetro de control temperatura (T_1). Se establece el contador de iteración a 1 (k=1), el límite de iteraciones máximas y se obtiene el coste de la solución inicial evaluándola en la función objetivo ($f(\mathbf{x}_0)$).
2) Aleatoriamente se genera una solución vecina $\mathbf{x}_k$ de acuerdo a un mecanismo generador establecido y se computa la función objetivo $f(\mathbf{x}_k)$.
3) Se calcula el valor Δ como la diferencia de costes entre la solución actual y la de la iteración anterior ($\Delta = f(\mathbf{x}_k) - f(\mathbf{x}_{k-1})$). Se genera un número aleatorio N uniformemente distribuido en el intervalo [0,1]. Se acepta la solución $\mathbf{x}_k$ si se cumple el **criterio de aceptación**, que en este caso se computa como $N \leq \min (1, e^{(-\Delta/ Tk)})$.
4) Si se alcanza el **criterio de parada**, se detiene la iteración. Si no es satisfecho, se aumenta el contador de iteración (k=k+1), se determina la temperatura T_k de acuerdo al programa de recocido y se salta al paso 2.

En el algoritmo SA tradicional empleado por R. V. Kacelenga et al. en 1990 [15] para el diseño de filtros FIR, se propone fijar una serie de parámetros con el fin de acelerar el tiempo de convergencia al óptimo global del algoritmo:

a) La Temperatura inicial T_1: El requerimiento para este parámetro, que establece White en 2007 [16], es que el parámetro T debe ser del mismo orden de magnitud que el cambio medio en la función objetivo. Con el objetivo de obtener T_1, el algoritmo comienza perturbando, tanto como sea posible, el espacio de búsqueda de coeficientes mientras computa el cambio en la función objetivo de cada punto. Se establece la temperatura inicial como 10 veces el máximo cambio producido en la función objetivo.

b) Esquema de decremento de la Temperatura: Si se emplea un decremento de la temperatura fijo, se ralentiza el tiempo de convergencia y, además, puede quedarse estancado el algoritmo en un mínimo local. Para evitar esto, algunos autores proponen que el decremento de la temperatura sea exponencial, reduciéndose rápidamente al principio y reduciendo el número de veces que se evalúa cada solución sin afectar a la convergencia del algoritmo. Se establece un valor máximo de decremento de la temperatura de 0,5.

c) Criterio del bucle interno: Muchos algoritmos SA no implementan un esquema de detección de equilibrio. El problema de no hacerlo es la facilidad de caer en un mínimo local. Una forma de evitar caer en estos mínimos locales es realizar muchas iteraciones para cada temperatura, lo que implica muchas evaluaciones de funciones (mayor carga computacional).

En este caso, el criterio de bucle interno empleado es, si suponemos que los coeficientes del filtro tienen 8 bits, el número de posibles estados que el sistema puede asumir si cambia un coeficiente son 256. En el algoritmo propuesto, el equilibrio se establece cuando el número de puntos aceptados es 1,14 veces 256, o cuando se llega al límite de iteraciones (1,86 veces 256). Es necesario establecer un límite de tiempo empleado con cada temperatura, especialmente para los valores bajos de temperatura donde el equilibrio tarda en alcanzarse.

d) Criterio de finalización: Un criterio de finalización simple es, por ejemplo, finalizar cuando la evaluación de la solución en la función objetivo no varía de manera significante durante varias reducciones de temperatura. El empleado por los autores es finalizar el algoritmo cuando el coste medio no tiene cambios en 4 reducciones de temperatura, o bien cuando el vector de pasos tiene el valor más pequeño en todas sus componentes.

e) Evaluación de las funciones: Para aumentar la velocidad del algoritmo, se evalúan las funciones empleando una tabla con un rango de frecuencias discretas de 0 a pi. Esto se puede hacer en este problema debido a la naturaleza discreta del mismo.

Se puede observar en la Fig. 10 un esquema del funcionamiento del algoritmo de recocido simulado o *simulated annealing*.

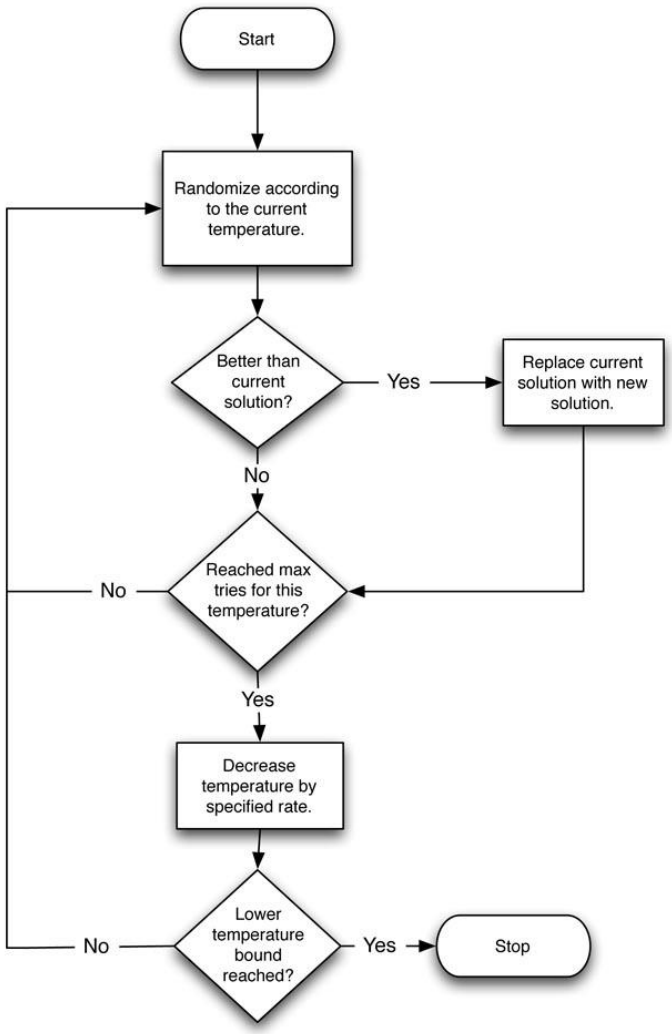

Figura 10. Diagrama de flujo del algoritmo de recocido simulado (SA).

Al igual que ocurre con el resto de algoritmos, a lo largo del tiempo, diversos autores han desarrollado técnicas y variantes al SA tradicional más eficientes que arrojan unos resultados mejores en términos de carga computacional y características del filtro. Se va a detallar el método empleado en [16], que utiliza una variante del algoritmo SA con lógica dispersa y adaptativo (*Fuzzy Adaptative Simulated Annealing*).

3.3.1.5. Algoritmo de Recocido Simulado Adaptativo con Lógica Difusa (FASA)

La dificultad que presenta el SA tradicional es la baja velocidad de convergencia (alta carga computacional), sin embargo, presentan una convergencia al mínimo global con una probabilidad muy elevada. Esta es una ventaja que presenta el SA frente a los GAs. Actualmente se han desarrollado técnicas para vencer estas limitaciones, llevando finalmente a la aproximación Recocido Simulado Muy Rápido o VFSR (Very Fast Simulated Re-annealing), desarrollada por L. Ingber en 1989 [17].

El Recocido Simulado Adaptativo (ASA), desarrollado por Jones et. al en 1995 [56], es una implementación del VFSR. Los problemas de optimización global estocástica, como este caso, comparten unas pocas características malas, como los largos periodos de poca mejora en el camino hacia el extremo global. En los SA, esto es causado por el programa de enfriado cuya velocidad es limitada por las características de la función de densidad de probabilidad utilizada para generar los nuevos puntos candidatos.

Si aplicamos el modelo de templado de Boltzman, la temperatura debe ser disminuida en una tasa máxima de T(k)= T(0) / ln(k) y en el caso del templado rápido, la tasa será T(k)= T(0)/k. La aproximación basada en ASA tiene un esquema de enfriado más optimizado incluso, de la forma:

$$T_i\,(k) = T_i\,(0) \cdot e^{-C_i \cdot k^{\frac{Q_i}{D}}} \qquad (23)$$

La constante C_i es un parámetro definido por el usuario. Los subíndices (i) indican la evolución independiente de las temperaturas para cada característica del filtro diseñado. Mientras que la medida Q_i, es denominada *quenching* o parámetro de templado. Si atribuimos un valor de Q mayor que 1, obtendremos una menor carga computacional, pero la convergencia a un óptimo global no está asegurada.

La aproximación empleada en [16] para acelerar el algoritmo ASA se basa en emplear un controlador de lógica difusa de Mamdani [18] que ajusta de manera dinámica algunos parámetros relacionados con el templado. Está demostrado que es posible acelerar el proceso de convergencia y además reducir considerablemente la necesidad de ajustar los parámetros por el usuario (incluso eliminar dicha necesidad) sin cambiar el funcionamiento del SA original, simplemente con el ajuste dinámico de parámetros.

3.3.1.6. Algoritmo de búsqueda Tabú

En el caso de los métodos basados en algoritmos genéticos [20], las regiones prometedoras de posibles soluciones se alcanzan rápidamente debido a la naturaleza simultánea del algoritmo. Sin embargo, debido a que las mismas soluciones pueden ser evaluadas muchas veces, hasta conseguir obtener la solución óptima, el coste computacional de esta clase de algoritmos es bastante elevado.

El algoritmo de búsqueda tabú (TSA) tradicional, presentado por Karaboga et al. en 1995 [11], en términos generales, se define como un mecanismo de búsqueda iterativo que comienza con una solución inicial factible y trata de obtener una solución mejor (empleando un algoritmo hill-climbing). Utiliza la información almacenada sobre las soluciones previamente vistas, en una memoria flexible para crear una mejor solución. El primer paso del algoritmo TS consiste en una solución presente que lleva asociado una serie de posibles soluciones que pueden obtenerse como una simple modificación (**movimiento**) de la solución actual. Los componentes principales del algoritmo TSA son la **lista Tabú** (TL) formada por una serie de restricciones y el **nivel de aspiración** (AL).

a) Restricciones de la Lista Tabú (TL)

La lista tabú almacena los distintos movimientos que no pueden ser aplicados a la solución actual. Los movimientos almacenados en la TL son aquellos más frecuentes y recientes, según una serie de criterios llamados **restricciones tabú**. El uso de esta lista reduce la posibilidad de un bucle infinito porque previene que, en un número de iteraciones, se vuelva a una solución reciente.

b) Nivel de Aspiración

Es posible eliminar las restricciones de la lista tabú cuando sea necesario para la búsqueda, utilizando un criterio llamado **criterio de aspiración**. Determina, por medio de una serie de condiciones, qué movimientos de la lista pueden ser evadidos si el movimiento es suficientemente bueno. La idea básica del TSA es elegir una solución posible y posteriormente obtener un vecino de la solución óptima. El movimiento o cambio al vecino es efectuado si dicho movimiento no pertenece a la lista tabú o, en el caso de pertenecer, aprueba el criterio de aspiración.

Para determinar lo buena que es una solución, es necesario establecer una **función objetivo**. Esto no es más que una función que determina cómo de buena es la solución obtenida al evaluarla en dicha función.

Durante estos procedimientos, la mejor solución es elegida y almacenada para la siguiente iteración hasta que el criterio de parada es alcanzado. Se pueden establecer diferentes condiciones para detener la búsqueda (criterio de parada). Un ejemplo de criterio de parada es el empleado en [12], donde se emplean dos criterios de parada: O bien, el número de iteraciones realizadas desde que la mejor solución se alcanzó es mayor a un número establecido previamente, o se alcanza el número máximo permitido de iteraciones.

El algoritmo de búsqueda tabú (TS) se puede resumir en su conjunto por una serie de pasos:

Antes de comenzar, se establece la Lista Tabú (TL) como vacía y el Nivel de Aspiración (AL) a 0.

1) Se inicializa el contador de iteraciones a 0 y se establece una solución inicial **x**, que será la mejor solución alcanzada hasta el momento (**x'' = x**).
2) Se genera aleatoriamente un conjunto de posibles soluciones **S** (vecinos de la solución actual **x**), aplicando un movimiento a dicha solución. Se ordenan las soluciones en sentido ascendente según su **función objetivo**, obteniendo el conjunto de soluciones Q. La posible mejor solución será la primera de la lista Q, es decir, la que tenga mejor función objetivo (valor más pequeño) y la denominaremos **x'**.

3) Si la función objetivo de **x**' es mayor que la función objetivo de **x**'' (es decir, que la solución **x**'' es mejor que la solución **x**'), saltamos al paso 4. Si no es mayor, se establece la mejor solución **x'' = x'** y se salta al paso 4.
4) Si **x**' no está en la lista Tabú, se acepta la x' como solución inicial (**x = x'**). Posteriormente se actualiza la lista tabú y el Nivel de Aspiración y saltamos al paso 7. Si **x**' está en la TL, se salta al paso 5.
5) Se efectúa el test de nivel de aspiración, si se satisface, se ignora la lista tabú y se acepta la **x**' como solución inicial (**x = x'**), se actualiza el AL y se va al paso 7. Si no se satisface el test de aspiración, se va al paso 6.
6) Si el final de la lista Q es alcanzado, se va al paso 7. Sino, se establece **x**' como la siguiente solución de Q y se salta al paso 4.
7) Se efectúa el test de finalización. Si se alcanza el criterio de parada, se para el algoritmo. En caso contrario, se incrementa el contador de iteración y se salta al paso 2.

En la Fig. 11 se muestra un diagrama de flujo de la implementación del algoritmo de búsqueda tabú.

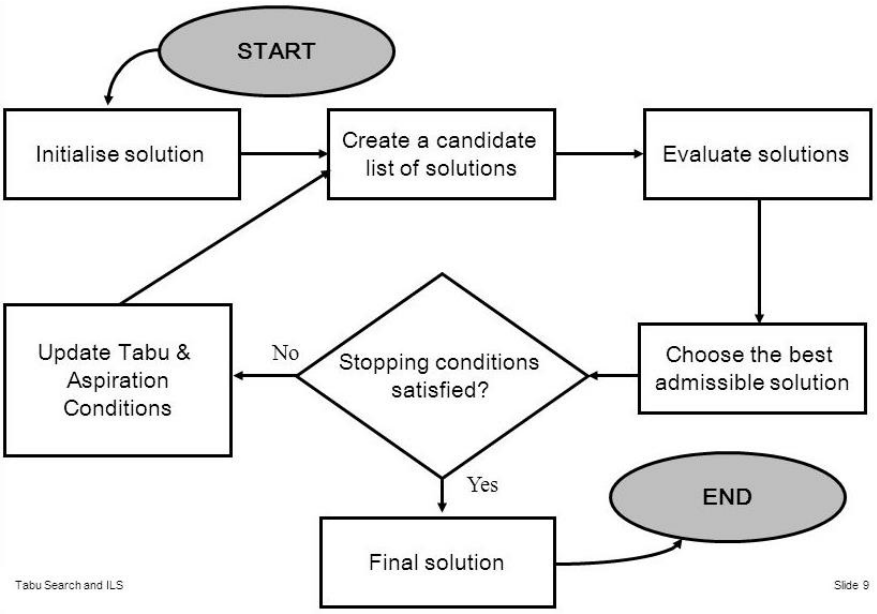

Figura 11. Diagrama de flujo del algoritmo TS tradicional [57].

Los algoritmos que se van a detallar a continuación son una variante del algoritmo de búsqueda tabú tradicional. En [12], Watcharasitthiwat et al., exponen una técnica basada en el algoritmo de búsqueda tabú, pero de forma múltiple, tratando de reducir el coste computacional. Y, en [13], Adem Kalinli y Nurhan Karaboga, proponen un método de diseño basado en el algoritmo de búsqueda Tabú simultánea (PTS) para el diseño de filtros FIR de longitud finita óptimos.

3.3.1.7. Algoritmo de búsqueda Tabú múltiple o paralelo

En el TSA hay al menos 4 características a las que se le puede aplicar el paralelismo:

1) En la evaluación de la función objetivo.
2) En la examinación de los vecinos.
3) En la descomposición del problema.
4) En la exploración del dominio de soluciones llevando a cabo distintas búsquedas.

El algoritmo de búsqueda tabú múltiple (MTSA) fue desarrollado por Karaboga et al. en 2005 [13] y posteriormente implementado por Watcharasitthiwat et al. en 2006 [12]. La característica sobre la que se aplica en [12] y [13] el paralelismo en el TSA es sobre la exploración del dominio de soluciones llevando a cabo distintas búsquedas. Multitud de Algoritmos de Búsqueda Tabú independientes son ejecutados empleando distintas soluciones iniciales. El empleo de varios TS de manera simultánea permite intercambiar información entre ellos. Dicho intercambio de información puede producirse en un mismo momento para todos los TS (síncrono) o en diferentes momentos (asíncrono). El intercambio de información entre los TSA ejecutados en paralelo está basado en el **mecanismo de cruce** de los GAs descrito en [13].

El procedimiento seguido por el algoritmo se puede observar en el diagrama de la Fig. 12. Consta de cinco mecanismos:

1) **Inicialización**. En el caso de que la solución óptima en el TSA tradicional esté muy alejada de la solución inicial establecida, puede llevar a un elevado tiempo de computación y por tanto una mayor carga computacional. Este problema se ve solventado con esta variante del algoritmo empleando múltiples soluciones iniciales independientes. La solución inicial para cada uno de los TS independientes se genera mediante un generador de números aleatorios con diferentes semillas (*seeds*).
2) **Búsqueda adaptativa**. El rango de variación de la solución actual es llamado *step size*. La utilización de un *step size* variable, ayuda a encontrar el área de búsqueda apropiado durante el proceso de búsqueda. Esta característica adaptativa continúa hasta que la solución cercana al óptimo global es alcanzada. Es necesario elegir correctamente el *step size*, que debe ser de un valor elevado cuando el mecanismo de búsqueda se encuentre lejano al óptimo y un valor pequeño cuando se acerque al óptimo global.
3) **Búsqueda múltiple**. Actualmente los ordenadores han evolucionado mucho, pueden realizar múltiples operaciones simultáneamente o lo que es lo mismo, múltiples procesos paralelos. Por esto, la variante MTSA se aprovecha de esta capacidad para ofrecer una búsqueda en paralelo y así encontrar las regiones prometedoras en un intervalo de tiempo muy pequeño.
4) **Mezcla o *Crossover*.** Cuando el número máximo de iteraciones es alcanzado, la ejecución de cada TSA autónomo se detiene. Los distintos TSA pueden tener distintos parámetros de control. Las *n* soluciones alcanzadas en los *n* TSA independientes representan **el primer nivel**, las denominadas soluciones preliminares. Se utiliza una operación a esas n soluciones con el fin de obtener las mejores primeras soluciones para **el segundo nivel**. Se pueden utilizar diversas operaciones, en el caso de [12], se emplea un operador de cruce o *crossover*.

   El operador de cruce es empleado para generar dos soluciones (**hijos**), a partir de dos soluciones existentes (**padres**) en la población obtenida por la operación de selección. En nuestro problema, vamos a representar las soluciones en binario, por lo que el mecanismo de cruce se efectúa de forma muy sencilla: Se eligen dos soluciones a cruzar y se selecciona una parte de cada solución para generar cada uno de las dos soluciones hijas. Por ejemplo:

   Solución inicial 1 (padre): **111100|001111** Nueva Solución 1: **111100|**111101
   Solución inicial 2 (padre): 101100|111101 Nueva Solución 2: 101100|**001111**

   (En este ejemplo se ha considerado el caso más sencillo, el cruce monopunto).

La mejor solución preliminar es directamente seleccionada para el segundo nivel y al resto de ellas se les aplica el cruce para alcanzar el segundo nivel [12], sin embargo, en [13], pueden emplearse distintos métodos, por ejemplo, la mejor solución preliminar se escoge como una solución inicial para el segundo nivel de TSs y el resto de soluciones iniciales se obtienen tras aplicar el mecanismo de cruce (de los GAs) al resto de soluciones preliminares y otras pueden ser generadas mediante la generación de números aleatorios. Los niveles indican el número de veces que se han ejecutado los TSs simultáneos. En todos los niveles, los TSs se aplican de la misma manera. Finalmente, la mejor solución es encontrada a través del proceso de búsqueda completo, obteniendo la solución óptima al problema.

5) **Proceso de Reinicio.** Cuando el proceso de búsqueda se queda estancado en una solución local durante mucho tiempo, se aplica el proceso de reinicio para evitar el problema y encontrar una mejor solución.

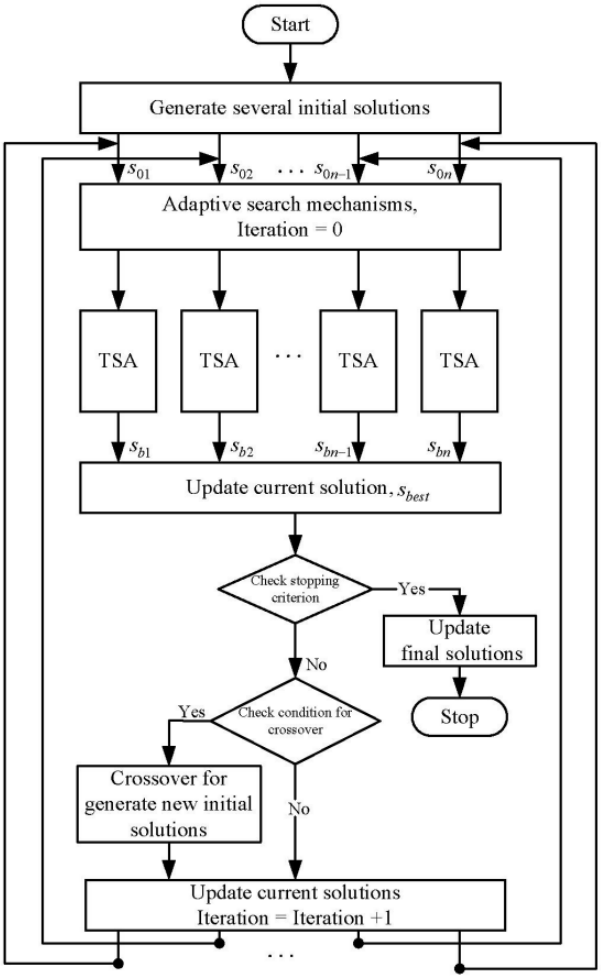

Figura 12.Diagrama de flujo del algoritmo de búsqueda tabú múltiple [12].

3.3.2. Métodos bioinspirados grupales

Tratan de replicar la forma en que la naturaleza se enfrenta al problema de optimización, con el fin de obtener un modelo iterativo para la resolución de problemas de optimización no lineales. En este grupo se incluirán los algoritmos de colonia de hormigas, de nube de partículas, entre otros, basados en el comportamiento en grupo de distintos seres vivos.

3.3.2.1. Algoritmo de colonia de hormigas (*Ant Colony Optimization*)

El Algoritmo de Colonia de Hormigas (ACO) es un algoritmo de optimización inspirado en el comportamiento natural de las hormigas, que encuentran los caminos más cortos entre su hormiguero y las zonas de alimentación. Ha sido demostrado que el ACO posee un excelente comportamiento a la hora de optimizar problemas de gran escala. El diseño de filtros FIR de orden elevado es un problema de gran escala que necesita un tiempo muy elevado para encontrar la solución óptima mediante técnicas tradicionales, sin embargo, el uso del ACO puede ser muy efectivo.

Este método de computación, empleado por Shuntaro Tsutsumi y Kenji Suyama en [6], se resume en 5 fases:

1) **Modelo de búsqueda.** En primer lugar, son generados muchos individuos (**hormigas**) preparadas para buscar las soluciones. La siguiente imagen muestra los distintos caminos de búsqueda para encontrar las soluciones al problema. Las **feromonas** $\tau_{n,k}^{00}, \tau_{n,k}^{01}, \tau_{n,k}^{10}\ y\ \tau_{n,k}^{11}$ son establecidas para cada camino de búsqueda. Donde $\tau_{n,k}^{lm}$ denota la cantidad de feromona depositada en el camino de *l* a *m*.

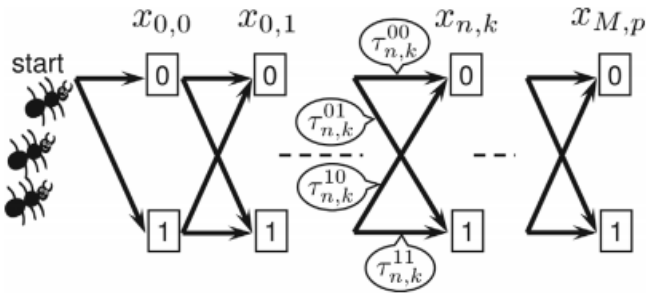

Figura 13. Resumen gráfico del algoritmo de colonia de hormigas [6].

2) **Inicialización de las feromonas.** Se deposita la cantidad de feromona inicial τ_{init} de un valor comprendido entre [0.5, 1] en los caminos correspondientes a soluciones redondeadas (Nota: En el diseño de filtros FIR con coeficientes discretos, la solución óptima se acerca a las soluciones redondeadas de los coeficientes continuos), y un total de 1- τ_{init} en los otros caminos. Como resultado de esto, grandes cantidades de feromonas son esparcidas en los caminos que tienen más probabilidad de producir buenas soluciones.
3) **Selección del camino.** Cada individuo parte del punto de partida *start* y se mueve por los distintos vértices seleccionando 0 o 1 para cada variable $x_{n,k}$ hasta llegar al punto final. ***x*** representa una posible solución al problema. Esta selección se realiza de manera estocástica basada en las feromonas. Por ejemplo, cuando es necesario determinar $x_{n,k}$ a partir del vértice anterior, cada individuo selecciona el vértice siguiente de acuerdo con *r*, un número aleatorio entre [0,1] según la siguiente fórmula:
    a) Si el vértice anterior es un 0:

$$x_{n,k} = \begin{cases} 0, & r < p_{n,k}^{00} \\ 1, & r > p_{n,k}^{00} \end{cases} \qquad (24)$$

    b) Si el vértice anterior es un 1:

$$x_{n,k} = \begin{cases} 0, & r < p_{n,k}^{10} \\ 1, & r > p_{n,k}^{10} \end{cases} \quad (25)$$

En el caso que $r = p_{n,k}^{00}$ *o* $r = p_{n,k}^{10}$, se vuelve a generar el valor aleatorio *r* y se vuelven a aplicar las fórmulas. Los valores de p son definidos a partir de las feromonas:

$$p_{n,k}^{00} = \frac{\tau_{n,k}^{00}}{\tau_{n,k}^{00} + \tau_{n,k}^{01}} \quad \text{y} \quad p_{n,k}^{10} = \frac{\tau_{n,k}^{10}}{\tau_{n,k}^{10} + \tau_{n,k}^{11}} \quad (26)$$

Cuando un individuo llega al punto final después de haber pasado por los vértices, el camino de dicho individuo corresponde a una solución.

4) **Actualización de las feromonas.** Cuando todos los individuos han completado la búsqueda, una cantidad de feromonas Δτ es añadida al camino correspondiente a la función objetivo con mejor valor. Además, las feromonas de todos los caminos se normalizan:

$$\tau_{n,k}^{00} = \frac{\tau_{n,k}^{00}}{\tau_{n,k}^{00} + \tau_{n,k}^{01}} \; ; \; \tau_{n,k}^{01} = \frac{\tau_{n,k}^{01}}{\tau_{n,k}^{00} + \tau_{n,k}^{01}}; \; \tau_{n,k}^{10} = \frac{\tau_{n,k}^{10}}{\tau_{n,k}^{10} + \tau_{n,k}^{11}}; \quad (27)$$

$$\tau_{n,k}^{11} = \frac{\tau_{n,k}^{11}}{\tau_{n,k}^{10} + \tau_{n,k}^{11}}$$

Esta normalización simula la **evaporación** de los niveles de feromonas en los caminos con malas soluciones. Cada individuo es más probable que tome los caminos con mayores feromonas y por ello, debido a la actualización de las feromonas, los vecinos de las soluciones con mejor valor de función objetivo son buscadas con mayor probabilidad.

5) **Algoritmo de búsqueda.** Los pasos que se siguen para alcanzar la solución a este problema de optimización son:

Paso 1. Las feromonas en los caminos son inicializadas.

Paso 2. Todas las hormigas son colocadas en el punto de partida.

Paso 3. Cada individuo repetidamente elige los vértices según el criterio de selección de camino hasta que el punto final de la búsqueda es alcanzado.

Paso 4. Una cantidad de feromonas Δτ es añadida los caminos con mejor valor de función objetivo.

Paso 5. Las feromonas son inicializadas (evaporación).

Paso 6. El algoritmo es finalizado cuando alcanza un número de iteraciones establecido. Si no acaba, se salta al paso 2.

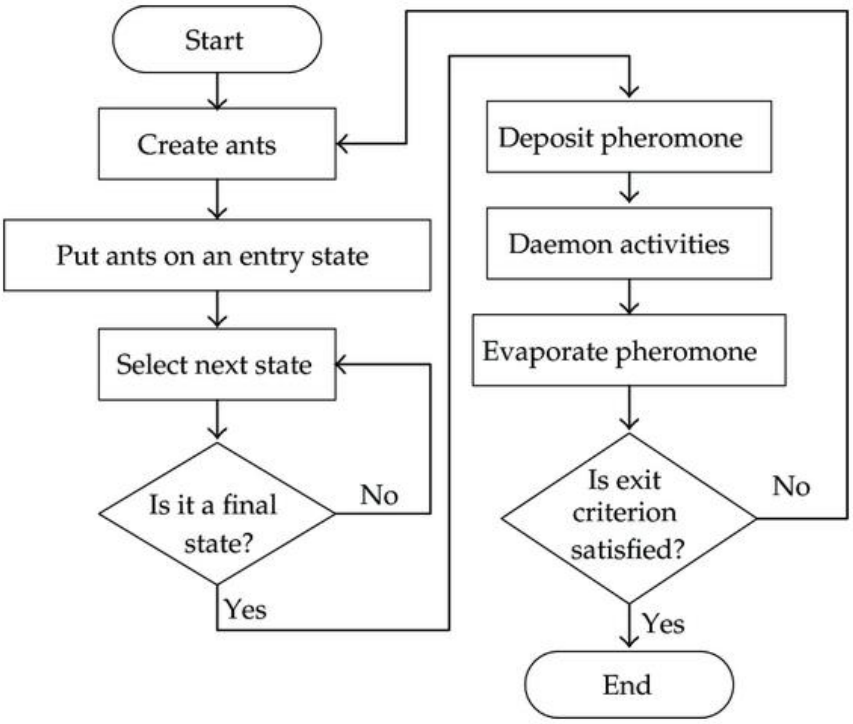

Figura 14. Diagrama de flujo del algoritmo de colonia de hormigas (ACO)

3.3.2.2. Algoritmo de nube de partículas (*Particle Swarm Optimization)*

La optimización por nube de partículas (PSO) es un algoritmo que pertenece al grupo de algoritmos evolutivos. El método de optimización por nube de partículas es un algoritmo de búsqueda basado en poblaciones, está inspirado en el comportamiento en grupo de bandadas de pájaros y bancos de peces. En el algoritmo PSO, una nube de partículas se mueve a través de un espacio de búsqueda D dimensional. Las partículas en el proceso de búsqueda son soluciones potenciales que se mueven por el espacio de búsqueda con una velocidad V hasta que el error se minimiza o la solución es alcanzada, decidido por la **función de aptitud**.

En 2014 [23], Naha hace una descripción del algoritmo PSO y su aplicación para el diseño de filtros. Las partículas alcanzan la solución deseada actualizando su posición y velocidad de acuerdo a las ecuaciones PSO. Cada individuo i-ésimo es tratado como una partícula sin volumen en un espacio D-dimensional, con su posición (vector de soluciones) y velocidad representadas como:

Posición → $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ *Velocidad* → $\mathbf{v}_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$

Cambio en la Velocidad:

$$\mathbf{v}_{ij} = w \cdot \mathbf{v}_{ij} + c_1 \cdot rand_1() \cdot (\mathbf{p}_{ij} - \mathbf{x}_{ij}) + c_2 \cdot rand_2() \cdot (\mathbf{p}_g - \mathbf{x}_{ij}) \quad (28)$$

Cambio en la Posición:

$$\mathbf{x}_{ij} = \mathbf{x}_{ij} + \mathbf{v}_{ij} \quad (29)$$

Las partículas son distribuidas aleatoriamente en el espacio de búsqueda con una posición y velocidad iniciales. Cambian su posición y velocidad con las ecuaciones respectivas donde c_1 y c_2

son dos constantes de aceleración cognitiva y social (típicamente de valor 2.05). $Rand_1$ () y $rand_2$ () son dos funciones uniformemente distribuidas en el rango [0,1] y w es el coeficiente de inercia introducido para acelerar la velocidad de convergencia del algoritmo. El vector $\mathbf{p}_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$ contiene la mejor posición previa (la posición dada por el mejor valor de aptitud) de la partícula i-ésima denominado pbest. El vector $\mathbf{p}_g = (p_{g1}, p_{g2}, \ldots\ldots\ldots, p_{gD})$ contiene la posición de la mejor partícula de toda la población y es denominada **gbest**. $\mathbf{x}_{ij}$, $\mathbf{v}_{ij}$ y $\mathbf{p}_{ij}$ son las posiciones, velocidad y mejor posición en la iteración j.

El flujo del algoritmo se puede describir en la siguiente serie de pasos:

1) Se inicializa cada una de las partículas con una posición y velocidad inicial aleatoriamente distribuidas y el contador de iteraciones.
2) Se calcula el valor de aptitud para cada partícula. Si el valor de *fitness* es mejor que el mejor almacenado en memoria (pbest), se establece dicho valor de aptitud como pbest, sino, se continúa al paso 3.
3) Se elige a la partícula con mejor valor de aptitud de todas las partículas existentes y se establece como **gbest**. Continúa el paso 4.
4) Se actualizan la posición y la velocidad de cada partícula de acuerdo a las fórmulas de cambio de velocidad y de posición.
5) Finaliza el algoritmo si se alcanza el número máximo de iteraciones o el criterio de mínimo error es satisfecho. Si no, se salta al paso 2.

En la Fig. 15, se muestra un diagrama de flujo del algoritmo PSO, donde se pueden observar los pasos descritos con anterioridad.

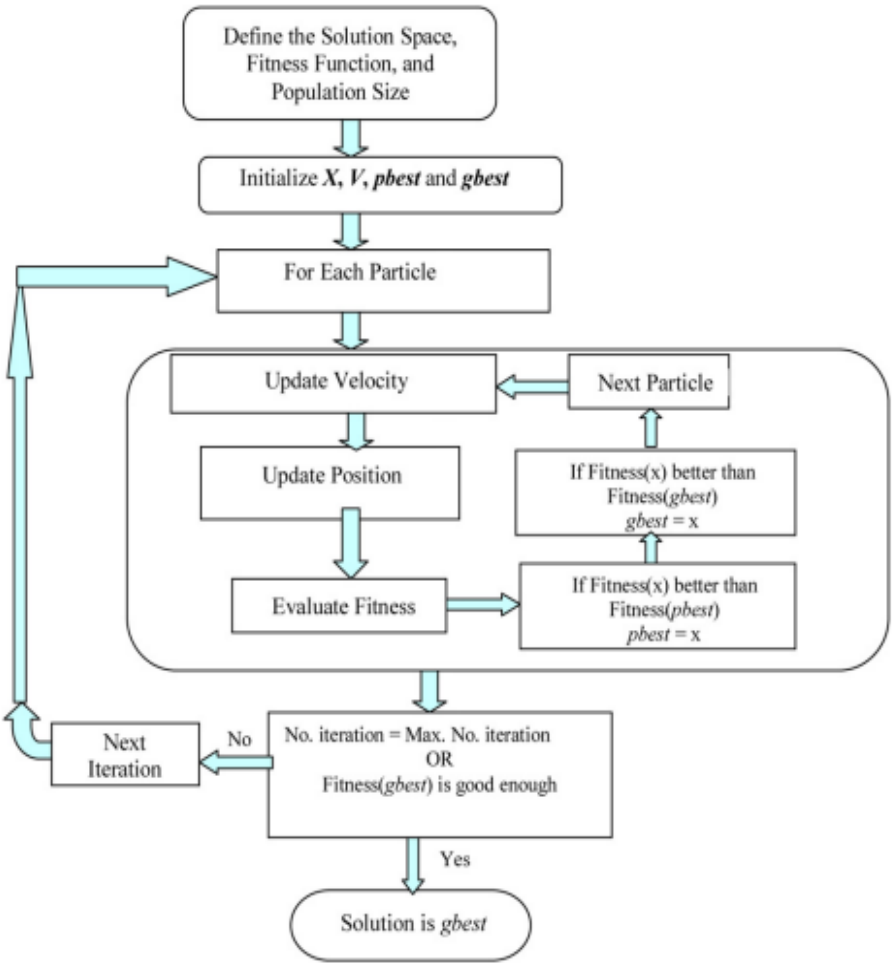

Figura 15. Diagrama de flujo del algoritmo de Nube de Partículas (PSO) [34].

3.3.2.3. Algoritmo de Manada de Felinos (CSO)

El Algoritmo de Manada de Felinos CSO (*Cat Swarm Optimization*) surge como una alternativa a algunos algoritmos evolutivos como PSO y DE con el fin de solucionar sus dos grandes problemas, el tiempo de convergencia y el estancamiento en mínimos locales.

En 2013 [25], Suman Kumar Saha et al. defienden el empleo del algoritmo de Manada de Felinos (CSO) como un método de optimización para el diseño de filtros FIR. Este algoritmo imita el comportamiento natural de los felinos.

Los felinos sienten una fuerte curiosidad por los objetos que se mueven y poseen buenas habilidades de caza. A pesar de que pasan la mayor parte del tiempo descansando, siempre permanecen alerta y se mueven muy lentamente. Sin embargo, cuando detectan la presencia de una presa, la persiguen muy rápidamente empleando gran cantidad de energía. Estas dos particularidades, (1) descansar con movimientos lentos y (2) cazar con alta velocidad, son caracterizadas matemáticamente por el **Modo de Búsqueda** y **Modo de Caza** respectivamente.

A. Modo de búsqueda

En primer lugar, se definen una serie de términos que se van a emplear para este modo.

- El número de copias de un felino producidas durante el modo de búsqueda se denomina Grupo de Memoria de Búsqueda (*SMP*).

- La máxima diferencia entre los valores nuevos y los antiguos en la dimensión seleccionada para mutar, se denomina Rango de Búsqueda de Dimensión Seleccionada (*SDR*).
- El número de dimensiones a mutar es conocido como Contador de Dimensiones a Cambiar (*CDC*).
- Para garantizar que los felinos pasan la mayor parte del tiempo descansando y observando, es decir, la mayor parte del tiempo en el Modo de Búsqueda, se introduce un término llamado Ratio de Mezcla (*MR*). Este Ratio es una fracción de la población total de individuos de valor muy pequeño.

Los pasos que se siguen en el Modo de Búsqueda son los siguientes:

1) Se selecciona aleatoriamente una fracción *MR* de la población total n_p que serán los felinos en estado de caza y el resto serán felinos en modo de búsqueda. Se establece un contador de iteración *i* que recorrerá todos los felinos en estado de búsqueda.
2) Se crean *SMP* copias del felino i-ésimo.
3) Basado en el *CDC*, se actualizan las posiciones de cada una de las copias del felino añadiendo o restando una fracción *SDR* a la posición actual.
4) Se evalúa la función de aptitud para todas las copias.
5) Se selecciona el mejor candidato de todas las copias y se establece la posición del felino i-ésimo como la de la mejor copia.
6) Se repite desde el paso 2 hasta que todos los felinos de búsqueda se han analizado.

B. Modo de caza

Este modo corresponde a una técnica de búsqueda local para la optimización del problema. En este modo, el felino caza a su objetivo empleando gran cantidad de energía. La rápida persecución del felino se modela matemáticamente como un cambio muy pronunciado en su posición. Se definen la posición y la velocidad del felino i-ésimo en el espacio D-dimensional como:

$$\textbf{Posición} \rightarrow \mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{iD}) \quad ; \quad \textbf{Velocidad} \rightarrow \mathbf{x}_i = (v_{i1}, v_{i2}, ..., v_{iD})$$

La mejor posición global de todos los felinos (la que presenta mejor función de aptitud) se representa como:

$$\mathbf{gbest} = (gbest_1, gbest_2, ..., gbest_D)$$

Las ecuaciones de actualización de Velocidad y Posición son:

Cambio en la Velocidad:

$$\mathbf{v}_{ij} = w \cdot \mathbf{v}_{ij} + C + r \cdot (\mathbf{gbest} - \mathbf{x}_{ij}) \qquad (30)$$

Cambio en la Posición:

$$\mathbf{x}_{ij} = \mathbf{x}_{ij} + \mathbf{v}_{ij} \qquad (31)$$

Donde w es el coeficiente de inercia, C es una aceleración constante, r un número aleatorio uniformemente distribuido entre [0,1] y j la generación actual. Como podemos ver, es una implementación del algoritmo de nube de partículas PSO, pero con algún cambio en el cálculo

de la velocidad y añadiendo la fase previa de división de los felinos en el modo de búsqueda y modo de caza.

El algoritmo CSO alcanza la solución óptima empleando dos grupos de felinos, un grupo contiene los gatos en modo de búsqueda y el otro en modo de caza. Los dos grupos emplean técnicas distintas que combinadas dan solución al problema de optimización.

