Acta de Correcciones al Proyecto de Grado
Matemáticas Aplicadas

Fecha: Abril 13 del 2023

Autores:
Jan Polanco Velasco

Nombre del Proyecto de Grado: ANÁLISIS COMPARATIVO ENTRE OPTIMIZACIÓN CLÁSICA Y OPTIMIZACIÓN METAHEURÍSTICA APLICADO AL BEAMFORMING ADAPTATIVO DE ARREGLOS PLANOS MIMO MASIVO.

Director:
Abel Alvarez Bustos

Codirector
Dimas Mavares

Como indica el artículo 2.27 de las Directrices de Trabajo de Grado, he verificado que los estudiantes indicados arriba han implementado todas las correcciones que los Jurados del Proyecto de Grado definieron que se efectuaran, como consta en el Acta de Calificación correspondiente.

Firma Directora del Proyecto de Grado

Nota de Aceptación

Aprobado por el Comité de Trabajo de Grado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana para optar el título de Profesional en Matemáticas Aplicadas.

Hernán Camilo Rocha Niño
Decano de la Facultad de Ingeniería y Ciencias

Andrés Felipe Amador Rodriguez
Directora Carrera de Matemáticas Aplicadas

Abel Álvarez Bustos
Director(a) Trabajo

Carlos Ramírez Ovalle
Jurado 1

Antonio Sánchez
Jurado 2

Santiago de Cali, .

Señores
Pontificia Universidad Javeriana Cali.
Dr. Andrés Felipe Amador Rodriguez
Directora de Carrera de Matemáticas Aplicadas.
Cali.

Por medio de la presente confirmamos que hemos revisado el proyecto de grado titulado **"Análisis comparativo entre optimización clásica y optimización metaheurística aplicado al Beamforming adaptativo de arreglos planos MIMO Masivo."**, escrito por el estudiante Jan Polanco Velasco del programa de Matemáticas Aplicadas identificado con el código: 2336323 y que en calidad de director y codirector aprobamos que sea entregado para su evaluación.

Gracias.

Atentamente,

Dr. Abel Álvarez Bustos

Dr. Dimas Mavares Terán

Santiago de Cali, .

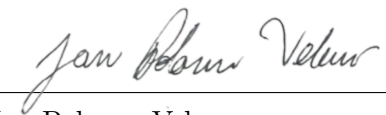
Señores
Pontificia Universidad Javeriana Cali.
Dr. Andrés Felipe Amador Rodriguez
Directora de Carrera de Matemáticas Aplicadas.
Cali.

Cordial Saludo.

Me permito presentar a su consideración el proyecto de grado titulado **"Análisis comparativo entre optimización clásica y optimización metaheurística aplicado al Beamforming adaptativo de arreglos planos MIMO Masivo."** con el fin de cumplir con los requisitos exigidos por la Universidad para optar por el título de Profesional en Matemáticas Aplicadas.

Al firmar aquí, doy fe que entiendo y conozco las directrices para la presentación de trabajos de grado de la Facultad de Ingeniería y Ciencias aprobadas el 26 de Noviembre de 2009, donde se establecen los plazos y normas para el desarrollo del anteproyecto y del trabajo de grado.

Atentamente,



Jan Polanco Velasco
Código: 2336323

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias.
Matemáticas Aplicadas.
Anteproyecto de Grado.

Análisis comparativo entre optimización clásica y optimización metaheurística aplicado al Beamforming adaptativo de arreglos planos MIMO Masivo.

Jan Polanco Velasco

Director: Dr. Abel Álvarez Bustos
Codirector: Dr. Dimas Mavares Terán

28 de marzo de 2023

Abstract

In recent years, wireless networks have experienced exponential growth due to increased research and new technologies in both hardware and software. This makes it increasingly easy to acquire devices that connect instantly to the network. However, this increase in devices poses several problems and challenges for the network, such as limited or non-existent coverage in rural or remote areas, signal interference due to electromagnetic interference or densely populated urban environments, data privacy, and poor quality of service offered by mobile operators. To address these issues, the industry and scientific community are investigating various techniques, with Beamforming being the most suitable strategy in the context of fifth and sixth generation mobile telephony.

This thesis performs a comparative analysis between classic optimization algorithms such as the Conjugate Gradient Method (CGM), Stochastic Gradient Descent (SGD), and Nelder-Mead Search (NMS), and metaheuristic algorithms such as Particle Swarm Optimization (PSO), Bat Algorithm (BA), and Cuckoo Search by Lévy Flights (CKLF). For this analysis, different test functions were selected in different dimensions and metrics. Stop criteria, maximum number of iterations, and success rate were established. In addition, the convergence order p was analyzed through the analysis of the Root Mean Squared Error time series.

The work also considers the Blind Adaptive Beamforming model, where information about the orientation of desired and interfering signals is not available to the algorithm. An antenna with a flat rectangular geometry of 64 radiating elements was used, and metrics such as the average power bandwidth in degrees, the radiation intensity of the main lobe, the depth of the first nulls, and the level of the side lobes were implemented, all in decibels. The results aim to improve spectral efficiency and service quality.

Keywords:: Wireless networks, Beamforming, Optimization algorithms, Blind Adaptive Beamforming, Root Mean Squared Error.

Resumen

En los últimos años, las redes inalámbricas han experimentado un crecimiento exponencial debido al aumento de investigaciones y nuevas tecnologías tanto en hardware como en software. Esto hace que sea cada vez más fácil adquirir dispositivos que se conecten instantáneamente a la red. Sin embargo, este aumento de dispositivos plantea varios problemas y desafíos para la red, como la cobertura limitada o inexistente en áreas rurales o remotas, interferencia en la señal debido a la interferencia electromagnética o a entornos urbanos densamente poblados, privacidad de los datos transmitidos y mala calidad del servicio ofrecido por los operadores móviles. Para abordar estos problemas, la industria y la comunidad científica están investigando diversas técnicas, siendo el Beamforming la estrategia más adecuada en el contexto de la telefonía móvil de quinta y sexta generación.

Este trabajo de grado realiza un análisis comparativo entre algoritmos clásicos de optimización, como el *Conjugate Gradient Method (CGM)*, *Stochastic Gradient Descent (SGD)* y *Nelder-Mead Search (NMS)*, y algoritmos metaheurísticos como *Particle Swarm Optimization (PSO)*, *Bat Algorithm (BA)* y *Cuckoo Search by Lévy Flights (CKLF)*. Para este análisis, se seleccionaron diferentes funciones de prueba en diferentes dimensiones y métricas. Se establecieron criterios de parada, máximo número de iteraciones y tasa de éxito. Además, se analizó el orden de convergencia p a través del análisis de la serie de tiempo del error *Root Mean Squared Error*.

En el trabajo, se también consideró el modelo de *Beamforming* Adaptativo Ciego, donde la información de la orientación de las señales deseadas e interferentes no está disponible para el algoritmo. Se utilizó una antena con una geometría rectangular plana de 64 elementos radiantes y se implementaron las métricas como el ancho de banda de la potencia media en grados, la intensidad de radiación del lóbulo principal, la profundidad de los primeros nulos y el nivel de los lóbulos laterales todas estas en decibeles. Los resultados buscan mejorar la eficiencia espectral y la calidad del servicio.

Palabras Clave: *Beamforming* Adaptativo, redes inalámbricas, procesamiento de señal, algoritmos de optimización, CGM, SGD, NMS, PSO, BA, CKLF, funciones de prueba, métricas, errores, criterios de parada, número máximo de iteraciones, tasa de éxito, orden de convergencia, *Singular Spectrum Analysis (SSA)*.

Índice general

Índice de figuras

Índice de cuadros

List of Algorithms

Introducción

En los últimos años las redes inalámbricas han presentado un crecimiento exponencial, esto relacionado con el incremento de investigaciones y de nuevas tecnologías tanto a nivel de *hardware* y *software* que hacen que cada vez sea más fácil adquirir dispositivos que de forma instantánea se agregan la red. Este incremento de dispositivos representa varios problemas y desafíos para la red, en los cuales se destacan las cobertura limitada o inexistente como lo es en el caso de algunas áreas rurales o remotas, interferencia de la señal que puede estar relacionada con alguna interferencia electromagnética o entornos urbanos densamente poblados que pueden afectar la calidad de la señal y la capacidad de la red, protección de privacidad relacionada con los datos transmitidos por las redes y en general una mala calidad del servicio ofrecida por los operadores móviles. Para mitigar este problema la industria y la comunidad científica están realizando investigaciones entre las cuales se destaca el *Beamforming* como la estrategia más adecuada en el contexto de la telefonía móvil celular de quinta y sexta generación [22].

El *Beamforming* es una técnica de procesamiento de señal que permite enfocar la señal de transmisión de manera precisa hacia los dispositivos móviles. Esto resulta en una mejor calidad de la señal y una reducción de los errores de transmisión. Al enfocar la señal hacia los dispositivos móviles deseados y evitar los interferentes, el *Beamforming* adaptativo permite una mejor utilización del espectro radioeléctrico y una mayor capacidad de la red, lo que resulta en una mayor cantidad de usuarios y servicios que pueden ser soportados en la red. En el sector energético, el Beamforming permite una reducción del consumo de energía porque mejora la calidad de la señal y aumenta la capacidad de la red, lo que implica una reducción en el consumo de energía de los dispositivos móviles, una mayor duración de la batería y una menor cantidad de emisiones de carbono. [9].