El proceso entero del algoritmo para el diseño de filtros uniendo ambos modos es el siguiente:

1) Se crean F felinos para el proceso.
2) Se esparcen aleatoriamente los felinos en el espacio de soluciones D-dimensional con valores entre [-2,2], es decir, se establece un valor aleatorio de la posición para cada una de las D-dimensiones de cada felino entre [-2,2]. Se constituye una velocidad con valor entre [-0.1, 0.1]. El número de dimensiones D será (N+1)/2 con N-1 el orden del filtro y la posición contendrá la mitad de coeficientes de la respuesta al impulso (**h**(n)).
3) De acuerdo con el valor de MR se extrae un número de felinos para el grupo en modo caza y el resto para el grupo en modo de búsqueda.
4) Se evalúa el valor de aptitud de cada felino aplicando a las posiciones de los felinos la función de aptitud. Esta función representa los criterios de nuestro objetivo. Se conserva al mejor felino en memoria como **gbest**. (Solamente es necesario conservar la posición del felino que representa la mejor solución hasta ahora).
5) Se desplazan los felinos de acuerdo al modo en que se encuentren (con los mecanismos expuestos anteriormente).
6) Se comprueba la condición de fin de bucle, en este caso se establece un número de iteraciones máximo. Si se satisface la condición, se termina el programa, sino, se repiten los pasos 3-6.

Una vez finalizado el bucle, se obtiene la posición óptima del felino, es decir, los coeficientes óptimos del filtro **h**(n). A partir del vector **gbest**, que contiene (N+1)/2 coeficientes, se concatena consigo mismo (considerando la simetría de los filtros FIR) para obtener la respuesta al impulso **h**(n) con N coeficientes.

3.3.2.4. Algoritmo de búsqueda del Cuco

El algoritmo de búsqueda del Cuco (CSA) ha sido desarrollado por X. S. Yang y S. Deb en 2009 [27]. Los algoritmos inspirados en el cuco son unos nuevos mecanismos de optimización evolutivos basados en el comportamiento parasitario de algunas especies de Cucos (un tipo de ave), combinado con el vuelo de Lévy de algunos pájaros y moscas de la fruta. Las siguientes subsecciones ilustran estos conceptos en detalle.

A. Reproducción de los Cucos

El algoritmo CSA está basado, como hemos dicho, en la vida de un ave denominada Cuco. Algunas especies del Cuco practican el parasitismo depositando sus huevos en los nidos de otras especies de aves. Las aves huéspedes pueden entrar en conflicto con sus ocupantes indeseados (cucos) y, por ejemplo, si descubren que los huevos no son suyos, los tiran del nido o, simplemente, abandonan el nido y construyen uno nuevo en otro lugar. Algunas especies de cuco como la Tapera Naevia han evolucionado hasta tal punto que el color y los patrones de los huevos de dichas especies son muy parecidos a los huevos de las especies huéspedes.

El algoritmo CSA emplea las siguientes representaciones [27]:

Cada huevo del nido representa una solución y el huevo de Cuco representa una nueva solución. El objetivo es emplear las nuevas y potencialmente mejores soluciones (huevos de Cuco) para reemplazar las no tan buenas soluciones de los nidos. En el caso más simple, cada nido tiene un huevo, pero se puede extender a casos más complicados donde cada nido tiene varios huevos representando un conjunto de soluciones.

CSA está basado en tres reglas básicas:

1. Cada Cuco deposita un huevo cada vez y deja su huevo en un nido elegido aleatoriamente.
2. Los mejores nidos con gran calidad de huevos se mantienen para la siguiente iteración.
3. Se fija el número total de nidos huéspedes y se establece que el huevo dejado por el Cuco es descubierto por el ave huésped con una probabilidad ($P_a \in [0,1]$).

Este algoritmo es formulado para obtener los resultados óptimos haciendo balance entre la exploración y explotación del espacio de búsqueda.

B. Vuelo de Lévy

Varios estudios han demostrado que el vuelo de varios animales e insectos siguen las características de los vuelos de Lévy. En la naturaleza se puede observar que los animales buscan comida de una manera aleatoria o cuasi-aleatoria. En general, la dirección que toman algunos animales depende directamente de una probabilidad que puede ser modelada matemáticamente.

Un estudio reciente muestra que las moscas de la fruta exploran el entorno utilizando una serie de caminos de vuelo rectos con giros de 90 grados, siguiendo lo que se denomina un patrón de vuelo Lévy. Dicho comportamiento ha sido empleado en problemas de optimización y de búsqueda del óptimo cuyos resultados han demostrado una prometedora capacidad.

La implementación del vuelo Levy en el Algoritmo de búsqueda del Cuco es utilizada para generar una nueva solución durante el proceso de exploración.

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \alpha \oplus \mathbf{Levy}(\lambda) \;;\; \text{donde,} \qquad (32)$$

α ($\alpha > 0$) es el tamaño de salto (relacionado con el tipo de problema a resolver), x_i^{t+1} es la nueva solución y x_i^t es la solución actual. Esta ecuación representa un paso aleatorio denominado cadena de Markov. Esto significa que la próxima solución depende de la solución actual y de la probabilidad de transición. $\mathbf{Levy}(\lambda)$ sigue una distribución Lévy con infinita media e infinita varianza ($1 < \lambda \leq 3$), Ec. (33). Esto permite a una parte de la generación alejarse de la actual solución, evitando que el algoritmo se quede atrapado en los mínimos locales.

$$\mathbf{Levy}(\lambda) \sim \mathbf{u} = \mathbf{t}^{-\lambda} \;, \qquad (1 < \lambda \leq 3) \qquad (33)$$

Los pasos que emplea dicho algoritmo y los parámetros elegidos por los autores de [28] para el diseño de filtros FIR paso alto y elimina banda son los siguientes:

1) Se inicializa el tamaño de población de nidos huéspedes ($n_c = 25$), la probabilidad de que el ave huésped descubra el huevo de Cuco ($P_a = 0.25$) y los límites superior e inferior de los coeficientes de los filtros HPF y BSF [-1,1]. Se establece el número máximo de iteraciones T=1000 y el contador de iteraciones t=1.

2) Se genera aleatoriamente una población de n_c nidos huéspedes, donde cada uno representa una solución candidata (conjunto de coeficientes del filtro). La solución se representa como $\mathbf{x}_i^t = [x_0, x_1, \ldots, x_N]$, donde cada componente del vector, representa uno de los N+1 coeficientes del filtro a optimizar.
3) Se evalúa la función de aptitud con los nidos huéspedes actuales $\mathbf{x}_i^t$, obteniendo F^t (El error respecto a la función objetivo en la iteración t).
4) Se generan nuevas soluciones (nidos) empleando la fórmula del vuelo Lévy descrito anteriormente ($\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \alpha \oplus \mathbf{Levy}(\lambda)$) y se calcula su aptitud F^{t+1}. Se comparan los valores de aptitud.
5) Para el problema de minimización del error, si $F^{t+1} < F^t$, los nidos actuales $\mathbf{x}_i^t$ son remplazados por los nuevos generados por el vuelo Lévy $\mathbf{x}_i^{t+1}$.
6) Una fracción P_a de los peores nidos obtenidos en el paso 5 son abandonados y nuevos nidos ($\mathbf{x}_n$) son generados empleando vuelo aleatorio.
7) Se computan las aptitudes de los nuevos nidos. Se actualiza el valor del mejor nido de la generación $\mathbf{x}_p$ (el que tenga mejor aptitud, o lo que es lo mismo, menor error).
8) El mejor nido obtenido hasta ahora $\mathbf{x}_b$ es remplazado por el mejor nido de la generación $\mathbf{x}_p$ si su función de aptitud $F_p < F_b$.
9) Se repiten los pasos de 3-8 hasta que el máximo número de iteraciones es alcanzado y la solución $\mathbf{x}_b$ es una solución óptima a nuestro problema.

3.3.2.5. Algoritmo de Búsqueda de Comida de Bacterias

La selección natural favorece la propagación de genes de los animales que poseen mejores técnicas de búsqueda, manejo y obtención de comida, eliminando aquellos que poseen técnicas más endebles. Con las características físicas propias de cada animal y medioambientales, los animales tratan de maximizar el consumo de energía por unidad de tiempo. Esta idea es la que inspira a Das et al., 2009 [30] para desarrollar el algoritmo de Búsqueda de Alimento de las Bacterias (BFO) para resolver problemas de optimización no lineales.

Para poder explicar la estrategia de búsqueda de comida de las bacterias (en concreto la Escherichia Coli, presente en el intestino de los seres humanos), son empleados cuatro procesos: **Chemotaxis, Inteligencia en Grupo de Bacterias (*Swarming*), Reproducción y Dispersión.**

Las bacterias poseen un conjunto de flagelos rígidos que permiten el movimiento. Dicho movimiento empleado en la búsqueda de comida se puede clasificar en dos maneras o modos de movimiento: **Natación** y la caída en conjunto denominada **Chemotaxis**. Un movimiento en una dirección determinada es llamado **Natación** o **Carrera**, mientras que, la Chemotaxis, es el movimiento del conjunto de Bacterias en diferentes direcciones. Durante la vida de las bacterias, se alternan ambos modos de movimiento. La rotación de los flagelos en el sentido de las agujas del reloj da como resultado la Chemotaxis y el movimiento de rotación en el sentido contrario a las agujas del reloj da lugar a la Natación.

La probabilidad de búsqueda de nutrientes concentrados, la capacidad de atacar en grupo a una presa muy grande para matarla y digerirla y la protección frente a los depredadores del grupo de Bacterias son los principales objetivos y motivaciones de la inteligencia social de las Bacterias para la búsqueda de alimento. El éxito de la búsqueda de comida de cada individuo del grupo de Bacterias deriva de la agrupación, el mecanismo de comunicación y la inteligencia en grupo. Para atraer al resto de bacterias a la óptima dirección de convergencia del algoritmo, es necesario que se comparta con otras bacterias información sobre la concentración de nutrientes (punto óptimo). Esto es denominado *swarming*. Para alcanzar esto, una función de penalización

es añadida a la función a optimizar. Dicha función de penalización está basada en la distancia relativa de cada Bacteria a la Bacteria con mejor aptitud (durante el proceso de búsqueda). La función de penalización se convierte en 0 cuando todas las Bacterias convergen a la solución deseada.

Después de verse involucrado en multitud de fases de Chemotaxis, el conjunto original de bacterias es apto para la **reproducción**. El proceso de reproducción consiste en dividir una Bacteria en dos idénticas. Esto se ve reflejado durante el proceso de optimización, reemplazando la mitad de las Bacterias con una peor función de aptitud, por la mitad de Bacterias con mejor aptitud. La población total de Bacterias se mantiene constante durante el proceso de evolución.

La **dispersión** de un conjunto de Bacterias a una posición nueva resulta una alteración al proceso natural de evolución. Este concepto es empleado para remplazar un conjunto nuevo de Bacterias cercanas a la localización de la comida para evitar el estancamiento en mínimos locales.

Las variables a optimizar son los coeficientes del filtro. El proceso empleado por el algoritmo BFO para la optimización del problema se resume en un conjunto de pasos que se pueden observar en las Fig. 16 y 17. Las partes que componen el diagrama de flujo se detallan a continuación.

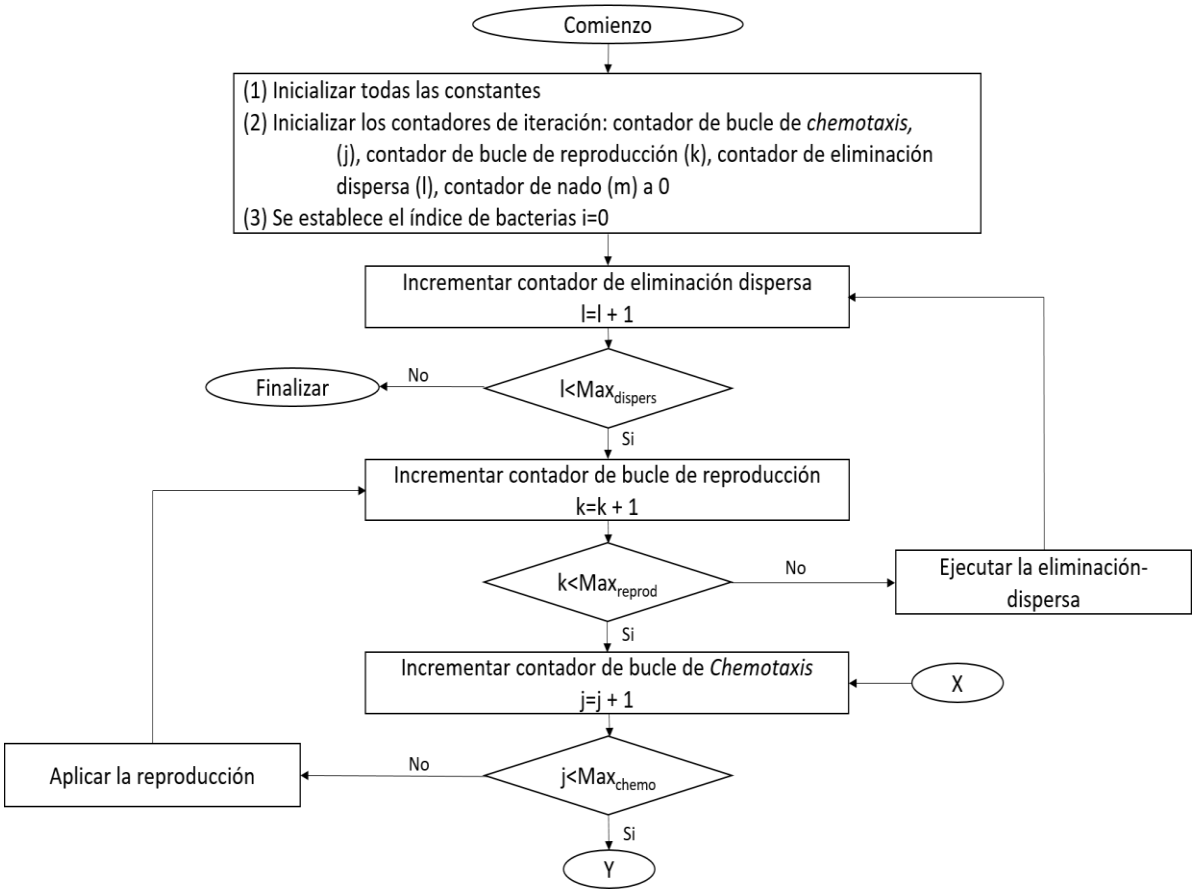

Figura 16. Primera parte del diagrama de flujo del algorimo BFO.

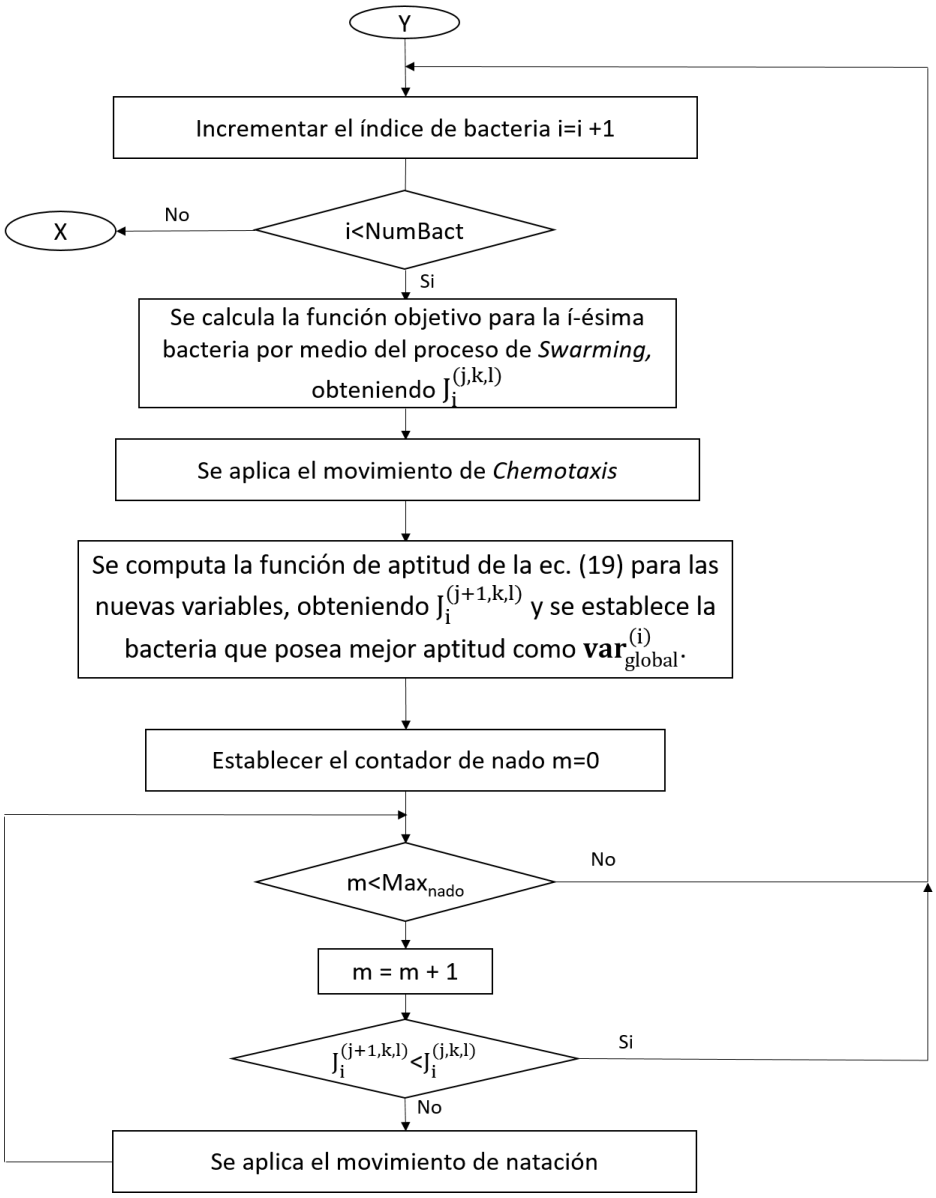

Figura 17. Segunda parte del diagrama de flujo del algorimo BFO.

Inicialización del número total de Bacterias (numBact) con (N+1)/2 soluciones (variables a optimizar) que satisfacen las restricciones de valores máximos y mínimos de las variables (var_{max} y var_{min}), se calculan sus aptitudes por medio de la Ec. (19) y se establece la bacteria que posea mejor aptitud como $\mathbf{var}_{global}^{(0)}$.

Se inicializan las distintas constantes empleadas por el algoritmo:

a. Máximos ciclos de *Chemotaxis* (Max_{chemo}).
b. Máxima longitud de nado (Max_{nado}).
c. Máximos ciclos de reproducción (Max_{reprod}).
d. Máximos ciclos de Dispersión ($Max_{dispers}$).
e. Número máximo de iteraciones totales max_cycle = $Max_{reprod} \cdot Max_{chemo}$.
f. Ratio de reproducción de las bacterias (Sr).
g. La profundidad del atrayente lanzado por la célula $d_{attract}$.
h. La anchura de la señal atrayente $w_{attract}$.

i. La altura del efecto repelente $d_{repellant}$.
j. La anchura del repelente $w_{repelant}$.
k. La probabilidad de eliminación P_{ed}.
l. Otras constantes positivas: c_{max}; c_{min}; d_1; d_2.

La eliminación-dispersa consiste en recorrer y seleccionar las bacterias de la población a las que aplicar la eliminación-dispersa con una probabilidad P_{ed}. Para hacer esto se eliminan dichas bacterias y se dispersan nuevas en una localización aleatoria, manteniendo el tamaño de la población.

El proceso de reproducción elimina la mitad de la población de bacterias que poseen la peor aptitud y duplica la mitad mejor adaptada para mantener el tamaño de población.

Se aplica el *Swarming* a las aptitudes de las bacterias a través de la siguiente ecuación:

$$J_i^{(j,k,l)} = J_i^{(j,k,l)} - \sum_{numBact} d_{attract} \, x \left(e^{-w_{attract} \, x \, \sigma_i^{(j+1,k,l)}} \right) + \sum_{numBact} d_{repelant} \, x \left(e^{-w_{repelant} \, x \, \sigma_i^{(j+1,k,l)}} \right) \quad (34)$$

Donde,

$$\sigma_i^{(j+1,k,l)} = \sigma_i^{(j,k,l)} + \sum_{numBact} \left(var_{global}^{(j,k,l)} - var_i^{(j,k,l)} \right)^2 \quad (35)$$

El movimiento de *Chemotaxis* que se aplica a las variables (soluciones) es el siguiente:

Se calcula las variables de la siguiente iteración imponiendo los límites establecidos de valor máximo y mínimo (var_{max} y var_{min}) como:

$$var_i{}^{(j+1,k,l)} = var_i{}^{(j,k,l)} + (c^{var_i} - d_2) \, x \frac{\Delta^{(var_i)}}{\Delta\Delta} \quad (36)$$

Donde,

$$\Delta^{(var_i)} = (2 \cdot round(rand()) - 1) \cdot rand(); \qquad \Delta\Delta = \sqrt{\Delta^{(var_i)} x \, \Delta'^{(var_i)}} \quad (37)$$

Y

$$c^{var_i} = c_{max} - (c_{max} - c_{min}) x \frac{j \, x \, k}{max\_cycle} \quad (38)$$

La natación se aplica obtienendo $\Delta^{(var)}$ y $\Delta\Delta$ a partir de la Ec. (37) y se calculan las nuevas variables con sus restricciones:

$$\mathrm{var}_i{}^{(\mathrm{j+1,k,l})} = \mathrm{var}_i{}^{(\mathrm{j,k,l})} + (\mathrm{c}^{\mathrm{var}_i} + \mathrm{d}_1)\ \mathrm{x}\frac{\Delta^{(\mathrm{var}_i)}}{\Delta\Delta} \qquad (39)$$

Y posteriormente se computa la función de error de aptitud $\mathrm{J_i}^{(\mathrm{j+1,k,l})}$ para cada bacteria con la Ec. (19).

3.3.3. Métodos Basados en Redes Neuronales

Un trabajo que supuso un antes y un después en la computación natural es [42], donde McCulloch y Pitts introdujeron en 1943 el primer modelo matemático de una neurona que dio lugar a las redes neuronales artificiales (RNAs). Las RNAs en inglés *ANNs* (*Artificial Neural Networks*) se pueden definir como sistemas de procesado de información basados en las células (neuronas) del sistema nervioso, en el mayor de los casos, el cerebro humano. La mayor parte de los usos de las RNAs es la resolución de problemas.

Los RNAs pueden definirse en el campo de la ingeniería como un mecanismo computacional capaz de adquirir, representar y mapear información de un espacio multivariable a otro, dando un conjunto de datos representando dicho mapeo [35].

3.3.3.1. **Red Neuronal Artificial (RNA)**

Las Redes Neuronales Artificiales (RNAs) son sistemas de computación formados por un número de elementos sencillos y altamente interconectados que procesan información gracias a su respuesta dinámica a las entradas externas. Una de las características distintivas de las RNAs es la capacidad de aprender de la experiencia y de ejemplos y la posibilidad de adaptarse a situaciones cambiantes.

Los RNAs son mucho más sencillos que el cerebro humano, están compuestos por menos elementos y operan de forma enormemente abstracta. El proceso de entrenamiento del RNA implica presentar un conjunto de ejemplos (patrones de entrada) con sus respectivas salidas conocidas. El sistema ajusta los pesos de las conexiones internas entre unidades de procesado (neuronas) con el fin de minimizar el error entre la salida de la red a los patrones de entrada y las salidas conocidas. Después de que el sistema está correctamente entrenado y probado, es capaz de predecir las salidas correspondientes a entradas no conocidas, dentro del dominio cubierto por los ejemplos de entrenamiento.

Las ventajas del empleo de RNAs son las siguientes:

- Una red neuronal puede realizar tareas que un programa lineal no puede.
- Cuando un elemento de la red neuronal falla, puede continuar sin ningún problema debido a su naturaleza paralela.
- Una red neuronal es capaz de aprender y no necesita reprogramarse.
- Puede ser implementado en múltiples aplicaciones.

Otro aspecto de las redes neuronales es que están formadas por diferentes estructuras y arquitecturas, que requieren diferentes implementaciones, pero a pesar de que parece un sistema complejo, es relativamente simple de implementar.

Como se ha descrito anteriormente, las redes neuronales son sistemas adaptativos, lo que significa que los parámetros del sistema durante la operación, normalmente en la fase de entrenamiento. Después de esta fase, los parámetros permanecen constantes durante la fase de test y el sistema se ejecuta para obtener la solución al problema planteado.

Para aplicar las RNAs al diseño de filtros FIR es necesario definir la respuesta en frecuencia del filtro como suma de cosenos, de la forma:

$$\mathrm{H(w)} = \sum_{\mathrm{m=0}}^{\frac{\mathrm{N-1}}{2}} \mathbf{a}(\mathrm{m}) \cdot \cos(\mathrm{mw}) \qquad (40)$$

$$\text{Donde, } \mathbf{a}(0) = \mathbf{h}\left(\tfrac{\mathrm{N-1}}{2}\right);\ \ \mathbf{a}(\mathrm{m}) = 2\mathbf{h}\left(\tfrac{\mathrm{N-1}}{2} - \mathrm{m}\right) \text{para } \mathrm{m} \neq 0 \qquad (41)$$

Siendo **h**(n) la respuesta al impulso del filtro diseñado. Para el diseño del filtro óptimo será necesario calcular los coeficientes de **a**(m) en la Ec.(40) que se acerquen a la respuesta en magnitud óptima.

Modelo de diseño de filtros óptimos a partir de redes neuronales

La Fig. 18 muestra el modelo basado en redes neuronales propuesto en [43] para el diseño de filtros FIR con una función de activación lineal basada en bases cosenoidales. La salida de la red neuronal se puede expresar en forma vectorial como:

$$\mathbf{h}_{\mathrm{N}}^{\mathrm{k}} = \mathrm{C}^{\mathrm{T}} \cdot \mathbf{a}_{\mathrm{k}} \qquad (42)$$

Donde $\mathbf{h}_{\mathrm{N}}^{\mathrm{k}}$ es el vector de salida de la red neuronal, C^{T}es la matriz de transformación de las unidades ocultas de la red neuronal, $\mathbf{a}_{\mathrm{k}}$ es el vector de pesos o coeficientes de la red neuronal, definidos como:

$$\mathbf{a} = \left[\mathrm{a}_0, \mathrm{a}_1, \dots, \mathrm{a}_{\frac{\mathrm{N-1}}{2}}\right]^{\mathrm{T}},\ \ \mathrm{w} = \left[\mathrm{w}_0, \mathrm{w}_1, \dots, \mathrm{w}_{\frac{\mathrm{N-1}}{2}}\right]^{\mathrm{T}}, \mathbf{h}_{\mathrm{N}} = \left[\mathrm{H}_{\mathrm{N}}(\mathrm{w}_0), \mathrm{H}_{\mathrm{N}}(\mathrm{w}_1), \dots, \mathrm{H}_{\mathrm{N}}(\mathrm{w}_{\frac{\mathrm{N-1}}{2}})\right]^{\mathrm{T}},$$

$$\mathrm{C} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \cos(\mathrm{w}_0) & \cos(\mathrm{w}_1) & \cdots & \cos(\mathrm{w}_{\frac{\mathrm{N-1}}{2}}) \\ \vdots & \vdots & \ddots & \vdots \\ \cos\left(\frac{\mathrm{N}-1}{2} \cdot \mathrm{w}_0\right) & \cos\left(\frac{\mathrm{N}-1}{2} \cdot \mathrm{w}_1\right) & \cdots & \cos\left(\frac{\mathrm{N}-1}{2} \cdot \mathrm{w}_{\frac{\mathrm{N-1}}{2}}\right) \end{bmatrix}$$

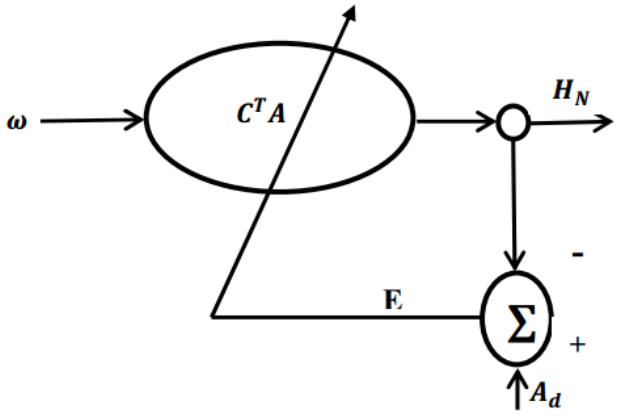

Figura 18. Modelo de diseño basado en Redes Neuronales.

Posteriormente se define la función de error **e** como:

$$\mathbf{e}(\mathrm{k}) = \mathbf{a}_\mathrm{d} - \mathbf{h}_\mathrm{N}^\mathrm{k} \qquad (43)$$

Donde el vector de error **e**(k) se calcula como el error entre la salida deseada ($\mathbf{a}_d$) y la salida actual de la red neuronal. Se calcula el índice de rendimiento P como:

$$\mathrm{P}(\mathbf{e}) = \frac{1}{\mathrm{J}} \cdot \sum_{\mathrm{i}=1}^{\frac{\mathrm{N}-1}{2}} \mathrm{W} \cdot \mathbf{e}^2(\mathrm{i}) \qquad (44)$$

Donde, W>0, y $\mathrm{J} = \sum_{\mathrm{l}=1}^{\frac{\mathrm{N}-1}{2}} \mathrm{W}$. El objetivo será minimizar el índice *P*, calculando recursivamente **a**, empleando:

$$\mathbf{a}_{\mathrm{k}+1} = \mathbf{a}_\mathrm{k} - \eta \frac{\partial \mathrm{p}}{\partial \mathbf{a}_\mathrm{k}} \qquad (45)$$

η es el ratio de aprendizaje de la red neuronal. Después de diferenciar respecto $\mathbf{a}_\mathrm{k}$ al rendimiento P, dado por la Ec. (44), obtenemos:

$$\frac{\partial \mathrm{p}}{\partial \mathbf{a}_\mathrm{k}} = \frac{\partial \mathbf{h}_\mathrm{k}^\mathrm{N}}{\partial \mathbf{a}_\mathrm{k}} \cdot \frac{\partial \mathbf{e}(\mathrm{k})}{\partial \mathbf{h}_\mathrm{k}^\mathrm{N}} \cdot \frac{\partial \mathrm{p}}{\partial \mathbf{e}(\mathrm{k})} = -\mathrm{C}\mathbf{e}(\mathrm{k}) \rightarrow \mathbf{a}_{\mathrm{k}+1} = \mathbf{a}_\mathrm{k} + \eta \mathrm{C}\mathbf{e}(\mathrm{k}) \qquad (46)$$

Es necesario elegir un ratio de aprendizaje adecuado para asegurar la convergencia de la red neuronal. El método de diseño se puede resumir en los siguientes pasos:

1. Se muestrea la respuesta en frecuencia deseada ($\mathbf{a}_d$) uniformemente para obtener el conjunto de muestras de entrenamiento de la red neuronal. Los puntos de muestreo son $\omega_\mathrm{i} = \frac{2\pi}{\mathrm{N}-1} \cdot \mathrm{i};\ \ \mathrm{i} = 0,1,\dots,\frac{\mathrm{N}-1}{2}$.
2. Se produce un vector de coeficientes aleatorio (**a**) empleando una función aleatoria, se define el índice de rendimiento mínimo ε y la tasa de aprendizaje η.
3. Se especifica el vector de pesos de coeficientes W de acuerdo con el siguiente esquema:

$$\mathbf{W} = \begin{cases} w_p > 1 & \text{en la banda de paso} \\ w_{pc} \geq 1 & \text{cuando } w_i \text{ está en el borde de la banda de paso} \\ w_t = 1 & \text{en la banda de transición} \\ w_s \geq 1 & \text{en la banda de corte} \end{cases}$$

4. Se genera la matriz de transformación C de las unidades ocultas de la red neuronal. Se produce el vector de salida predicho $\mathbf{h}_N$ de la red neuronal empleando (42).
5. Se calcula el vector de error **e**(k) y el índice de rendimiento P a partir de (43) y (44).
6. Actualizamos el vector de pesos $\mathbf{a}_{k+1}$ usando (45).
7. Si P > ε, se salta al paso 4, en caso contrario, se detiene el entrenamiento de la red neuronal.

3.3.4. Métodos híbridos y otras aproximaciones de Computación Natural

Estos métodos se basan en la combinación de diferentes métodos o técnicas, pueden ser de carácter evolutivo, bioinspiradas, neuronales o cualquier otro algoritmo de optimización con el fin de obtener un modelo nuevo capaz de superar las limitaciones que presentan los métodos por separado. Además, se incluirán otros métodos de optimización que no se pueden clasificar en ninguno de los grupos anteriores.

3.3.4.1. **Algoritmo genético Híbrido (HGA)**

A pesar de que se ha probado que los algoritmos genéticos superan las limitaciones de los métodos tradicionales, son comparativamente ineficientes en términos del tiempo que tarda en alcanzar una solución aceptable y de calidad. El uso de esquemas híbridos es interesante, aunque normalmente limitado debido a la gran carga computacional de estos. El HGA que se va a detallar, propuesto por K. Boudjelaba et al. en 2014 [5], combina un proceso genético puro (GA) combinado con una aproximación local de una manera innovadora y eficiente para el diseño de filtros FIR.

El proceso genético incluye múltiples mecanismos de interacción para hacer al algoritmo genético auto adaptativo en términos de diversidad (variedad genética) y elitismo (sólo los cromosomas mejor adaptados sobreviven). Mientras que, la aproximación local, implica una convergencia del algoritmo de forma más efectiva, implicando un menor coste computacional. Solamente los cromosomas más prometedores (sometidos a un proceso de selección específico), se someten a la aproximación local por lo que también reducimos carga al algoritmo original.

Manejando estos mecanismos específicos de una forma innovadora y eficiente, y optimizando el papel de la búsqueda local, el proceso evolutivo para alcanzar el óptimo global puede acelerarse sin perder eficacia. Estos mecanismos también contribuyen en la reducción del espacio de búsqueda y ayudan al algoritmo genético a escapar de los óptimos locales. De esta manera, se previene la convergencia prematura del GA. Como hemos visto, ambas características (calidad y coste computacional) son optimizadas con el HGA.

Podemos establecer los pasos que sigue este algoritmo al igual que hemos hecho con el algoritmo genético tradicional:

1) **Inicialización:** Se define un tamaño de población y un número de iteraciones del algoritmo.

2) Se evalúa la población mediante la función de aptitud.
3) Ejecutamos el proceso de **selección** sobre la población actual.
4) Se decide si aplicar el método ***refresh*** a la población. Consiste en aplicar una mutación muy fuerte a los cromosomas. La decisión de aplicarlo o no, se establece con un algoritmo con el fin de evitar la convergencia prematura hacia un óptimo local.
5) Se aplica el algoritmo de **cruce.**
6) Aplicamos el proceso de **mutación** con o sin los mecanismos de diversificación. específica a la mutación, que deciden el nivel de esta.
7) Se ejecuta el algoritmo de aproximación local con los parámetros apropiados en los cromosomas seleccionados. Dicho algoritmo de aproximación local se detalla en [5].
8) Se detiene porque ya ha cumplido las restricciones impuestas sobre la función de aptitud o se salta al paso 2.

3.3.4.2. Algoritmo de Evolución diferencial con Nube de Partículas (DEPSO)

Como se ha explicado anteriormente, la optimización por nube de partículas (PSO) es un algoritmo de optimización evolutivo, basado en el comportamiento de grupos de animales como los bancos de peces o bandadas de pájaros. En [22], B. Luitel y G. K. Venayagamoorthy aplican dicho algoritmo combinado con el Algoritmo de Evolución Diferencial para el diseño de filtros FIR. Ambos algoritmos los hemos detallado anteriormente en este documento.

A. Algoritmo de Evolución Diferencial (DE)

Es otra técnica de búsqueda para la minimización de funciones basada en poblaciones. Su utilización como algoritmo único para el diseño de filtros ha sido descrito en [24]. En el algoritmo DE, la diferencia entre dos vectores de poblaciones es añadida a un tercer vector y optimizada empleando selección cruce y mutación al igual que en los GAs. Cada individuo (**padre**) es, en primer lugar, mutado de acuerdo al operador Diferencia. Este individuo mutado, denominado **descendiente**, es recombinado con el padre bajo unos criterios como la tasa de cruce. Se evalúa tanto la aptitud del padre como del descendiente y este se selecciona para la siguiente generación solamente si presenta mejor aptitud que su padre.

La combinación del DE con el PSO confiere un nuevo método de optimización denominado Algoritmo de Evolución Diferencial con Nube de Partículas (DEPSO).

B. Algoritmo de Evolución Diferencial con Nube de Partículas (DEPSO)

En el DEPSO, un descendiente es creado a partir de la mutación de un individuo (padre). En [24] el padre utilizado para la mutación es **gbest** (variable que contiene la posición de la partícula con mejor función de aptitud). Para la mutación, 4 partículas son elegidas aleatoriamente de la población. El **error ponderado** ($\delta_{2,d}$) entre las posiciones de las partículas es empleado para mutar el padre y obtener el descendiente según la siguiente fórmula:

Condición, *Si (rand()<CR o d == k):*

$$\mathbf{t}_{id} = \mathbf{p}_{gd} + \boldsymbol{\delta}_{2,d} \text{ , siendo } \boldsymbol{\delta}_{2,d} = \frac{(\mathbf{p}_{1,d} - \mathbf{p}_{2,d}) + (\mathbf{p}_{3,d} - \mathbf{p}_{4,d})}{2} \quad (47)$$

Donde $\mathbf{t}_{id}$ es el descendiente y $\mathbf{p}_{gd}$ el padre empleado (mejor posición de la partícula). El error $\boldsymbol{\delta}_{2,d}$ se calcula a partir de las posiciones de las 4 partículas seleccionadas aleatoriamente ($\mathbf{p}_{1,d}$, $\mathbf{p}_{2,d}$, $\mathbf{p}_{3,d}$, $\mathbf{p}_{4,d}$). La mutación solamente se lleva a cabo si un número aleatorio entre [0,1] es menor que la tasa de reproducción CR, o si la posición de las partículas en alguna dimensión escogida

aleatoriamente (k) es mutada. Esto asegura que el descendiente nunca es igual que el padre. Posteriormente se evalúa la aptitud del descendiente y el hijo sustituye al padre solamente si tiene una mejor aptitud que el padre. Si no, el padre se retiene para la siguiente iteración. El mecanismo del DEPSO se puede resumir en los siguientes pasos:

1) Se inicializa cada una de las partículas con una posición y velocidad inicial aleatoriamente distribuidas y el contador de iteraciones.
2) Se calcula el valor de aptitud para cada partícula. Si el valor de *fitness* es mejor que el mejor almacenado en memoria (pbest), se establece dicho valor de aptitud como pbest; si no, se continúa al paso 3.
3) Se elige a la partícula con mejor valor de aptitud de todas las partículas existentes y se establece como **gbest**. Continúa el paso 4.
4) Se actualizan la posición y la velocidad de cada partícula de acuerdo a las fórmulas de cambio de velocidad y de posición. Salto al paso 5.
5) Para cada partícula padre se seleccionan 4 partículas aleatoriamente de la población y se calcula el error ponderado. Se muta el padre siguiendo el criterio anteriormente expuesto y se calculan las aptitudes del padre y del descendiente. Se continúa al paso 6.
6) Si la aptitud del padre es peor que la del hijo, se reemplaza el padre por el hijo; en caso contrario, se mantiene el padre con el mismo valor y se continúa al paso 7.
7) Finaliza el algoritmo si se alcanza el número máximo de iteraciones o el criterio de mínimo error es satisfecho. Si no, se salta al paso 2.

Se puede observar el diagrama de flujo del algoritmo DEPSO combinando DE y PSO en la Fig. 19.

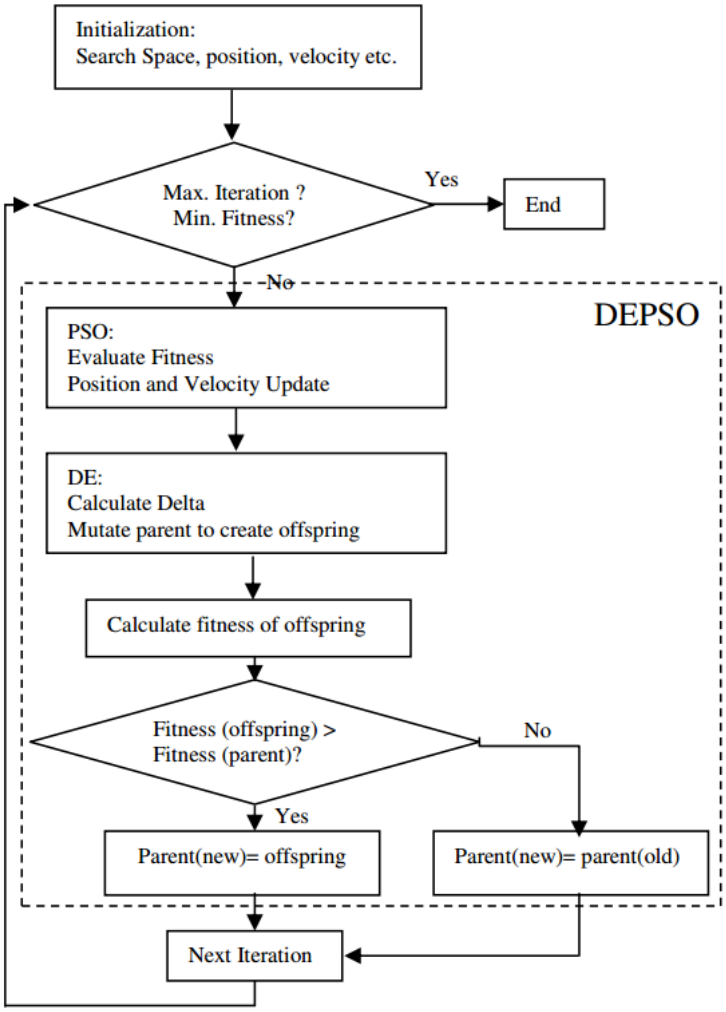

Figura 19. Diagrama de flujo del Algoritmo DEPSO [22].

3.3.4.3. Algoritmo Híbrido Genético, Recocido Simulado y Búsqueda Tabú

El Algoritmo Genético, propuesto por J. Holland [20], es un sistema genético artificial basado en el principio de la selección natural donde los individuos más fuertes y mejor adaptados sobreviven en un entorno de competitividad. Con el desarrollo de ordenadores más potentes y baratos, los GAs se han impuesto como una herramienta madura de búsqueda y optimización. Muchos autores han empleado los GAs para el diseño de filtros digitales [3,7] aprovechando sus propiedades multimodales y de variables codificadas.

Sin embargo, el algoritmo genético posee tres principales desventajas que reducen su aplicación en ciertos campos. En primer lugar, el GA puede ser visualizado como una combinación equilibrada de la exploración de nuevas regiones en el espacio de búsqueda y la explotación de las regiones descubiertas. Este equilibrio está controlado críticamente por la elección correcta de los parámetros de control del algoritmo, como pueden ser el tamaño de población y las probabilidades de mutación y de cruce. Los parámetros óptimos dependen del tipo de problema de optimización al que se enfrente el algoritmo. Normalmente se necesitan múltiples intentos hasta hallar el ajuste de parámetros correcto; además, funciona mejor para las fases iniciales del algoritmo, pero se deteriora a medida que aumenta el número de iteraciones. El segundo problema es la prematura convergencia del algoritmo genético convencional. Sobre todo, cuando el espacio de búsqueda es extenso y el problema tiene múltiples dimensiones, se

obtiene una solución sub-óptima. Tercero, la tasa de convergencia es muy baja para espacios de búsqueda grandes.

Para solventar estos problemas, L. Cen propone en 2007 [44], un algoritmo híbrido denominado GST, que mezcla el algoritmo genético junto con los algoritmos de recocido simulado y de búsqueda tabú. Se denomina de esta manera porque mezcla: genético (G), simulado (S) y tabú (T).

El proceso de optimización del algoritmo GST está gobernado por el Algoritmo Genético Adaptativo (AGA), en el cual, el tamaño de población y las probabilidades de cruce y mutación son ajustadas adaptativamente. De esta manera no solo mejora el rendimiento del algoritmo, sino que evita emplear múltiples intentos para determinar los parámetros mencionados. El algoritmo de recocido simulado (SA) se emplea cuando hay un indicador de que el AGA se ha quedado "atrapado" en un óptimo local. La función del SA es crear variaciones de la población cambiando el orden de los dígitos para encontrar configuraciones óptimas. Si suponemos que los cromosomas están formados por tres tipos de dígitos: 1, 0, -1; el recocido simulado no cambia el número de 1s ó 0s, sino que los ordena de otra manera para encontrar una mejor solución. Algunos de los mejores individuos de la población son elegidos para aplicar el recocido simulado y las nuevas soluciones obtenidas por el SA remplazan a los peores miembros de la generación. Con la ayuda del SA, mejora la calidad de la población añadiendo mejores miembros y eliminando los peores.

Un espacio de búsqueda grande es la principal razón de la lenta convergencia del GA. Esto nos hace considerar si podemos acotar el espacio de búsqueda, en nuestro problema de diseño de filtros FIR, eliminando las soluciones imposibles. Generalmente, los coeficientes localizados en el centro de la respuesta al impulso tienen mayores valores absolutos que los que se encuentran al comienzo o al final. Así, si consideramos que los coeficientes están representados por 10 bits, es imposible que el valor del primer coeficiente h(0) sea cercano a 1023 y los coeficientes centrales menores a 100. De acuerdo con estas propiedades, el concepto de Búsqueda Tabú es introducido en el algoritmo para delimitar el espacio de búsqueda. En cada generación, la búsqueda tabú es aplicada a la población. Si la nueva solución pertenece a la lista Tabú, será aceptada o rechazada según el criterio de aspiración. Este criterio de aspiración permite saltarse la restricción Tabú. En este caso, si el valor de aptitud es mejor que el valor de aptitud de la mejor solución encontrada hasta el momento, se acepta; en caso contrario, se acepta de acuerdo con una probabilidad de aceptación.

En el Algoritmo GST, hay dos niveles de criterio de convergencia. El primero, se emplea para determinar si el AGA se ha quedado estancado en un óptimo local de acuerdo a la mejora de la aptitud. Si se cumple la condición, se aplica el SA. Se aplica el SA cuando no se pueden encontrar mejores soluciones en un número de iteraciones máximo preestablecido. El segundo nivel controla la convergencia de todo el algoritmo si se satisface una de estas condiciones:

A. El valor de aptitud objetivo se ha alcanzado.
B. No se pueden encontrar mejores soluciones para un número máximo de iteraciones prefijado.

Los pasos principales del algoritmo se pueden resumir en los siguientes:

1) Crear una población inicial.
2) Se evalúa el valor de aptitud para cada cromosoma en la población.

3) Se Aplican las operaciones del Algoritmo Genético: reproducción, cruce y mutación para generar descendencia.
4) Se comprueba el criterio del primer nivel de convergencia y se decide si el AGA ha alcanzado un óptimo local. Si satisface la condición, se aplica el SA a un número prefijado de individuos de la descendencia. Los miembros con peor aptitud se sustituyen por los nuevos obtenidos a partir del SA.
5) Se comprueba la lista tabú para cada individuo de esta generación. Si es rechazado por la búsqueda tabú, se aplica el mecanismo de reparación.
6) Se copian los dos mejores cromosomas de la población antigua para remplazar los dos peores de la descendencia. Se actualiza la nueva población con la descendencia.
7) Se comprueba el segundo criterio de convergencia para decidir si el proceso de optimización continúa o se detiene.

La Fig. 20 muestra el diagrama de flujo del algoritmo GST.

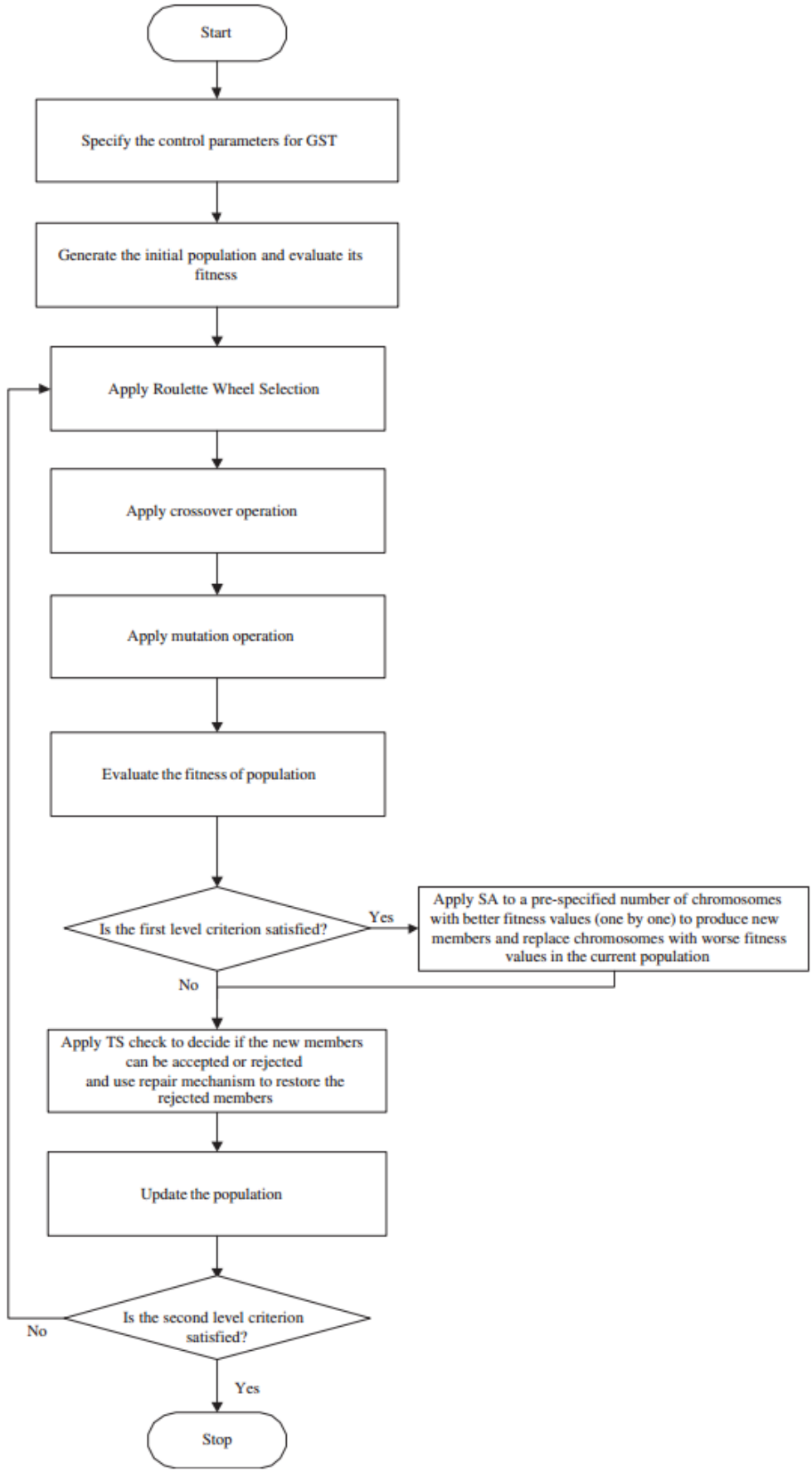

Figura 20. Diagrama de Flujo del Algoritmo GST [44].

4. Estado del Arte

En este capítulo vamos a exponer y analizar los diferentes resultados obtenidos por los distintos métodos de computación natural empleados en la literatura para el diseño de filtros FIR óptimos. Las técnicas evaluadas se pueden clasificar en cuatro grandes grupos: Métodos algorítmicos, métodos bioinspirados "de grupo", métodos basados en redes neuronales y métodos híbridos y otras aproximaciones de computación natural. Se comparan 4 parámetros esenciales de estos filtros obtenidos: Rizado de la banda de paso, atenuación de la banda eliminada, ancho de banda de transición y coste computacional.

4.1. Métodos evolutivos

4.1.1. Algoritmos genéticos (GA)

La implementación de los GAs para la resolución de nuestro problema ya ha sido estudiada por diferentes autores dando unos resultados bastante óptimos en términos de respuesta al impulso y de una carga computacional mínima.

Un primer estudio que emplea los Algoritmos Genéticos para el diseño de filtros FIR, fue propuesto por Suckley en 1991 [7], donde compara la respuesta en frecuencia y la carga computacional que obtiene al diseñar un filtro utilizando AGs, con un estudio realizado por Wade en 1990 [8], donde elabora dicho filtro a partir de un algoritmo de síntesis (método tradicional). Los resultados que obtiene son bastante parecidos en términos de eficiencia y de carga computacional. Si bien es cierto que el algoritmo de Wade al estar exclusivamente diseñado para el propósito de nuestro problema, obtiene resultados ligeramente superiores. Sin embargo, Suckley propone este método evolutivo como una clara alternativa a los métodos tradicionales para el diseño eficiente de filtros digitales tipo FIR.

En su tesis doctoral, Sabbir U. Ahmaden (2008) [3] presenta el diseño de diferentes tipos de filtros digitales utilizando técnicas evolutivas, en concreto, algoritmos genéticos y algunas variantes híbridas con otros métodos. Uno de los usos de los métodos genéticos que muestra el autor es para el diseño de filtros FIR simétricos y lo compara con un método tradicional denominado WLS (Weightened Least Square) [39], una generalización del método de mínimos cuadrados. En este estudio, se comparan tres características importantes de los filtros generados con ambos métodos, el rizado y el retardo de grupo de la banda de paso y la atenuación de la banda de corte.

En concreto, el autor plantea el diseño de dos filtros; un paso bajo de longitud 28 muestras y un paso alto de 35 muestras. En su estudio obtiene dos soluciones empleando algoritmos genéticos y la solución con el método WLS, consiguiendo los siguientes resultados:

	Method	*Solution a*	*Solution b*
Passband edge (rad/s)	0.25		
Stopband edge (rad/s)	0.4		
Filter length, N	28		
Passband ripple (dB)	WLS	0.686	
	ENSGA	0.3025	0.2789
Stopband attenuation (dB)	WLS	44.001	
	ENSGA	43.90	43.823
Passband group-delay (Q)	WLS	1.2002	
	ENSGA	0.3506	0.1298

Tabla 3. Características del filtro paso bajo obtenidas por el GA y por el método WLS [3].

	Method	*Solution a*	*Solution b*
Stopband edge (rad/s)	0.40		
Passband edge (rad/s)	0.55		
Filter length, N	35		
Maximum Passband ripple (dB)	WLS	0.275	
	ENSGA	0.197	0.210
Minimum stopband attenuation (dB)	WLS	52.376	
	ENSGA	53.670	53.000
Passband group-delay (Q)	WLS	0.984	
	ENSGA	0.621	0.524

Tabla 4. Características del filtro del filtro paso alto obtenidas por el GA y por el método WLS [3].

Como se puede apreciar en las Tablas 3 y 4, se logran unos mejores resultados en términos de rizado de la banda de paso y de retardo de grupo mediante el uso de técnicas evolutivas frente al algoritmo WLS. Lo que es lo mismo, con el mismo orden del filtro, se obtiene una mejor función de transferencia empleando métodos genéticos, aunque notamos que la atenuación de la banda de paso es ligeramente superior en el método de WLS (un 0,2 y 0,4%).

Posteriormente, multitud de investigaciones han sido realizadas en este campo, ya que, como he comentado con anterioridad, el problema que inicialmente presentan estos métodos es la elevada carga computacional para alcanzar una solución óptima. Con el fin de reducir este coste computacional, se implementan diversas variantes al algoritmo genético tradicional y surgen los métodos híbridos [5].

4.1.2. Algoritmo de Evolución Diferencial

En la Fig. 19, se puede observar la respuesta al impulso en frecuencia de un filtro FIR paso bajo obtenidas mediante algoritmo genético (GA), algoritmo de evolución diferencial (DE) y el método de mínimos cuadrados y ecuaciones dispersas (LSQ) [24].

Se puede comprobar cuál es el comportamiento de dichos algoritmos en cuanto al rizado de banda de paso y de atenuación de la banda de corte obteniéndose resultados muy similares para los tres algoritmos. Queda demostrado en [24] que se pueden obtener respuestas con menor rizado con el coste de aumentar el ancho de banda de transición del filtro, pero combinando la función de aptitud con algún otro método, se podría reducir la anchura de la banda de transición.

Cabe destacar la carga computacional del DE es menor que la del algoritmo GA, obteniendo una respuesta muy similar y es mucho menor que la del método de mínimos cuadrados (método tradicional). Además, la aplicación de los genes reservados reduce el tiempo de convergencia del algoritmo, sobre todo para órdenes de filtro elevados, como se puede ver en la Fig. 22. Resumiendo, se puede deducir que este algoritmo es una buena alternativa para el diseño de filtros en cuanto a la reducción de la carga computacional si lo comparamos con el método tradicional LSQ y el algoritmo GA, obteneniendo unos resultados satisfactorios.

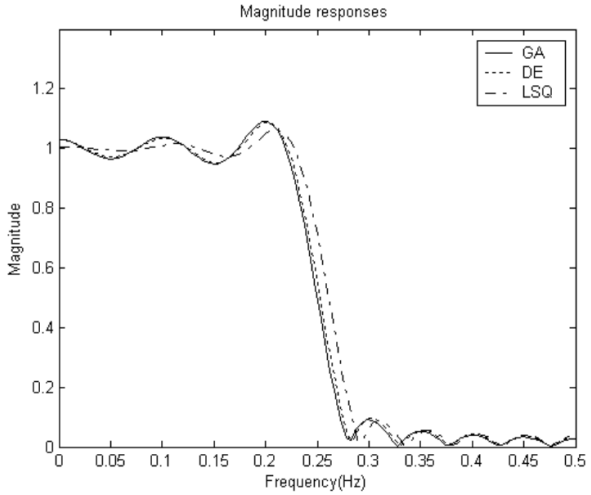

Figura 21. Respuesta en frecuencia de los filtros de orden 20 obtenidos mediante GA, DE y LSQ [24]

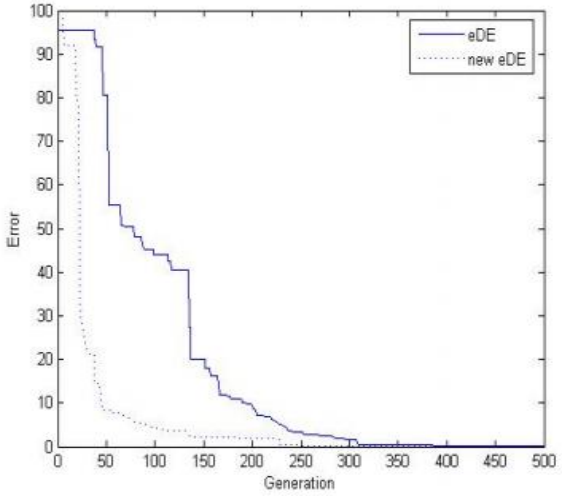

Figura 22. Evolución del error con las diferentes generaciones para un filtro de orden 30 empleando DE (eDE) y DE con genes reservados (new eDE)

4.1.3. Algoritmo de Recocido Simulado (Simulated Annealing)

En un primer estudio, Kacelenga et al. en 1990 [15], proponen el recocido simulado para el diseño de filtros digitales, tanto FIR como IIR. Los resultados obtenidos en [15] se pueden observar en la Tabla 5, teniendo en cuenta el año en que fueron desarrollados e implementados, son bastante buenos en términos de rizado de la banda de paso y de atenuación en la banda de corte. Sin embargo, el tiempo de CPU es muy elevado, lo que implica una elevada carga computacional. Este elevado tiempo computacional es necesario para filtros de un orden grande, que, seguramente, se verá reducido mediante el empleo de ordenadores más potentes

que puedan procesar datos más rápidamente. De 1990 hasta hoy los ordenadores han evolucionado radicalmente y no necesitan tanto tiempo para obtener una solución aceptable.

	CASE I	CASE II	CASE III	CASE IV
Filter Order (N)	11	21	31	41
Passband Edge	0.7854	0.7854	0.7854	0.7854
Stopband Edge	1.9635	1.9635	2.3562	2.3562
CPU(s)	650.4	1158.2	1769.3	1836.8
Passband Ripple	0.043dB	0.043dB	0.017dB	0.017dB
Stopband Atten.	26.00dB	26.00dB	46.00dB	46.00dB

Tabla 5. Resultados obtenidos para el diseño de filtros FIR de orden 11,21,31 y 41 mediante Recocido Simulado [15].

Al igual que ocurre con el resto de algoritmos, a lo largo del tiempo, diversos autores han desarrollado técnicas y variantes al SA tradicional más eficientes que arrojan unos resultados mejores en términos de carga computacional y características del filtro. En concreto, el método desarrollado y empleado por Aguiar e Oliveira et al. en 2007 [16], que utiliza una variante del algoritmo SA con lógica dispersa y adaptativo (*Fuzzy Adaptative Simulated Annealing,* FASA).

Para demostrar el funcionamiento del algoritmo FASA para el diseño de filtros FIR, se realiza el diseño en [16] de un filtro paso banda de orden 32, obteniendo los resultados de la Tabla 6. Se puede apreciar que los resultados en cuanto al rizado máximo de la banda de paso (δ_p) y el error de retardo de grupo son mejores para el algoritmo FASA que el método empleado en la ref. 1, que emplea una aproximación basada en métodos neuronales [45].

Method	Peak magnitude (δ_p)	Group delay error (δ_τ)
This paper	6.089e-02	0.5489
Ref. [1]	6.266e-02	1.080

Tabla 6. Comparación del rizado máximo de la banda de paso y el error de retardo de grupo empleado el algoritmo FASA y redes neuronales [16].

4.1.4. Algoritmo de Búsqueda Tabú

Karaboga et al. en 1997 [51] propusieron un nuevo método de diseño para optimizar los coeficientes de los filtros FIR llamado algoritmo de búsqueda tabú. En esta primera implementación para el diseño de filtros, se consiguen unos resultados bastante buenos en términos de carga computacional y de características del filtro diseñado como se puede ver en la Tabla 7; mejorando las características obtenidas mediante otra técnica de computación natural, el algoritmo genético.

4.1.5. Algoritmo de Búsqueda Tabú Múltiple o Paralelo

Posteriormente, surge una mejora de la búsqueda tabú, llamada algoritmo de búsqueda tabú múltiple o paralelo (MTSA o PTS) desarrollado por Watcharasitthiwat el al. en 2005 [12] y

Karaboga et al. en 2006 [13]. Esta mejora implica la inclusión de nuevos mecanismos de búsqueda como la inicialización, búsqueda adaptativa, búsqueda múltiple, cruce y reinicio. Los resultados obtenidos mediante este algoritmo múltiple mejoran con creces los resultados del TSA convencional, así como los del Algoritmo Genético clásico (GA). En la Tabla 3 se aprecian los diferentes valores obtenidos de atenuación de la banda eliminada, del rizado máximo de la banda de paso, el error medio, así como el tiempo computacional en el diseño de un filtro FIR paso bajo de orden 40. Demostrando así, que el MTSA obtiene una mejor solución que los métodos tradicionales del GA y del TSA en todas las características comparadas. En concreto mejora una característica muy importante que es la carga computacional.

Methods	δ_{pass}	A_{stop} ***(dB)***	***ME***	***Computational Time (sec)***
GA	0.016752	-39.2177	0.00277026	5.3104
TSA	0.016839	-39.9127	0.00266861	3.6522
MTSA	0.013880	-43.7810	0.00200900	2.4271

Tabla 7. Comparativa del rizado de la banda de paso, atenuación de la banda de corte, ME (Error medio) y tiempo computacional en varias técnicas naturalistas [12].

En Tabla 8, se observan las diferencias entre diferentes métodos de diseño de un mismo filtro FIR paso bajo de orden 40, en términos de atenuación de la banda de corte y rizado de la banda de paso. Se obtienen mejores características en todos los campos empleando el algoritmo PTS que el PGA (Algoritmo Genético Paralelo) y el TS convencional.

Methods	Pass-band deviation	Stop-band deviation
Round-off	0.020087	0.01747 (− 35.15 dB)
Integer programming	0.016144	0.01380 (− 37.20 dB)
TS	0.014606	0.01136 (− 38.89 dB)
PGA	0.016470	0.01105 (− 39.13 dB)
PTS	0.014304	0.01011 (− 39.90 dB)

Tabla 8. Comparativa de las características del filtro diseñado con métodos tradicionales y con métodos de computación natural [13].

4.2. Métodos bioinspirados “de grupo”

4.2.1. Algoritmo de colonia de hormigas (ACO)

El empleo del ACO en el diseño de filtros FIR surge como necesidad para encontrar el óptimo global de filtros de orden elevado en un tiempo relativamente pequeño. Para comprobar el desempeño del algoritmo de colonia de hormigas para el diseño de filtros FIR, Trusumi et al. diseñan siete filtros con condiciones y características diferentes en [6]. Dichas características se pueden observar en las Tablas 9 y 10.

	A	B	C	D	E	F	G
N	32	40	48	60	64	80	200
p	8	8	8	8	8	8	9
f_p	0.2	0.2	0.2	0.2	0.2	0.2	0.2
f_s	0.262	0.25	0.24	0.235	0.232	0.225	0.21
S	160	200	240	300	320	400	1000

Tabla 9. Parámetros de los filtros diseñados (A-G) de órdenes diferentes (N) con frecuencias de la banda de paso f_p y de la banda de corte f_s, S es el número de muestras frecuenciales y p la longitud de la palabra [6].

	A	B	C	D	E	F	G
$\Delta\tau$	0.05	0.05	0.05	0.05	0.05	0.05	0.05
τ_{init}	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Number of ants	100	200	200	200	250	300	500
Number of iterations	200	200	200	200	200	250	500
Number of trials	10	10	10	10	10	10	10

Tabla 10. Parámetros del algoritmo ACO empleados, la cantidad de feromona inicial (τ_{init}), la cantidad de feromona que se añade al camino ($\Delta\tau$) y el número de hormigas, iteraciones e intentos empleados [6].

Para comparar con otros métodos de diseño, se han empleado otras técnicas como los algoritmos genéticos (con mismo número de individuos), el método de ramificación y poda para hallar el filtro óptimo, un método de redondeo y otro ACO con distinto método de actualización de feromonas. Los resultados obtenidos en cuanto al error máximo y el tiempo computacional son comparados en las Tablas 11 y 12.

$\times 10^{-2}$	A	B	C	D	E	F	G
δ_{opt}	1.226	1.299	1.364	1.263	1.180	1.385	–
δ_{rou}	1.579	1.535	1.953	1.913	1.836	1.856	1.768
δ_{GA}	1.335	1.501	1.548	1.458	1.306	**1.476**	1.293
$\delta_{ACO\ (conv.)}$	1.472	1.489	1.444	1.375	1.367	1.552	1.553
$\delta_{ACO\ (prop.)}$	**1.235**	**1.398**	**1.442**	**1.339**	**1.180**	1.481	**1.255**
$\delta_{rou}/\delta_{opt}$	1.288	1.182	1.432	1.515	1.556	1.340	–
δ_{GA}/δ_{opt}	1.089	1.156	1.135	1.154	1.107	1.066	–
$\delta_{ACO\ (conv.)}/\delta_{opt}$	1.201	1.146	1.059	1.089	1.158	1.121	–
$\delta_{ACO\ (prop.)}/\delta_{opt}$	**1.007**	**1.076**	**1.057**	**1.060**	**1.000**	1.070	–

Tabla 11. Error máximo obtenido con los distintos métodos de diseño, el filtro óptimo, el de redondeo, el algoritmo genético y los dos algoritmos de colonia de hormigas [6].

Se puede observar que los errores obtenidos por el ACO son menores para casi todos los órdenes de los filtros, sobre todo para los filtros de mayor orden, donde el algoritmo propuesto obtiene un menor error que el resto de métodos comparados. Además, para un orden muy elevado (200) hay algún algoritmo como el óptimo que por coste computacional no es capaz de obtener una solución.

	A	B	C	D	E	F	G
Optimal	13	59	779	7423	6862	535411	–
GA	5	16	19	24	34	68	905
ACO (conv.)	**1**	**3**	**6**	**10**	**13**	**28**	685
ACO (prop.)	2	6	9	14	18	32	**463**

Tabla 12. Tiempo que tarda el algoritmo en converger y obtener la respuesta al impulso óptima del filtro diseñado [6].

En cuanto al tiempo computacional, representado en la tabla anterior, se observa que este método, a pesar de dar una mejor solución que otros métodos en términos del error máximo, el tiempo computacional es bastante menor, sobre todo en el filtro G (orden 200) donde el método de ramificación y poda (óptimo), por ejemplo, no alcanza ni siquiera una solución en un tiempo de 2 semanas.

4.2.2. Algoritmo de nube de partículas (*Particle Swarm Optimization)*

Como ocurre con otros métodos algorítmicos para el diseño de filtros, esta técnica de optimización se ha empleado para el diseño de filtros FIR en numerosas ocasiones. En primer lugar, Ababneh y Bataineh en 2008 [34] presentaron este algoritmo como una alternativa a otros métodos de computación natural para el diseño de filtros FIR paso bajo. Posteriormente, Neha y Singh en Julio de 2014 [23], mejoraron la implementación de este método consiguiendo mejores resultados. En la Tabla 13, se muestra una comparativa del resultado obtenido del empleo de este algoritmo, para el diseño de un filtro FIR paso bajo, comparado con otros métodos de diseño. Se emplea en [23] un coeficiente de inercia optimizado para el diseño de filtros, de la forma:

$$w = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \text{con } \varphi = c_1 + c_2 > 4 \qquad (48)$$

Donde c_1 y c_2 son los coeficientes de restricción cuyo valor típico es 2,05.

Parámetros a comparar (Todos para Filtro Paso Bajo)

Método Empleado	**Orden del filtro (N-1)**	**Atenuación máxima de la banda de corte (dB)**	**Máximo rizado de la banda de paso**	**Máximo Rizado de la banda de corte**	**Ancho de banda de transición**
Algoritmo PSO 1[34]	30	<30	0.15	0.031	0.05
Algoritmo DE [46]	20	*	>0.08	>0.09	>0.06
Algoritmo eDE [24]	20	<28	0.04	0.04(aprox.)	>0.06
Algoritmo DEPSO [22]	20	<27	0.291	0.270	>0.13
Algoritmo PSO 2[47]	20	31.09	0.118	0.0279	0.0904
Algoritmo PSO y LMS [33]	33	<29	*	*	*
Algoritmo PSO 3[23]	20	31.6	0.0390	0.0260	0.0632

Tabla 13. Resultados obtenidos de atenuación de la banda de corte y rizado máximo para los diferentes algoritmos comparados con el algoritmo PSO. (*) No se especifican los valores en los respectivos artículos.

Los resultados obtenidos para el algoritmo PSO mejora en la atenuación de la banda de corte y el máximo de rizado de la banda de paso con respecto a otros algoritmos como el de Evolución Diferencial de Karaboga [46] o el Ababneh que emplea el PSO [34] con otro coeficiente de inercia en combinación con AG, para incluso un orden del filtro más pequeño. Además, como se puede observar en la Tabla 13, el ancho de banda de transición no se ve afectado. El tiempo computacional empleado para computar este algoritmo es mucho menor que el de otros algoritmos expuestos como el PSO y GA o el DE.

4.2.3. Algoritmo de Manada de Felinos (CSO)

Saha et al. realizan en 2013 [25] un estudio de la aplicación del algoritmo CSO para el diseño de 4 tipos de filtros FIR (paso bajo o LP, paso alto o HP, paso banda o BP y elimina banda BS) y lo comparan con otros 4 algoritmos, 3 de ellos evolutivos como son el algoritmo de evolución diferencial DE, el algoritmo genético real RGA y el algoritmo de nube de partículas PSO y un algoritmo tradicional del que ya hemos hablado, el Parks-MacClellan (PM).

En la Tabla 14 se puede observar la atenuación de la banda de corte de los filtros diseñados con los diferentes algoritmos y comprobamos que con el algoritmo de manada de felinos (CSO) se obtiene una mayor atenuación que con el resto de métodos para todos los filtros diseñados.

Filter type	Maximum stop band attenuation (dB)				
	PM	RGA	PSO	DE	CSO
LP	23.56	26.11	28.03	29.53	33.99
HP	23.55	25.25	28.1	29.16	33.62
BP	22.37	30.8	32.03	32.58	34.47
BS	21.65	29.73	30.56	30.96	32.11

Tabla 14.Atenuación máxima de la banda de corte para los distintos algoritmos [25].

A parte de obtener muy buenos resultados en cuanto a atenuación de la banda de corte, en la Tabla 15, se pueden observar distintos parámetros obtenidos en el diseño de un filtro paso bajo de orden 20 con distintas técnicas naturales.

ALGORITMO	RIZADO DE BANDA DE CORTE MÁXIMO	ANCHO DE BANDA DE TRANSICIÓN	TIEMPO DE EJECUCIÓN DE 100 ITERACIONES (S)
PM	0.06636	0.0839	-
RGA	0.04949	0.0853	5.7174
PSO	0.03967	0.0869	3.3286
DE	0.03339	0.0948	3.9543
CSO	0.01998	0.0946	3.0624

Tabla 15. Distintos parámetros del filtro obtenidos para las distintas técnicas y tiempo de ejecución empleado.

Se puede observar también que la carga computacional en segundos es menor para el CSO que para el resto de algoritmos, mientras que la banda de transición, es la segunda mejor del resto de algoritmos. Aun así, es un resultado bastante bueno, consolidándose como una alternativa al resto de algoritmos evolutivos comparados.

4.2.4. Algoritmo de búsqueda del Cuco (CSO)

El método de búsqueda de vuelo del Cuco es un algoritmo desarrollado bastante recientemente. Aggarwal et al. en 2016 [28], realizan una comparación cuantitativa entre varios métodos para diseño de filtros, en concreto compara el CSA con el algoritmo de nube de partículas (PSO), con el Algoritmo Genético GA y con el PM (Técnica de Parks y MacClellan's para diseñar filtros con el mismo rizado en banda de paso y banda de corte). Este último (PM), está basado en técnicas clásicas de optimización y tiene una mayor tendencia a quedarse estancado en óptimos locales,

ya que depende enormemente de la solución inicial. En la Tabla 16 se puede observar la mínima atenuación de la banda de corte y la máxima atenuación de la banda de paso para un filtro elimina-banda de orden 20 diseñado con los 4 métodos.

Algorithm	Minimum stopband attenuation (dB)	Maximum passband attenuation (dB)	Algorithm execution time (s) per 100 cycles
PM	−15.31	1.375	–
RCGA	−17.76	0.910	3.4284
PSO	−16.76	0.798	2.5242
CSA	−28.19	0.320	0.4956

Tabla 16. Atenuación máxima de la banda de paso y mínima de la banda de corte, tiempo computacional para cada uno de los algoritmos empleados [28].

Cabe destacar el algoritmo CSA como el que diseña el filtro con mejor respuesta en cuanto a atenuación de la banda de corte y con menor rizado en la banda de paso. Además, si observamos el coste computacional, vemos que el tiempo que tarda en realizar 100 iteraciones del bucle es notablemente menor que el de otros algoritmos naturales como el GA y el PSO. El método PM no se puede comparar en este sentido ya que no pertenece al grupo de algoritmos evolutivos iterativos.

En la Fig. 23 se muestra la respuesta del filtro en dB para los 4 métodos de diseño, observándose que en el caso del CSA si obtiene una mayor atenuación a causa de aumentar el ancho de banda de transición.

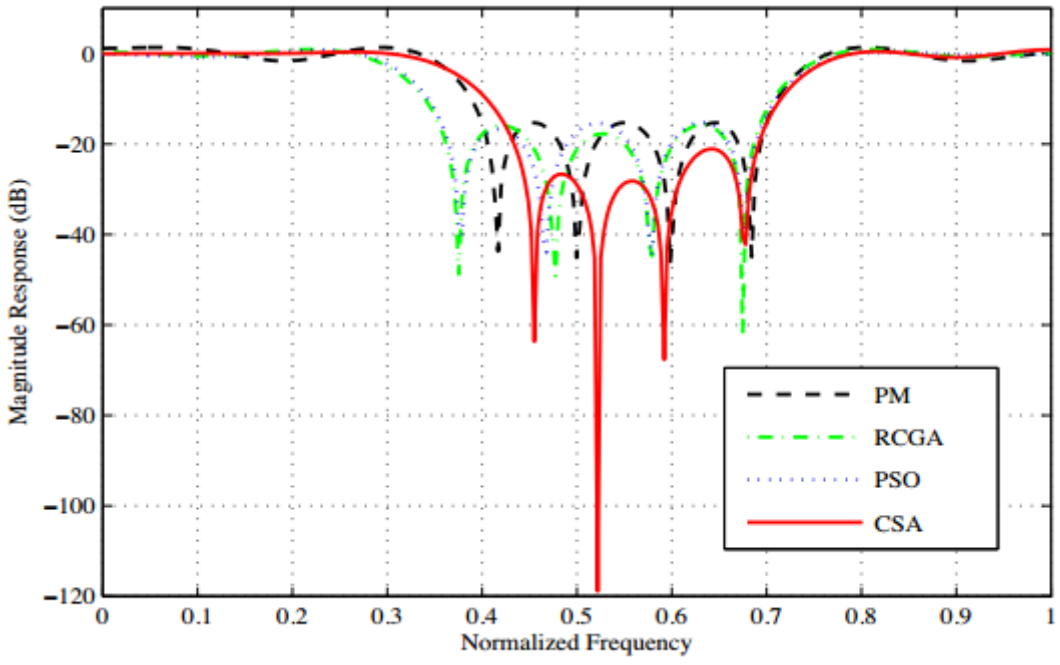

Figura 23. Respuesta en frecuencia del filtro en decibelios para los distintos métodos de diseño [28].

4.2.5. Algoritmo de Búsqueda de Comida de Bacterias

S. K. Saha et. al desarrollaron en 2013 [29], un novedoso algoritmo de búsqueda de comida de bacterias para alcanzar una solución al problema multimodal del diseño de filtros FIR con

coeficientes óptimos. Se obtiene un filtro paso bajo de orden 20, para compararlo con otros métodos de diseño en cuanto a coste computacional, rizado de la banda de corte y ancho de la banda de transición. En la Tabla 17 se pueden ver representados los parámetros conseguidos frente a los obtenidos con el algoritmo tradicional PM, y frente a dos métodos evolutivos, los GAs y el PSO.

Algorithm	*FIR LP filter of order 20*		
	Maximum, average stop band ripple (normalised)	*Transition width (normalised)*	*Execution time for 100 cycles (s)*
PM	0.06636, 0.06599	0.1000	-
RGA	0.04949, 0.0255	0.0965	3.3286
PSO	0.03967, 0.0198	0.0970	3.0416
BFO	0.02773, 0.0177	0.0918	5.7174

Tabla 17. Rizado de la banda de corte, ancho de banda de transición y tiempo computacional (en 100 iteraciones) para los distintos algoritmos de optimización.

Se puede comprobar que los resultados conseguidos por el algoritmo BFO superan a los otros métodos en el rizado de la banda de corte, así como en el ancho de banda de la banda de transición. Pero como podemos apreciar en la última columna de la Tabla 17, el tiempo computacional empleado en 100 iteraciones del algoritmo es mucho mayor que en otros métodos evolutivos como el PSO o el GA, lo que implica un mayor coste computacional. Además, como se puede apreciar en la Fig. 24, la convergencia del BFO es mucho más lenta que en los algoritmos GA y muy parecida al PSO, luego harán falta un mayor número de iteraciones para obtener los coeficientes del filtro óptimo, lo que implica una mayor carga que para otros algoritmos.

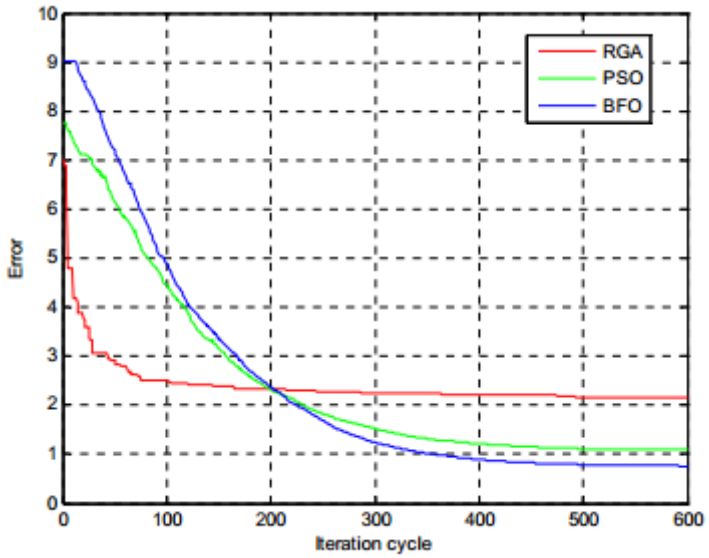

Figura 24. Evolución del error (aptitud) para el algoritmo genético, de nube de partículas y de comida de bacterias en función de las iteraciones empleadas [29].

En la Tabla 18 se realiza una comparación de los métodos naturales más importantes empleados hasta la fecha para el diseño de filtros FIR con el novedoso algoritmo de búsqueda de comida de bacterias [29].

Parámetros a comparar						
Método Empleado	**Tipo de filtro**	**Orden del filtro (N-1)**	**Atenuación mínima de la banda de corte (dB)**	**Máximo rizado de la banda de paso**	**Máximo Rizado de la banda de corte**	**Ancho de banda de transición**
Algoritmo DE [46]	LP	20	*	>0.08	>0.09	>0.06
Algoritmo eDE [24]	LP	20	<28	0.04	0.04(aprox.)	>0.06
Algoritmo PSO y LMS [33]	LP	33	<29	*	*	*
	BP	33	<25	*	*	*
Algoritmo PSO y GA [34]	LP	30	<30	0.15	0.031	0.05
Algoritmo DEPSO [48]	LP	20	<27	>0.1	>0.06	>0.15
	BP	20	<8	>0.2	>0.05	>0.07
Algoritmo DEPSO [22]	LP	20	<27	0.291	0.270	>0.13
Algoritmo FASA [16]	BP	30	<33	*	*	>0.1
Algoritmo PSO 3[23]	LP	20	31.6	0.0390	0.0260	0.0632
Algoritmo BFO [29]	LP	20	31.09	0.129	0.02773	0.0918
	HP	20	33.44	0.129	0.02129	0.0937
	BP	20	34.63	0.142	0.01856	0.1014
	BS	20	33.84	0.161	0.02033	0.1080

Tabla 18. Comparativa de los parámetros de los filtros óptimos diseñados con distintas técnicas naturalistas [Elaboración propia].

Podemos observar que los parámetros de los filtros diseñados son muy buenos en comparación con el resto de métodos empleados, superando a la mayoría, y posicionándose así el BFO como una alternativa a otros métodos de diseño, teniendo en cuenta que su carga computacional es más elevada.

4.3. Métodos basados en redes neuronales (RNAs)

Hui Zhao et al. en 1997 [49], proponen una novedosa técnica basada en redes neuronales para el diseño de filtros FIR óptimos. Los resultados obtenidos, aunque buenos, podían ser mejorados. Posteriormente, K. Pachori et al. desarrollaron en 2012 [43] una red neuronal de una sola capa para el diseño de filtros FIR óptimos. Está basado en el concepto de minimización de la suma de los errores cuadráticos entre la respuesta del filtro deseada y la del filtro diseñada mediante redes neuronales, y mejoraron los resultados obtenidos por Zhao et al. [49]. Esta mejora surge de no emplear la operación de inversión a la matriz. Los rizados en las bandas de paso y de corte son menores como se puede observar en la Tabla 19, donde se compara, para un filtro paso bajo de orden 36, el rizado de la banda de paso y atenuación de la banda de corte.