El presente trabajo de grado consideró un análisis comparativo entre algoritmos de optimización clásica ampliamente usados en la literatura como lo es el caso de *Conjugate Gradient Method (CGM)*, *Stochastic Gradient Descent (SGD)* y *Nelder-Mead Search(NMS)* y Algoritmos Metaheurísticos (AM) como *Particle Swarm Optimization (PSO)*, *Bat Algorithm (BA)* y *Cuckoo Search by Lévy Flights (CKLF)*, este análisis consideró una selección de funciones prueba en distintas dimensiones ($\mathbb{R}^2$ y $\mathbb{R}^{10}$), distintas métricas como *Mean Squared Error (MSE:)*, *Mean Absolute Error (MAE)*, etc. Se estableció criterios de parada para las funciones de prueba considerando perturbaciones en las soluciones, MSE, máximo número de iteraciones y tasa de éxito por ejecución. Por último, se analizó el orden de convergencia p, para esto se consideró las curvas del error *Root Mean Squared Error (RMSE:)* y para obtener el valor p se realizó una descomposición de la serie de tiempo de cada curva RMSE y se calculó la tendencia a través del algoritmo *Singular Spectrum Analysis (SSA*.

Para el caso del *Beamforming* se consideró el modelo adaptativo ciego, en el cual la información sobre la orientación de las señales deseadas e interferentes no está disponible para para el algoritmo, los algoritmos solo tienen a su disposición una función objetivo y un proceso iterativo para

encontrar el óptimo. Para este análisis, se consideró una antena con geometría rectangular plana de 64 elementos radiantes de dimensión 8×8, se implementó el criterio de parada máximo número de iteraciones por la dificultad del problema y las métricas empleadas en los diagramas de radiación son la intensidad de radiación del lóbulo principal en decibeles, el ancho de banda de la potencia media del lóbulo principal en grados, la profundidad de los primeros nulos del lóbulo principal en decibeles y el nivel de los lóbulos laterales también en decibeles. Estos resultados obtenidos son importantes porque buscan mejorar la eficiencia espectral y a su vez mejorar la calidad del servicio *(Quality of Service QoS)*.

En el año 2022 se realizaron 6960 investigaciones relacionadas con *Beamforming* en Google Académico y diariamente salen nuevos artículos que buscan mejorar la eficiencia espectral, la calidad del servicio, etc. Con estas investigaciones el principal desafío está relacionado con minimizar las interferencias y como se puede optimizar la capacidad de los sistemas de comunicaciones inalámbricos. Para esto el *Beamforming* ha ganado un reconocimiento extraordinario en la telefonía móvil celular, micrófonos, SONAR, GPS y sistemas de radar modernos A pesar de sus ventajas en el cálculo del vector de pesos óptimo y la estimación de la dirección de llegada de las señales, estas técnicas son difíciles de aplicar en entornos cambiantes, donde los emisores se mueven y los pesos deben ser calculados continuamente.

En la actualidad las investigaciones se orientan con aplicaciones de *Machine Learning*, *Deep Learning* para los problemas más realistas del *Beamforming* analizando métricas de desempeño como tasas de convergencia, potencia de las señales deseadas, detecciones de fallos y respuestas en tiempo real. Por otro lado, los algoritmos de optimización clásicos y metaheurísticos son la base para modelos más avanzados que permiten mejorar el *Beamforming* y maximizar la eficiencia espectral de los sistemas de comunicación inalámbricos y también ser un insumo para el entrenamiento de modelos basados en inteligencia artificial.

El presente trabajo de grado está organizado de la siguiente forma. En la primera parte se revisaron los antecedentes más importantes y significativos para la investigación. Se revisó el factor de arreglo con geometría rectangular plana MIMO Masivo, de dimensión 8×8 y sus restricciones para la implementación. Se reviso el esquema de *Beamforming* Adaptativo y su diagrama de bloques identificando las variables como la señal deseada, las señales interferente, la señal de referencia y el algoritmo adaptativo. Se definieron las 4 funciones de prueba, con sus características, su dominio y el valor teórico de sus óptimos y al final se revisan los algoritmos, revisando su historia, los parámetros establecidos y el pseudocódigo.

En la segunda parte se identificaron las métricas a usar y su distinta tipología, los criterios de parada implementados en los códigos y el orden de convergencia p a través de un análisis de serie de tiempo aditiva, luego se realizó el diseño del *Beamforming* Adaptativo teniendo cuenta las restricciones asociadas al problema, identificando la función multiobjetivo, la cantidad de señales a considerar y los parámetros que debe tener cada algoritmo para tener un buen desempeño. Los

resultados se dividen en dos partes, la primera asociada a las funciones de prueba donde el ganador es el algoritmo CKLF con la mejor tasa de éxito en las dos dimensiones y el segundo resultado asociado al *Beamforming* Adaptativo donde el algoritmo CKLF tiene el mejor desempeño en las distintas métricas empleadas.

Capítulo 1

Descripción del Problema

1.1. Definición del problema

En la actualidad las telecomunicaciones son de vital importancia para la sociedad, la necesidad de estar siempre conectados ha generado nuevas dinámicas en la sociedad afectando la economía, la educación, salud y en general todo lo que nos rodea. Desde mensajes de texto, llamadas, redes sociales, servicios de *streaming* de audio y vídeo como es el caso de Spotify y Netflix pasando por comunicaciones Humano/Maquina o Maquina/Maquina con el internet de las cosas.

Estos cambios en el paradigma de las telecomunicaciones han traído consigo nuevos retos en las telecomunicaciones, de los cuales se destacan una mayor cantidad de dispositivos conectados a la red. Al término del primer trimestre del 2021 según las cifras del ministerio de las TIC, el total de líneas de telefonía móvil celular alcanzo 69,4 millones lo que representa un incremento de más de 2.9 en el mismo trimestre del año anterior, esto a su vez representa que por cada 100 colombianos existen 136 líneas de telefonía móvil [6].

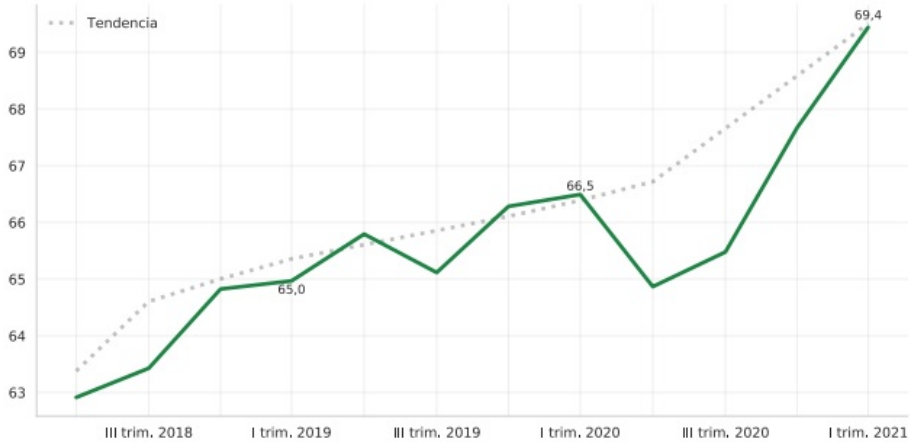


Figura 1.1: Datos reportados por los proveedores de redes y servicios a Colombia TIC. *fuente:[6]*.

Estas cifras denotan que las telecomunicaciones en el país ya sobrepasaron la población colombiana y este sobredimensionamiento de la población representa nuevos retos para las empresas de telefonía móvil, desde aumento de la infraestructura, nuevas tecnologías, compra de más espectro radioeléctrico, optimización del espectro vigente (eficiencia espectral), velocidad de trasmisión

throughput, etc. Todo esto para aumentar la transmisión de datos inalámbrica de alta velocidad, capacidad de la red y mejorar la calidad del servicio a los usuarios.

En el presente trabajo de grado se abordará la eficiencia espectral a través del Beamforming adaptativo. El *Beamforming* usa un arreglo de antenas para controlar el diagrama de radiación de una onda electromagnética (ver Figura 2.2), considerando las señales entrantes, interferentes y ruido que ingresan al arreglo. Se calcula la diferencia entre la señal de salida y la señal de referencia y a través de los algoritmos adaptativos se realiza una ponderación adecuada de la magnitud y la fase de cada elemento de la antena.

Para realizar el *Beamforming Adaptativo* se establece un arreglo de antenas bajo cierta geometría (linear, circular, plano, etc.). El *Beamforming* proporciona una mejor área de cobertura a ciertas áreas en los límites de la celda, aumenta la calidad de la señal al usuario y reduce la interferencia entre celdas dado que cada antena del arreglo hace su contribución a la señal dirigida mejorando la eficiencia espectral.

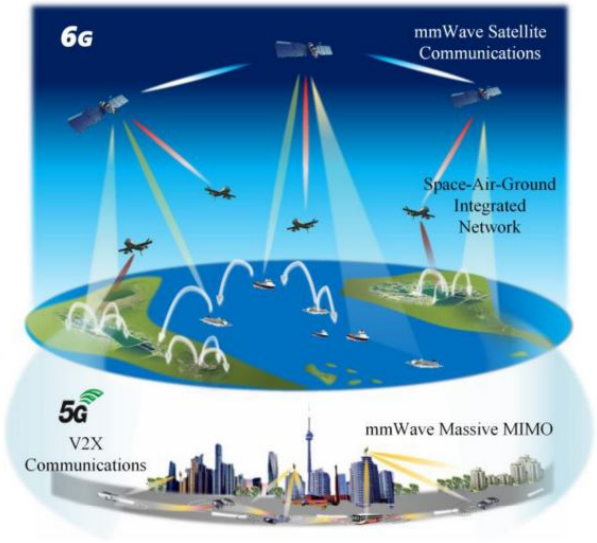


Figura 1.2: Esquemas de Telecomunicaciones de 5G y 6G. *fuente:[8]*.