Type of Filter	Filter Length	$\omega_p(\pi)$	$\omega_s(\pi)$	$\delta_p(dB)$	$\delta_s(dB)$	Method
Low-pass (*Example 1*)	37	0.3	0.4	0.1218	34.11	NNO [9]
				0.0849	36.02	Proposed

Tabla 19. Comparación de características del filtro diseñado por Zhao [49] y el diseñado por Kushboo [43]

De los resultados obtenidos con ambos métodos, destaca la evolución desarrollada por Kushboo en 2012 [43] como una mejora en cuanto a un menor rizado de la banda de paso (δ_p) y una mayor atenuación de la banda de corte (δ_s). Además, al no incluir el cálculo de una matriz inversa, se reduce considerablemente la carga computacional.

En [50], Zhe-Zhao Zeng et al. proponen, por otro lado, en 2006, emplear las redes neuronales para el diseño de filtros digitales óptimos de orden muy elevado. Al igual que ocurre con el método de Kushboo, no emplea la operación de matriz inversa, por esto, se reduce considerablemente la carga computacional y permite el empleo en el diseño de filtros FIR óptimos de orden elevado. Una ventaja a los algoritmos naturalistas evolutivos es que estos últimos emplean una gran carga computacional y necesitarían mucho tiempo para poder obtener una respuesta al impulso óptima, mientras que con las redes neuronales este coste no es tan elevado y es factible su empleo para filtros de orden tan grande. Se puede observar en la Fig. 25 la respuesta en frecuencia del filtro diseñado de orden 2000 que obtiene una atenuación en la banda de corte de 230 decibelios y una frecuencia de corte fácilmente controlable con exactitud.

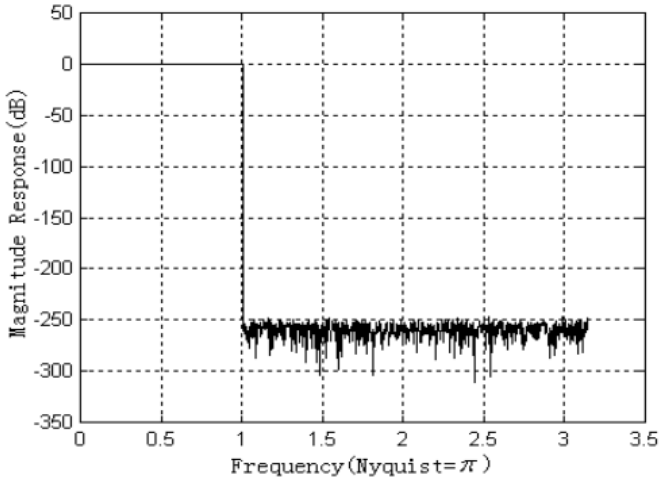

Figura 25. Respuesta en frecuencia del filtro paso bajo óptimo de orden 2000 diseñado con redes neuronales [50].

4.4. Métodos híbridos y otras aproximaciones de Computación Natural

4.4.1. Algoritmo Genético Híbrido (HGA)

Boudjelaba et al. en 2014 [5] presentan una variante del algoritmo genético tradicional que emplea una técnica de aproximación local que reduce la carga computacional y acelera la convergencia del algoritmo. En la Tabla 20, se realiza una comparación de este HGA con otras aproximaciones utilizadas para el diseño de filtros FIR. Se contrastan el error (desviación media), el error de pico (desviación máxima), el ancho de la banda de transición y el tiempo de CPU empleado para el diseño de un filtro FIR paso bajo de orden 40.

Podemos observar que los mejores resultados en cuanto a error y ancho de la banda de transición se obtienen con el algoritmo genético híbrido (HGA), aunque con un tiempo computacional muy elevado. Es cierto también que, comparado con el algoritmo genético

tradicional (SGA), se reduce ampliamente el tiempo de CPU empleado. Los métodos que menor coste computacional tienen son los tradicionales como el de los mínimos cuadrados o el enventanado, aunque el filtro obtenido es peor.

PARÁMETROS A COMPARAR (FILTRO PASO BAJO DE ORDEN 40)

MÉTODO EMPLEADO	Error Medio	Error de pico	Ancho de banda de transición	Tiempo de CPU (s)
ALGORITMO HGA [5]	0.0017	0.0084	0.05	130.54
ALGORITMO GA	0.0048	0.0295	0.064	175.61
ALGORITMO PM [31]	0.0499	0.0573	0.0908	<2
MÉTODO MÍNIMOS CUADRADOS	0.0207	0.0827	0.0904	<2
MUESTREO EN FRECUENCIA	0.0107	0.0954	0.1476	<2
ENVENTANADO	0.0094	0.0779	0.1351	<2

Tabla 20. Características de error medio, error de pico, ancho de banda de transición y tiempo computacional obtenido del diseño de filtros con distintos métodos [Elaboración propia].

Los resultados obtenidos por diferentes aproximaciones y diferentes métodos muestran que los algoritmos evolutivos son más eficientes que las técnicas convencionales. Los algoritmos evolutivos son mejores en términos de error medio, error de pico y banda de transición. El único criterio en el que los métodos tradicionales son mejores a los evolutivos es en la eficiencia de uso de CPU, puesto que los primeros no son iterativos. La inclusión de las aproximaciones locales y los algoritmos implementados para mejorar los GAs, reducen el tiempo de convergencia entre un 5-20% en comparación con los algoritmos genéticos tradicionales, lo que se traduce en un menor coste computacional. Se puede mejorar también la calidad de las soluciones a costa de un aumento de la carga computacional, lo que nos recuerda el problema inicial de los GAs, el coste computacional.

4.4.2. Algoritmo de Evolución diferencial con Nube de Partículas (DEPSO)

Bipul Luitel et al. en 2008 [22] propusieron una combinación del algoritmo de nube de partículas y el algoritmo de evolución diferencial para el diseño de filtros FIR óptimos. Los resultados obtenidos se pueden observar en la Tabla 21, que muestra los resultados obtenidos en cuanto a rizado de banda de paso y rizado de banda de corte de dos filtros paso bajo de orden 20 diseñados con PSO, con DEPSO y con el PSO de [34]. Para su cálculo, se emplean dos funciones de aptitud diferentes; para el caso 1 se utiliza la función dada por la Ec. (14) y para el caso 2 se utiliza el error cuadrático medio entre el filtro ideal y el diseñado. Se emplean 40 iteraciones del algoritmo.

		PSO		DEPSO		Ref [2]
		Case I	Case II	Case I	Case II	
Time	Avg.	3.108	3.077	4.573	3.015	<60
	Min.	3.021	2.954	3.200	2.875	<60
Passband (δ_p)	Avg.	0.169	0.275	0.172	0.269	0.073
	Min.	0.124	0.256	0.152	0.253	0.071
	Max.	0.266	0.290	0.194	0.291	0.075
	Std.	0.041	0.016	0.018	0.016	0.0013
Stopband (δ_s)	Avg.	0.124	0.263	0.203	0.245	0.073
	Min.	0.190	0.246	0.169	0.207	0.071
	Max.	0.262	0.275	0.257	0.270	0.075
	Std.	0.063	0.012	0.041	0.027	0.0013

Tabla 21. Características de rizado de la banda de paso y de la banda de corte del filtro diseñado con los algoritmos PSO y DEPSO [22].

Observamos que la optimización híbrida implica las mejores características de ambos algoritmos y ayuda a reducir el tiempo de diseño. Además, como se puede ver en la respuesta en frecuencia de ambos filtros de la Fig. 26, DEPSO se comporta de una mejor manera obteniendo un menor rizado de la banda eliminada con un tiempo de computación similar al empleado con PSO. Además, si observamos en la figura el ancho de banda de transición del filtro diseñado con ambos métodos, el filtro obtenido con DEPSO obtiene una ligera mejora en cuanto a la reducción de dicho ancho de banda de transición en comparación con el algoritmo PSO tradicional.

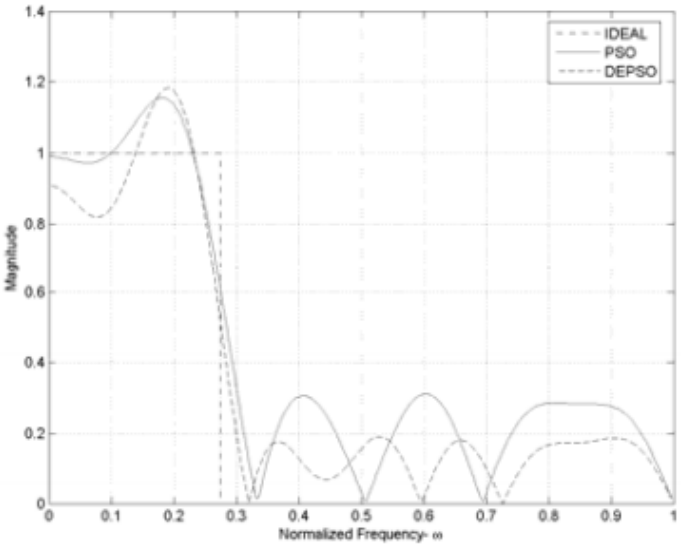

Figura 26.Respuesta en frecuencia del filtro óptimo obtenido con los algoritmos PSO y DEPSO [22].

4.4.3. Algoritmo Híbrido Genético, Recocido Simulado y Búsqueda Tabú

En 2006 [44], Ling Cen propone un algoritmo genético híbrido denominado GST para el diseño de filtros FIR óptimos. Integra las principales características del algoritmo genético adaptativo (AGA), del recocido simulado (SA) y de la búsqueda Tabú (TS). Los resultados de simulación observables en la Tabla 22 muestran que los rizados de pico normalizados pueden ser reducidos con la ayuda del algoritmo híbrido GST. En comparación con el GA tradicional, el GST no solamente mejora la calidad de la solución, sino que reduce considerablemente el coste

computacional. Esto se ve reflejado en el número de evaluaciones medio de la función de aptitud necesario para alcanzar la solución óptima.

Longitud del Filtro (N)	Algoritmo Genético		Algoritmo GST	
	Rizado de pico normalizado (dB)	Número de evaluaciones medio	Rizado de pico normalizado (dB)	Número de evaluaciones medio
31	-39.79	387	-41.17	219.736
33	-39.84	429.5	-41.43	224.745
35	-40.85	441	-42.59	245.529
37	-41.91	478.5	-45.02	274.372
39	-42.03	534.5	-45.27	290.53
41	-43.46	569	-48.12	312.949

Tabla 22. Comparación de los valores medios obtenidos de rizado de pico normalizado y de evaluaciones medias de la función de aptitud para el GA y el GST en 10 ejecuciones del algoritmo en el diseño de un filtro paso bajo [44].

En la Fig. 27 se pueden observar las respuestas en frecuencia de los filtros óptimos diseñados con el algoritmo GST para distintas longitudes del filtro.

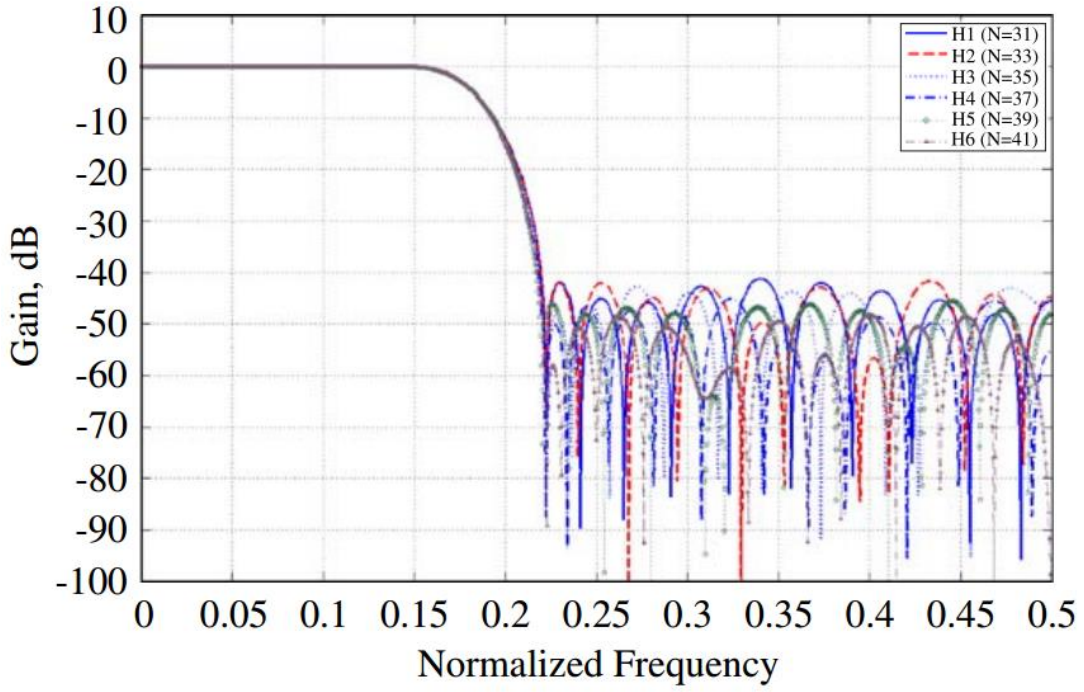

Figura 27. Respuesta en frecuencia de los filtros diseñados con el algoritmo GST para distintas longitudes del filtro (N) [44].

5. Método de diseño propuesto

5.1. Algoritmo de Polinización de Flores

Se estima que existen 250000 tipos diferentes de plantas con flores en la naturaleza y que el 80% de todas las especies de plantas tienen flores. Las plantas con flores han ido evolucionando desde hace más de 125 millones de años y las flores han ocupado un papel fundamental en la evolución. El principal objetivo de las flores es la reproducción mediante la polinización. Este es un proceso en el cual se transfiere el polen de las flores de unas a otras.

Hay dos tipos de polinización: biótica y abiótica. Cerca del 90% de las plantas con flores pertenecen al grupo de bióticas, es decir, el polen es transferido mediante polinizadores como insectos o animales. Sobre el 10% de la polinización no requiere ningún tipo de polinizador, es denominada polinización abiótica. El viento o la difusión mediante el agua llevan a cabo el proceso de polinización.

La polinización se puede clasificar en auto-polinización o polinización local y polinización cruzada (alogamia). En la polinización cruzada, el polen es transferido desde la flor de otra planta; mientras que, en la auto-polinización, el polen es recibido de la misma flor o de flores de la misma planta, normalmente sin la acción de un polinizador.

La polinización cruzada, habitualmente es biótica y el polen recorre largas distancias gracias a la acción de los polinizadores como abejas, insectos, pájaros y murciélagos que se mueven obedeciendo una distribución Lévy [53].

Podemos idealizar las características descritas del proceso de polinización y el comportamiento de los polinizadores con las siguientes reglas:

1. La polinización cruzada biótica es considerada como un proceso de polinización global. Los polinizadores emplean vuelos Lévy para llevar el polen de unas flores a otras.
2. La auto-polinización abiótica se puede interpretar como una polinización local.
3. Debido a la proximidad física y al viento, la polinización local posee ventajas frente a la polinización global. Ambas son controladas mediante una variable P definida en el intervalo [0,1].

Obviamente, en realidad, cada planta puede tener múltiples flores y cada flor millones e incluso billones de gametos de polen. Sin embargo, para simplificarlo, vamos a asumir que cada planta tiene solamente una flor y dicha flor un gameto de polen. No hay necesidad de distinguir entre el polen, una flor y una planta. En futuros estudios se puede extender este método a múltiples gametos y múltiples flores para resolver problemas de optimización. Cada flor, gameto o planta es representada por un vector de soluciones $\mathbf{x}_i$.

$$\mathbf{x}_i^t = [h_i(1), h_i(2), h_i(3), \dots, h_i(\frac{N+1}{2})] \quad (49)$$

Cada vector de soluciones está formado por (N+1) / 2 coeficientes del filtro, aprovechando la simetría de los filtros FIR diseñados y reduciendo la dimensión del problema a poco más de la mitad; el índice *i* indica una flor o polen de la población total de NumFlor (1 ≤ i ≤ *NumFlor*) y el superíndice t, la generación actual. La población estará formada por un total de *NumFlor* posibles soluciones, siendo este un parámetro para ajustar en la resolución de nuestro problema.

$$\begin{cases} \mathbf{x}_1^t = [h_1(1), h_1(2), h_1(3), \dots, h_1(\frac{N+1}{2})] \\ \mathbf{x}_2^t = [h_2(1), h_2(2), h_2(3), \dots, h_2(\frac{N+1}{2})] \\ \vdots \\ \mathbf{x}_{\mathrm{NumFlor}}^t = [h_{\mathrm{NumFlor}}(1), h_{\mathrm{NumFlor}}(2), h_{\mathrm{NumFlor}}(3), \dots, h_{\mathrm{NumFlor}}(\frac{N+1}{2})] \end{cases} \quad (50)$$

De las características y discusión anteriores se puede diseñar un algoritmo basado en las flores denominado algoritmo de polinización de flores (FPA), desarrollado inicialmente por X. S. Yang en [58]. Hay dos pasos principales en este algoritmo, la polinización global y la polinización local.

En el modo de polinización global, la reproducción óptima es garantizada debido a que los insectos pueden viajar grandes distancias; se puede formular con la Ec. (51).

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \gamma \mathbf{L}(\mathbf{g}^* - \mathbf{x}_i^t) \quad (51)$$

Donde $\boldsymbol{x}_i^t$ representa el polen i; en otras palabras, $\boldsymbol{x}_i^t$ es el i-ésimo vector solución en la iteración t; **g*** es la mejor solución encontrada hasta la iteración t, γ representa el factor de escala del *step size* o tamaño de paso y L es la "fuerza" de polinización, que es proporcional al *step size*. Los largos movimientos de los polinizadores (L) son modelados empleando un vuelo Lévy, descrito por Pavlyukevich en 2007 [53]. La función de vuelo Lévy (L) se calcula con la siguiente ecuación:

$$\mathbf{L} = \left(\frac{\mathbf{u} \cdot \left(\frac{\Gamma(1+\beta) \cdot \sin(\pi \cdot \frac{\beta}{2})}{\Gamma\left(\frac{1+\beta}{2}\right) \cdot \beta \cdot 2^{\frac{\beta-1}{2}}} \right)^{\frac{1}{\beta}}}{\mathbf{v}} \right)^{\frac{1}{\beta}} \quad (52)$$

Donde Γ es la función gamma $\left(\Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, dt\right)$; β una constante que establecemos a 1,5 por defecto. **u** y **v** son dos vectores aleatorios con valores comprendidos entre 0 y 1 y de longitud la dimensión del problema.

La auto-polinización, descrita en la regla 2, se puede formular de la siguiente manera:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \epsilon(\mathbf{x}_j^t - \mathbf{x}_k^t) \quad (53)$$

Donde $\mathbf{x}_j^t$ y $\mathbf{x}_k^t$ representan el polen j y k de la generación t; en otras palabras, dos pólenes o vectores de solución elegidos aleatoriamente de la población. Estas flores pertenecen a una misma planta, simulando la polinización local. La variable ϵ es obtenida mediante una distribución uniforme [0,1].

La mayor parte de las actividades de polinización de las flores pueden ocurrir en una escala local y global. En la práctica, es más probable que las flores que se encuentran próximas entre sí se polinicen por gametos de flores locales y no por aquellas que están más alejadas. Se emplea una probabilidad de cambio P_c (Regla 3) o probabilidad de proximidad para alternar entre la polinización global y la local. Para comenzar, podemos establecer el valor de P_c a 0,5; aunque el valor de 0,8 está demostrado que funciona mejor para la mayoría de los casos.

El algoritmo de Polinización de Flores se puede resumir en su conjunto por una serie de pasos:

1) Se inicializa el contador de iteraciones a 0 y se establece una población inicial de n flores o gametos (**x**) con valores aleatorios. Se encuentra la mejor solución **g*** de la población inicial, que será la que menor valor de aptitud tenga. Se define la probabilidad de cambio P con un valor comprendido entre 0 y 1.
2) Se calcula un valor aleatorio comprendido entre [0,1], si dicho valor es menor que la probabilidad P se salta al paso 3, sino, se salta al paso 4.
3) Se calcula el vector de salto **L** que obedece la distribución Lévy a través de la Ec. (52). Posteriormente se aplica la polinización global con la Ec. (51).
4) Se calcula ϵ a partir de una distribución uniforme en el intervalo [0,1]. Aleatoriamente se elige el gameto j y k de la población. Posteriormente se aplica la polinización local a través de la Ec. (53).
5) Se calcula el valor de aptitud para cada flor. Si el valor de *fitness* de cada flor es mejor que el valor de *fitness* anterior, se actualiza en la población. Si no, se mantiene el gameto (solución) de la iteración anterior.
6) Si ya se han recorrido todas las *n* flores de la población y se las ha aplicado la polinización, se elige al gameto con mejor valor de aptitud de todas las flores existentes y se establece como **g***. Continúa el paso 7.
7) Finaliza el algoritmo si se alcanza el número máximo de iteraciones preestablecido (*MaxGeneration*). Si no, se salta al paso 2.

En la Fig. 28 se puede observar el diagrama de flujo del algoritmo de polinización de flores desarrollado anteriormente.

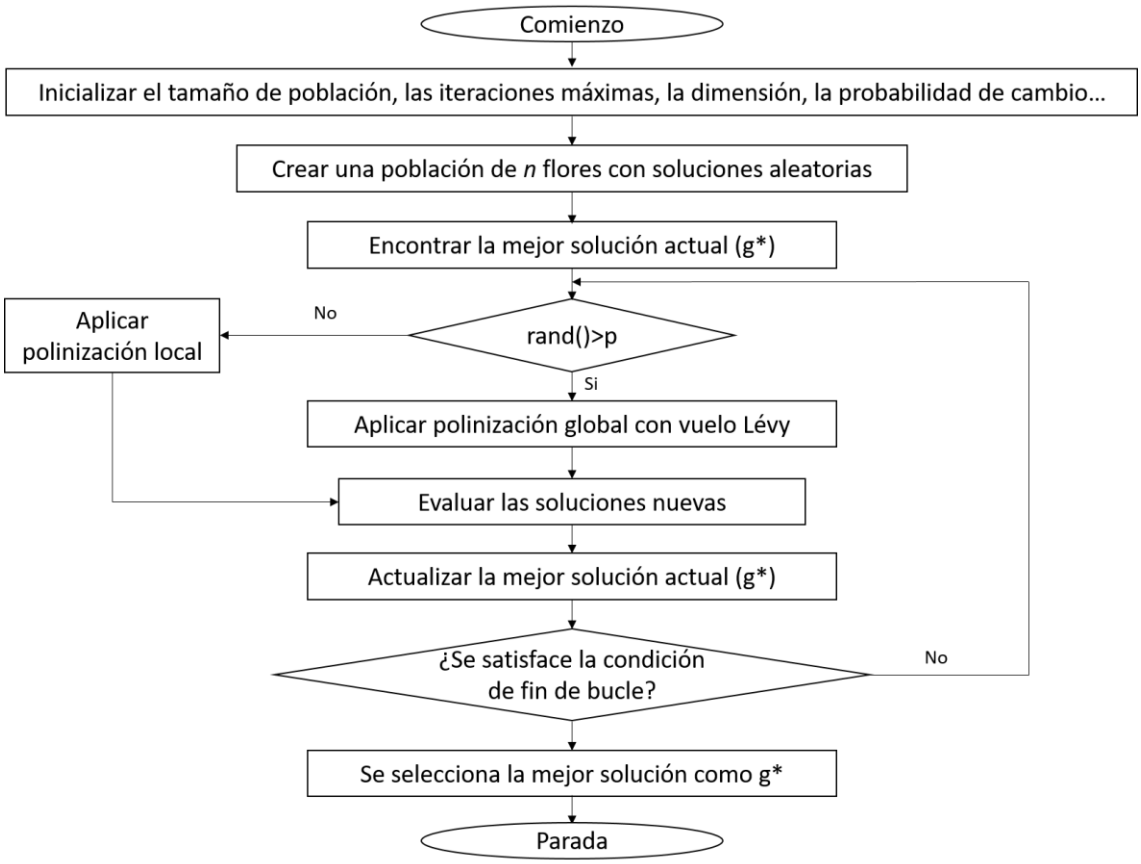

Figura 28. Diagrama de flujo del algoritmo FPA (Algoritmo de Polinización de Flores).

5.2. Función de Aptitud

Como se ha detallado anteriormente, una de las mayores dificultades a la hora de diseñar un filtro óptimo mediante computación natural es la necesidad de emplear una función de aptitud que proporcione buenos resultados. En este caso hemos decidido emplear dos funciones de aptitud diferentes para comparar los resultados obtenidos.

En primer lugar, hemos utilizado la función de aptitud dada por la Ec. (19) introducida en [25] y [29] que permite ajustar la banda de transición correctamente y al calcularse con la suma de todos los errores, se obtiene un menor rizado de la banda de paso y mayor atenuación de la banda eliminada.

En segundo lugar, se ha empleado la función de aptitud definida en la Ec. (18), desarrollada en [34]; dicha función objetivo permite ajustar los rizados de las bandas y el ancho de banda de transición con facilidad.

Como novedad en este documento se ha añadido una tercera función de aptitud o función objetivo realizada como una combinación de las dos anteriores. Es denominada *función de aptitud múltiple.*

5.2.1. Función de Aptitud Múltiple

Existen algunas aproximaciones basadas en computación natural que buscan dar solución a problemas multi-objetivo, es decir, deben minimizar o maximizar varias funciones de aptitud

con el fin de que cumpla múltiples objetivos. Sin embargo, existe una manera muy simple de hacerlo; realizar una suma ponderada combinando múltiples objetivos (aptitudes) en uno solo:

$$f = \sum_{i=1}^{m} w_i \cdot f_i, \qquad \sum_{i=1}^{m} w_i = 1, \quad w_i > 0 \tag{54}$$

Donde *m* es el número de objetivos; en nuestro caso 2 y w_i son la ponderación de cada función de aptitud no negativos. La función de aptitud múltiple desarrollada para nuestro diseño es J_3 (56).

$$J_1 = \max_{w \le w_p} (|\mathbf{E}(w) - \delta_p|) + \max_{w \ge w_s} (|\mathbf{E}(w) - \delta_s|) \qquad (18)$$

$$J_2 = \sum \text{abs}[\text{abs}(|\mathbf{H}_d(w)| - 1) - \delta_p] + \sum \text{abs}(|\mathbf{H}_d(w)| - \delta_s) \qquad (19)$$

$$J_3 = w_1 \cdot J_1 + w_2 \cdot J_2 \tag{55}$$

Los coeficientes w_1 y w_2 los fijaremos más adelante donde se consiga un filtro con mejores características frecuenciales. δ_p es el rizado de la banda de paso y δ_s el rizado de la banda eliminada. La función de error *E(w)* empleada para el cálculo de J_2 es la propuesta por Parks y McClellan en 1972 [31]. La función de ponderación *G(w)* se va a determinar posteriormente para asegurar el diseño de un buen filtro.

$$\mathbf{E}(w) = \mathbf{G}(w) \, [\, \mathbf{H}_d(w) - \mathbf{H}_i(w) \,] \tag{56}$$

$$\mathbf{G}(w) = \begin{cases} \frac{8}{k} & ; \quad 0 \le w \le w_p \\ k & ; \quad w_s \le w \le \pi \end{cases} \tag{57}$$

Siendo $\mathbf{H}_d(w)$ la respuesta en frecuencia del filtro diseñado y $\mathbf{H}_i(w)$ la respuesta en frecuencia del filtro ideal; $k = \delta_p / \delta_s$.

6. Resultados Numéricos

Esta sección muestra los resultados de las simulaciones realizados con la herramienta *Matlab* R2012a y la efectividad del método de diseño propuesto empleando el algoritmo FPA para el diseño óptimo de filtros LPF, HPF, BPF y BSF.

El orden del filtro elegido para los filtros es N-1, siendo N el número de muestras establecido. Para todos los filtros diseñados en esta memoria se han empleado un número impar de muestras y suponiendo la condición de simetría de los coeficientes. Por lo tanto, podemos diseñar cualquier tipo de filtro, ya sea paso alto, paso bajo, elimina banda y paso banda sin restricciones. Además, la dimensión del problema se reduce a la mitad debido a esta simetría de los Filtros FIR de fase lineal. Se han elegido un total de 21 muestras para la mayor parte de los filtros diseñados debido a que la mayoría de los artículos analizados realizan el estudio con este número de muestras y será más sencillo compararlos. El algoritmo es aplicado un total de 20 veces para extraer el mejor resultado.

En primer lugar, se van a seleccionar los parámetros del algoritmo. Es necesario comprobar su desempeño con distintos valores y elegir los que obtengan una mejor aptitud de la solución con un tiempo de convergencia no muy elevado. Se analiza, cambiando los distintos parámetros, la evolución de la aptitud con el tiempo y el valor de aptitud alcanzado para establecer qué parámetros del algoritmo son los ideales para el diseño de filtros FIR. En primer lugar, probamos con los valores recomendados en [2], que es una probabilidad de cambio = 0.5, β para el cálculo del vuelo Lévy = 1.5 y el salto (γ) = 0.01, y modificamos el número de flores de la población y el total de iteraciones del bucle (MaxGeneration).

En la Tabla 23 y 24 se muestra la aptitud media y desviación típica conseguida en 20 implementaciones del algoritmo FPA para distintos valores de los parámetros *NúmeroFlores* y *MaxGeneration*.

	MaxGeneration = 500	**MaxGeneration = 1000**	**MaxGeneration = 1500**	**MaxGeneration = 2000**
NUMFLOR = 10	1.0238 ± 0.4498	0.8144 ± 0.3767	0.7037 ± 0.2350	0.6445 ± 0.1383
NUMFLOR = 15	0.5502 ± 0.1663	0.4531 ± 0.1280	0.4415 ± 0.1232	0.4386 ± 0.1228
NUMFLOR = 20	0.3775 ± 0.1335	0.2799 ± 0.1512	0.2741 ± 0.1498	0.2720 ± 0.1483
NUMFLOR = 25	0.2763 ± 0.0960	0.1721 ± 0.0626	0.1688 ± 0.0630	0.1677 ± 0.0630
NUMFLOR = 30	0.2987 ± 0.0613	0.1621 ± 0.0040	0.1550 ± 0.0016	0.1541 ± 0.0013
NUMFLOR = 40	0.3212 ± 0.0586	0.1645 ± 0.0033	0.1563 ± 0.0019	0.1547 ± 0.0012
NUMFLOR = 50	0.3399 ± 0.0711	0.1672 ± 0.0039	0.1579 ± 0.0013	0.1550 ± 0.0008
NUMFLOR = 80	0.4029 ± 0.0616	0.1729 ± 0.0041	0.1597 ± 0.0019	0.1565 ± 0.0010
NUMFLOR = 100	0.3780 ± 0.0667	0.1744 ± 0.0062	0.1606 ± 0.0018	0.1567 ± 0.0008
NUMFLOR = 120	0.3998 ± 0.0484	0.1750 ± 0.0049	0.1611 ± 0.0016	0.1564 ± 0.0012
NUMFLOR = 180	0.3909 ± 0.0558	0.1760 ± 0.0043	0.1616 ± 0.0013	0.1571 ± 0.0008
NUMFLOR = 250	0.4162 ± 0.0449	0.1779 ± 0.0044	0.1623 ± 0.0013	0.1575 ± 0.0009

Tabla 23. Aptitud obtenida por el algoritmo FPA para distintos valores de tamaño de población (NumFlor) y de generaciones máximas (MaxGeneration). Se muestra el valor medio ± desviación estándar de la aptitud en 20 implementaciones del algoritmo.

	MaxGeneration = 2500	MaxGeneration = 3000	MaxGeneration = 3500	MaxGeneration = 4000
NUMFLOR = 10	0.6229 ± 0.1369	0.6116 ± 0.1388	0.6013 ± 0.1366	0.5967 ± 0.1393
NUMFLOR = 15	0.4278 ± 0.1191	0.4261 ± 0.1184	0.4246 ± 0.1180	0.4235 ± 0.1177
NUMFLOR = 20	0.2715 ± 0.1480	0.2713 ± 0.1479	0.2712 ± 0.1478	0.2712 ± 0.1478
NUMFLOR = 25	0.1673 ± 0.0631	0.1671 ± 0.0631	0.1671 ± 0.0632	0.1671 ± 0.0631
NUMFLOR = 30	0.1533 ± 0.0010	0.1530 ± 0.0009	0.1528 ± 0.0008	0.1527 ± 0.0007
NUMFLOR = 40	0.1536 ± 0.0010	0.1532 ± 0.0010	0.1529 ± 0.0009	0.1527 ± 0.0007
NUMFLOR = 50	0.1539 ± 0.0007	0.1534 ± 0.0007	0.1531 ± 0.0007	0.1529 ± 0.0006
NUMFLOR = 80	0.1549 ± 0.0008	0.1539 ± 0.0005	0.1534 ± 0.0004	0.1531 ± 0.0003
NUMFLOR = 100	0.1552 ± 0.0008	0.1543 ± 0.0006	0.1538 ± 0.0004	0.1534 ± 0.0003
NUMFLOR = 120	0.1551 ± 0.0008	0.1543 ± 0.0006	0.1538 ± 0.0005	0.1534 ± 0.0003
NUMFLOR = 180	0.1552 ± 0.0008	0.1543 ± 0.0004	0.1537 ± 0.0003	0.1534 ± 0.0003
NUMFLOR = 250	0.1554 ± 0.0007	0.1545 ± 0.0005	0.1539 ± 0.0003	0.1535 ± 0.0002

Tabla 24. Aptitud obtenida por el algoritmo FPA para distintos valores de tamaño de población (NumFlor) y de generaciones máximas (MaxGeneration). Se muestra el valor medio ± desviación estándar de la aptitud en 20 implementaciones del algoritmo.

Si seleccionamos una población muy pequeña de 10 flores, la diversidad será muy pequeña y, aunque el algoritmo converja como se observa en las Fig. 29 y 30, el valor de aptitud medio alcanzado para 4000 iteraciones (0.5967) es muy elevado; es decir, posee un error muy grande. Esto es porque el algoritmo converge hacia un mínimo local debido a la poca población con la que se puede mezclar cada solución (falta de diversidad) y al escaso número de soluciones iniciales.

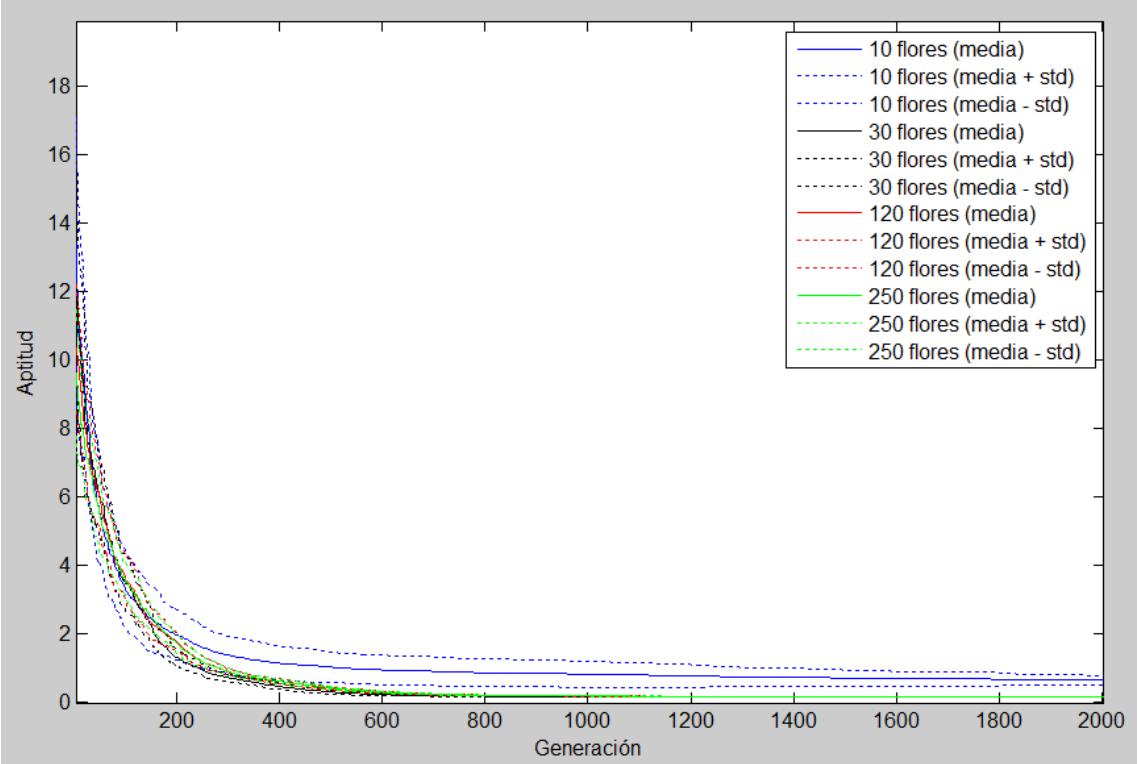

Figura 29. Evolución de la aptitud media y desviación estándar a medida que avanzan las iteraciones (generación) para una población de 10, 30, 120 y 250 flores.

Se eleva la población hasta observar que los mejores valores de aptitud se obtienen a partir de 25. En concreto, los mejores valores de aptitud se obtienen para una población de 30 flores. Por lo tanto, se establece un total de 30 flores para asegurar la convergencia y no estancamiento en un mínimo local. En la Fig. 29 y 30 se puede observar que el algoritmo con una población (30 flores) converge más rápidamente y el valor de aptitud conseguido es mucho menor, incluso menor que en las poblaciones con mayor número de flores.

En cuanto a las iteraciones necesarias, se puede ver en la Fig. 30 que, para una población de 30 flores, a partir de 1500 iteraciones el algoritmo no mejora sustancialmente; solamente se produce una mejora de 0.02 en la aptitud media en 2500 iteraciones. Por esto se elige una población de 30 flores y un total de 1500 generaciones o realizaciones del bucle para reducir el coste computacional, que se incrementa considerablemente cuanto mayor es el número de flores y el máximo de generaciones.

Si aumentamos el número de flores a más de 30, la diversidad de la población será mucho más elevada, y el espacio de búsqueda será cubierto con mayor facilidad. Sin embargo, si queremos obtener una aptitud similar a la mínima alcanzada con 30 flores o gametos, es decir, que el algoritmo converja hacia un óptimo global, tendremos que aumentar el número de iteraciones del algoritmo (más de 3000 iteraciones para 50 flores y más de 4000 para 120 flores); por tanto, aumentará considerablemente el coste computacional. Se busca un equilibrio entre la carga computacional y la calidad de la solución obtenida.

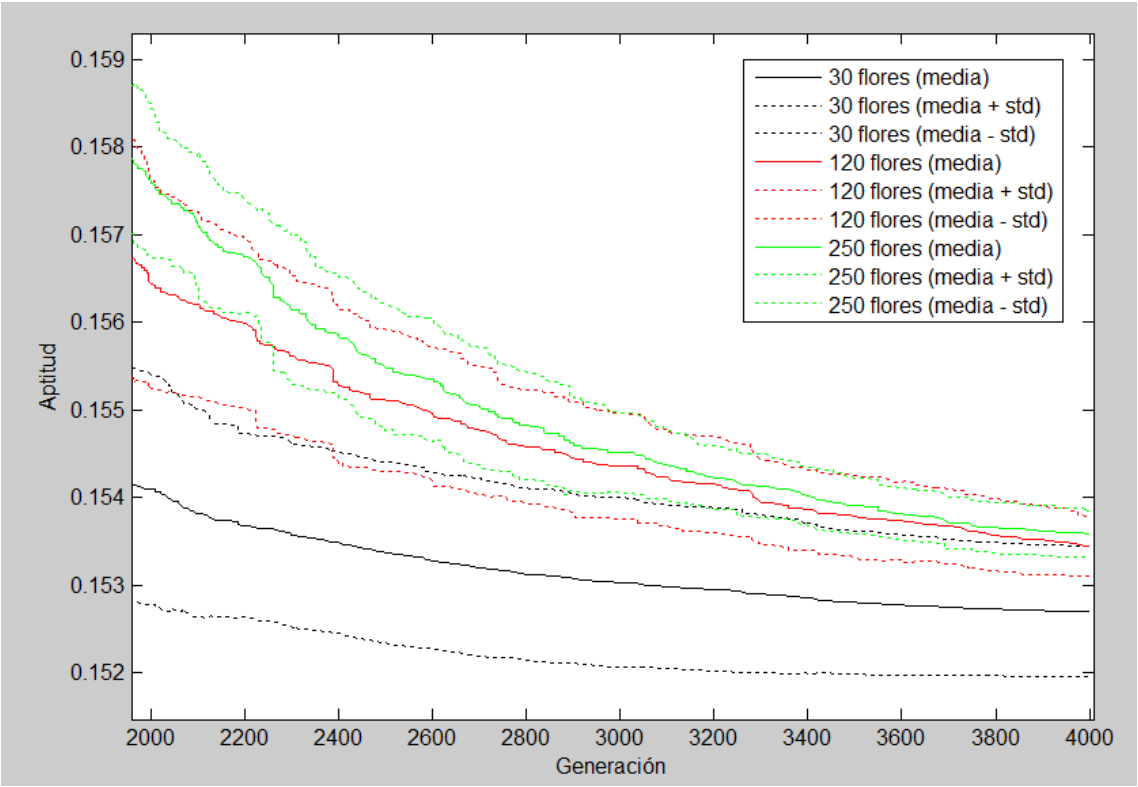

Figura 30. Evolución de la aptitud media y desviación estándar a medida que avanzan las iteraciones (generación) para una población de 30, 120 y 250 flores.

En segundo lugar, una vez establecido el tamaño de población y el máximo de generaciones, procedemos a realizar diferentes simulaciones con el fin de determinar los parámetros óptimos de probabilidad de cambio, β de la función Lévy y salto (γ). Los tres parámetros influyen en la polinización global del algoritmo. La polinización global se emplea para aumentar la diversidad de las soluciones y evitar el estacionamiento del algoritmo, por lo cual, es necesario establecer

un valor de salto grande con la finalidad de variar las soluciones lo suficiente para evitar ese estancamiento. La β elegida es la que se propone en [53] y vamos a variar el salto, comenzando por el propuesto en [2] de 0.01, y la probabilidad de cambio, comenzando en 0.5. La Tabla 25 muestra los valores de aptitud medios y desviación típica, empleando la función de aptitud (56), conseguidos en 20 implementaciones del algoritmo FPA para distintos valores de salto y de probabilidad de cambio.

	p_{salto} = 0.3	p_{salto} = 0.4	p_{salto} = 0.5	p_{salto} = 0.6	p_{salto} = 0.7	p_{salto} = 0.8
γ = 1	0.171 ± 0.071	0.200 ± 0.108	0.157 ± 0.003	0.162 ± 0.005	0.224 ± 0.141	0.272 ± 0.102
γ = 0.5	0.165 ± 0.062	0.170 ± 0.069	0.161 ± 0.057	0.159 ± 0.041	0.163 ± 0.025	0.188 ± 0.078
γ = 0.1	0.168 ± 0.001	0.167 ± 0.063	0.153 ± 0.001	0.154 ± 0.002	0.158 ± 0.003	0.169 ± 0.034
γ = 0.05	0.164 ± 0.001	0.155 ± 0.002	0.156 ± 0.002	0.156 ± 0.002	0.159 ± 0.003	0.174 ± 0.061
γ = 0.01	0.183 ± 0.089	0.168 ± 0.062	0.155 ± 0.001	0.160 ± 0.002	0.163 ± 0.005	0.192 ± 0.079
γ = 0.001	0.199 ± 0.106	0.184 ± 0.087	0.157 ± 0.003	0.173 ± 0.078	0.217 ± 0.089	0.279 ± 0.095

Tabla 25. Valores de aptitud obtenidos para 1500 iteraciones del algoritmo FPA y distintos parámetros. Valor de aptitud (medio ± s.d.) en 1500 iteraciones

Los mejores resultados de aptitud se obtienen con un salto relativamente grande, de 0.1 y 0.05; con una tasa de cambio (probabilidad de polinización global) de 0.4 y 0.5. De esto se puede deducir que un salto grande produce grandes cambios en las soluciones, evitando alcanzar mínimos locales, pero disminuyendo la convergencia; sin embargo, la probabilidad de salto de 0.4 y 0.5 proporcionan una mayor convergencia que otras probabilidades de salto debido a que ocurre una mayor polinización local. Por tanto, hemos seleccionado un salto de 0.1 y una probabilidad de salto de 0.5; ya que producen el equilibrio entre velocidad de convergencia y estancamiento en mínimos locales. En la Fig. 31 se puede observar la rápida convergencia del algoritmo con estos valores y el mínimo valor de aptitud conseguido. Por un lado, se aprecia que la convergencia es mucho menor para probabilidades de cambio mayores. Y, por otro lado, se observa que, con una probabilidad de salto pequeña, lo que mayormente ocurre es la polinización local y la rápida convergencia del algoritmo, pero hacia un mínimo local.

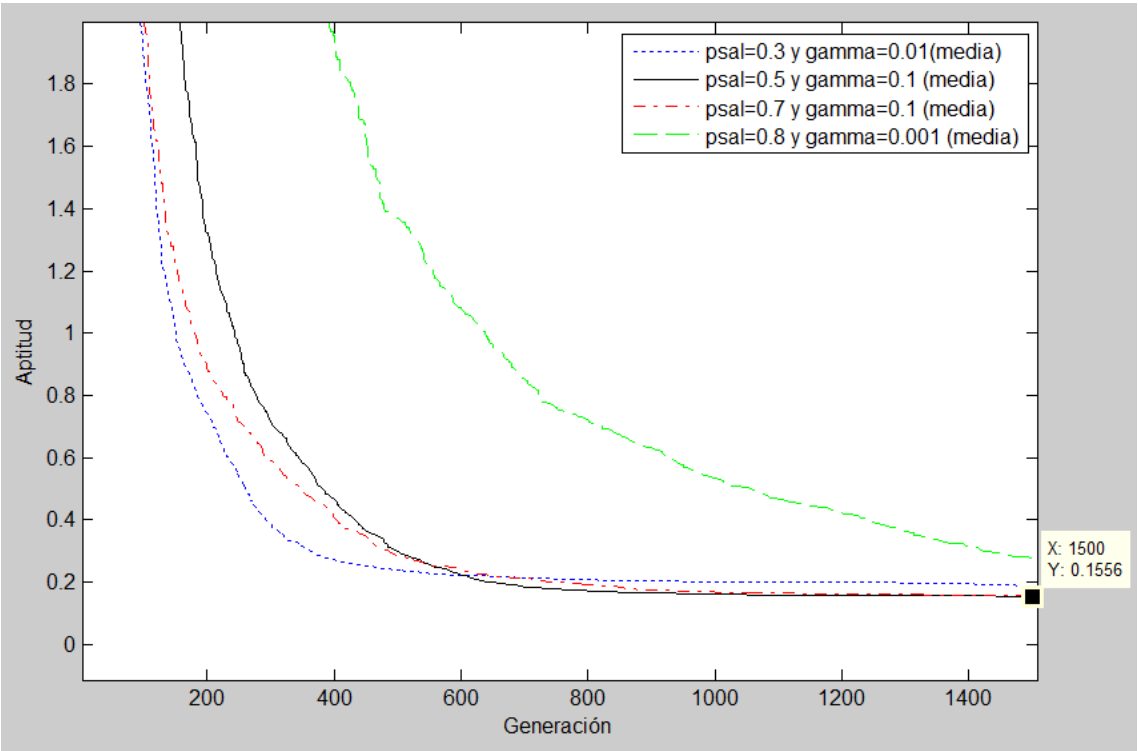

Figura 31. Evolución de la aptitud múltiple (54) para distintos valores de salto γ y distinta probabilidad de salto p_{sal} y β=1.5 empleando el algoritmo FPA.

Después de las distintas simulaciones, los valores seleccionados para los distintos parámetros del algoritmo FPA son: Número de flores (población) = 30, número de iteraciones = 1500, probabilidad de cambio = 0.5, β utilizada para el cálculo del vuelo Lévy = 1.5 y el salto (γ) = 0.1.

Para determinar con qué función de aptitud se obtiene una mejor respuesta al impulso se van a analizar las características de los filtros paso bajo obtenidos con el algoritmo FPA empleando distintas funciones de aptitud. En la Tabla 26 se pueden observar los parámetros obtenidos de rizado máximo de la banda de paso, atenuación máxima de la banda eliminada y ancho de banda de transición empleando el algoritmo FPA para las funciones de aptitud diferentes. Los resultados son los valores máximo, medio y desviación típica de los parámetros calculados tras 20 implementaciones del algoritmo.

Los parámetros del filtro paso bajo deseado que hemos establecido para el diseño son: Frecuencia de muestreo (Fs)=2Hz, número de muestras frecuenciales (L) =512, orden del filtro (N-1) = 20, rizado de la banda de paso (δ_p) = 0.1, rizado de la banda eliminada (δ_s) = 0.01, frecuencia de corte de la banda de paso (w_p) = 0.45, frecuencia de corte de la banda eliminada (w_s) = 0.55 y el ancho de banda de transición = 0.1.

PARÁMETROS OBTENIDOS CON FPA DE FILTRO PASO BAJO DE ORDEN 20

	Coeficientes w_1 y w_2	**Rizado máximo de la banda de paso normalizado**		**Aten. máxima de la banda eliminada (dB)**		**Ancho de banda de transición**	
		Máximo	*Medio ± s.d.*	*Máximo*	*Medio ± s.d.*	*Máximo*	*Medio ± s.d.*
FUNCIÓN DE APTITUD 1 (18)	-	0.265	0.249 ± 0.006	39.68	38.80 ± 0.39	0.102	0.098 ± 0.003
FUNCIÓN DE APTITUD 2 (19)	-	0.142	0.131 ± 0.004	33.34	31.93 ± 0.69	0.102	0.097 ± 0.002
FUNCIÓN DE APTITUD MÚLTIPLE (56)	$w_1 = 0.1$ $w_2 = 0.9$	0.144	0.141 ± 0.005	36.59	36.40 ± 0.36	0.098	0.096 ± 0.002
	$w_1 = 0.2$ $w_2 = 0.8$	0.160	0.148 ± 0.010	36.70	36.56 ± 0.15	0.098	0.096 ± 0.002
	$w_1 = 0.3$ $w_2 = 0.7$	0.169	0.160 ± 0.006	36.85	36.74 ± 0.06	0.098	0.096 ± 0.002
	$w_1 = 0.4$ $w_2 = 0.6$	0.191	0.171 ± 0.013	37.34	36.77 ± 0.29	0.098	0.096 ± 0.002
	$w_1 = 0.5$ $w_2 = 0.5$	0.244	0.205± 0.027	38.14	37.27 ±0.47	0.098	0.096 ± 0.002
	$w_1 = 0.6$ $w_2 = 0.4$	0.247	0.237 ± 0.013	38.41	37.99 ± 0.33	0.098	0.096 ± 0.002
	$w_1 = 0.7$ $w_2 = 0.3$	0.252	0.244 ± 0.004	38.66	38.26 ± 0.28	0.102	0.097 ± 0.002
	$w_1 = 0.8$ $w_2 = 0.2$	0.247	0.245 ± 0.002	38.80	38.60 ± 0.12	0.098	0.097 ± 0.002
	$w_1 = 0.9$ $w_2 = 0.1$	0.250	0.247 ± 0.002	39.09	38.77 ±0.18	0.098	0.096 ± 0.002

Tabla 26. Valores máximos, medios y varianza del rizado de la banda de paso, atenuación de la banda de corte y ancho de banda de transición obtenidos tras 20 realizaciones del algoritmo FPA para las tres funciones de aptitud.

Se puede observar en la Tabla 26 que la función de aptitud 1 por sí sola presenta unos resultados muy buenos en cuanto a la atenuación de la banda de corte que es muy elevada. Sin embargo, el ancho de banda de transición en algunos casos supera el 0.1 establecido y presenta un rizado muy elevado en la banda de paso; llegando en algunos casos a tener una atenuación de la banda de paso de 2,68 dB. En las implementaciones en que dicha atenuación de la banda de paso no sea muy crítica y necesitemos una mayor atenuación de la banda eliminada esta función de aptitud funcionará correctamente.

Para la función de aptitud 2, el ancho de banda de transición también supera en algunos casos el 0.1. Si se tiene en cuenta la atenuación de la banda eliminada, la función de aptitud 2 presenta una menor atenuación de la banda de corte en comparación con la función de aptitud 1. El rizado de la banda de paso es mucho menor que con cualquier otra función de aptitud empleada, lo cual es una ventaja para no atenuar mucho la señal en la banda de paso.

Por último, la nueva función de aptitud múltiple propuesta que combina las dos anteriores, obtiene las ventajas del menor rizado de la función de aptitud 2 y la gran atenuación de la banda eliminada de la función de aptitud 1. Se puede ajustar según los pesos w_1 y w_2 empleados. Hemos elegido los pesos $w_1 = 0.3$ y $w_2 = 0.7$ que producen un equilibrio entre estas dos características. Se reduce el rizado de la banda de paso comparándola con la función de aptitud 2 sin reducirse mucho la atenuación de la banda eliminada. Esta función de aptitud es la más equilibrada en cuanto a los tres parámetros frecuenciales analizados.

La función de aptitud que vamos a emplear a partir de ahora para el cálculo de los coeficientes óptimos de los filtros es la dada por la Ec. (55) con los coeficientes $w_1 = 0.3$ y $w_2 = 0.7$ debido a sus ventajas frente a las otras dos.

La comparación se va a realizar con otros métodos algorítmicos como el BFO y el RGA y se realizará la simulación de otros métodos tradicionales como el Parks and McClellan [31] y el método de enventanado para ver en qué características frecuenciales se produce una mejora.

6.1. Filtro Paso Bajo

Los parámetros del filtro paso bajo deseado que hemos establecido para el diseño son: Frecuencia de muestreo (Fs)=2Hz, número de muestras frecuenciales (L) =512, orden del filtro (N-1) = 20, rizado de la banda de paso (δ_p) = 0.1, rizado de la banda eliminada (δ_s) = 0.01, frecuencia de corte normalizada de la banda de paso (w_p) = 0.45, frecuencia de corte normalizada de la banda eliminada (w_s) = 0.55 y el ancho de banda de transición = 0.1.

Se va a realizar la obtención de los coeficientes óptimos del filtro paso bajo con distintas técnicas de diseño para determinar con qué método se obtienen unas mejores características frecuenciales del filtro diseñado. Los métodos que se van a simular son dos métodos tradicionales: el método de enventanado y el método de Parks y McClellan [31]; y el algoritmo FPA desarrollado en este trabajo. En la Fig. 32 se pueden comparar las respuestas en frecuencia de los tres filtros diseñados y del filtro ideal. Cabe destacar que el filtro que posee un menor ancho de banda de transición es el diseñado mediante el algoritmo FPA destacando claramente sobre los otros dos métodos, a costa de aumentar ligeramente el rizado de la banda de paso. Además, podemos observar que las respuestas en frecuencia de los tres filtros pasan por el mismo punto; que corresponde a la pulsación w=0.45π, donde la amplitud del filtro cae a 0.7071.

Para poder determinar exactamente la atenuación mínima de la banda eliminada y el ancho de banda de transición, se calcula la respuesta en frecuencia en decibelios de cada filtro. Se puede observar, en la Fig. 33, como el filtro diseñado con el algoritmo FPA presenta una mayor atenuación que el filtro diseñado con el algoritmo PM y que el diseñado con el método de enventanado. Además, si nos centramos en la fase, podemos comprobar que tanto el filtro diseñado con el algoritmo FPA como los obtenidos mediante técnicas tradicionales presentan una fase lineal en la banda de paso, cumpliendo la característica de linealidad requerida.

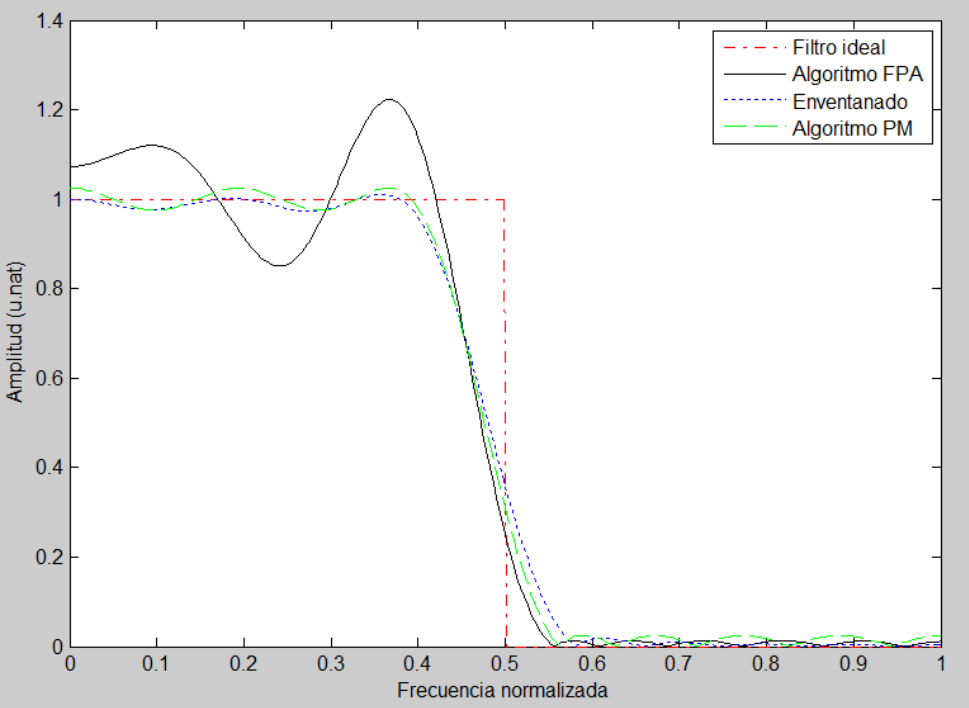

Figura 32.Respuesta en frecuencia de los filtros obtenidos con el algoritmo PM, el algoritmo FPA y el método de enventanado (v. Káiser con β=2.48) en comparación con el filtro ideal.

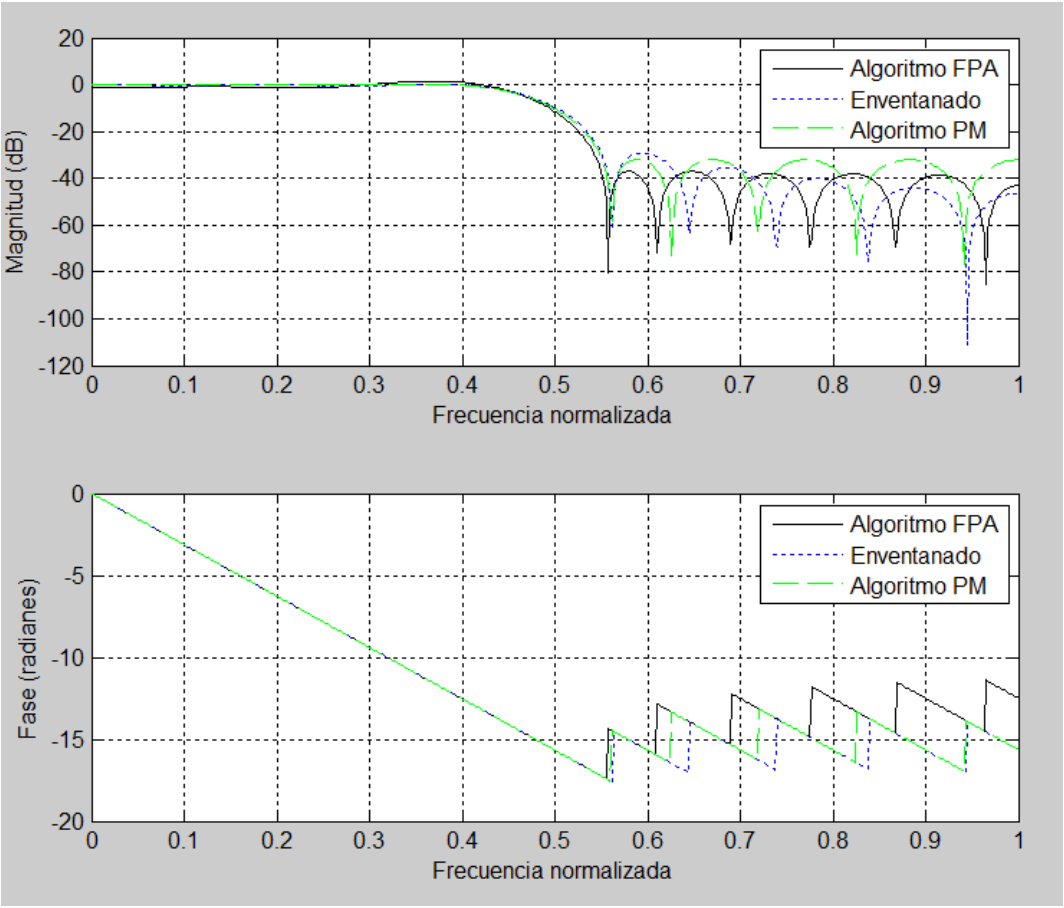

Figura 33. Respuesta en frecuencia y fase de los filtros paso bajo de orden 20 obtenidos con el algoritmo PM, el algoritmo FPA y el método de enventanado (v. Káiser con β=2.48).

Los resultados obtenidos en cuanto a ancho de banda de transición, rizado máximo de la banda eliminada, rizado y atenuación máxima de la banda de paso, se pueden observar en la Tabla 27. Los valores son calculados tras 20 implementaciones de los algoritmos FPA y PM; y con los filtros calculados con el método de enventanado (ventana Káiser, β=2.48, β=1.7).

Método Empleado	*Máximo rizado de la banda de paso (0-0.45π)*		*Máximo Rizado de la banda de corte (0.55π-π)*		*Atenuación mínima de la banda de corte (dB)*		*Ancho de banda de transición*	
	Máx	**Medio ± s.d.**	**Máx**	**Medio ± s.d.**	**Máx**	**Medio ± s.d.**	**Máx**	**Medio ± s.d.**
Algoritmo PM [31]	0.025	0.025±0	0.025	0.025±0	32.03	32.03± 0	0.102	0.102 ± 0
Enventanado con v. Káiser y β=2.48	0.031	0.031±0	0.018	0.018±0	34.68	34.68± 0	0.121	0.121 ± 0
Enventanado con v. Káiser y β=1.7	0.052	0.052±0	0.034	0.034±0	29.38	29.38± 0	0.098	0.098 ± 0
Algoritmo FPA	0.174	0.158±0.01	0.016	0.015±0	36.82	36.51±0.24	0.098	0.096 ± 0.001

Tabla 27. Valores de ancho de banda de transición, rizado de las bandas de paso y de corte y atenuación de la banda eliminada obtenidos tras 20 realizaciones de los algoritmos FPA y PM. Corresponden a un filtro paso bajo de orden 20.

De la Tabla 26 se deduce que, para el caso del método de las ventanas, si se desea reducir el ancho de banda de transición, se debe disminuir la atenuación de la banda eliminada (β más

pequeña). Al hacer esto, el rizado de la banda de paso aumenta, así como la máxima atenuación que se produce en la misma. Además, podemos extraer que el algoritmo FPA implementado junto a la función de aptitud múltiple desarrollada en este trabajo proporciona una mejor respuesta en frecuencia que el algoritmo PM en términos de ancho de banda de transición y atenuación de la banda eliminada. Sin embargo, aumenta el rizado de la banda de paso y, por consiguiente, la atenuación máxima producida en esta banda.

Todos los cálculos que hemos realizado han sido con el mismo ancho de banda de transición y mismo orden del filtro. Vamos a ver qué ocurre si modificamos esas dos medidas del filtro y qué valores de los distintos parámetros frecuenciales se obtienen para determinar con qué características funciona mejor nuestro método de diseño. La Tabla 27 muestra cómo cambian los distintos parámetros frecuenciales para distintos órdenes de filtro; todos ellos diseñados con el algoritmo FPA y las siguientes especificaciones: Frecuencia de muestreo (Fs)=1Hz, número de muestras frecuenciales (L) = 512, rizado de la banda de paso (δ_p) = 0.1, rizado de la banda eliminada (δ_s) = 0.01, frecuencia de corte de la banda de paso (w_p) = 0.45π, frecuencia de corte normalizada de la banda eliminada (w_s) = 0.55π y el ancho de banda de transición = 0.1π.

Cuanto mayor es el orden del filtro diseñado, disminuye el ancho de banda de transición; sin embargo, lo que buscamos es obtener una gran atenuación de la banda eliminada sin perder dicho ancho de banda de transición. Por esto, decidimos aumentar la atenuación de la banda eliminada reduciendo el rizado deseado a δs = 0.003 para órdenes de filtros mayores. Los resultados obtenidos son bastante buenos, sin embargo, el coste computacional para órdenes de filtro altos es bastante elevado. Esto es por la necesidad de un aumento de la población para aumentar la diversidad y ampliar el espacio de búsqueda y un aumento del máximo de iteraciones empleadas para asegurar la convergencia (Tabla 28).

Orden del Filtro	*Máximo rizado de la banda de paso (0-0.45π)*	*Atenuación máxima de la banda de paso (dB)*	*Máximo Rizado de la banda de corte (0.55π-π)*	*Atenuación mínima de la banda de corte (dB)*	*Ancho de banda de transición*	*Coste computacional (segundos)*
14	0.108544	-0.998	0.0590356	-24.6	0.103	4.5030
18	0.074999	0.845	0.0343026	-29.29	0.102	4.5335
20	0.110798	-1.015	0.0140452	-36.77	0.099	4.5644
22	0.130039	-1.21	0.0100246	-40.05	0.095	4.4735
24	0.137808	-1.07	0.0110314	-39.12	0.088	4.4510
*26**	0.125519	-1.165	0.0039993	-48.04	0.098	12.1878
*28**	0.138510	-1.295	0.0035933	-48.92	0.099	12.1266
*30**	0.149329	-1.279	0.0034007	-49.35	0.086	12.3207
*40***	0.149180	-1.224	0.0032484	-49.77	0.059	137.2548
*50***	0.187169	-1.7992	0.0035572	-49	0.049	489.2778
*60***	0.143045	-1.13	0.0001379	-77.23	0.068	751.2194
*70***	0.172343	-1.643	0.0000064	-103.7	0.077	1250.1448

Tabla 28. Coste computacional y parámetros frecuenciales obtenidos para distintos órdenes de filtros diseñados con el algoritmo FPA.

**Es necesario ampliar el número de flores a 50 y el número de iteraciones a 2500 para que converja el algoritmo (aumenta la carga computacional), además disminuimos el rizado de la banda eliminada a δ_s = 0.003 para obtener mayor atenuación.*

*** Para estos órdenes de filtro, es necesario aumentar considerablemente el tamaño de población (60-200) y por tanto el número de iteraciones (6000-18000) para que el algoritmo converja en un óptimo global.*

Si se aumentara el orden a uno mucho mayor, el número de iteraciones en conjunto con el aumento del tamaño de población y la dimensión del problema harían necesario el empleo de varias horas para conseguir que el algoritmo converja a una solución aceptable, por lo tanto, hemos restringido el uso de este método para filtros de orden bajo y medio ((N-1) ≤ 70).

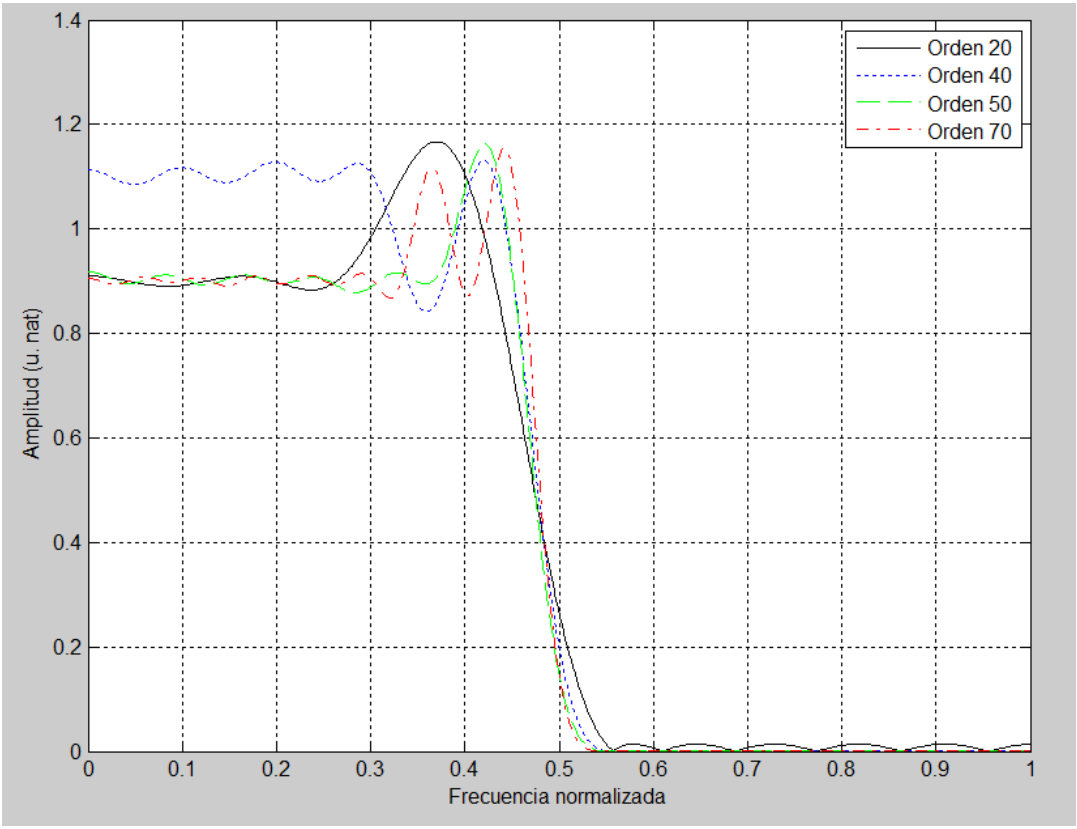

Figura 34. Respuesta en frecuencia de los filtros diseñados con el algoritmo FPA y la función de aptitud múltiple para distintos órdenes del filtro.

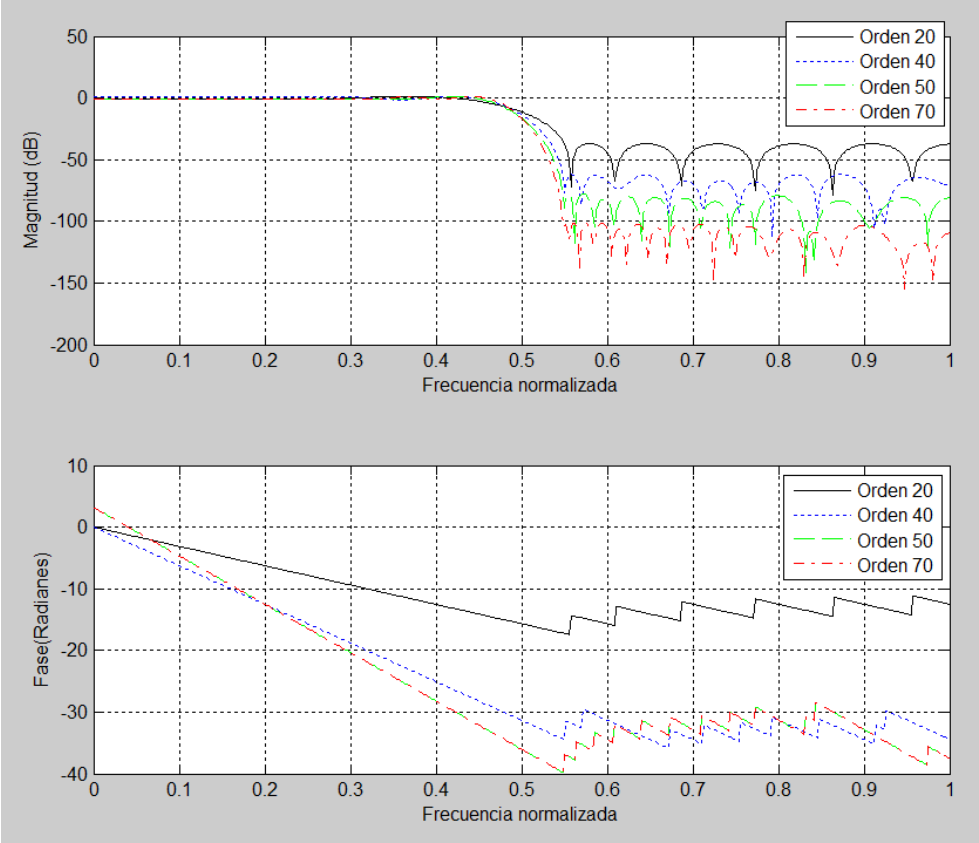

Figura 35. Respuesta en frecuencia (en dB) y fase de los filtros diseñados con el algoritmo FPA y la función de aptitud múltiple para distintos órdenes.

En las Fig. 34 y 35 se pueden observar las respuestas en frecuencia de los filtros diseñados con el algoritmo FPA para distintos órdenes. El ancho de banda de transición es menor para el filtro de orden mayor, al igual que ocurre con la atenuación de la banda eliminada.

Se ha podido comprobar que cuanto mayor es el orden del filtro que se desea diseñar, el problema se complica considerablemente, añadiendo un mayor número de coeficientes a optimizar y necesitando una población y un número de iteraciones mayor para converger en una solución aceptable. Esto era de esperar, puesto que, si añadimos más coeficientes a optimizar, se va a necesitar de un mayor número de flores en la población para poder abarcar todo el espacio de búsqueda, y si es mayor el número de flores, es necesario un mayor número de iteraciones para su convergencia. En la Fig. 36 se puede ver la relación entre el orden de filtro diseñado y el número de flores e iteraciones necesarias para su convergencia.

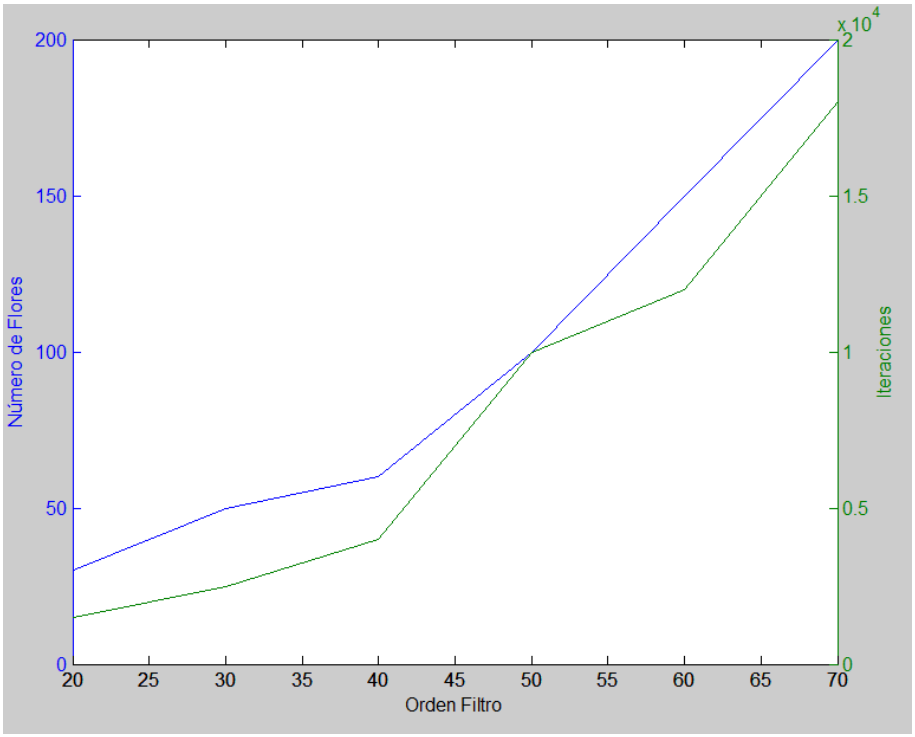

Figura 36. Número de flores e iteraciones necesarias para la convergencia del algoritmo FPA para distintos órdenes de filtros paso bajo diseñados.

A continuación, se realizarán distintas simulaciones con los siguientes parámetros constantes: Frecuencia de muestreo (Fs)=2Hz, número de muestras frecuenciales (L) = 512, rizado de la banda de paso (δ_p) = 0.1. Variaremos el ancho de banda de transición y el rizado de la banda de corte para comprobar cómo se comporta el algoritmo FPA junto con la función de aptitud múltiple. Los parámetros frecuenciales obtenidos se pueden observar en la Tabla 29. De estos resultados se deduce que el algoritmo FPA junto con la función de aptitud múltiple propuesta priorizan la minimización de los rizados de las bandas. Se obtiene, en la mayor parte de los casos, la atenuación establecida en ambas bandas sacrificando el ancho de banda de transición. Si en una primera implementación vemos que este ancho de banda es mayor al establecido, se deberá aumentar el orden del filtro o el rizado de una de las bandas para que se obtenga una menor atenuación de la banda eliminada o mayor atenuación en la banda de paso y se reduzca el ancho de banda de transición.

Ancho de banda de transición / δ_s	**Máximo rizado de la banda de paso**	**Atenuación máxima de la banda de paso (dB)**	**Máximo Rizado de la banda de corte**	**Atenuación mínima de la banda de corte (dB)**	**Ancho de banda de transición**
(0.45-0.55)π / 0.01	0.130338989	-1.082	0.01446662155	-36.8	0.099
(0.45-0.55)π / 0.03	0.1154955952	-1.066	0.0305056082	-30.32	0.087
(0.45-0.55)π / 0.05	0.1258298756	-1.084	0.0506146606	-25.94	0.079
(0.45-0.55)π / 0.008	0.115291907	-1.064	0.01347669911	-37.41	0.101
(0.45-0.6)π / 0.01	0.132369328	-1.132	0.0056577722	-44.95	0.1216
(0.45-0.6)π / 0.001	0.1390062478	-1.3	0.0020769739	-53.65	0.1485
(0.45-0.6)π / 0.0005	0.1223931588	-1.134	0.00055146825	-65.17	0.1686
(0.45-0.5)π / 0.01	0.1477833792	-1.215	0.0127317011	-37.9	0.1010
(0.45-0.5)π / 0.05	0.1237056733	-1.147	0.0757103148	-22.53	0.06
(0.45-0.5)π / 0.08	0.1285348987	-1.195	0.133675025	-17.48	0.05

Tabla 29.Parámetros frecuenciales de los filtros obtenidos mediante el algoritmo FPA con distintos valores de rizado de la banda de paso y ancho de banda de transición. Para un filtro paso bajo de orden 20.

Una vez hecha la comparativa con los métodos tradicionales, se va a realizar una comparativa entre los filtros realizados por otros autores para distintos algoritmos naturalistas y los filtros obtenidos con el algoritmo FPA. En la Tabla 30 se muestran las características frecuenciales obtenidas por medio de diferentes técnicas de computación natural. De la tabla se puede extraer que, el algoritmo propuesto en este trabajo, obtiene unas características frecuenciales del filtro diseñado algo superiores a otras técnicas de diseño. Se obtiene una mayor atenuación de la banda de corte que con cualquier otro método y un rizado de la banda de paso no muy elevado. La banda de transición es algo superior a la obtenida con otros métodos; sin embargo, se adapta al ancho de banda establecido en el diseño.

La Fig. 37 muestra las respuestas en frecuencia de los filtros diseñados con diferentes tipos de algoritmos evolutivos, posicionándose el diseñado con el algoritmo FPA como el que posee una mayor atenuación de la banda eliminada sin un aumento de la banda de transición. A su vez, el rizado de la banda de paso es muy similar al obtenido con otros métodos de diseño como se puede observar en la Tabla 30. Si nos fijamos en la fase de los filtros diseñados, todos ellos presentan una fase lineal en la banda de paso, cumpliendo los requisitos de diseño.

Método Empleado	**Tipo de filtro**	**Orden del filtro (N-1)**	**Atenuación mínima de la banda de corte (dB)**	**Máximo rizado de la banda de paso**	**Máximo Rizado de la banda de corte**	**Ancho de banda de transición**
Algoritmo DE [46]	LP	20	*	>0.08	>0.09	>0.06
Algoritmo eDE [24]	LP	20	<28	0.04	0.04(aprox.)	>0.06
Algoritmo PSO y LMS [33]	LP	33	<29	*	*	*
Algoritmo PSO y GA [34]	LP	30	<30	0.15	0.031	0.05
Algoritmo DEPSO [48]	LP	20	<27	>0.1	>0.06	>0.15
Algoritmo DEPSO [22]	LP	20	<27	0.291	0.0321	>0.13
Algoritmo PSO 3[23]	LP	20	31.6	0.0390	0.0260	0.0632
Algoritmo BFO [29]	LP	20	31.09	0.129	0.02773	0.0918
Algoritmo CSO [25]	LP	20	33.99	0.164	0.01998	0.0946
Algoritmo FPA	LP	14	24.6	0.108	0.0590	0.103
	LP	20	36.77	0.111	0.01405	0.099
	LP	26	48.04	0.125	0.00399	0.098
	LP	30	49.35	0.149	0.00340	0.086
	LP	40	-49.77	0.149	0.00324	0.059
	LP	50	-49	0.187	0.00355	0.049
	LP	60	-77.23	0.143	0.00013	0.068
	LP	70	-103.7	0.172	0.0000064	0.077

Tabla 30. Características frecuenciales obtenidas por distintos métodos de diseño en comparación con las obtenidas con el algoritmo FPA.

**No se especifican en el artículo.*

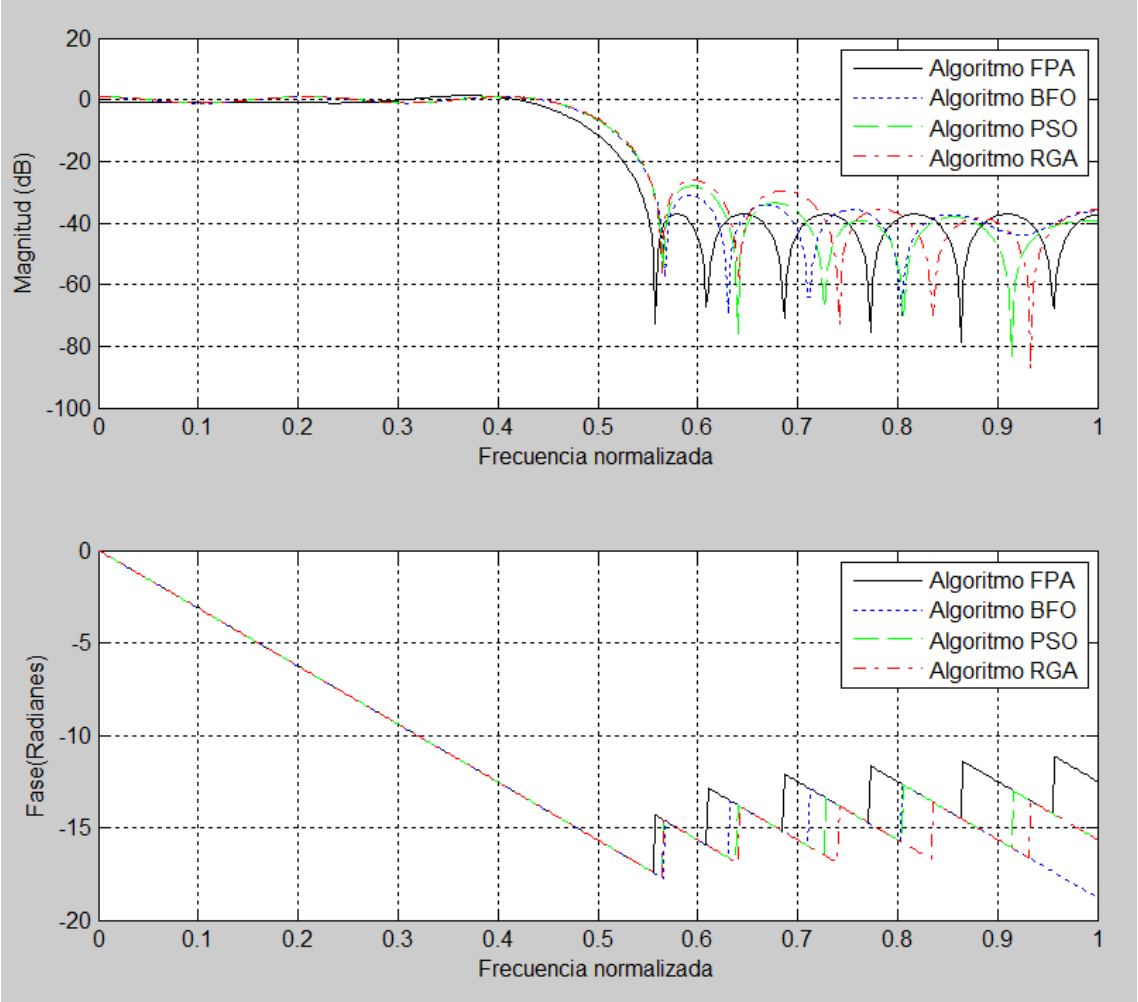

Figura 37. Respuesta en frecuencia (en dB) y fase de los filtros de orden 20 diseñados mediante distintos tipos de algoritmos evolutivos.