Aunque se habla del *Beamforming* desde 1970, hace muy poco empezó a ser considerado en el problema de la eficiencia espectral y en el desarrollo de la nueva telefonía móvil de quinta generación y sexta generación, el problema del *Beamforming* se debe a: cantidad de receptores (usuarios), desplazamiento de los receptores y expresiones matemáticas complejas como el *factor de arreglo* por su dimensionalidad, separabilidad y escalabilidad, es decir, esto implica que a mayor número de elementos del arreglo de antenas implica una dimensión mayor del problema (mayor cantidad

de magnitudes y fases, por ejemplo una antena de 256 elementos, implica un problema de 256 dimensiones), esto ha generado que estas tecnologías sean muy costosas para implementarlas a nivel comercial por tal motivo se requiere potencia computacional y algoritmos de optimización ya sean tradicionales o metaheurísticos [20].

En este trabajo de grado se considerará una de las representaciones más básicas el *Beamforming Adaptativo*, que a su vez se puede abordar desde dos frentes, desde una perspectiva de optimización clásica: determinando sus condiciones suficientes y necesarias, basados en métodos analíticos y en el cálculo diferencial para encontrar los puntos óptimos de la función. Por el otro lado desde una perspectiva de optimización metaheurística: que es una estrategia heurística de solución de ensayo y error, para encontrar soluciones buenas y factibles a problemas complejos en tiempos razonablemente prácticos [25]. Los algoritmos de optimización han sido la forma que se ha considerado para realizar la optimización del *Beamforming* y sus distintas configuraciones, donde se han considerado desde enfoques clásicos (gradiente, mínimos cuadrados, etc.) hasta algoritmos heurísticos, inspirados en la biología [25] y aunque no se puede garantizar las mejores soluciones, se pueden considerar algoritmos eficientes que se ajusten al problema y a su complejidad. Además, que sean capaces de reproducir soluciones de buena calidad [25].

1.2. Objetivos

1.2.1. Objetivo General

- Comparar los algoritmos de optimización en el problema del Beamforming adaptativo de un arreglo plano de antenas 8×8 MIMO masivo.

1.2.2. Objetivos Específicos

- Identificar características, propiedades y variables a considerar dentro del análisis comparativo entre Optimización Clásica y Optimización metaheurística.
- Establecer los cálculos necesarios de la función objetivo para verificar los diagramas de radiación de arreglos planos.
- Identificar características relevantes de cada algoritmo como; tiempos de ejecución, tasas de éxito, convergencia, etc y recursos computacionales necesarios para ejecutar el algoritmo y obtener una solución óptima o cercana a la óptima para un problema específico.
- Implementar características relevantes de cada algoritmo respecto al *Beamforming Adaptativo* como diagramas de radiación optimizados, HPBW, Iteraciones, profundidad de los nulos, etc.
- Diseñar experimento del *Beamforming adaptativo* para analizar el desempeño de cada uno de los algoritmos.

- Realizar análisis estadístico sobre los resultados obtenidos de cada algoritmo de optimización.

Capítulo 2

Desarrollo del proyecto

2.1. Marco Teórico

2.1.1. Antecedentes

Para desarrollar el marco teórico del trabajo de grado fue necesario definir los antecedentes, es decir, aquellas investigaciones que fueron relevantes como punto de partida para el trabajo de grado, a continuación, se detalla la forma de cómo se consultó esta información, en que se diferenció de los antecedentes del anteproyecto y cuál fue el aporte más relevante para el trabajo grado. Primero se consultó a través de Google académico y se utilizó como base de estudio los libros *"Engineering Optimization: An Introduction with Metaheuristic Applications"* de Xin-She Yang, con 1138 citas, este libro es uno de los más importantes actualmente en el área de optimización metaheurística con información relevante para PSO. "Antenna Theory: Analysis and Design" de Constantine A. Balanis, con 26644 citas el cual ofrece información sobre el *Beamforming* Adaptativo de arreglos lineales con filtros y algoritmos adaptativos como *Least Mean Squares (LMS)* y "Cuckoo Search via Lévy Flights" también del profesor Xin-She Yang, con 3477 citas. Se consideró una ventana temporal del 2016 al 2022 y se realizaron filtros de búsqueda utilizando palabras clave como: *adaptative Beamforming*, *metaheuristics*, *algorithm*, CGM, SGD, NMS, PSO, BA y CKLF.

- **Beamforming de arreglos planos MIMO Masivo usando optimización metaheurística**, realizado por Jan Polanco Velasco (Autor), en el año 2022. Este trabajo de grado considera un tipo de *Beamforming* usando cinco algoritmos metaheurísticos, entre ellos se distingue: Optimización de enjambre de partículas (Particle Swarm Optimization (PSO)), algoritmo del Murciélago (Bat Algorithm (BA)) y el algoritmo del Cuco con vuelos de Levy (Cuckoo Search vía Levy Flights (CKLF)). Para validar el perfecto funcionamiento de los algoritmos se implementaron funciones de prueba y se verificaron sus tasas de éxito y demás estadísticos. Lo que pretende el trabajo de grado es realizar el Beamforming de una antena en condiciones 5G MIMO Masivo con tres escenarios: Usuarios estáticos, móviles y mixtos. Este antecedente es de vital importancia porque sirve de insumo y provee los tres algoritmos metaheurísticos: PSO, BA y CKLF y códigos referentes al factor de arreglo URA [22].

- **Performance Analysis of Adaptive Beamforming using Particle Swarm Optimization** realizado por Smita Banerjee y Ved Vyas Dwivedi, en el año 2016. Este articulo analiza el desempeño de PSO en el *Beamforming* adaptativo en una antena con geometría lineal, al final analiza la cantidad de decibeles en el máximo obtenido por la señal deseada y los nulos

de las señales interferentes. Este antecedente es de vital importancia porque ofrece la función objetivo y la metodología del experimento del Beamforming [3].

- **A comparison between different adaptive beamforming techniques** realizado por Rajen Kumar Patra y Chinmay Kumar Nayak, en el año 2019. Este articulo compara distintas técnicas clásicas para realizar el *Beamforming* adaptativo con señales moduladas, entre las cuales se destacan *Least Mean Square (LMS)*, *Recursive Least Square (RLS)*, *Sample Matrix Inversion (SMI)* y *Conjugate Gradient Method (CGM)*. En los resultados plantean la complejidad computacional a través del número de elementos de la antena y la cantidad de operaciones involucradas en los algoritmos, la estrategia para realizar la comparación consiste en analizar el número de iteraciones, el ancho de banda en grados, el nivel de los lóbulos laterales y la profundidad de los nulos en decibeles. Este antecedente es importante porque permite descartar algoritmos clásicos que se consideraron en el anteproyecto como es el caso del filtro MSI y el algoritmo NLMS, pero de acuerdo con recomendaciones del director del trabajo de grado, el mejor ajuste fue considerar un algoritmo derivativo determinístico, en este caso CGM, otro derivativo pero estocástico como lo es SGD y uno no derivativo como lo es NMS [16].
- **Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting and Prognostics: Properties and Typology** realizado por Alexei Botchkarev, en el año 2018. Este articulo analiza el desempeño de distintas métricas que son utilizadas en *Machine Learning*, para esto desarrolla una tipología considerando la distancia, normalización y agregación. Este artículo fue de vital importancia como antecedente para resolver el problema del *Beamforming* Adaptativo y las funciones de prueba, ya que proporciona una comprensión detallada de las métricas más adecuadas lo que permitió mejorar la evaluación del desempeño [4].

En general estos antecedentes nos indican como se debe hacer la investigación del trabajo de grado, teniendo en cuentas aspectos clave como la función objetivo, los algoritmos a implementar, las métricas a considerar, el análisis de convergencia de los algoritmos y los resultados obtenidos por los mismos en los dos escenarios planteados: Funciones de prueba en $\mathbb{R}^2$ y $\mathbb{R}^{10}$ y el problema del *Beamforming* Adaptativo en $\mathbb{R}^{64}$.

Aunque hay información disponible sobre el *Beamforming*, todavía se requiere más investigación para conocer en profundidad la implementación y las características más relevantes de la función objetivo, los AM y sus respectivas configuraciones, la construcción de los experimentos y las métricas para evaluar el desempeño de los AM en el contexto del *Beamforming*.

2.1.2. Factor de Arreglo

Para comprender el problema del Beamforming Adaptativo en un arreglo de antenas de geometría rectangular, primero tenemos que entender la expresión del factor de arreglo. Basados en la figura 2.1, se considera la unión de M arreglos lineales, cada uno de los cuales tiene N elementos

en el plano x-y. Este arreglo rectangular es de tamaño $M \times N$, y está definido por dos ángulos: el ángulo Azimutal θ y el ángulo de Elevación ϕ.

En la figura 2.1 se presenta además información sobre el arreglo, considerando tres variables críticas de construcción: su altura (*height* h), ancho (*width* W), y longitud (*length* L) de la microcinta. Cabe aclarar que un arreglo rectangular no necesariamente tiene que estar hecho de microcinta con parches rectangulares; se pueden hacer arreglos considerando otros tipos de antenas, por ejemplo, dipolos, entre otros [2].

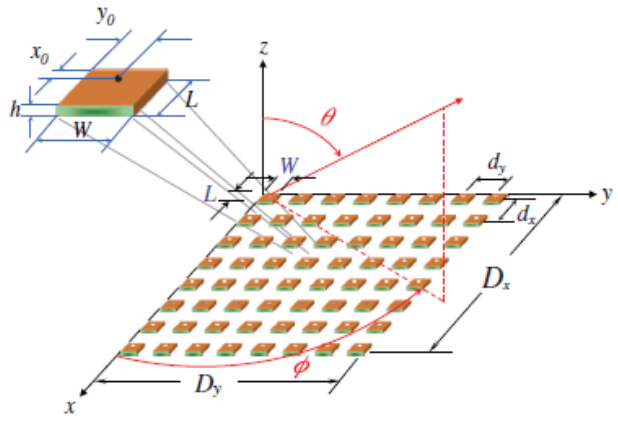


Figura 2.1: Diseño de un Arreglo 8x8 con parches rectangulares *fuente: [2].*

Para construir la expresión matemática del arreglo rectangular vamos a considerar información adicional en términos del *Beamforming* que puede ser usada para controlar la forma del patrón de radiación del arreglo [2]. Iniciamos el planteamiento con un arreglo lineal de M elementos sobre el eje X, basados en [2], sabemos que el factor de arreglo se define como:

$$AF_x = \sum_{m=1}^{M} a_{m1} e^{j(m-1)(kd_x \sin\theta\cos\phi+\beta_x)}, \tag{2.1}$$

donde a_{m1} es el coeficiente de excitación de cada elemento en x, d_x es la distancia entre elementos en x, β_x el cambio de fase progresivo entre los elementos de x, M es la cantidad de elementos del arreglo lineal y k el número de onda, $2\pi/\lambda$. Si definimos N arreglos lineales ubicados de forma consecutiva en dirección del eje y la ecuación (2.1) tenemos:

$$AF = \sum_{n=1}^{N} b_{1n} \left[\sum_{m=1}^{M} a_{m1} e^{j(m-1)(kd_x \sin\theta\cos\phi+\beta_x)} \right] e^{j(n-1)(kd_y \sin\theta\sin\phi+\beta_y)}, \tag{2.2}$$

donde b_{1n} es el coeficiente de excitación de cada elemento en y, d_y es la distancia entre elementos en y, β_y el cambio de fase progresivo entre los elementos en y y N la cantidad de elementos del

arreglo en la dirección y. Cuando la distancia entre elementos es mayor o igual a $\lambda/2$, se obtienen varios lóbulos de igual magnitud, el primero se le llama lóbulo principal y a los demás se les conoce como lóbulos de rejilla. Las fases progresivas β_x y β_y son independientes entre ellas:

$$\beta_x = -kd_x \sin\theta_0 \cos\phi_0, \qquad \psi_x = (kd \sin\theta \cos\phi + \beta_x), \tag{2.3}$$

$$\beta_y = -kd_y \sin\theta_0 \sin\phi_0, \qquad \psi_y = kd \sin\theta \sin\phi + \beta_y. \tag{2.4}$$

El factor de arreglo plano lo podemos expresar como $AF = AF_x \cdot AF_y$ y la distancia entre elementos como $d_x = d_y = d$:

$$AF_x = \sum_{m=1}^{M} I_{m1} e^{j(m-1)(kd \sin\theta \cos\phi + \beta_x)}, \tag{2.5}$$

$$AF_y = \sum_{n=1}^{N} I_{1n} e^{j(n-1)(kd \sin\theta \sin\phi + \beta_y)}. \tag{2.6}$$

El factor de arreglo será:

$$AF = \sum_{m=1}^{M} a_{m1} e^{j(m-1)(kd \sin\theta \cos\phi + \beta_x)} \sum_{n=1}^{N} b_{1n} e^{j(n-1)(kd \sin\theta \sin\phi + \beta_y)}, \tag{2.7}$$

$$AF = \sum_{m=1}^{M} \sum_{n=1}^{N} W_{mn} e^{j[(m-1)(kd \sin\theta \cos\phi) + (n-1)(kd \sin\theta \sin\phi)]}, \tag{2.8}$$

donde $W_{mn} = \sum_{m=1}^{M} \sum_{n=1}^{N} a_{m1} \cdot b_{1n} \cdot e^{j[(m-1)\beta_x + (n-1)\beta_y]}$ es la matriz de pesos complejos de cada elemento $m-n$ del arreglo rectangular. Considerando los términos ψ_x y ψ_y podemos simplificar la ecuación (2.8) de la siguiente forma:

$$AF = W_{mn} \sum_{m=1}^{M} \sum_{n=1}^{N} e^{j[(m-1)\psi_x + (n-1)\psi_y]}. \tag{2.9}$$

La validación indirecta del factor de arreglo en coordenadas cilíndricas y esféricas y su implementación en *MATLAB* se realizó en [22] considerando diferente cantidad de elementos (cinco y ocho), distancia entre elementos $\lambda/4, \lambda/2$ y λ y distintos valores para θ_0 y ϕ_0 lo que indicó que el código del factor de arreglo implementado es correcto y corresponde a lo planteado en el libro de *Balanis* [22].

La ecuación (2.9) es la que vamos a usar en el *Beamforming adaptativo* dentro del problema de optimización sin restricciones donde al final se obtendrá la matriz de pesos W_{mn} adecuada con los distintos tipos de algoritmos (clásicos o metaheurísticos) que maximiza la energía a cierto lugar deseado y/o minimiza la energía en direcciones no deseadas.

2.1.3. Beamforming Adaptativo

El *Beamforming* se ha convertido en tecnología clave para los modernos sistemas de comunicación inalámbrica y en los últimos años ha sido objeto de un intenso campo de estudio especialmente en las redes de 5G y 6G donde es necesaria la capacidad de direccionar la energía hacia cierta ubicación deseada, para esto se considera un Arreglo de antenas (véase ecuación (2.9)) y ciertas técnicas de procesamiento para direccionar el haz [1].

Los sistemas de antenas adaptativas son ampliamente usados en sistemas de antenas inteligentes para comunicaciones móviles inalámbricas y sistemas de radar de defensa. El Beamforming considera las señales entrantes al arreglo y orienta el haz hacia la dirección de llegada (Direction Of Arrival (DOA)), generando los pesos de cada elemento de la antena para que se pueda orientar a la dirección deseada y ubicar los nulos sobre las señales interferentes [1].

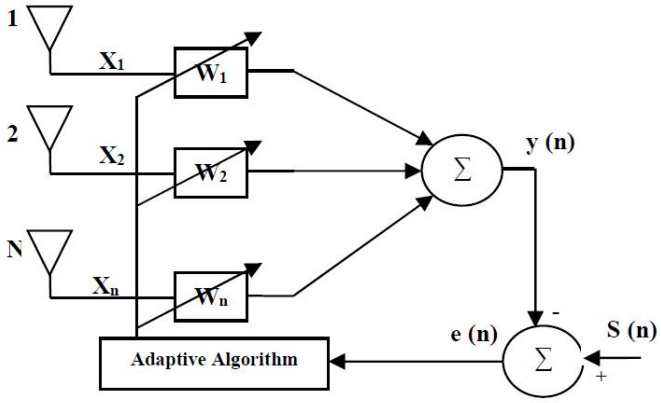


Figura 2.2: Diagrama de bloques de sistema de Antena Adaptativa *fuente: [11]*.

Del diagrama de bloques de la figura (2.2) se puede establecer el concepto del *Beamforming adaptativo*, para esto considere una señal entrante deseada $s(k)$ de longitud k y con ciertos ángulos de incidencia (θ_{0s}, ϕ_{0s}) y V señales no deseadas de longitud k ciertos ángulos de incidencia (θ_{1i}, ϕ_{1o}) al arreglo de N elementos. La señal recibida x_{mn}, se expresa de la siguiente forma:

$$x = a_0 s(k) + \begin{bmatrix} a_1 & a_2 & \cdots & a_V \end{bmatrix} + n(k), \tag{2.10}$$

$$x = a_0 s(k) + Ai(k) + n(k), \tag{2.11}$$

$$a_v = \begin{bmatrix} 1 & e^{j[(m-1)(kd\sin\theta_v\cos\phi_v)+(n-1)(kd\sin\theta_v\sin\phi_v)]} & \cdots & e^{j[(M-1)(kd\sin\theta_v\cos\phi_v)+(N-1)(kd\sin\theta_v\sin\phi_v)]}, \end{bmatrix}^T \tag{2.12}$$

donde a_v es el vector dirección del arreglo con $v = 0, 1, \cdots, V$ y A es la matriz dirección del arreglo de tamaño $(M \times N) \times V$. A la señal entrante la multiplicamos por un vector de pesos inicial w, donde $w_1, w_2, \cdots, w_{m \times n}$ corresponden a los pesos complejos del factor de arreglo (véase ecuación (2.9)), con esta información creamos $y(k)$ como la suma de todos los productos entre la señal de entrada y el vector de pesos. Por último, se establece la diferencia entre $y(k)$ y la señal deseada o señal de referencia $d(k)$ definiendo el error $e(k)$. El error ingresa al algoritmo Adaptativo (clásico o metaheurístico) el cual busca minimizar el error y/o otra función objetivo (optimización multiobjetivo) repitiendo este proceso de forma iterativa hasta cumplir cierto criterio de parada.

$$y(k) = w^H x(k), \tag{2.13}$$
$$e(k) = s(k) - w^H x(k), \tag{2.14}$$
$$\tag{2.15}$$

donde w es el vector de pesos complejos, H es el transpuesto conjugado complejo y $x(n)$ es la señal entrante/recibida. Para las simulaciones se considera el *Beamforming adaptativo* de geometría de arreglo plano 8×8 y distancia entre elementos $d = \lambda/2$.

2.1.4. Funciones de prueba

Una forma para determinar la fiabilidad, eficiencia y validar la correcta implementación de los algoritmos de optimización es definiendo una prueba de referencia usando funciones test ampliamente conocidas en la literatura. Se compararán los resultados de distintos tipos algoritmos entre clásicos y heurísticos con funciones test en distintas dimensiones [10].

Estas funciones test son usadas frecuentemente para estudiar la capacidad que tiene un algoritmo de escapar de un mínimo local. Si el proceso exploratorio del algoritmo sobre la superficie esta pobremente diseñado o no lo considera, el algoritmo se puede quedar atascado en un óptimo local. Las funciones multimodales suelen ser las más difíciles para los algoritmos, seguidas de funciones planas donde no se obtiene mucha información hacia dónde dirigir la búsqueda y/o la elección del tamaño de paso *(learning rate)*.