La Tabla 31 muestra los coeficientes del mejor filtro paso bajo obtenido con el algoritmo FPA y los coeficientes obtenidos con los algoritmos RGA, PSO y BFO en [29].

h(N)	RGA [29]	PSO [29]	BFO [29]	FPA
h(1)=h(21)	0.020644508012550	0.025116793352393	0.028302685047878	-0.017226782178695
h(2)=h(20)	0.048721413185106	0.047219259300299	0.046719898206481	-0.004869947065430
h(3)=h(19)	0.005868601564964	0.003546242723169	0.001859568002839	0.030508046178040
h(4)=h(18)	-0.040966865300227	-0.040094047283599	-0.040498019404735	0.030025794626746
h(5)=h(17)	-0.000863506780022	-0.000520432067214	0.001013961054997	-0.033750850915487
h(6)=h(16)	0.059796031265565	0.060907207778672	0.058649649609037	-0.066173068267902
h(7)=h(15)	-0.001408842862974	-0.001759240756773	-0.000384440011894	0.021432852698407
h(8)=h(14)	-0.103117834700311	-0.103613994946693	-0.105609977954995	0.113369483745420
h(9)=h(13)	-0.000440644382089	0.000627623037422	0.001355134966428	-0.001842291602112
h(10)=h(12)	0.317600651261946	0.318119036548684	0.315197573380121	-0.294787941753771
h(11)	0.500018538901556	0.500018538901556	0.500018538901556	-0.456542225556973

Tabla 31. Coeficientes del filtro óptimo diseñado mediante cuatro técnicas naturalistas diferentes.

6.2. Filtro Paso Alto

Los parámetros del filtro paso alto deseado que hemos establecido para el diseño son: Frecuencia de muestreo (Fs)=2Hz, número de muestras frecuenciales (L) = 512, orden del filtro (N-1) = 20, rizado de la banda de paso (δ_p) = 0.1, rizado de la banda eliminada (δ_s) = 0.01, frecuencia de corte normalizada de la banda de paso (w_p) = 0.55, frecuencia de corte normalizada de la banda eliminada (w_s) = 0.45 y el ancho de banda de transición = 0.1.

Se obtienen los coeficientes óptimos del filtro paso alto con distintas técnicas de diseño para determinar con qué método se obtienen unas mejores características frecuenciales del filtro diseñado. Los métodos que se van a simular, son dos métodos tradicionales: el método de enventanado con ventana de Káiser (β=1.7) y el método de Parks y McClellan [31]; y el algoritmo FPA desarrollado en este documento.

En las Fig. 38 y 39 se pueden comparar las respuestas en frecuencia de los tres filtros diseñados. Cabe destacar que el filtro que posee un menor ancho de banda de transición es el diseñado mediante el algoritmo FPA. Sin embargo, el rizado de la banda de paso es más elevado que en los otros dos métodos. La Fig. 38 muestra que el filtro diseñado con el algoritmo FPA presenta una mayor atenuación que el filtro diseñado con el algoritmo PM y que el diseñado con el método de enventanado.

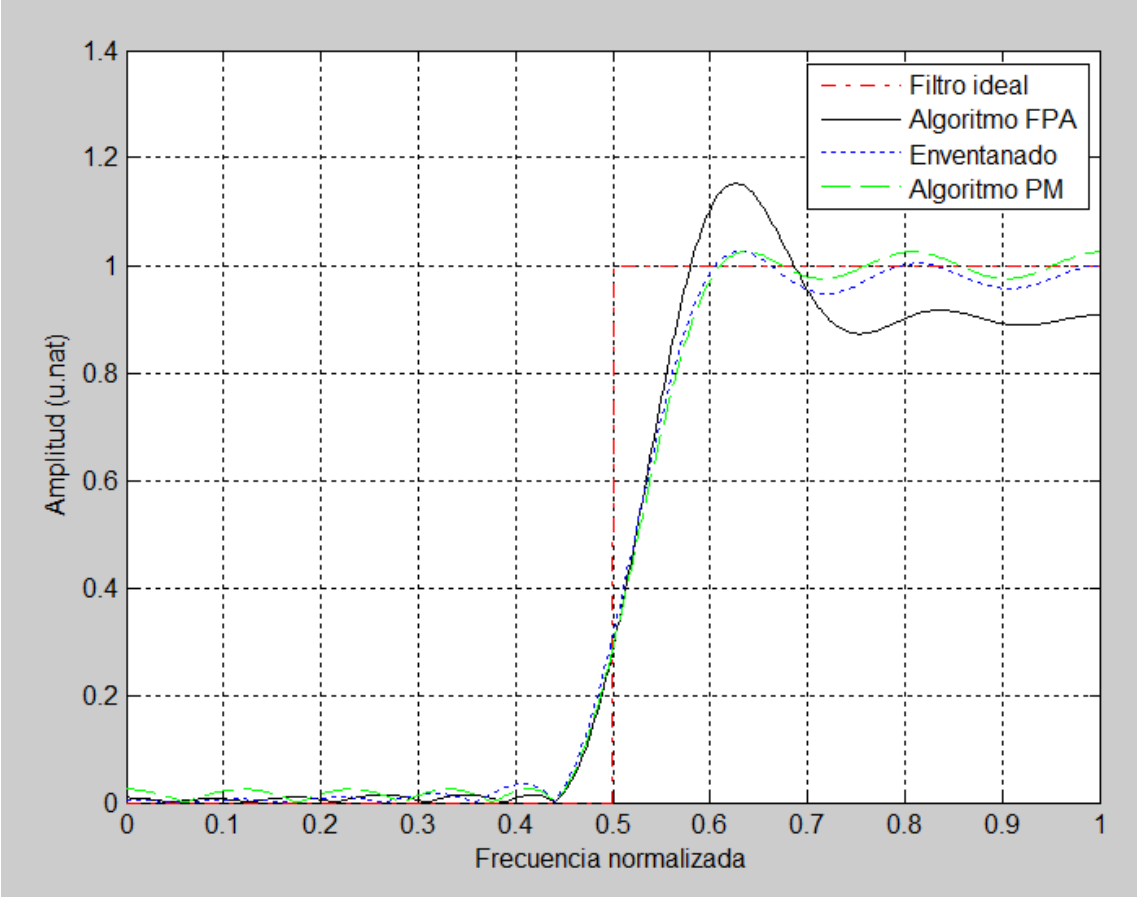

Figura 38.Respuesta en frecuencia de los filtros paso alto obtenidos con el algoritmo PM, el algoritmo FPA y el método de enventanado (v. Káiser con β=1.7) en comparación con el filtro ideal.

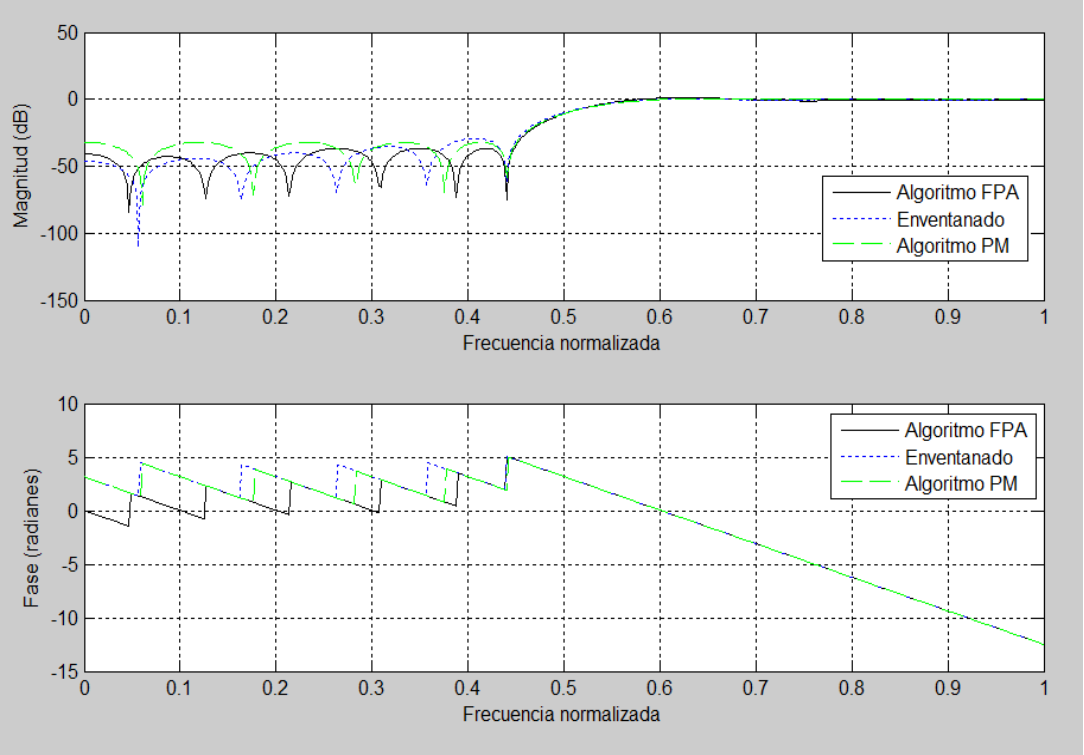

Figura 39. Respuesta en frecuencia y fase de los filtros paso alto obtenidos con el algoritmo PM, el algoritmo FPA y el método de enventanado (v. Káiser con β=1.7).

La Tabla 32 muestra los parámetros frecuenciales de rizado y atenuación en las bandas de corte y de paso, así como el ancho de banda de transición obtenido con los tres métodos de diseño simulados. Se obtienen unas muy buenas características con el método propuesto en este documento superando a las técnicas convencionales. Se consigue una mayor atenuación de la banda de corte a costa de aumentar el rizado de la banda de paso.

Método Empleado	*Máximo rizado de la banda de paso (0-0.45π)*		*Máximo Rizado de la banda de corte (0.55π-π)*		*Atenuación mínima de la banda de corte (dB)*		*Ancho de banda de transición*	
	Máx	**Medio ± s.d.**	**Máx**	**Medio ± s.d.**	**Máx**	**Medio ± s.d.**	**Máx**	**Medio ± s.d.**
Algoritmo PM [31]	0.025	0.025 ± 0	0.025	0.025 ± 0	32.03	32.03± 0	0.104	0.1039 ± 0
Enventanado con v. Káiser y β=1.7	0.0524	0.0524 ± 0	0.034	0.034 ± 0	29.38	29.38 ± 0	0.104	0.1039 ± 0
Algoritmo FPA	0.1828	0.164 ± 0.014	0.014	0.013 ± 0.001	37.70	37.49 ± 0.14	0.098	0.096 ± 0.002

Tabla 32. Valores de ancho de banda de transición, rizado y atenuación de las bandas de paso y de corte obtenidos tras 20 realizaciones de los algoritmos FPA y PM.

En la Tabla 33 se puede ver el resultado obtenido en el diseño del filtro paso alto óptimo con el método FPA comparado con las técnicas empleadas por otros autores mediante métodos de computación natural. El algoritmo propuesto es el que presenta una mayor atenuación de la banda de corte; o lo que es lo mismo, un menor rizado de la banda eliminada que los algoritmos comparados. Sin embargo, el rizado de la banda de paso es un poco mayor que con otros métodos de diseño. El ancho de banda de transición se ajusta perfectamente a los requisitos de diseño, al igual que ocurre con el filtro paso bajo diseñado.

Método Empleado	*Atenuación mínima de la banda de corte (dB)*	*Máximo rizado de la banda de paso*	*Máximo Rizado de la banda de corte*	*Ancho de banda de transición*
Algoritmo BFO [29]	33.44	0.129	0.02129	0.0937
Algoritmo CSO [25]	33.62	0.132	0.02085	0.0941
Algoritmo FPA	37.49	0.1643	0.0133	0.098

Tabla 33. Características frecuenciales obtenidas por distintos métodos de diseño naturalistas en comparación con las obtenidas con el algoritmo FPA para el diseño de un filtro paso alto de orden 20.

La Fig. 40 muestra las respuestas en frecuencia de los filtros diseñados con diferentes tipos de algoritmos evolutivos, posicionándose el diseñado con el algoritmo FPA como el que posee una mayor atenuación de la banda eliminada sin un aumento de la banda de transición.

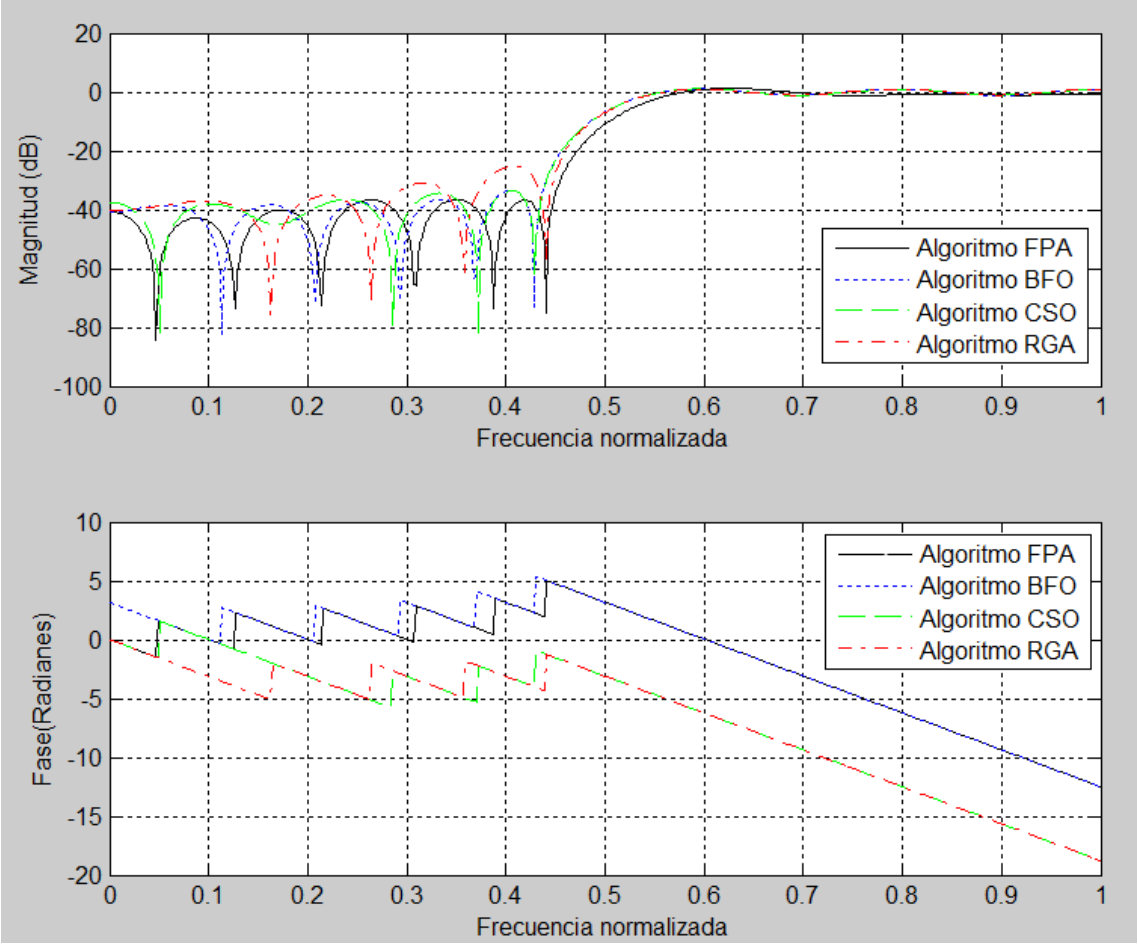

Figura 40. Respuesta en frecuencia (en dB) y fase de los filtros diseñados distintos tipos de algoritmos evolutivos.

La Tabla 34 contiene los coeficientes del filtro FIR paso alto óptimo conseguido mediante el algoritmo FPA en comparación con los obtenidos por los algoritmos RGA, BFO [29] y CSO [25].

h(N)	RGA [29]	BFO [29]	CSO [25]	FPA
h(1)=h(21)	0.021731353326545	0.027824923367289	0.027538232958190	-0.01610375734904
h(2)=h(20)	-0.048131602227058	-0.046597665739077	-0.044380496079130	0.008952989653055
h(3)=h(19)	0.006298189918824	0.004224538931035	0.003299313166652	0.025879152138873
h(4)=h(18)	0.041895345956760	0.038757904733956	0.042933406737536	-0.033024764023322
h(5)=h(17)	0.000879943669486	-0.002348409804654	0.000099323498881	-0.027489302717608
h(6)=h(16)	-0.059027866591514	-0.059691053526051	-0.058234539320593	0.063584246952844
h(7)=h(15)	-0.000013559660394	0.000185499704015	0.003181307723221	0.015698385190627
h(8)=h(14)	0.104257677520726	0.103659087803552	0.102402693086120	-0.110076416145845
h(9)=h(13)	0.003823743541217	-0.003720248355056	-0.002236399396214	-0.000022330210536
h(10)=h(12)	-0.316631427282300	-0.316800834726383	-0.317925532944817	0.294133800052195
h(11)	0.499468012025621	0.499981461098444	0.499981461098444	-0.455924680501984

Tabla 34. Coeficientes del filtro paso alto óptimo diseñado mediante cuatro técnicas naturalistas diferentes.

6.3. Filtro Paso Banda

Los parámetros del filtro paso banda deseado que hemos establecido para el diseño son: Frecuencia de muestreo (Fs)=2Hz, número de muestras frecuenciales (L) = 512, orden del filtro (N-1) = 20, rizado de la banda de paso (δ_p) = 0.1, rizado de la banda eliminada (δ_s) = 0.01, frecuencia de corte de la banda eliminada 1 (w_{s1}) = 0.25, frecuencia de corte normalizada de la banda de paso 1 (w_{p1}) = 0.35, frecuencia de corte normalizada de la banda de paso 2 (w_{p2}) = 0.65, frecuencia de corte de la banda eliminada 2 (w_{s2}) = 0.75 y el ancho de banda de transición = 0.1.

Se van a obtener de los coeficientes óptimos del filtro paso banda con distintas técnicas de diseño para determinar con qué método se obtienen unas mejores características frecuenciales del filtro diseñado. Los métodos que se van a simular, son dos métodos tradicionales: el método de enventanado con ventana de Káiser (β=1.9) y el método de Parks y McClellan [31]; y el algoritmo FPA desarrollado en este documento.

En las Fig. 41 y 42 se pueden comparar las respuestas en frecuencia de los tres filtros diseñados. Se comprueba que el método que presenta una mayor atenuación de las bandas eliminadas es el algoritmo FPA, con un ancho de banda de transición muy similar al del resto de los métodos, siendo, por el contrario, peor en cuanto al rizado de la banda de paso que es ligeramente superior al rizado obtenido con otras técnicas de diseño.

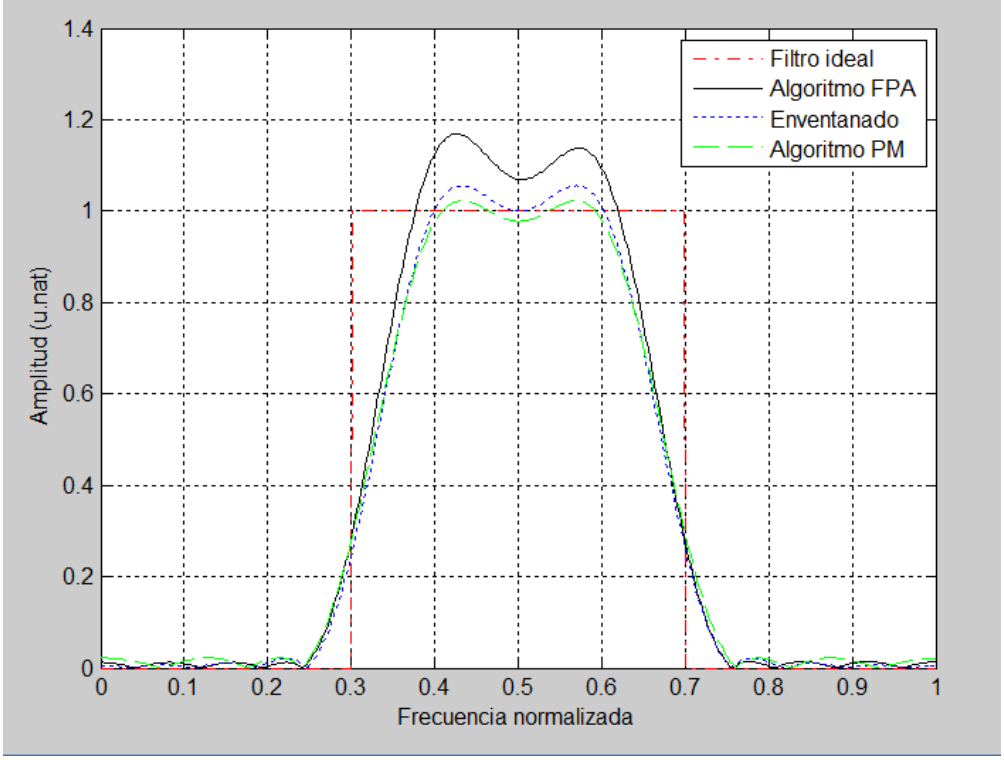

Figura 41. Respuesta en frecuencia de los filtros paso banda obtenidos con el algoritmo PM, el algoritmo FPA y el método de enventanado (v. Káiser con β=1.9) en comparación con el filtro ideal.

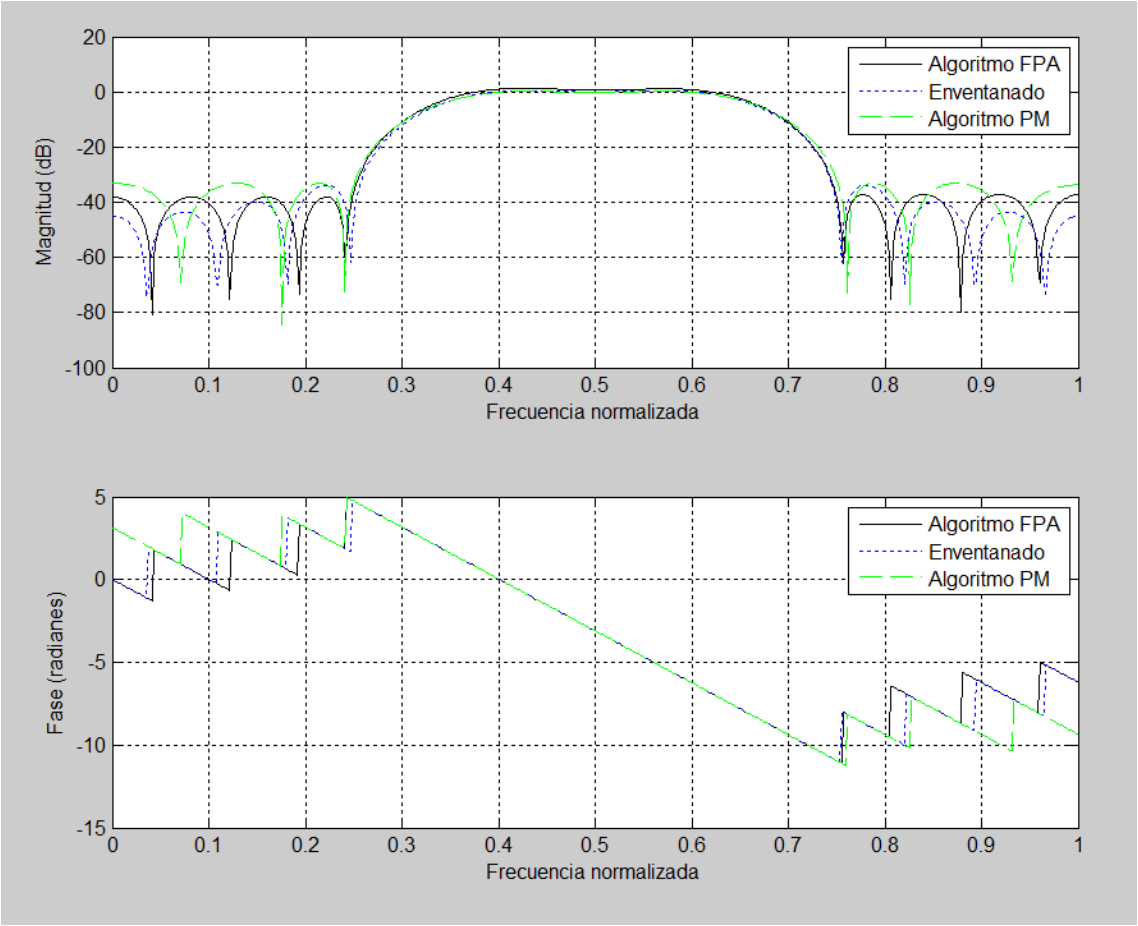

Figura 42. Respuesta en frecuencia y fase de los filtros paso banda obtenidos con el algoritmo PM, el algoritmo FPA y el método de enventanado (v. Káiser con β=1.9).

La Tabla 35 muestra la comparación de los parámetros frecuenciales obtenidos con los tres métodos, consolidándose el algoritmo FPA junto con la función de aptitud múltiple como una clara alternativa a las técnicas tradicionales para el diseño de filtros digitales paso banda. Se obtiene una mayor atenuación de la banda eliminada con un ancho de banda de transición mucho más pequeño que se ajusta a las especificaciones de diseño. Esto no ocurre con las técnicas tradicionales que, a pesar de tener una menor atenuación, el ancho de banda de transición es mucho mayor; sin embargo, el rizado en la banda de paso es mucho menor.

Método Empleado	*Máximo rizado de la banda de paso (0-0.45π)*		*Máximo Rizado de la banda de corte (0.55π-π)*		*Atenuación mínima de la banda de corte (dB)*		*Ancho de banda de transición*	
	Máx	**Medio ± s.d.**	**Máx**	**Medio ± s.d.**	**Máx**	**Medio ± s.d.**	**Máx**	**Medio ± s.d.**
Algoritmo PM [31]	0.040	0.0402 ± 0	0.034	0.0336 ± 0	29.46	29.46 ± 0	0.102	0.102±0
Enventanado con v. Káiser y β=1.9	0.055	0.0553 ± 0	0.020	0.0203 ± 0	33.82	33.82 ± 0	0.102	0.102±0
Algoritmo FPA	0.167	0.159±0.017	0.016	0.015±0.001	36.94	36.75 ± 0.4	0.094	0.094±0.001

Tabla 35. Valores de ancho de banda de transición, rizado y atenuación de las bandas de paso y de corte obtenidos tras 20 realizaciones de los algoritmos FPA y PM.

La Tabla 36 muestra el resultado obtenido en el diseño del filtro paso banda óptimo con el método FPA. Se compara con las técnicas empleadas por otros autores mediante métodos de computación natural. El algoritmo propuesto es el que presenta una mayor atenuación de la banda de corte; o lo que es lo mismo, un menor rizado de la banda eliminada que los algoritmos comparados. Además, aunque el rizado de la banda de paso es un poco mayor que con otros métodos, el ancho de banda de transición se ajusta perfectamente a los requisitos de diseño, obteniéndose un 10% menos de banda de transición que la requerida. Está en la media de los otros métodos de diseño, que tampoco se ajustan a los requerimientos de diseño.

Método Empleado	***Orden del filtro (N-1)***	***Atenuación mínima de la banda de corte (dB)***	***Máximo rizado de la banda de paso***	***Máximo Rizado de la banda de corte***	***Ancho de banda de transición***
Algoritmo DEPSO [48]	20	<8	>0.2	>0.05	>0.07
Algoritmo FASA [16]	30	<33	*	*	>0.1
Algoritmo PSO y LMS [33]	33	<25	*	*	*
Algoritmo BFO [29]	20	34.63	0.142	0.01856	0.1014
Algoritmo CSO [25]	20	32.11	0.144	0.02479	0.1034
Algoritmo FPA	20	36.94	0.1679	0.016	0.0937

Tabla 36. Características frecuenciales obtenidas por distintos métodos de diseño naturalistas en comparación con las obtenidas con el algoritmo FPA en el diseño de un filtro paso banda.

**No se especifican en el artículo.*

La Fig. 43 muestra las respuestas en frecuencia de los filtros diseñados con diferentes tipos de algoritmos evolutivos, posicionándose el diseñado con el algoritmo FPA como el que posee una mayor atenuación de la banda eliminada sin un aumento de la banda de transición.

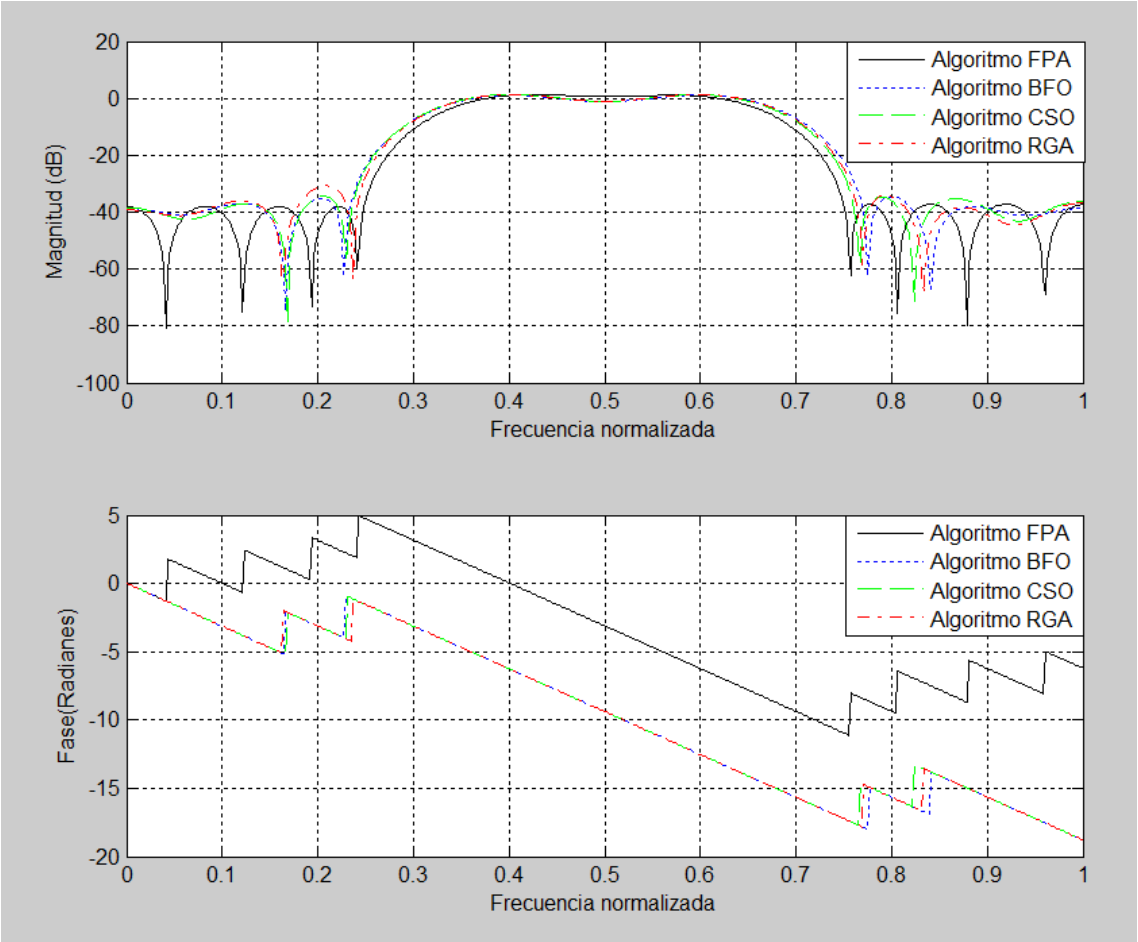

Figura 43. Respuesta en frecuencia (en dB) y fase de los filtros paso banda diseñados mediante distintos tipos de algoritmos evolutivos.

La tabla 37 contiene los coeficientes del filtro FIR paso banda óptimo conseguido mediante el algoritmo FPA en comparación con los obtenidos por los algoritmos RGA, BFO [29] y CSO [25].

h(N)	RGA [29]	BFO [29]	CSO [25]	FPA
h(1)=h(21)	0.028502857888104	0.025312619938173	0.029219726125746	-0.030500168683099
h(2)=h(20)	-0.001893868108392	0.000452170238718	-0.002853754882048	0.001595178617595
h(3)=h(19)	-0.076189026154460	-0.072644354421667	-0.075276944714603	0.051523797012138
h(4)=h(18)	0.000994123920259	-0.000654013990320	0.003288541979801	0.000300734469315
h(5)=h(17)	0.053196793860741	0.055262319583904	0.051130186585096	-0.007921282728573
h(6)=h(16)	-0.000639149080848	-0.000806958587929	0.001521235029526	-0.005273577617971
h(7)=h(15)	0.100057194730152	0.093750954252203	0.100018377612239	-0.131373999519684
h(8)=h(14)	0.001409980793664	0.001652190182879	-0.005673995620418	0.008191490607776
h(9)=h(13)	-0.299380312728113	-0.296008990498813	-0.298355272098757	0.301376853910954
h(10)=h(12)	-0.000752480372393	-0.000668163668605	0.002785499453246	-0.004043578936641
h(11)	0.400369877077545	0.400369877077545	0.400369877077545	-0.378973929525794

Tabla 37. Coeficientes del filtro paso banda óptimo diseñado mediante cuatro técnicas naturalistas diferentes.

6.4. Filtro Elimina Banda

Los parámetros del filtro paso banda deseado que hemos establecido para el diseño son: Frecuencia de muestreo (Fs)=2Hz, número de muestras frecuenciales (L) = 512, orden del filtro (N-1) = 20, rizado de la banda de paso (δ_p) = 0.1, rizado de la banda eliminada (δ_s) = 0.01, frecuencia de corte de la banda de paso 1 (w_{p1}) = 0.25, frecuencia de corte normalizada de la banda de corte 1 (w_{s1}) = 0.35, frecuencia de corte normalizada de la banda de corte 2 (w_{s2}) = 0.75, frecuencia de corte de la banda de paso 2 (w_{p2}) = 0.85 y el ancho de banda de transición = 0.1. Para asegurar la convergencia del algoritmo con este diseño es necesario aumentar la población a 40 flores y el máximo de generaciones a 2000.

Se obtienen los coeficientes óptimos del filtro elimina banda con distintas técnicas de diseño para determinar con qué método se obtienen unas mejores características frecuenciales del filtro diseñado. Los métodos que se van a simular, son dos métodos tradicionales: el método de enventanado con ventana de Káiser (β=1.9) y el método de Parks y McClellan [31]; y el algoritmo FPA desarrollado en este documento.

En las Fig. 44 y 45 se pueden comparar las respuestas en frecuencia de los tres filtros diseñados. Cabe destacar que el filtro que posee un menor ancho de banda de transición es el diseñado mediante el algoritmo FPA. Sin embargo, el rizado de la banda de paso es más elevado que en los otros dos métodos. La Fig. 45 muestra que el filtro diseñado con el algoritmo FPA presenta una mayor atenuación que el filtro diseñado con el algoritmo PM y que el diseñado con el método de enventanado.

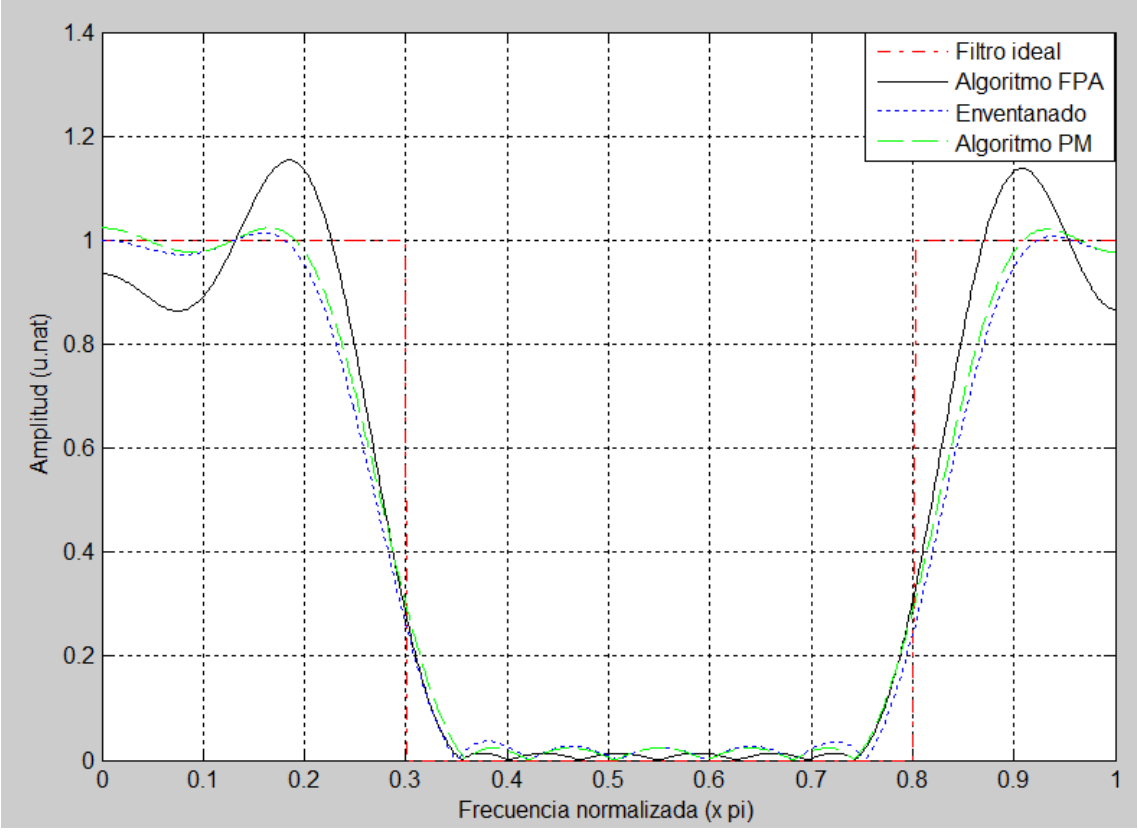

Figura 44. Respuesta en frecuencia de los filtros elimina banda obtenidos con el algoritmo PM, el algoritmo FPA y el método de enventanado (v. Káiser con β=1.9) en comparación con el filtro ideal.

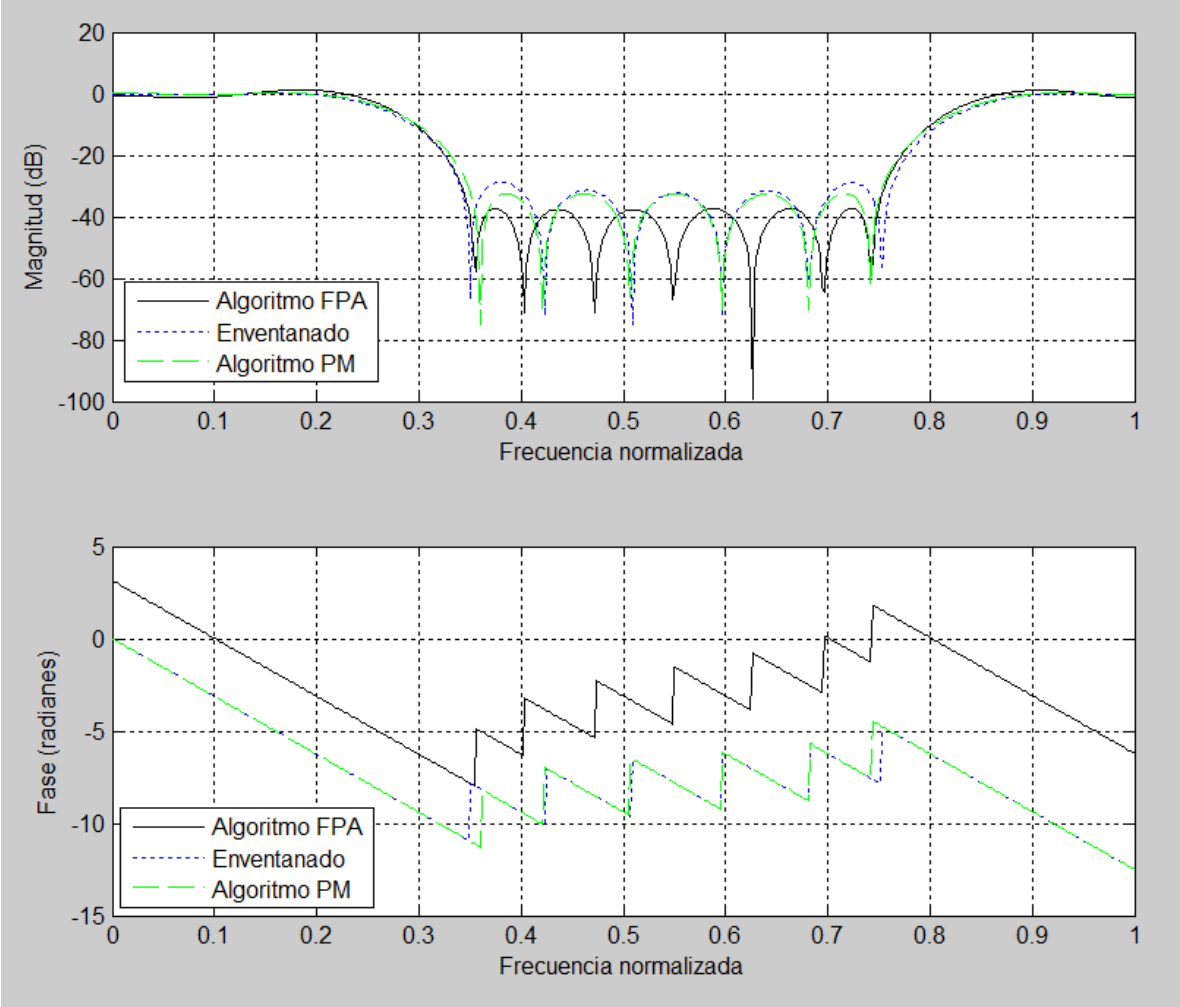

Figura 45. Respuesta en frecuencia y fase de los filtros paso banda obtenidos con el algoritmo PM, el algoritmo FPA y el método de enventanado (v. Káiser con β=1.9).

La Tabla 38 muestra la comparación de los parámetros frecuenciales obtenidos con los tres métodos, consolidándose también el algoritmo FPA junto con la función de aptitud múltiple como una clara alternativa a las técnicas tradicionales para el diseño de filtros digitales elimina banda. Se obtiene una mayor atenuación de la banda eliminada con un ancho de banda de transición también mucho menor al conseguido con otros métodos de diseño.

Método Empleado	*Máximo rizado de la banda de paso (0-0.45π)*		*Máximo Rizado de la banda de corte (0.55π-π)*		*Atenuación mínima de la banda de corte (dB)*		*Ancho de banda de transición*	
	Máx	**Medio ± s.d.**	**Máx**	**Medio ± s.d.**	**Máx**	**Medio ± s.d.**	**Máx**	**Medio ± s.d.**
Algoritmo PM [31]	0.035	0.035 ± 0	0.031	0.0305 ± 0	30.28	30.28 ± 0	0.099	0.099 ± 0
Enventanado con v. Káiser y β=1.9	0.061	0.061± 0	0.021	0.0209 ± 0	33.58	33.58± 0	0.095	0.095 ± 0
Algoritmo FPA	0.178	0.151±0.011	0.014	0.013 ± 0.001	38.03	37.53 ± 0.34	0.090	0.090 ± 0.001

Tabla 38. Valores de ancho de banda de transición, rizado y atenuación de las bandas de paso y de corte obtenidos tras 20 realizaciones de los algoritmos FPA y PM para el diseño de un filtro elimina-banda de orden 20.

La Tabla 39 muestra el resultado obtenido en el diseño del filtro elimina banda óptimo con el método FPA. Se compara con los parámetros frecuenciales de los filtros obtenidos mediante el empleo de otros métodos de computación natural. El algoritmo propuesto es el que presenta una mayor atenuación de la banda de corte; o lo que es lo mismo, un menor rizado de la banda eliminada que los algoritmos comparados. Además, el rizado de la banda de paso es un poco menor que con otros métodos y el ancho de banda de transición es bastante mejor al de otras técnicas, consolidándose como una muy buena alternativa a otros métodos para el diseño de cualquier tipo de filtro digital.

Método Empleado	***Atenuación mínima de la banda de corte (dB)***	***Máximo rizado de la banda de paso***	***Máximo Rizado de la banda de corte***	***Ancho de banda de transición***
Algoritmo BFO [29]	33.84	0.161	0.02033	0.1080
Algoritmo CSO [25]	34.47	0.163	0.01891	0.1006
Algoritmo FPA	37.72	0.1518	0.0132	0.090

Tabla 39. Características frecuenciales obtenidas por distintos métodos de diseño naturalistas en comparación con las obtenidas con el algoritmo FPA en el diseño de un filtro elimina banda de orden 20.

La Fig. 46 muestra las respuestas en frecuencia de los filtros diseñados con diferentes tipos de algoritmos evolutivos, posicionándose el diseñado con el algoritmo FPA como el que posee una mayor atenuación de la banda eliminada sin un aumento de la banda de transición.

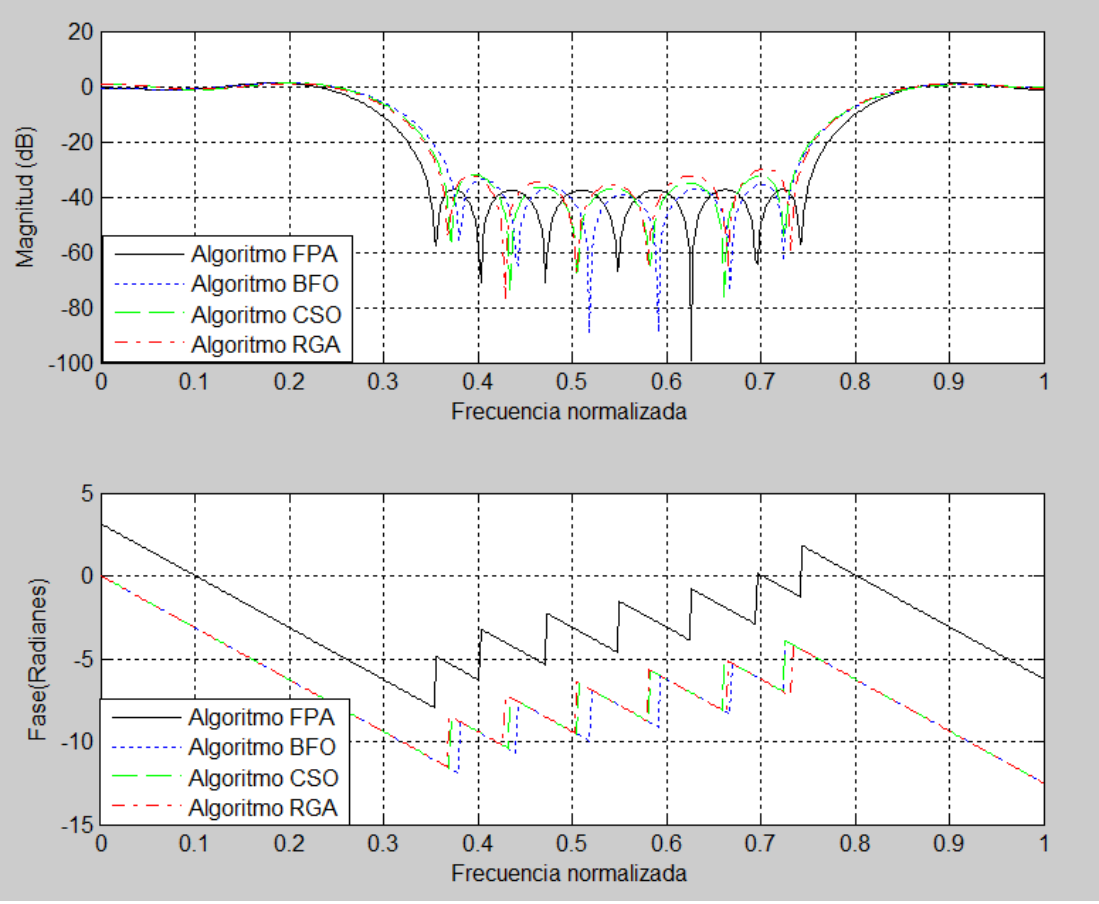

Figura 46. Respuesta en frecuencia (en dB) y fase de los filtros elimina banda diseñados mediante distintos tipos de algoritmos evolutivos.

La Tabla 40 contiene los coeficientes del filtro FIR elimina banda óptimo conseguido mediante el algoritmo FPA en comparación con los obtenidos por los algoritmos RGA, BFO [29] y CSO [25].

h(N)	RGA [29]	BFO [29]	CSO [25]	FPA
h(1)=h(21)	0.008765244188382	0.001215676211074	0.011453242233574	-0.041413454999685
h(2)=h(20)	0.054796923249762	0.045537959243453	0.052437411749113	0.002764601104204
h(3)=h(19)	0.001796419983890	-0.002144609358493	0.011428069462394	-0.055761794053191
h(4)=h(18)	0.048911654246731	0.039021065188363	0.047411160505444	0.049166741491508
h(5)=h(17)	-0.054718457691943	-0.059827163056132	-0.049098141901288	0.026931802515620
h(6)=h(16)	-0.060963142228236	-0.076247946667813	-0.065254829042326	0.057159855845370
h(7)=h(15)	0.004293459264617	-0.012399887285566	-0.001020332055708	0.089854847574650
h(8)=h(14)	-0.065342448643273	-0.082031203556439	-0.068705355567975	-0.139812747073478
h(9)=h(13)	0.300682045893488	0.289729300694508	0.296714629509332	0.000991113585408
h(10)=h(12)	0.069036675664641	0.068426397965933	0.074218022294613	-0.415499167403741
h(11)	0.499582536276171	0.500000357523254	0.500000357523254	-0.083024485043927

Tabla 40. Coeficientes del filtro elimina banda óptimo diseñado mediante cuatro técnicas naturalistas diferentes.

7. Filtrado de una señal biomédica- EEG

El electroencefalograma (EEG) es una señal biomédica que registra la actividad eléctrica del cerebro. Esta señal eléctrica es una suma temporal y espacial de los potenciales postsinápticos generados por las neuronas en la corteza cerebral. La amplitud del campo eléctrico generado oscila entre 5 y 200µV.

El registro de esta actividad se realiza mediante electrodos que se colocan en una serie de marcas en el cráneo según el sistema internacional 10-20. Las letras indican cada región de la cabeza: frontopolar (Fp), frontal (F), temporal (T), central (C), parietal (P), occipital (O).

La Fig. 47 se muestra un esquema de la colocación de los electrodos en las distintas partes de la cabeza y de los nombres empleados según el sistema internacional 10-20.

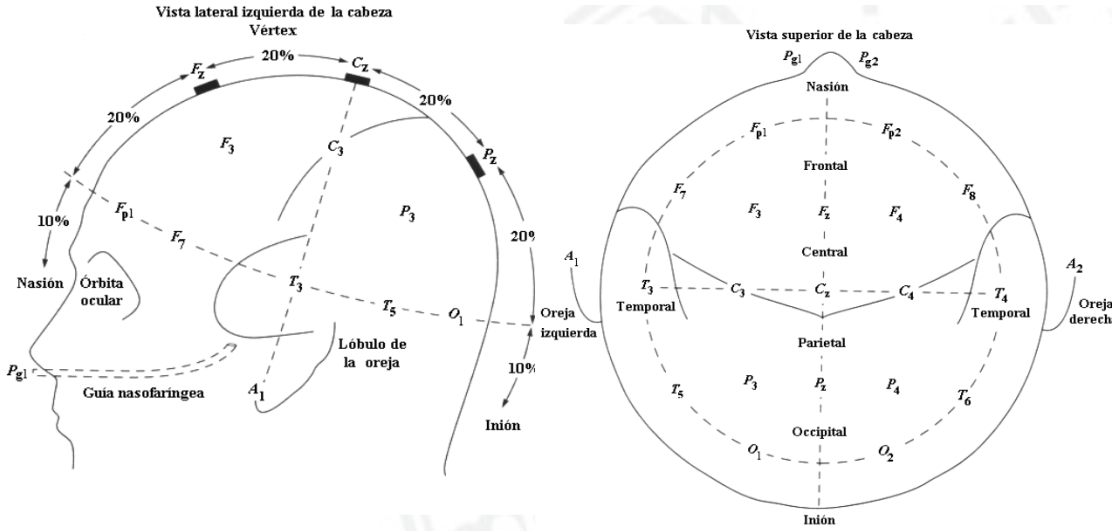

Figura 47. Colocación de los electrodos según el sistema internacional 10-20.

El EEG posee varias bandas de frecuencia que poseen una mayor o menor actividad según la acción que está realizando el sujeto. Con el método de diseño de filtros propuesto en este documento, se van a realizar distintos tipos de filtros digitales con el objetivo de separar las bandas de frecuencia del EEG para facilitar su procesado posterior

La señal EEG empleada, que se puede observar en la Fig. 48, ha sido extraída de la base de datos pública physionet [55]. Dicha señal pertenece a un sujeto dormido y tiene los siguientes parámetros: Número de muestras (N) = 1000000, frecuencia de muestreo (Fs) = 100 Hz, electrodos empleados: Fpz-Cz. De las 1000000 muestras, se extraen 1000 que son las pertenecientes a los primeros 10 segundos de señal.

La Fig. 49 muestra la transformada de Fourier de la señal donde se pueden ver las componentes frecuenciales que la componen. La DFT se ha realizado con L=1024 muestras.

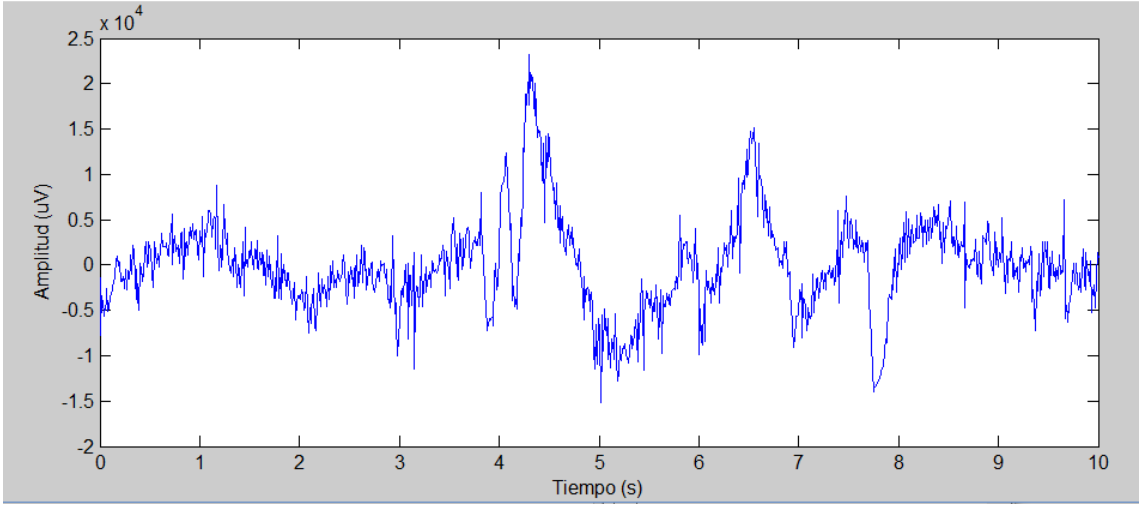

Figura 48. 10 segundos de la señal del EEG obtenida de un sujeto dormido.

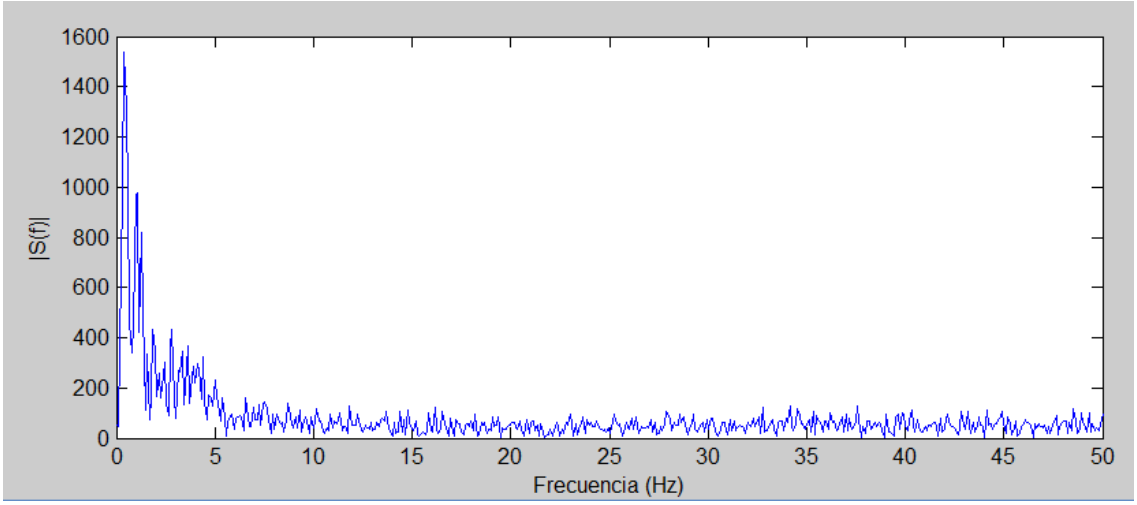

Figura 49. Transformada de Fourier de la señal de Electroencefalograma

7.1. Banda Delta

La banda Delta está formada por frecuencias inferiores a 4 Hz. Es propia de sujetos en estado de sueño profundo, respiración forzada (hiperventilación), en lactantes, niños y adolescentes, así como en pacientes con enfermedades cerebrales graves.

Predomina la actividad Delta en las zonas frontal y parieto-occipital.

Para obtener esta banda de frecuencias se plantea el problema de diseñar un filtro paso bajo digital que permita el paso de las frecuencias de 0 a 4 Hz y elimine el resto. Los parámetros que hemos elegido para este filtro son los siguientes: Frecuencia de muestreo (Fs) = 100Hz, frecuencia de corte de la banda de paso (fp) = 4Hz, frecuencia de corte de la banda eliminada (fs) = 5Hz, orden del filtro diseñado (N-1) = 70, número de muestras frecuenciales (L) = 1024.

La Fig. 50 muestra la transformada de Fourier discreta de la señal de EEG filtrada mediante convolución con el filtro paso bajo de orden 70, cuya respuesta en frecuencia se puede observar también en la parte inferior de la Fig. 50. Al realizar el filtrado, la señal resultante tendrá solamente las componentes frecuenciales de la banda Delta (0-4 Hz) como puede apreciarse. La

Fig. 51 muestra la banda Delta de la señal en el dominio temporal, con componentes de muy baja frecuencia.

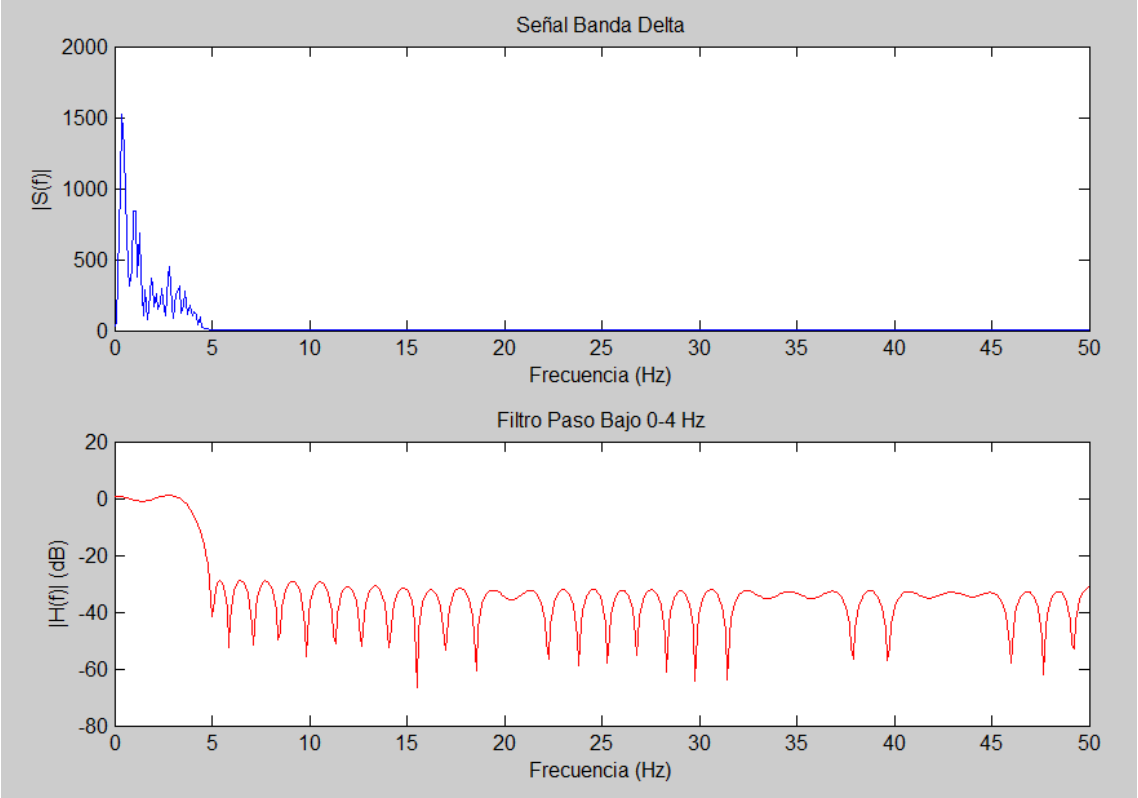

Figura 50. DFT de la señal EEG filtrada (0-4Hz) y respuesta en frecuencia del filtro empleado.

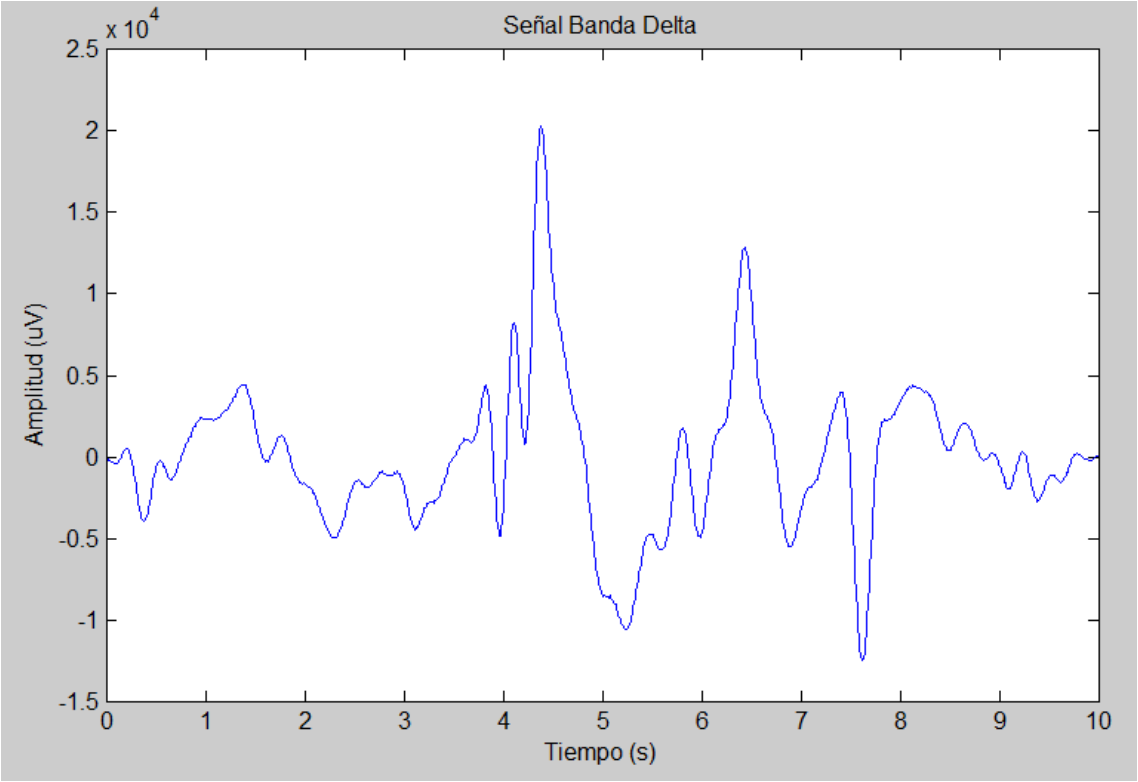

Figura 51. Señal de EEG filtrada, solamente con componentes frecuenciales de la banda Delta.

Para comprobar el funcionamiento del filtro diseñado en la extracción de la banda deseada y la supresión del resto de bandas de frecuencia; se realiza el filtrado de la señal con otros dos filtros digitales del mismo orden (N-1 = 70), diseñados con el algoritmo PM y con el método de enventanado. Sobre esa señal filtrada se calcula la potencia de señal en la banda deseada y la potencia de la señal en las bandas eliminadas. De esta manera podemos determinar qué filtro elimina una mayor potencia de la señal no deseada, dejando pasar solamente las frecuencias esperadas. Los resultados obtenidos se pueden observar en la tabla 40, conseguidos al calcular la potencia sobre 500 fragmentos diferentes (de 15 segundos de duración) de la señal de un mismo sujeto en estado de sueño. Se obtienen la potencia media de los fragmentos de la señal en la banda deseada, en este caso la banda Delta; y la potencia media de los fragmentos de la señal en las bandas eliminadas.

La Tabla 41 muestra que el filtro que elimina una mayor potencia de las bandas no deseadas es el diseñado mediante el algoritmo FPA a costa de atenuar un poco más la banda deseada. Esto era de esperar debido a que la técnica desarrollada presenta un mayor rizado en la banda de paso y por tanto una mayor atenuación de la misma. Sin embargo, la potencia de la banda eliminada es muy inferior a la potencia de la banda eliminada de la señal filtrada con cualquiera de los otros dos filtros, consolidándose como una muy buena alternativa en la eliminación de las bandas no deseadas sin atenuar en exceso la banda de paso.

El filtro FPA deja pasar aproximadamente un 15% menos de la potencia de la señal deseada, sin embargo, elimina en torno a 3-5 veces más de potencia de señal indeseada que los filtros diseñados con técnicas convencionales.

Filtro Empleado	*Potencia media en la banda Delta de la señal filtrada (mW)*	*Potencia media en la banda eliminada de la señal filtrada (mW)*	*Ratio entre la potencia media de señal en la banda deseada y en la banda eliminada*
Técnica Enventanado	11.730477	0.116907	100.340067
Algoritmo PM	13.236105	0.155902	84.900119
Método Mínimos Cuadrados	12.546249	0.122413	102.491014
Algoritmo FPA	10.819727	0.035935	301.085125

Tabla 41. Potencia media de las bandas deseada (Delta) y eliminada en 500 fragmentos de la señal de EEG con distintos filtros empleados.

7.2. Banda Theta

La banda Theta está formada por frecuencias comprendidas entre 4 y 8 Hz. Es la principal componente del EEG infantil. Aparecen al mantener los ojos abiertos, bajo respiración forzada (hiperventilación), al dormirse, con cansancio y en ligeras alteraciones generales. Predominan en la zona occipital.

Para obtener esta banda de frecuencia se plantea el problema de diseñar un filtro paso banda digital que permita el paso de las frecuencias de 4 a 8 Hz y elimine el resto. Los parámetros que hemos elegido para este filtro son los siguientes: Frecuencia de muestreo (Fs) = 100Hz, frecuencia de corte de la banda eliminada 1 (fs1) = 3Hz, frecuencia de corte de la banda de paso

1 (fp1) = 4Hz, frecuencia de corte de la banda de paso 2 (fp2) = 8Hz, frecuencia de corte de la banda eliminada 2 (fs2) = 9Hz, orden del filtro diseñado (N-1) = 70, número de muestras frecuenciales (L) = 1024.

La Fig. 52 muestra la transformada de Fourier discreta de la señal de EEG filtrada mediante convolución con el filtro paso banda de orden 70, cuya respuesta en frecuencia se puede observar también en la parte inferior de la Fig. 52. Al realizar el filtrado, la señal resultante tendrá solamente las componentes frecuenciales de la banda Theta (4-8 Hz) como puede apreciarse. La Fig. 53 muestra la banda Theta de la señal en el dominio temporal.

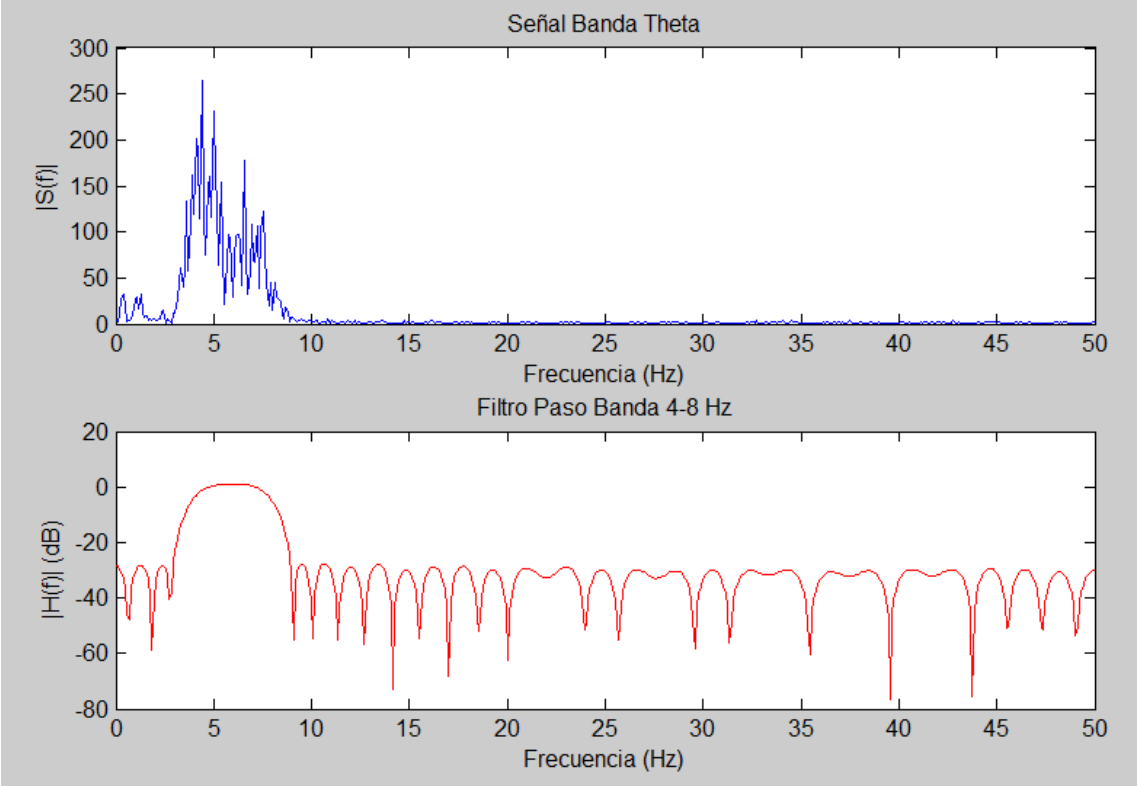

Figura 52. DFT de la señal EEG filtrada (4-8Hz) y respuesta en frecuencia del filtro empleado.

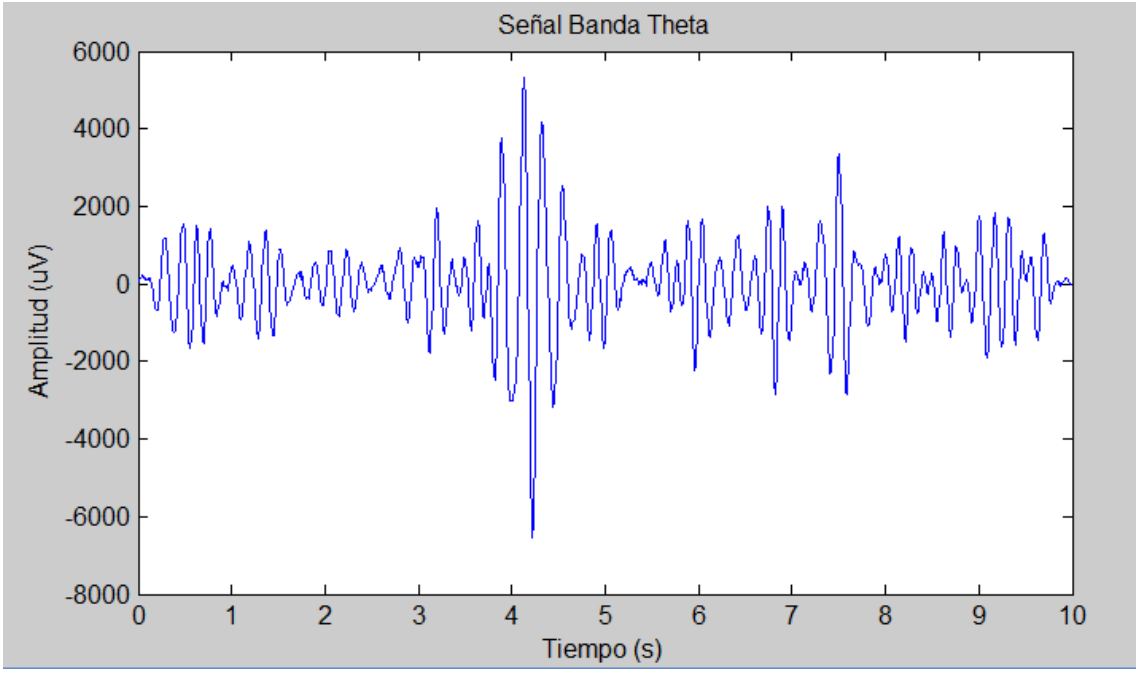

Figura 53. Señal de EEG filtrada, solamente con componentes frecuenciales de la banda Theta.

Para comprobar el funcionamiento del filtro diseñado en la extracción de la banda deseada y la supresión del resto de bandas de frecuencia; se realiza el filtrado de la señal con otros dos filtros digitales del mismo orden (N-1 = 70), diseñados con el algoritmo PM y con el método de enventanado. Sobre esa señal filtrada se calcula la potencia de señal en la banda deseada y la potencia de la señal en las bandas eliminadas. De esta manera podemos determinar qué filtro elimina una mayor potencia de la señal no deseada, dejando pasar solamente las frecuencias esperadas. Los resultados obtenidos se pueden observar en la tabla 41, conseguidos al calcular la potencia sobre 500 fragmentos diferentes (de 15 segundos de duración) de la señal de un mismo sujeto en estado de sueño. Se obtienen la potencia media de los fragmentos de la señal en la banda deseada, en este caso la banda Theta; y la potencia media de los fragmentos de la señal en las bandas eliminadas.

La Tabla 42 muestra que el filtro que elimina una mayor potencia de las bandas no deseadas es el diseñado mediante el algoritmo FPA a costa de atenuar un poco más la banda deseada. Esto era de esperar debido a que la técnica desarrollada presenta un mayor rizado en la banda de paso y por tanto una mayor atenuación de la misma. Sin embargo, la potencia de la banda eliminada es muy inferior a la potencia de la banda eliminada de la señal filtrada con cualquiera de los otros dos filtros, consolidándose como una muy buena alternativa en la eliminación de las bandas no deseadas sin atenuar en exceso la banda de paso.

El filtro FPA deja pasar un 5-8% menos de la potencia de la señal deseada; pero elimina 5 veces más de la potencia indeseada.

Filtro Empleado	*Potencia media en la banda Theta de la señal filtrada (mW)*	*Potencia media en la banda eliminada de la señal filtrada (mW)*	*Ratio entre la potencia media de señal en la banda deseada y en la banda eliminada*
Técnica Enventanado	1.104075	0.569910	1.853568
Algoritmo PM	0.987332	0.509106	1.841944
Método Mínimos Cuadrados	0.973266	0.491964	1.978328
Algoritmo FPA	0.932947	0.100701	9.264454

Tabla 42. Potencia media de las bandas deseada (Theta) y eliminada en 500 fragmentos de la señal de EEG con distintos filtros empleados.

7.3. Banda Alfa

La banda Alfa está formada por frecuencias comprendidas entre 8 y 13 Hz. Aparecen en sujetos normales despiertos, sin ninguna actividad y con los ojos cerrados. También aparecen en estado de conciencia meditativo, en fase de sueño REM, en intoxicaciones por medicamentos y en estado comatoso. Predominan en la zona occipital.

Para obtener esta banda de frecuencia se plantea el problema de diseñar un filtro paso banda digital que permita el paso de las frecuencias de 8 a 13 Hz y elimine el resto. Los parámetros que hemos elegido para este filtro son los siguientes: Frecuencia de muestreo (Fs) = 100Hz, frecuencia de corte de la banda eliminada 1 (fs1) = 7Hz, frecuencia de corte de la banda de paso 1 (fp1) = 8Hz, frecuencia de corte de la banda de paso 2 (fp2) = 13Hz, frecuencia de corte de la banda eliminada 2 (fs2) = 14Hz, orden del filtro diseñado (N-1) = 70, número de muestras frecuenciales (L) = 1024.

La Fig. 54 muestra la transformada de Fourier discreta de la señal de EEG filtrada mediante convolución con el filtro paso banda de orden 70, cuya respuesta en frecuencia se puede observar también en la parte inferior de la Fig. 54. Al realizar el filtrado, la señal resultante tendrá solamente las componentes frecuenciales de la banda Alfa (8-13 Hz) como puede apreciarse. La Fig. 55 muestra la banda Alfa de la señal en el dominio temporal.

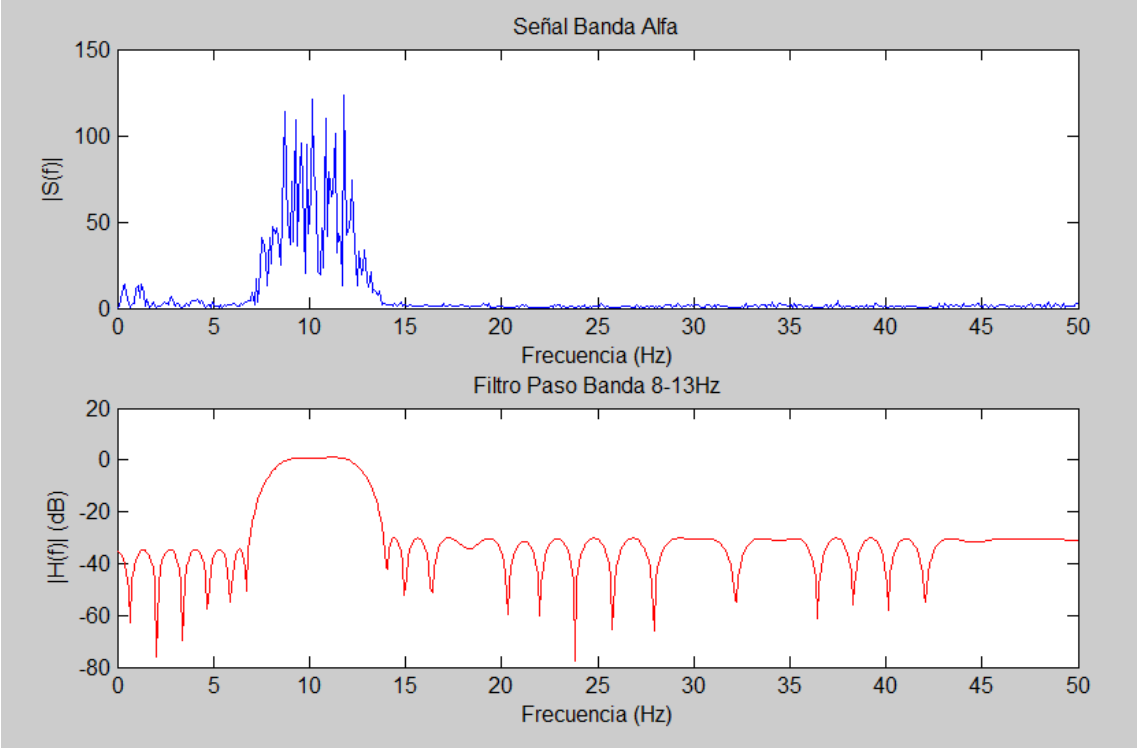

Figura 54. DFT de la señal EEG filtrada (8-13Hz) y respuesta en frecuencia del filtro empleado.

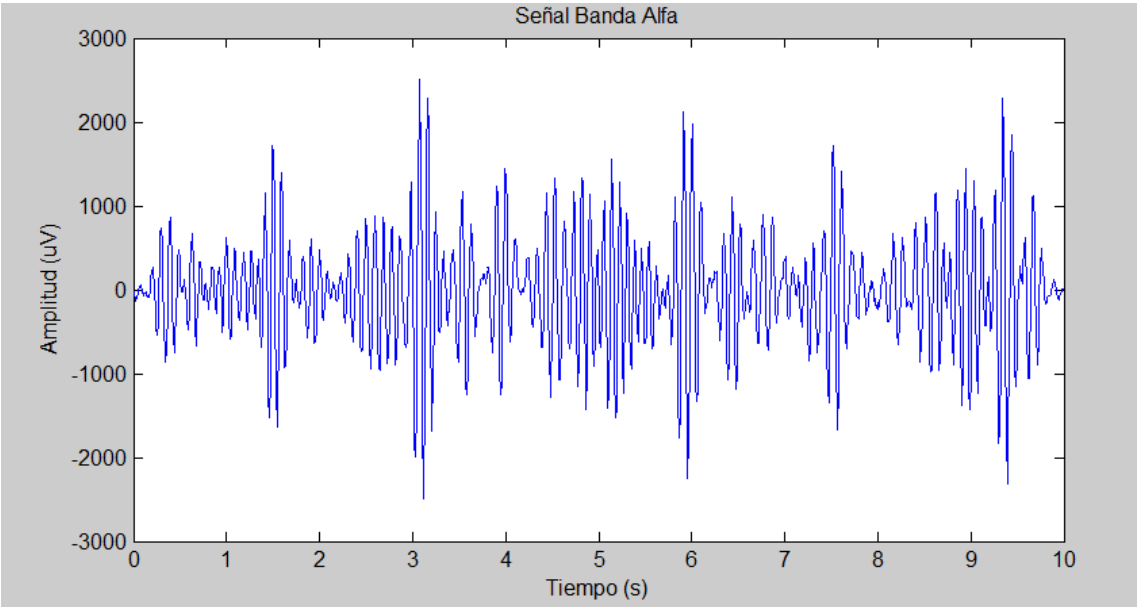

Figura 55. Señal de EEG filtrada, solamente con componentes frecuenciales de la banda Alfa.