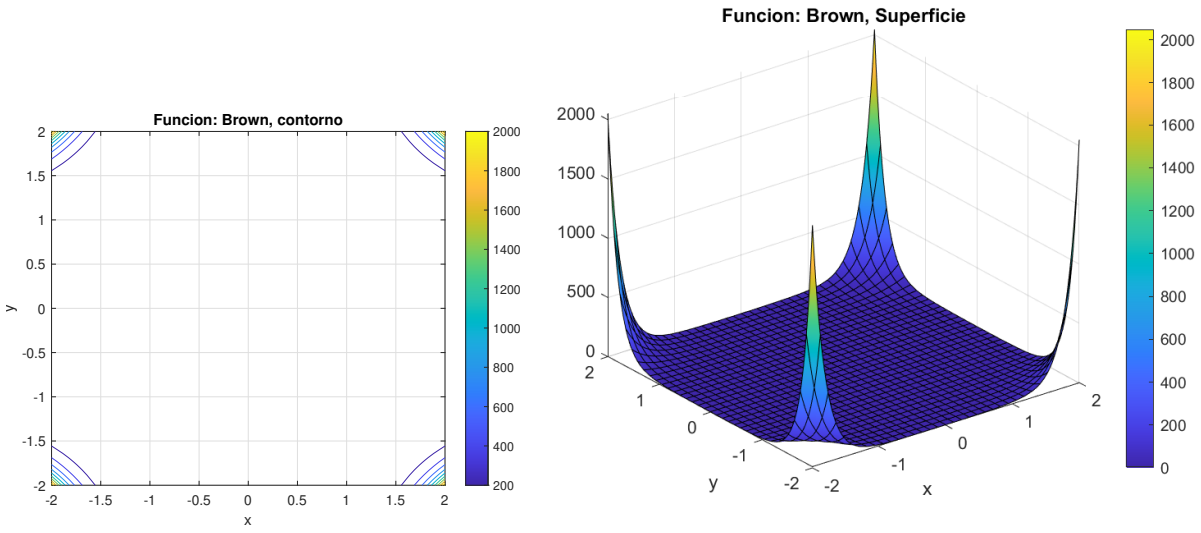
Se las agrupa según sus similitudes en sus propiedades, por ejemplo: modalidad es decir cantidad de picos u óptimos locales, separabilidad, dimensionalidad, diferenciabilidad, escalabilidad y las formas de sus superficies: presencia de valles o presencia de cuencas [10]. Otras características importantes que pueden afectar la eficiencia de los algoritmos son la dimensionalidad del espacio de búsqueda, la separabilidad de las funciones, escalabilidad y características relacionadas con el orden de magnitud del dominio y la función. Para este trabajo de grado se considera el uso de 4 funciones test en las cuales se destaca la función *Brown*, *Pathological*, *Streched V Sine Wave* y *Wavy* [10]. Estas funciones se implementaron en $\mathbb{R}^2$ y $\mathbb{R}^{10}$.

2.1.4.1. Función Brown

Esta función es continua, diferenciable, no separable, escalable y unimodal (véase ecuación 2.16). Esta función tiene un único mínimo global en una superficie plana, con zonas que presentan un fuerte descenso (véase figura en 3D 2.3), se seleccionó por la dificultad de estancamiento que tienen los algoritmos al momento de encontrar el óptimo [10].

$$f_{25}(x) = \sum_{i=1}^{n-1} \left(x_i^2\right)^{\left(x_{i+1}^2+1\right)} + \left(x_{i+1}^2\right)^{\left(x_i^2+1\right)}, \tag{2.16}$$

con $x \in \mathbb{R}^n$, donde $-5 \leq x_i \leq 5$. El mínimo global se encuentra en $x^* = (0, \cdots, 0)$, $f(x^*) = 0$.



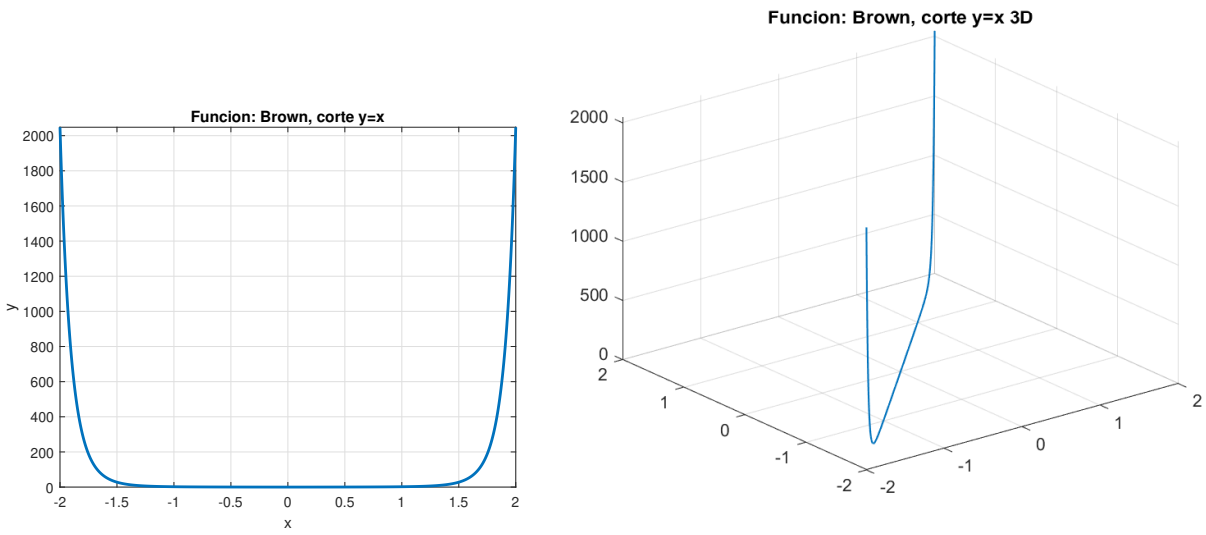
(a) Contorno (b) Superficie

Figura 2.3: Función Brown Izq.: Contorno, Der: Superficie. Fuente: *Autor*.

2.1.4.2. Función Pathological

Esta función es continua, diferenciable, no separable, no escalable y unimodal (véase ecuación 2.17). Esta función presenta un largo valle con una serie de crestas que tienen máximos y mínimos locales (véase figura en 3D 2.5), esta característica representa un problema para los algoritmos escaladores y una gran dificultad para poder explorar de forma efectiva la superficie [10].

$$f_{87}(x) = \sum_{i=1}^{n-1} \left(0{,}5 + \frac{\sin^2\left(\sqrt{100x_i^2 + x_{i+1}^2}\right) - 0{,}5}{1 + 0{,}001\left(x_i^2 - 2x_i x_{i+1} + x_{i+1}^2\right)^2}\right), \tag{2.17}$$



(a) Contorno (b) Plot

Figura 2.4: Función Brown Izq.: Corte 2D, Der: Corte 3D. Fuente: *Autor*.

con $x \in \mathbb{R}^n$, donde $-100 \leq x_i \leq 100$. El mínimo global se encuentra en $x^* = (0, \cdots, 0)$, $f(x^*) = 0$.

2.1.4.3. Función Streched V Sine Wave

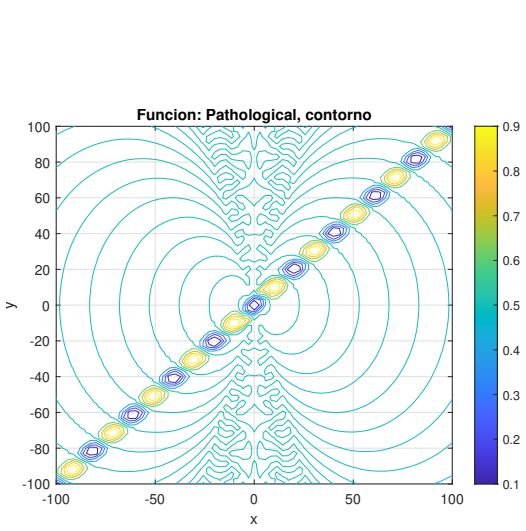
Esta función es continua, diferenciable, no separable, escalable y unimodal (véase ecuación 2.18). Esta función con forma de onda sinusoidal estirada presenta un mínimo global en centro (véase figura en 3D 2.7). Se selecciona por dificultad que tienen los algoritmos al estancarse en óptimos locales (curvas de nivel) [10].

$$f_{141}(x) = \sum_{i=1}^{n-1} \left(x_{i+1}^2 + x_i^2\right)^{0,25} \left[\sin^2\left\{50\left(x_{i+1}^2 + x_i^2\right)^{0,1}\right\} + 0{,}1\right], \tag{2.18}$$

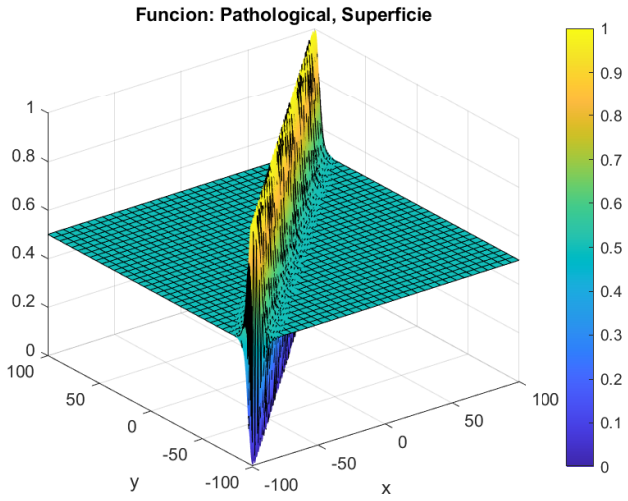
con $x \in \mathbb{R}^n$, donde $-10 \leq x_i \leq 10$. El mínimo global se encuentra en $x^* = (0, \cdots, 0)$, $f(x^*) = 0$

2.1.4.4. Función Wavy

Esta función (véase ecuación 2.19) es continua, diferenciable, separable, escalable y multimodal (véase figura en 3D 2.9). Se selecciona por el parecido que tiene a un panal de huevos. Los algoritmos se pueden estancar en los distintos óptimos locales, en este caso la constante se establece como $k = 10$. [10].

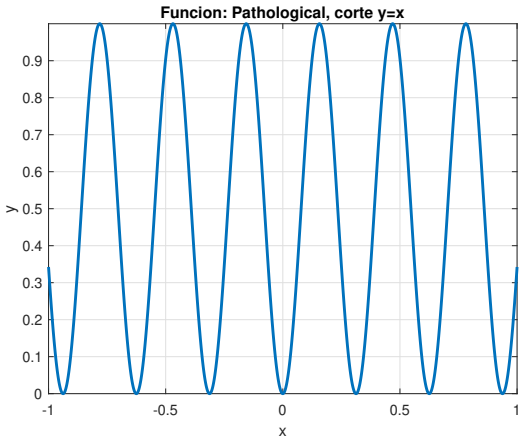


(a) Contorno

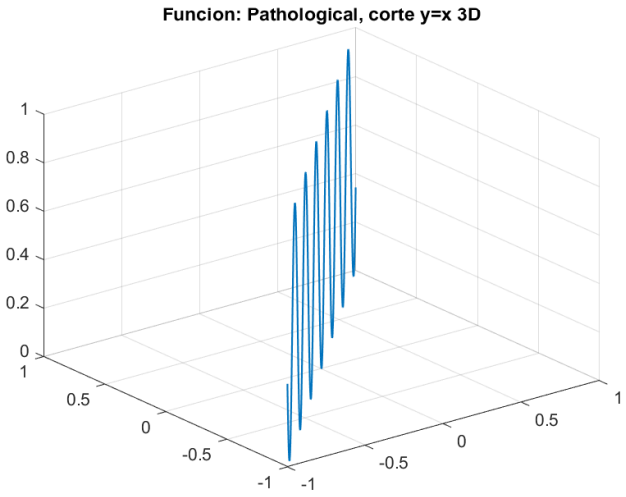


(b) Superficie

Figura 2.5: Función Pathological Izq: Contorno, Der: Superficie. Fuente: *Autor*.

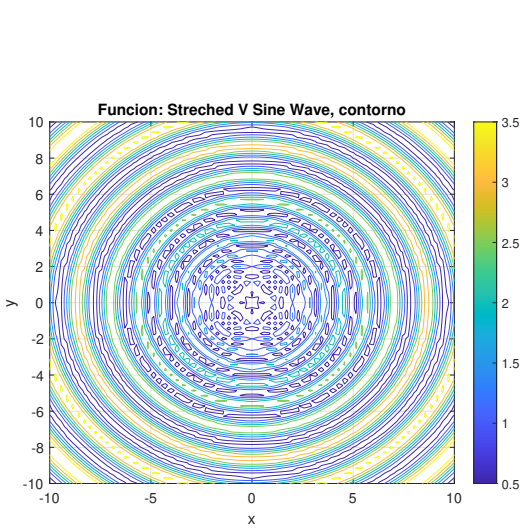


(a) Corte 2D

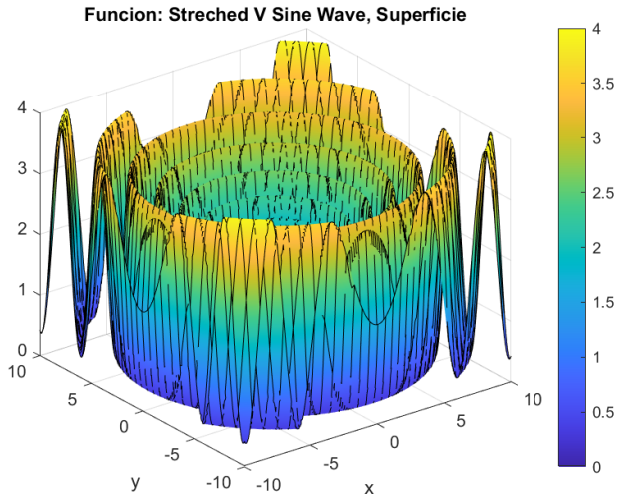


(b) Corte 3D

Figura 2.6: Función Pathological Izq: Corte 2D, Der: Corte 3D. Fuente: *Autor*.

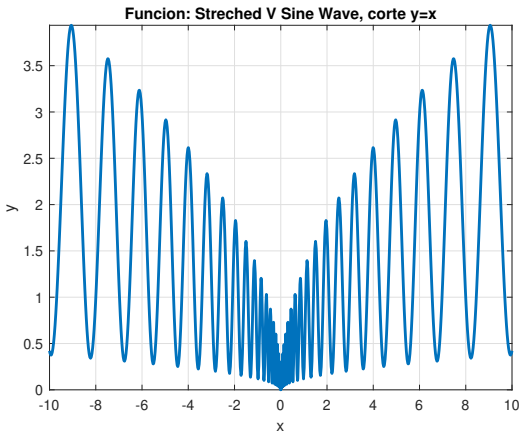


(a) Contorno

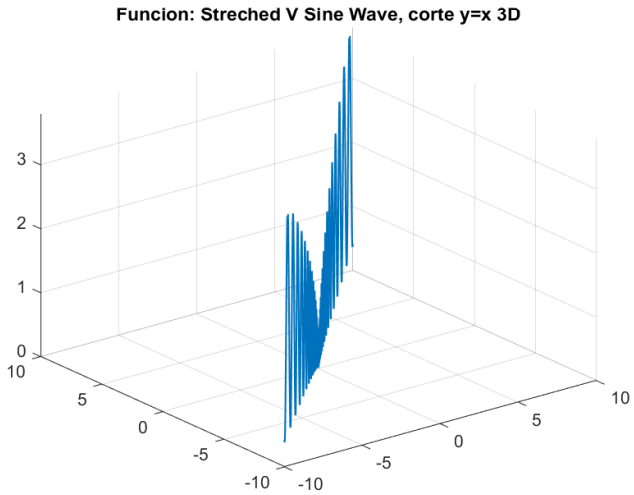


(b) Superficie

Figura 2.7: Función Streched V Sine Wave Izq: Contorno, Der: Superficie. Fuente: *Autor*.



(a) Corte 2D

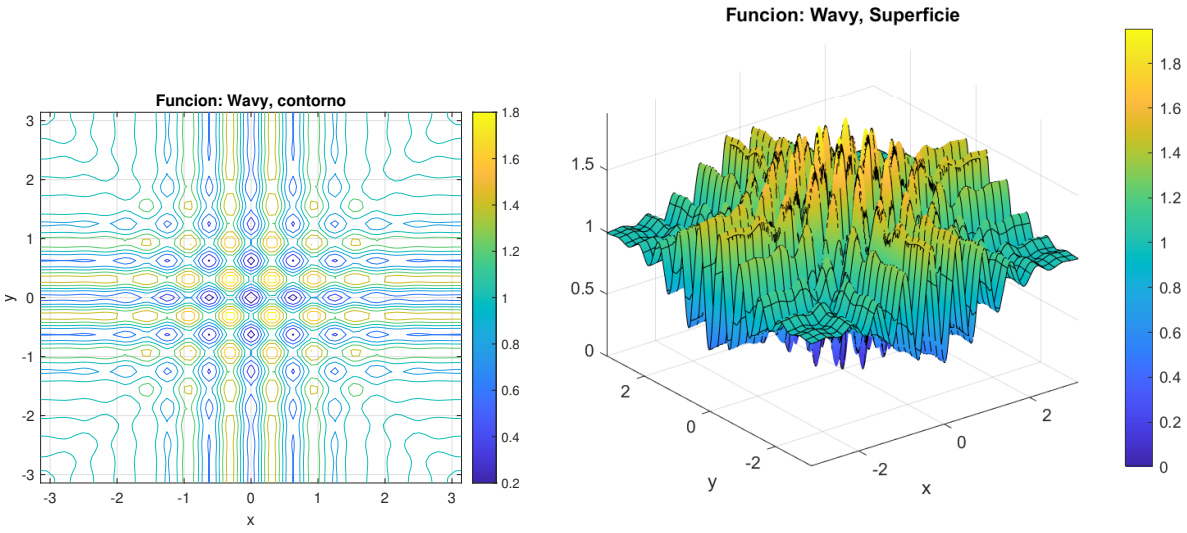


(b) Corte 3D

Figura 2.8: Función Streched V Sine Wave. Izq: Corte 2D, Der: Corte 3D. Fuente: *Autor*.

$$f_{164}(x) = 1 - \frac{1}{n}\sum_{i=1}^{n}\cos(kx_i)e^{\frac{-x_i^2}{2}}, \tag{2.19}$$

con $x \in \mathbb{R}^n$, donde $-\pi \leq x_i \leq \pi$. El mínimo global se encuentra en $x^* = (0, \cdots, 0)$, $f(x^*) = 0$. El número de mínimos locales son k^n cuando k impar y $(k+1)^n$ cuando k par.