Para comprobar el funcionamiento del filtro diseñado en la extracción de la banda deseada y la supresión del resto de bandas de frecuencia; se realiza el filtrado de la señal con otros dos filtros digitales del mismo orden (N-1 = 70), diseñados con el algoritmo PM y con el método de enventanado. Sobre esa señal filtrada se calcula la potencia de señal en la banda deseada y la potencia de la señal en las bandas eliminadas. De esta manera podemos determinar qué filtro elimina una mayor potencia de la señal no deseada, dejando pasar solamente las frecuencias esperadas. Los resultados obtenidos se pueden observar en la Tabla 43, conseguidos al calcular la potencia sobre 500 fragmentos diferentes (de 15 segundos de duración) de la señal de un mismo sujeto en estado de sueño. Se obtienen la potencia media de los fragmentos de la señal

en la banda deseada, en este caso la banda Alfa; y la potencia media de los fragmentos de la señal en las bandas eliminadas.

La tabla 42 muestra que el filtro que elimina una mayor potencia de las bandas no deseadas es el diseñado mediante el algoritmo FPA a costa de atenuar un poco más la banda deseada. Esto era de esperar debido a que la técnica desarrollada presenta un mayor rizado en la banda de paso y por tanto una mayor atenuación de la misma. Sin embargo, la potencia de la banda eliminada es muy inferior a la potencia de la banda eliminada de la señal filtrada con cualquiera de los otros dos filtros, consolidándose como una muy buena alternativa en la eliminación de las bandas no deseadas sin atenuar en exceso la banda de paso.

El filtro FPA deja pasar aproximadamente un 10% menos de la potencia de la señal deseada, sin embargo, elimina en torno a 5-10 veces más de potencia de señal indeseada que los filtros diseñados con técnicas tradicionales.

Filtro Empleado	*Potencia media en la banda Alfa de la señal filtrada (mW)*	*Potencia media en la banda eliminada de la señal filtrada (mW)*	*Ratio entre la potencia media de señal en la banda deseada y en la banda eliminada*
Técnica Enventanado	0.355976	0.054544	6.526345
Algoritmo PM	0.340549	0.168617	2.019663
Método Mínimos Cuadrados	0.322813	0.045735	7.058337
Algoritmo FPA	0.316775	0.016662	19.012061

Tabla 43. Potencia media de las bandas deseada (Alfa) y eliminada en 500 fragmentos de la señal de EEG con distintos filtros empleados.

7.4. Banda Beta

La banda Beta está formada por frecuencias comprendidas entre 14 y 30 Hz. Aparecen en adultos en estado de vigilia, cuando hay mala relajación y al comienzo del sueño. También aparecen bajo la administración de ciertos medicamentos, cuando el sujeto está bajo tensión, en ciertos estados comatosos y en relación con algunas enfermedades internas.

Para obtener esta banda de frecuencia se plantea el problema de diseñar un filtro paso banda digital que permita el paso de las frecuencias de 14 a 30 Hz y elimine el resto. Los parámetros que hemos elegido para este filtro son los siguientes: Frecuencia de muestreo (Fs) = 100Hz, frecuencia de corte de la banda eliminada 1 (fs1) = 13Hz, frecuencia de corte de la banda de paso 1 (fp1) = 14Hz, frecuencia de corte de la banda de paso 2 (fp2) = 30Hz, frecuencia de corte de la banda eliminada 2 (fs2) = 31Hz, orden del filtro diseñado (N-1) = 70, número de muestras frecuenciales (L) = 1024.

La Fig. 56 muestra la transformada de Fourier discreta de la señal de EEG filtrada mediante convolución con el filtro paso banda de orden 70, cuya respuesta en frecuencia se puede observar también en la Fig. 56. Al realizar el filtrado, la señal resultante tendrá solamente las componentes frecuenciales de la banda Beta (14-30 Hz) como puede apreciarse. La Fig. 57 muestra la banda Beta de la señal en el dominio temporal.

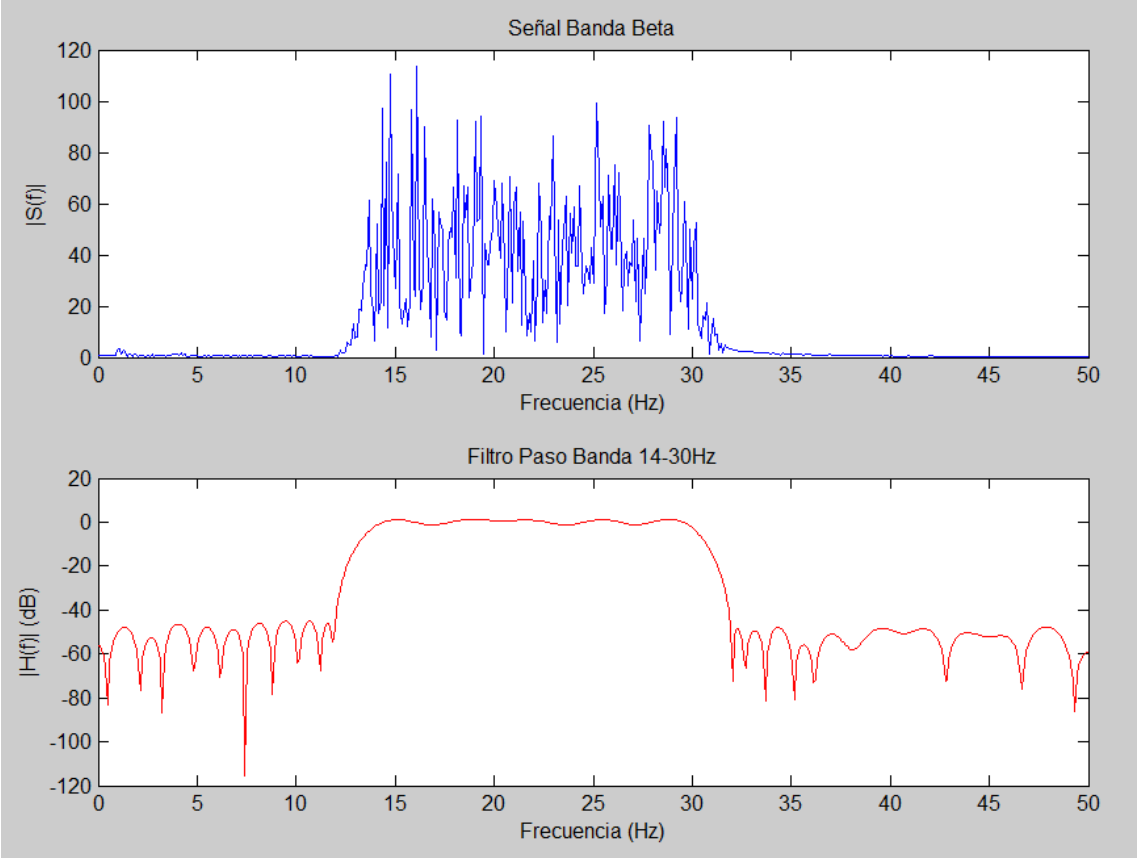

Figura 56. DFT de la señal EEG filtrada (14-30Hz) y respuesta en frecuencia del filtro empleado.

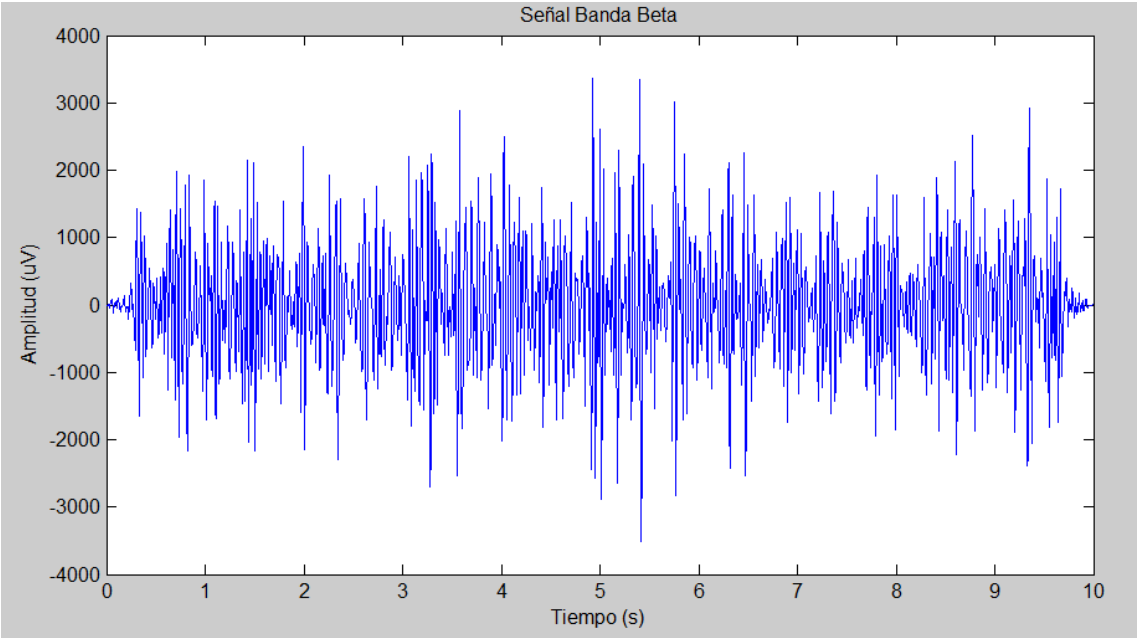

Figura 57. Señal de EEG filtrada, solamente con componentes frecuenciales de la banda Beta.

Para comprobar el funcionamiento del filtro diseñado en la extracción de la banda deseada y la supresión del resto de bandas de frecuencia; se realiza el filtrado de la señal con otros dos filtros digitales del mismo orden (N-1 = 70), diseñados con el algoritmo PM y con el método de enventanado. Sobre esa señal filtrada se calcula la potencia de señal en la banda deseada y la potencia de la señal en las bandas eliminadas. De esta manera podemos determinar qué filtro elimina una mayor potencia de la señal no deseada, dejando pasar solamente las frecuencias

esperadas. Los resultados obtenidos se pueden observar en la Tabla 44, conseguidos al calcular la potencia sobre 500 fragmentos diferentes (de 15 segundos de duración) de la señal de un mismo sujeto en estado de sueño. Se obtienen la potencia media de los fragmentos de la señal en la banda deseada, en este caso la banda Beta; y la potencia media de los fragmentos de la señal en las bandas eliminadas.

La Tabla 44 muestra que el filtro que elimina una mayor potencia de las bandas no deseadas es el diseñado mediante el algoritmo FPA y además posee una mayor potencia de la banda deseada. Por esto, se consolida el algoritmo FPA como una muy buena alternativa para el diseño de filtros digitales, separando con mayor éxito la banda de frecuencia deseada.

En este caso, el filtro diseñado con el algoritmo FPA consigue una menor atenuación (en torno al 5%) de la señal deseada en la banda de paso y 3-4 veces mayor atenuación de la potencia de la señal indeseada en la banda eliminada que los filtros diseñados con técnicas tradicionales.

Filtro Empleado	*Potencia media en la banda Beta de la señal filtrada (mW)*	*Potencia media en la banda eliminada de la señal filtrada (mW)*	*Ratio entre la potencia media de señal en la banda deseada y en la banda eliminada*
Técnica Enventanado	0.686911	0.064739	10.610453
Algoritmo PM	0.668623	0.068201	9.803642
Método Mínimos Cuadrados	0.666624	0.059735	11.159578
Algoritmo FPA	0.705367	0.025704	27.441407

Tabla 44. Potencia media de las bandas deseada (Beta) y eliminada en 500 fragmentos de la señal de EEG con distintos filtros empleados.

7.5. Banda Gamma

La banda Gamma está formada por frecuencias comprendidas de 30 Hz en adelante. Se encuentra actualmente en investigación. Se ha teorizado que las ondas gamma podrían estar implicadas en el proceso de percepción consciente, pero no hay acuerdo unánime al respecto.

Para obtener esta banda de frecuencia se plantea el problema de diseñar un filtro paso alto digital que permita el paso de las frecuencias de 30 Hz en adelante y elimine el resto. Los parámetros que hemos elegido para este filtro son los siguientes: Frecuencia de muestreo (Fs) = 100 Hz, frecuencia de corte de la banda eliminada (fs) = 28 Hz, frecuencia de corte de la banda de paso (fp) = 30 Hz, orden del filtro diseñado (N-1) = 70, número de muestras frecuenciales (L) = 1024.

La Fig. 58 muestra la transformada de Fourier discreta de la señal de EEG filtrada mediante convolución con el filtro paso alto de orden 70, cuya respuesta en frecuencia se puede observar también en la Fig. 58. Al realizar el filtrado, la señal resultante tendrá solamente las componentes frecuenciales de la banda Gamma (>30 Hz) como puede apreciarse. La Fig. 59 muestra la banda Gamma de la señal en el dominio temporal.

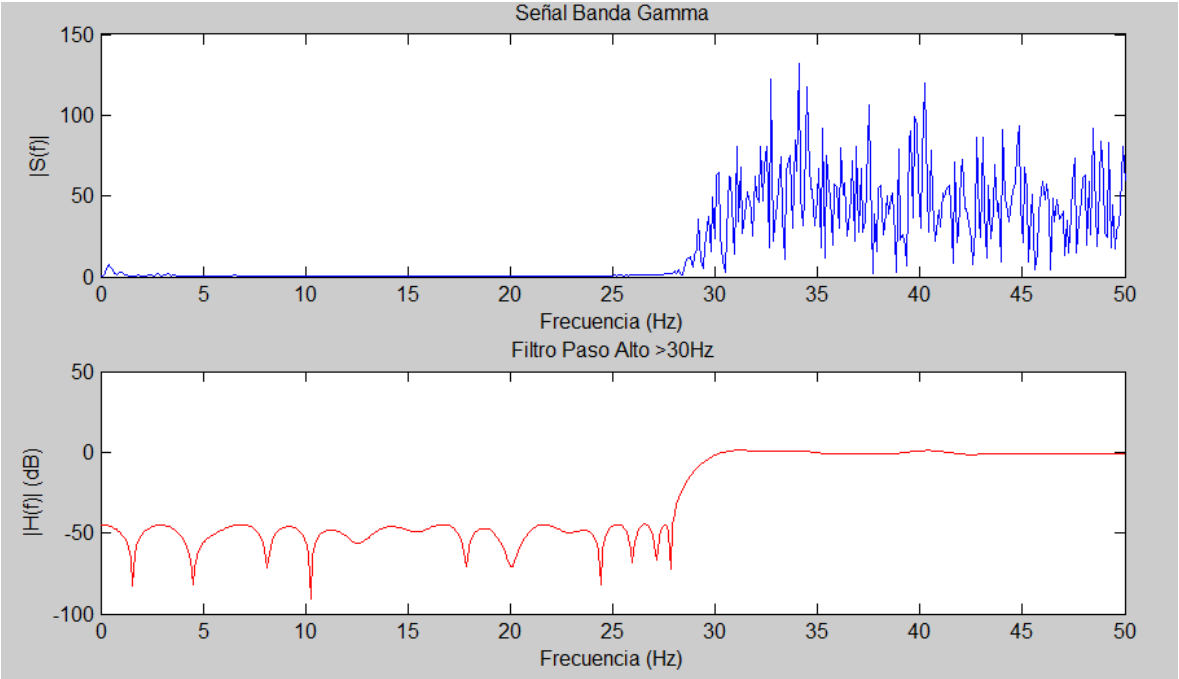

Figura 58. DFT de la señal EEG filtrada (>30Hz) y respuesta en frecuencia del filtro empleado.

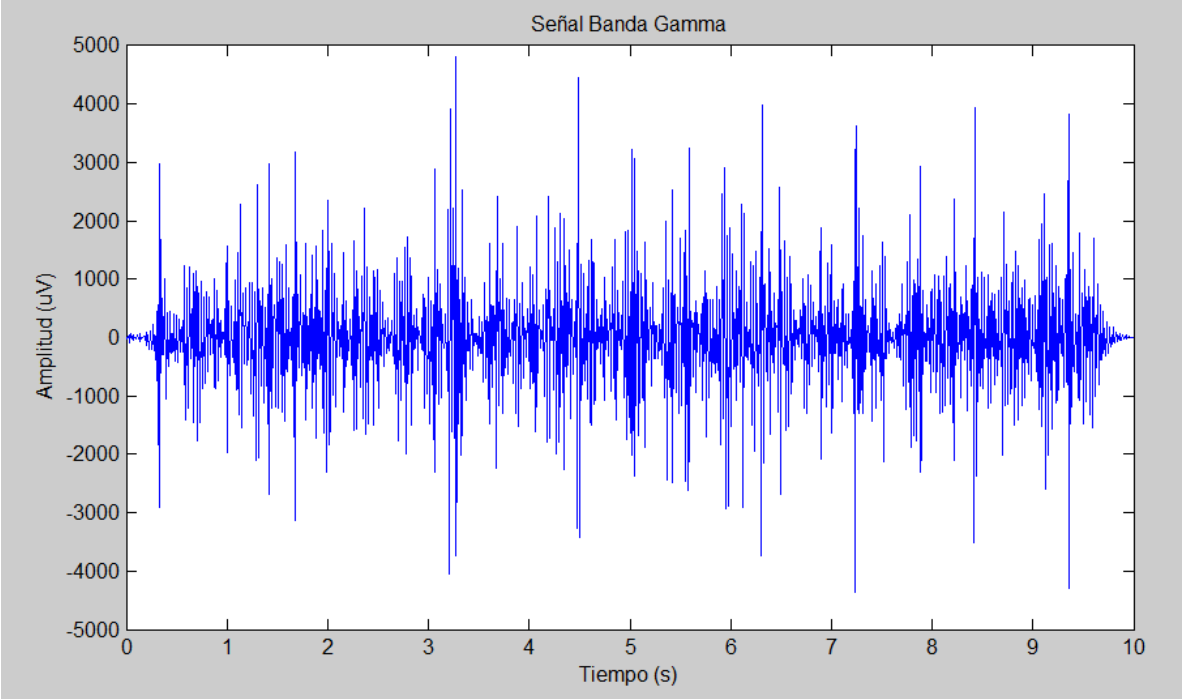

Figura 59. Señal de EEG filtrada, solamente con componentes frecuenciales de la banda Gamma.

Realizamos el filtrado de la señal con otros dos filtros digitales del mismo orden (N-1 = 70), diseñados con el algoritmo PM y con el método de enventanado. Sobre esa señal filtrada se calcula la potencia de señal en la banda deseada y la potencia de la señal en las bandas eliminadas. De esta manera podemos determinar qué filtro elimina una mayor potencia de la señal no deseada, dejando pasar solamente las frecuencias esperadas. Los resultados obtenidos se pueden observar en la Tabla 45, conseguidos al calcular la potencia sobre 500 fragmentos diferentes (de 15 segundos de duración) de la señal de un mismo sujeto en estado de sueño. Se obtienen la potencia media de los fragmentos de la señal en la banda deseada, en este caso la banda Gamma; y la potencia media de los fragmentos de la señal en las bandas eliminadas.

La Tabla 45 muestra que el filtro que elimina una mayor potencia de las bandas no deseadas es el diseñado mediante el algoritmo FPA a costa de atenuar un poco más la banda deseada. Esto era de esperar debido a que la técnica desarrollada presenta un mayor rizado en la banda de paso y por tanto una mayor atenuación de la misma. Sin embargo, la potencia de la banda eliminada es muy inferior a la potencia de la banda eliminada de la señal filtrada (50% menos) con cualquiera de los otros dos filtros, consolidándose como una muy buena alternativa en la eliminación de las bandas no deseadas sin atenuar en exceso la banda de paso.

El filtro FPA deja pasar aproximadamente un 7% menos de la potencia de la señal deseada, sin embargo, elimina en torno a 2-3 veces más de potencia de señal indeseada que los filtros diseñados con técnicas convencionales.

Filtro empleado	***Potencia media en la banda Gamma de la señal filtrada (mw)***	***Potencia media en la banda eliminada de la señal filtrada (mw)***	***Ratio entre la potencia media de señal en la banda deseada y en la banda eliminada***
Técnica Enventanado	0.894812	0.030119	29.708935
Algoritmo PM	0.907726	0.035289	25.722373
Método Mínimos Cuadrados	0.906794	0.028278	32.067148
Algoritmo FPA	0.841039	0.015623	53.831453

Tabla 45. Potencia media de las bandas deseada (Gamma) y eliminada en 500 fragmentos de la señal de EEG con distintos filtros empleados.

8. Conclusiones

El principal objetivo de este trabajo ha sido explorar las distintas técnicas de computación natural y sus características para encontrar soluciones prometedoras en el problema multimodal del diseño de filtros FIR digitales. Para plantear el diseño de filtros como un problema de optimización es necesario elegir una *función de aptitud* que sirva para obtener correctamente las características frecuenciales deseadas. Además, es necesario encontrar un algoritmo que converja en una solución aceptable explorando al máximo el espacio de búsqueda empleando un tiempo de simulación no demasiado alto.

Se ha podido comprobar que el algoritmo FPA, junto con la función de aptitud múltiple propuesta en este trabajo, funcionan muy bien a la hora de encontrar los coeficientes óptimos de los filtros FIR de bajo orden sin necesidad de aumentar mucho la carga computacional. En el Capítulo 6, se muestran los resultados obtenidos por este método de diseño comparados con los métodos tradicionales de diseño de filtros digitales y otros métodos basados en computación natural.

En primer lugar, los métodos tradicionales empleados para la comparación son: el método de enventanado y el método de Parks-McClellan. Se obtienen unos resultados simulados empleando el algoritmo de FPA que mejoran las características frecuenciales de los filtros obtenidos por técnicas convencionales. Para comenzar, la respuesta en frecuencia cumple con bastante exactitud las características frecuenciales de ancho de banda de transición y de frecuencias de corte de las bandas; priorizando la atenuación de las bandas sobre el ancho de banda de transición. Se obtiene una mayor atenuación de la banda eliminada (4-5 dB) en comparación con los métodos tradicionales, reduciendo además el ancho de banda de transición. Sin embargo, se produce un ligero aumento del rizado de la banda de paso y por tanto un aumento de la atenuación en dicha banda en torno a 0.6-0.8 dB.

Por otro lugar, si se comparan con otras técnicas de computación natural como los algoritmos RGA, PSO, CSO y BFO, los resultados son bastante satisfactorios. Las simulaciones efectuadas indican que el método propuesto obtiene un mejor desempeño que otras técnicas en cuanto a los parámetros frecuenciales obtenidos: una mayor atenuación de las bandas eliminadas, un ancho de banda de transición adecuado y un rizado de la banda de paso muy similar al obtenido con otras técnicas de diseño.

Por esto, se puede afirmar que el algoritmo FPA junto con la función de aptitud múltiple puede ser empleado como una buena alternativa a otras técnicas para obtener los coeficientes de un filtro FIR (paso bajo, paso alto, elimina banda y paso banda) para órdenes de filtros bajos y medios ((N-1) < 70). Lo que ocurre para órdenes más elevados es que el coste computacional necesario para que el algoritmo converja en una solución aceptable es muy elevado. Esto es debido a que es necesario aumentar el tamaño de la población (NumFlor) y, por tanto, el número de iteraciones. Esto conlleva un aumento la diversidad y evita que el algoritmo se estanque en óptimos locales; sin embargo, el aumento de carga computacional es tal que no se aconseja su empleo.

Un campo de posible aplicación de los filtros FIR digitales es en el filtrado de señales biomédicas (Capítulo 7), donde, muchas veces, se requiere que los filtros empleados cumplan determinadas

características de linealidad de la fase en la banda de paso y de eliminación de bandas no deseadas.

La primera característica de fase lineal en la banda de paso, la cumplen los filtros diseñados debido a la naturaleza simétrica de los coeficientes impuesta en el trabajo. Sin embargo, en el Capítulo 7, se emplean filtros diseñados con el algoritmo FPA para el filtrado de una señal de electroencefalograma (EEG) real y se compara con las señales filtradas con filtros obtenidos mediante dos técnicas convencionales.

Uno de los usos de los filtros digitales con esta señal biomédica es el desarrollado en este trabajo; consiste en la realización de una serie de filtros paso bajo, paso banda y paso alto con el fin de obtener las componentes de las distintas bandas de frecuencia del EEG (alfa, tetha, beta…) para su posterior procesado.

Con el fin de determinar qué filtro realiza un mejor filtrado de la señal, se obtiene la potencia de la señal filtrada en la banda de interés y la potencia de la señal indeseada en el resto de frecuencias. Los resultados obtenidos muestran que el filtro diseñado mediante el algoritmo FPA atenúa las frecuencias no deseadas 5-10 veces más que los filtros diseñados con técnicas convencionales. En cuanto a la potencia de la señal en la banda deseada, el filtro diseñado mediante el algoritmo FPA atenúa ligeramente (entre 5-15%) la señal más que los filtros. Como conclusión; aunque nuestro filtro diseñado atenúa más la señal en la banda deseada, la supresión de potencia que se produce de la señal indeseada es mucho mayor, posicionándose como una clara alternativa para el filtrado de EEG.

En resumen, el algoritmo FPA junto con la función de aptitud múltiple empleados para el diseño de filtros FIR digitales obtiene unas características frecuenciales del filtro que mejoran las obtenidas con otros métodos de diseño. Si se compara con las técnicas convencionales, se obtienen una mayor atenuación de la banda de corte y menor ancho de banda de transición, aunque el rizado de la banda de paso es ligeramente superior. Si se compara con otras técnicas de computación natural, los resultados obtenidos son mejores en cuanto a atenuación de la banda de corte y muy similares en cuanto a rizado de la banda de paso y ancho de banda de transición.

9. Futuras líneas de trabajo

En esta sección se elaborarán dos posibles líneas de investigación para el futuro sobre las cuales poder trabajar para mejorar los resultados obtenidos en este trabajo. En primer lugar, se detallará un posible uso de las técnicas de computación natural para el filtrado adaptativo. En segundo lugar, la modificación del algoritmo y de la función de aptitud para solventar los problemas encontrados.

9.1. Filtrado Adaptativo

En el filtrado de señales digitales, los algoritmos naturalistas han resultado ser una buena alternativa para la síntesis de filtros FIR óptimos. Sin embargo, dentro del mundo del procesado

de señales hay más problemas a los que pueden dar solución. Se propone, para un futuro trabajo de investigación, el empleo del algoritmo FPA para la cancelación de ruido adaptativa.

La cancelación de ruido es un problema básico que tiene importantes aplicaciones en multitud de áreas como el procesado de voz, la cancelación de ecos, la mejora de señales, el procesado de arrays de antenas, señales biomédicas y el procesado de imagen. La cancelación de ruido consiste en la extracción de la señal deseada de una señal corrupta con mucho ruido cancelando dicho ruido. La Fig. 60 muestra la estructura de un cancelador de ruido adaptativo o filtro adaptativo.

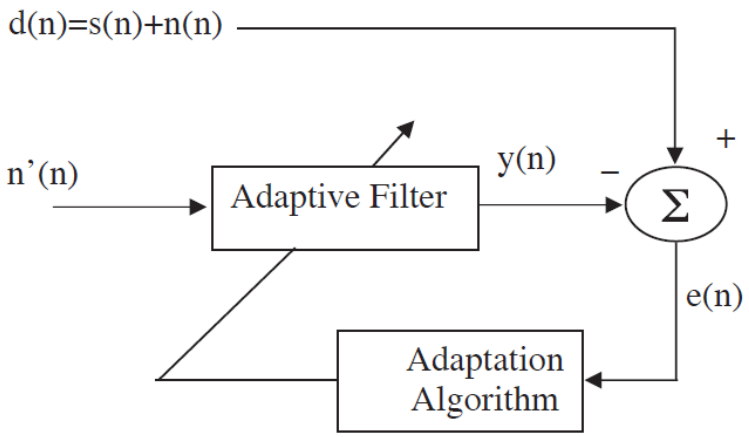

Figura 60. Cancelador de ruido adaptativo o Filtrado Adaptativo.

Como se puede ver en la Fig. 60, hay cuatro señales implicadas en este diseño: la señal de información **s**(n), la señal de ruido **n**(n) que corrompe a la señal deseada **d**(n) y la señal de referencia **n'**(n). Esta señal de referencia (señal de entrada) **n'**(n) es parecida a una versión correlada de la señal de ruido **n**(n). La señal de error **e**(n), que es igual a la diferencia entre la señal ruidosa **d**(n) y la salida del filtro adaptativo **y**(n).

El algoritmo de adaptación, en este caso sería el algoritmo FPA propuesto, se computa para ajustar los coeficientes del filtro adaptativo para minimizar la función de costes. Una posible función de costes es el error cuadrático medio (MSE) de la señal de error.

$$J(n) = \mathbf{E}[|e(n)|^2] \qquad (58)$$

En esta ecuación, **E** denota el valor esperado. La función de aptitud del algoritmo se puede calcular a partir de la Ec. (56); siendo i la iteración actual y J la función de costes.

$$fit(i) = \frac{1}{1 + J_i(w)} \qquad (59)$$

9.2. Mejorar las prestaciones del método empleado.

Los elementos fundamentales de las técnicas de computación natural para el diseño de filtros FIR que determinan la calidad de la solución encontrada son la función de aptitud que determina las características del filtro diseñado y el algoritmo empleado que determina la convergencia a una solución óptima.

Por esto, una posible futura línea de trabajo es la elaboración de distintas funciones de aptitud que traten de solventar los problemas encontrados en este trabajo. El principal problema que tiene la función de aptitud empleada en este trabajo es el elevado rizado de la banda de paso. Por ello, en una futura línea de trabajo se puede buscar una función de aptitud que no eleve tanto el rizado de la banda de paso sin sacrificar la atenuación de la banda eliminada. Otro de los factores que se puede tratar de solventar es que esta función de aptitud prioriza la atenuación de la banda eliminada en lugar del ancho de banda de transición.

Se ha observado que, si se desea obtener la respuesta al impulso del filtro eficiente para órdenes elevados, es necesario un considerable aumento del tamaño de población y de las iteraciones, lo que incrementa el coste computacional. Por ello, otro posible campo de actuación es el control de la diversidad del algoritmo y de la búsqueda en todo el espacio de soluciones (evitar estancamiento en óptimos locales) sin elevar tanto la carga computacional del algoritmo. Esto se puede conseguir realizando modificaciones al algoritmo FPA, empleando alguna hibridación con el objetivo de mejorar la diversidad y evitar quedarse atrapado en estas soluciones sub-óptimas.

10. Referencias

[1] Z. Milivojević, "Digital filter design", January 1, 2009, *MikroElektronika (1st ed)*.

[2] X. S. Yang, "Flower Pollination Algorithm for Global Optimization", *11th International Conference, UCNC 2012, Orléan, France, September 3-7, 2012. Proceedings,* pp. 240-249

[3] S. U. Ahmad, "Design of Digital Filters Using Genetic Algorithms", Thesis, (2008).

[4] A. Antoniou, "Digital Signal Processing: Signals, Systems, and Filters". *New Jersey: McGraw-Hill,* 2005.

[5] K. Boudjelaba, F. Ros and D. Chikouche, "An efficient Hybrid Genetic Algorithm to design FIR filters", 2014.

[6] S. Tsutsumi and K. Suyama, "Design of FIR Filters with Discrete Coefficients Using Ant Colony Optimization", *Electronics and Communications in Japan*, Vol. 97, No. 4, 2014.

[7] D. Suckley, "Genetic algorithm in the design of FIR filters," IEE Proc., pt. G, vol. 138, pp. 234-238, Apr. 1991.

[8] G. Wade, P. Van-Eetvelt and H. Darwen: "Synthesis of efficient low-order FIR filters from primitive sections". IEE Proc. G.1990,137, pp. 367-372

[9] D. E. Goldberg, "Genetic Algorithms in Search Optimization and Machine Learning". San Francisco: Addison-Wesley, 1989.

[10] D. E. Goldeberg: "Simple genetic algorithms and the minimal, deceptive problem", in DAVIS, L., (Ed.): "Genetic algorithms and simulated annealing" (Pitman, 1987).

[11] D. Karaboga and A. Kaplan (1995), "Optimising multivariable functions using tabu search algorithms", paper presented at the *Tenth Int. Symp. on Computer and Information Sciences*, Turkey Vol. II, pp. 793-799.

[12] K. Watcharasitthiwat, J. Koseeyaporn and P. Wardkein (2006). "Designing Digital FIR Filters Using Multiple Tabu Search Algorithm", *2006 International Conference on Communications, Circuits and Systems*, Vol. 1, pp. 171-175.

[13] A. Kalinli, N. Karaboga, "A parallel tabu search algorithm for digital filter design", *The international journal for computation and mathematics in electrical and electronic engineering*, Vol. 24 No. 4, 2005 pp. 1284 – 1298

[14] A. Aggarwal, T. K. Rawat, D. K. Upadhyay (2016), "Design of optimal digital FIR filters using evolutionary and swarm optimization techniques," *Int. J. Electron. Commun. (AEÜ)* 70 (2016), pp. 373–385

[15] R. V. Kacelenga, P. J. Graumann, and L. E. Turner, "Design of digital filters using simulated annealing," in Proc. IEEE Int. Symp. Circuits Syst. (New Orleans, LA), May 1990, pp. 642-645.

[16] H. Aguiar e Oliveira, A. Petraglia, and M. R. Petraglia, "Frequency Domain FIR Filter Design Using Fuzzy Adaptive Simulated Annealing", *2007 IEEE International Symposium on Signal Processing and Information Technology*, pp.884-888.

[17] L. Ingber, "Very fast simulated re-annealing", Mathl. Comput. Modeling, Vol. 12, pp. 967-973, Ago. 1989.

[18] H. A. Oliveira ," Fuzzy *control of stochastic global optimization algorithms and VFSR*", Naval Research Magazine, No. 16, pp. 103-113, Oct. 2003 (in Portuguese). Draft available at www.optimizationonline.org (in English).

[19] J. Pittman and C. A. Murthy, "Fitting optimal picewise linear functions using genetic algorithms," *IEEE Tarns. on Pattern Analysis Machine Intelligence*, vol. 22, no. 7, pp. 701–718, July 2000.

[20] J. H. Holland, '*Adaptation in Natural and Artificial Systems'*. Ann Arbor, MI: University of Michigan Press, 1975.

[21] K. D. Jong, "Hybrid methods using genetic algorithms for global optimization," *IEEE Trans. Systems Man Cybernatics*, vol. 10, no. 9, pp. 566 –574, September 1980.

[22] B. Luitel and G. K. Venayagamoorthy, "Differential Evolution Particle Swarm Optimization Digital Filter Design". *2008 IEEE Congress on Evolutionary Computation (CEC 2008)*, pp. 3954-3961.

[23] A. P. S. Neha, "Design of Linear Phase Low Pass FIR Filter using Particle Swarm Optimization Algorithm", *International Journal of Computer Applications (0975 – 8887) Vol.98*– No.3, July 2014, pp.40-44.

[24] Q. Zhao, G. Meng, "Design of Digital FIR Filters Using Differential Evolution Algorithm Based on Reserved Gene", *International Conference of Information Science and Management Engineering* in *2010,* pp. 177-180.

[25] S. K. Saha, S. P. Ghoshal, R. Kar, D. Mandal, "Cat Swarm Optimization algorithm for optimal linear phase FIR filter design", *ISA Transactions 52 (2013)* pp. 781–794

[26] M. Kumar, T. K. Rawat, "Optimal design of FIR fractional order differentiator using cuckoo search algorithm", *Expert Systems with Applications 42, 2015*, pp. 3433–3449.

[27] X.S. Yang and S. Deb, "Cuckoo Search via Levy Flights", Proceedings of IEEE World Conference on Nature and Biologically Inspired Computing (2009), pp. 210–214.

[28] Ap. Aggarwal, T. K. Rawat, D. K. Upadhyay, “Design of optimal digital FIR filters using evolutionary and swarm optimization techniques”, *International Journal of Electronics and Communications (AEÜ)* 70 (2016), pp. 373–385.

[29] S. K. Saha, R. Kar, D. Mandal, S. P. Ghoshal, “Bacteria foraging optimisation algorithm for optimal FIR filter design”, *International Journal of Bio-Inspired Computation · April 2013*, pp. 52-66.

[30] S. Das, B.K. Panigrahi and S. Pattnaik, “Nature-inspired algorithms for multi-objective optimization”, *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods and Techniques*, IGI Global (2009), Vol. 1, pp.95–104.

[31] T.W. Parks and J.H. McClellan (1972) “Chebyshev approximation for non-recursive digital filters with linear phase”, *IEEE Trans. Circuits Theory CT-19*, pp.189–194.

[32] N. Karaboga, B. Cetinkayal “Design of digital FIR filters using differential Evolution algorithm”. *Circuits Systems Signal Processing 2006*; pp. 649–660.

[33] M. Najjarzadeh, A. Ayatollahi, “FIR digital filters design: particle swarm optimization utilizing LMS and minimax strategies”. *IEEE International Symposium on Signal Processing and Information Technology*, ISSPIT 2008; pp. 129–32.

[34] J.I. Ababneh and M.H. Bataineh (2008) “Linear phase FIR filter design using particle swarm optimization and genetic algorithms”, *Digital Signal Processing*, Vol. 18, No. 4, pp.657–668

[35] H. Kaur and B. Dhaliwa “Design of Low Pass FIR Filter Using Artificial Neural Network*”, International Journal of Information and Electronics Engineering*, Vol. 3, No.2, March 2013, pp.204-207

[36] A. V. Oppenheim, R. W. Schafer, J. R. Buck, “Discrete-time signal processing (2nd ed.)”. *Prentice-Hall, Inc*. Upper Saddle River, NJ, USA 1999

[37] B. Gold and K. L. Jordan Jr., "A direct search procedure for designing finite duration impulse response filters", IEEE Trans. Audio Electroacoust. (1969), vol. AU-17, pp. 33-36.

[38] E. Hofstetter, A. V. Oppenheim and J. Siegel, "A new technique for the design of non-recursive digital filters", 5th Annu. Princeton Conf. Information Sciences and Systems, 1971.

[39] Y. C. Lim, J. H. Lee, C. K. Chen, and R. H. Yang, "A weighted least squares algorithm for quasi-equiripple FIR and IIR digital filter design", IEEE Trans. Signal Proces., vol. 40, no. 3, pp. 551-558, 1992.

[40] R. Storn and K. Price (1997), “Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces”, Journal of Global Optimization, 11, pp. 341–359.

[41] K. S. Reddya, S. K. Sahooa, “An approach for FIR filter coefficient optimization using differential evolution algorithm”, Int. J. Electron. Commun. (AEÜ) (2015), pp. 101–108.

[42] W. McCulloch, WH. Pitts, “A logical calculus of the ideas immanent in nervous activity”, *Bulletin of Mathematical Biophysics (1943),* pp. 115–33.

[43] K. Pachori, Dr. A. Mishra, “Design of FIR Digital Filters using ADALINE Neural Network”, *Fourth International Conference on Computational Intelligence and Communication Networks (2012)*, pp.800-803.

[44] L. Cen, “A hybrid genetic algorithm for the design of FIR filters with SPoT coefficients”, *Signal Processing*, 87 (2006), pp. 528–540

[45] Y. D. Jou, “Design of real FIR filters with arbitrary magnitude and phase specifications using a neural-based approach”, *IEEE Transactions on Circuits and Systems-II: Express Briefs*, Vol. 53, pp.1068-1072, Oct. 2006.

[46] N. Karaboga and B. Cetinkaya, “Design of Digital FIR Filters Using Differential Evolution Algorithm”, *Circuits System Signal Processing*, vol. 25, pp. 649-660D, 2006.

[47] S. Mondal, D. Chakraborty, R. Kar, D. Mandal and S.P. Ghoshal, “Novel Particle Swarm Optimization for Low Pass FIR Filter Design”, *WSEAS Transactions on Signal Processing*, vol. 8, pp. 111-120, 2012.

[48] A. Sarangi, R.K. Mahapatra and S.P. Panigrahi (2010) “DEPSO and PSO-QI in digital filter design”, *Expert Systems with Applications*, Vol. 38, No. 9, pp.10966–10973.

[49] H. Zhao & J. Yu, “A Novel Neural Network-based Approach for Designing Digital Filters”, *IEEE International Symposium on Ciruits and Systems*, vol. no.4, pp. 2272-2275, June 1997.

[50] Z. Z. Zeng, Y. Chen, Y. N. Wang, “Optimal design study of high-order fir digital filters based on neural-network algorithm”, *Fifth International Conference on Machine Learning and Cybernetics, Dalian*, 13-16 of August 2006, pp. 3157-3161.

[51] D. Karaboga, D. H. Horrocks, N. Karaboga, A. Kalinli, “Designing digital FIR filters using Tabu search algorithm”, *Circuits and Systems, 1997. Proceedings of 1997 IEEE International Symposium,* vol.4, pp. 2236 – 2239

[52] E. Nabil, “A Modified Flower Pollination Algorithm for Global Optimization”, *Expert Systems With Applications, 57,* 29 of March 2016*,* pp. 192–203

[53] I. Pavlyukevich, “Lévy flights, non-local search and simulated annealing”, *Journal of Computational Physics (2007),* 226, pp. 1830-1844.

[54] N. Karaboga, M. B. Centikaya, “A novel and efficient algorithm for adaptive filtering: artificial bee colony algorithm”, *Turk. J. Electr. Eng. Comput. Sci. 19* (2011), pp.175-190

[55] *https://www.physionet.org/physiobank/database/*

[56] A.E.W. Jones and G.W. Forbes, “An adaptive simulated annealing algorithm for global optimization over continuous variables”, *Journal of Global Optimization*, *January 1995, Volume 6, Issue 1*, pp 1–37

[57] Dr. S. Swift, “Algorithms and their Applications” CS2004 (2012-2013) available at http://slideplayer.com/slide/7798979/.

[58] X. S. Yang, “Flower pollination algorithm for global optimization”, *Unconventional Computation and Natural Computation 2012*, *Computer Science*, Vol. 7445, pp. 240-249 (2012).