(a) Contorno (b) Superficie

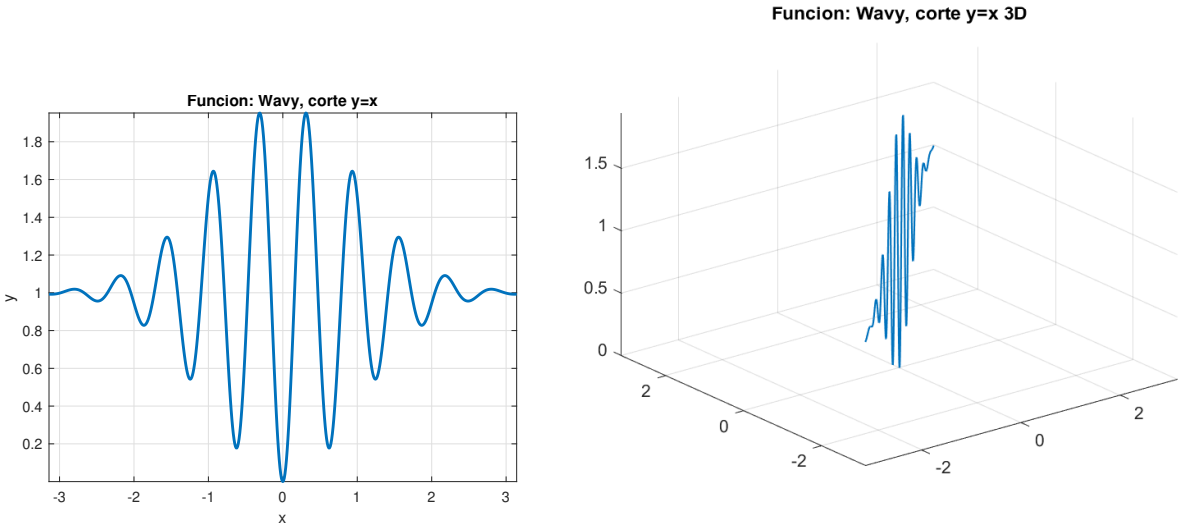
Figura 2.9: Función Wavy. Izq: Contorno, Der: Superficie. Fuente: *Autor*.

2.2. Optimización Clásica

Identificamos algunos de los algoritmos clásicos más relevantes implementados en el Beamforming, esto nos da un punto de partida, pues los algoritmos identificados son pertinentes para el objeto de investigación.

2.2.1. Conjugate Gradient Method (CGM)

Conjugate Gradient Method CGM o algoritmo de gradiente conjugado, es un método iterativo de gradiente creado por Hestenes and Stiefel en el año 1952 en el *Journal of Research of the NBS*. [7]. Este método ha sido utilizado ampliamente para resolver problemas de optimización sin restricciones. Para la construcción del método consideremos la función $f : \mathbb{R}^n \to \mathbb{R}$ continua y diferenciable, $\mathbb{R}^n$ un espacio euclidiano n-dimensional y el problema de optimización se expresa de la siguiente forma:



(a) Corte 2D (b) Corte 3D

Figura 2.10: Función Wavy. Izq: Corte 2D, Der: Corte 3D. Fuente: *Autor*.

$$\min_{x \in \mathbb{R}^n} f(x).$$

De acuerdo con la línea de búsqueda se puede expresar el problema como:

$$x_{k+1} = x_k - \alpha_k d_k, \qquad k = 0, 1, 2, \cdots, \tag{2.20}$$

donde x_{k+1} es la nueva posición actualizada, x_k la posición actual, $\alpha_k > 0$ un escalar que indica el tamaño de paso *(Learning rate)* y d_k un vector unitario o dirección en la cual x debería actualizarse/moverse [14]. De modo que podemos expresar la función como $\psi(\alpha_k) = f(x_k + \alpha_k d_k)$ y el problema de optimización queda expresado como:

$$\lim_{\alpha_k \to 0} f(x_k + \alpha_k d_k), \tag{2.21}$$

$$\min_{\alpha_k > 0} f(x_k + \alpha_k d_k). \tag{2.22}$$

De acuerdo con las condiciones de *Wolfe* (Regla de Armijo y condición de curvatura) [14] tenemos que:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta \alpha_k d_k^T \nabla f(x_k), \tag{2.23}$$

$$\left| \nabla f(x_k + \alpha_k d_k)^T d_k \right| \leq -\sigma \nabla f(x_k)^T d_k, \tag{2.24}$$

donde $\nabla f(x_k)$ es el gradiente de $f(\cdot)$ evaluado en x_k con $0 < \delta < \sigma < 1$. Sin pérdida de generalidad y con la intención de simplificar la notación vamos a expresar el gradiente evaluado en x_k como $g_k = \nabla f(x_k)$ [14]. Ahora de la primera condición de *Wolfe* consideramos la ecuación 2.23 y consideramos el siguiente problema de optimización:

$$\min_d d_k^T g_k, \tag{2.25}$$

como d_k^T y g_k son vectores entonces podemos expresarlos en función del producto punto:

$$d_k^T g_k = |d_k| \cdot |g_k| \cos\theta, \tag{2.26}$$

donde θ es el ángulo entre los dos vectores y se minimizará cuando $\theta = \pi$, es decir $\cos\theta = -1$ y el vector unitario d es:

$$d_k = -\frac{g_k}{\|g_k\|}. \tag{2.27}$$

De forma general podemos decir que la dirección del vector d se da en el sentido contrario del gradiente $d = -g_k$ también conocido como gradiente descendiente, reemplazando la ecuación 2.27 en la ecuación 2.20 de línea de búsqueda tenemos el método de descenso más pronunciado (*Steepest Descent Method*):

$$x_{k+1} = x_k - \alpha_k g_k. \tag{2.28}$$

El gran problema del método de descenso más pronunciado está relacionado con su baja convergencia al solo considerar la dirección negativa del gradiente $\nabla f(x_k)$ y el tamaño de paso α_k por otro lado su gran ventaja es el uso de gradientes cuando el problema así lo permite, pero no necesita cálculos de segundas derivadas que pueden representar un costo computacional adicional al problema de optimización.

En el caso del método del gradiente conjugado vamos a considerar para la primera iteración k=0, el gradiente descendido $d = -g_k$, para $k > 1$ vamos a utilizar combinaciones lineales de la línea de búsqueda [14].

$$d_k = \begin{cases} -g_k, & k = 0 \\ -g_k + \beta_k d_{k-1}, & k \geq 1, \end{cases} \tag{2.29}$$

donde $\beta_k \in \mathbb{R}$ el cual es un escalar conocido como coeficiente del gradiente conjugado. En la actualidad son muchas las fórmulas planteadas para calcular el coeficiente β_k, las más conocidas

son Hestenes-Stiefel (HS), Fletcher-Reeves (FR), Polak-Ribiere-Polyak (PRP), Conjugate Descent (CD), Dai-Yuan (DY) y WYL (Wei-Yao-Liu) [14].

$$\beta_k^{HS} = \frac{g_k^T (g_k - g_{k-1})}{d_{k-1}^T (g_k - g_{k-1})}, \qquad \beta_k^{FR} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}, \qquad \beta_k^{PRP} = \frac{g_k^T (g_k - g_{k-1})}{\|g_{k-1}\|^2}, \tag{2.30}$$

$$\beta_k^{CD} = -\frac{\|g_k\|^2}{d_{k-1}^T g_k}, \qquad \beta_k^{DY} = \frac{\|g_k\|^2}{d_k^T (g_k - g_{k-1})}, \qquad \beta_k^{WYL} = \frac{d_k^T \left(g_k - \frac{\|g_k\|}{\|g_{k-1}\|} g_{k-1}\right)}{\|g_{k-1}\|^2}. \tag{2.31}$$

Algorithm 2.1 Pseudocódigo de *Conjugate Gradient Method (CGM)*

Require: *target:* Minimizar función objetivo
Ensure: F_{best}, Pos_{best}:
  Función Objetivo $f(x)$ con $x \in \mathbb{R}^n$
  Escoger x_0 arbitrario en el dominio
  while Criterio de parada: Tolerancia **do**
    Calcular coeficiente del gradiente conjugado β_k usando Ec. (2.30-2.31)
    Calcular d_k usando Ec. (2.29)
    Actualizar x_k usando Ec. (2.20) y (2.29)
    Incremento de la iteración $k = k + 1$
  end while
  Procesar información y visualizar x_{k+1}

2.2.2. Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent SGD o algoritmo de gradiente estocástico descendente, es un método iterativo de gradiente creado por Herbert Robbins y Sutton Monro en el año 1951 en la universidad de Carolina del Norte [18]. Sea f una función tal que $f : \mathbb{R}^n \to \mathbb{R}$ y el problema de optimización:

$$\min_{x \in \mathbb{R}^n} f(x),$$

El primer acercamiento para este tipo de problemas fue propuesto por Cauchy a través del método clásico de gradiente descendiente, para lo cual se requiere calcular el gradiente de $f(\cdot)$, $\nabla f(x)$ en cualquier punto x [15]. Empezando desde un vector inicial x_0 y haciendo las siguientes actualizaciones basados en la ecuación 2.28 de CGM tenemos:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k).$$

Para $k = 0, \cdots$, y α_k el tamaño de paso del algoritmo o tasa de aprendizaje *Learning rate*. De acuerdo Herbert Robbins y Sutton Monro,Más concretamente, supongamos que para cada k, tenemos acceso a un vector aleatorio ζ_k que en promedio es igual a $\nabla f(x_k)$, es decir, $\mathbb{E}[\zeta_k] = \nabla f(x_k)$ [15]. donde $\mathbb{E}[\cdot]$ es el operador Esperanza matemática definido como $\mathbb{E}(X) = \sum_x xP(X = x)$ cuando la variable es discreta o $\mathbb{E}(X) = \int_{-\infty}^{\infty} xf(x)dx$ cuando es continua y las actualizaciones del proceso iterativo son:

$$x_{k+1} = x_k - \alpha_k \zeta_k. \tag{2.32}$$

Esta configuración es conocida como aproximación estocástica y esta da origen al gradiente estocástico descendiente, en la década de los ochenta los métodos de gradiente CGM y SGD alcanzaron mucha popularidad en problemas de *Machine Learning* y procesamiento de señales donde se podía calcular de forma exacta el gradiente, pero la gran dificultad radicaba en los problemas donde no era posible calcular gradientes exactos debido a experimentos o mediciones [15].

Uno de los problemas más evidentes de los métodos de gradiente está relacionados con la lentitud de las funciones que tienen valles pronunciados como es el caso de la función *Brown y Pathological* debido a su tamaño de paso, una de las formas más comunes utilizadas para resolver este inconveniente por ejemplo en *Machine Learning* es estandarizar los datos (*dataset*) con $\mu = 0$ y $\sigma = 1$.

$$z = \frac{x - \mu}{\sigma}. \tag{2.33}$$

Algorithm 2.2 Pseudocódigo de *Stochastic Gradient Descent (SGD)*

Require: *target:* $f(x)$
Ensure: F_{best}, Pos_{best}:
  Escoger x_0 arbitrario en el dominio
  Definir α
  while Criterio de parada: Tolerancia **do**
    Actualizar x_{k+1} basado en Ec. (2.32)
  end while
  Procesar información y Visualizar

2.2.3. Nelder Mead Search (NMS)

Nelder Mead Search NMS o Algoritmo Simplex, es un algoritmo de búsqueda directa creado por J. A. Nelder y R. Mead en el año de 1965, conocido por resolver problemas de optimización multidimensional, sin restricciones y sin usar derivadas. No se debe confundir con el método Simplex (Dantzig) para programación lineal. Nelder y Mead se inspiraron en el método de búsqueda

directa planteado por Spendley en 1962 que consideraba dos movimientos: Reflejar sobre el peor vértice (x_r) y comprimir las soluciones respecto al mejor vértice [13].

En el año 1965 Nelder-Mead consideran dos movimientos adicionales: Expandir el vértice por afuera de la reflexión (x_r) o contraer por fuera del poliedro (x_{oc}) o por adentro (x_{ic}) permitiendo cambiar el tamaño y la forma del poliedro. En la actualidad goza de bastante popularidad dentro de los algoritmos de búsqueda directa [13].

Para la implementación considere una función $f(X)$ de valor real a minimizar de n variables, $x_1, x_1, \cdots, x_n \in \mathbb{R}^n$. El método considera $n+1$ vértices que corresponden a las soluciones que conforman el poliedro, sus soluciones se evalúan y se ordenan para determinar el mejor (x_1) y el peor vértice (x_h) es decir $y_1 < y_2 < \cdots < y_n < y_h$. Luego se considera $\bar{x}$ como el promedio de todas las soluciones sin considerar x_h y consideramos los condicionales en el pseudocódigo 2.3 para reemplazar x_h y así repetir el proceso de forma iterativa basado en su criterio de parada [19].

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{2.34}$$

Se definen 4 parámetros (coeficientes) para completar el método:

$$\rho > 0, \qquad 0 < \psi < 1, \tag{2.35}$$

$$\chi > 1, \qquad 0 < \sigma < 1, \tag{2.36}$$

los coeficientes son: reflexión (ρ), expansión (χ), contracción (ψ) y compresión (σ). Los valores recomendados para los coeficientes son $\rho = 1$, $\chi = 2$, $\psi = 0{,}5$ y $\sigma = 0{,}5$. Se considera x_r como punto de partida del primer condicional para determinar un posible punto de reflexión:

$$x_r = \bar{x} + \rho(\bar{x} - x_h) = (1+\rho)\bar{x} - \rho x_h. \tag{2.37}$$

Si se cumple la condición $y_1 < y_r < y_n$, entonces se acepta el punto de reflexión x_r que reemplaza al peor vértice x_h y termina la iteración, por otra parte, si $y_r < y_1$, se calcula el punto de expansión x_e:

$$x_e = \bar{x} + \chi(x_r - \bar{x}) = \bar{x} + \rho\chi(\bar{x} - x_h) = (1+\rho\chi)\bar{x} - \rho\chi x_h. \tag{2.38}$$

Si $y_e < y_r$ entonces se acepta el punto de expansión x_e que reemplaza al peor vértice x_h y se termina la iteración, cuando sucede lo contrario, es decir $y_r \leq y_e$ se acepta el punto de reflexión x_r que reemplaza al peor vértice x_h y termina la iteración. Para las contracciones se considera el caso si $y_n < y_r < y_h$ donde se calcula el posible punto de contracción hacia afuera x_{oc}:

$$x_{oc} = \bar{x} + \psi(x_r - \bar{x}) = \bar{x} + \rho\psi(\bar{x} - x_h) = (1 + \rho\psi)\bar{x} - \rho\chi x_h. \tag{2.39}$$

Si $y_{oc} < y_r$ se acepta el punto de contracción hacia afuera y se termina la iteración, en el caso contrario $y_r \leq y_{oc}$ se encogen las soluciones respecto al mejor vértice. Si $y_r \geq y_h$ entonces se calcula el posible punto de contracción hacia adentro:

$$x_{ic} = \bar{x} - \psi(\bar{x} - x_h) = (1 - \psi)\bar{x} + \psi x_h. \tag{2.40}$$

En el último caso si $y_{ic} < y_h$ entonces se acepta el punto de contracción hacia afuera x_{ic} que reemplaza al peor vértice x_h y termina la iteración, en el caso contrario $y_h \leq y_{ic}$ se encogen las soluciones respecto al mejor vértice. Para encoger los vértices consideramos:

$$v_i = x_1 + \sigma(x_i - x_1), \tag{2.41}$$

donde $i = 2, 3, \cdots, n + 1$ y $v_2, v_3, \cdots, v_{n+1}$ los nuevos vértices. Al final del proceso iterativo las soluciones se evalúan y se ordenan para determinar el mejor (x_1) y el peor vértice (x_h) de la próxima iteración.

2.3. Optimización Metaheurística

En la actualidad, los algoritmos metaheurísticos, que son estrategias de solución de ensayo y error inspiradas en la biología, se han vuelto muy populares debido a su capacidad para encontrar soluciones óptimas en tiempos razonables. Ejemplos de ello son los algoritmos BA y CKLF, que han demostrado ser muy eficientes. Desde hace millones de años, la evolución de la naturaleza ha sido una fuente de inspiración para los investigadores que buscan desarrollar algoritmos capaces de resolver problemas de optimización matemática. En esta investigación se estudiaron tres de estos algoritmos: Particle Swarm Optimization (PSO), Bat Algorithm (BA) y Cuckoo Search by Levy Flights (CKLF), los cuales se destacan por su inteligencia de enjambre, capacidad de selección de los individuos más aptos (elitismo), algoritmo evolutivo, población, entre otras características importantes.

La investigación en algoritmos metaheurísticos está en constante evolución y los investigadores están trabajando para mejorar la eficacia de estos algoritmos en términos de convergencia, tasa de éxito y esfuerzo computacional. Los nuevos artículos que se publican en este campo buscan desarrollar nuevos enfoques y técnicas para mejorar la capacidad de estos algoritmos para encontrar soluciones óptimas [27].

Además, se están estudiando diferentes configuraciones del algoritmo para optimizar su rendimiento en diferentes situaciones. Por ejemplo, se pueden ajustar los parámetros del algoritmo para adaptarse a diferentes tipos de problemas, o se pueden combinar diferentes algoritmos metaheurísticos para crear estrategias híbridas más eficientes. También se están explorando estrategias

Algorithm 2.3 Pseudocódigo de *Nelder Mead Search (NMS)*

Require: *target:* $f(x)$
Ensure: F_{best}, Pos_{best}:
  Escoger x_0 arbitrario en el dominio
  Crear poliedro y ordenarlo de acuerdo con su evaluación
  Crear poliedro con $n+1$ soluciones
  Definir ρ, χ, ψ y σ basados en Ec.(2.36)
  while Criterio de parada: Tolerancia **do**
    Crear $\bar{x}$ basado en Ec. (2.34)
    x_h ultima solución, $y_h = f(x_h)$
    $x_r = (1+\rho)\bar{x} - \bar{x}x_h$, $y_r = f(x_r)$ basado en Ec. (2.37)
    if $y_r < y_1$ **then**
      $x_e = (1+\rho\chi)\bar{x} - \rho\chi x_h$, $y_e = f(x_e)$ basado en Ec. (2.38)
      if $y_e < y_r$ **then**
        Acepto x_e: Expandir
      else
        Acepto x_r: Reflejar
      end if
    else
      if $y_r < y_n$ **then**
        Acepto x_r: Reflejar
      else
        if $y_r < y_h$ **then**
          $x_{oc} = (1+\rho\psi)\bar{x} - \rho\psi x_h$, $y_{oc} = f(x_{oc})$ basado en Ec. (2.39)
          if $y_{oc} < y_r$ **then**
            Acepto x_{oc}: Contraer hacia afuera
          else
            Encoger soluciones basado en Ec. (2.41)
          end if
        else
          $x_{ic} = (1-\psi)\bar{x} + \psi x_h$, $y_{ic} = f(x_{ic})$ basado en Ec. (2.40)
          if $y_{ic} < y_h$ **then**
            Acepto x_{ic}: Contraer hacia Adentro
          else
            Encoger soluciones basado en Ec. (2.41)
          end if
        end if
      end if
    end if
    Contador $t = t+1$
    Ordenar soluciones de acuerdo con su evaluación
  end while
  Procesar información y Visualizar

colaborativas en las que varios algoritmos trabajan juntos para resolver un problema de manera más efectiva. Por ejemplo, un algoritmo podría ser responsable de explorar el espacio de búsqueda y otro podría ser responsable de explotar las soluciones encontradas [27].

Se están investigando diferentes técnicas de sintonización de parámetros para optimizar el rendimiento de los algoritmos metaheurísticos en una variedad de situaciones. Estas técnicas pueden incluir la optimización de parámetros a través de métodos de búsqueda, como la búsqueda aleatoria *(Random Search)*, la búsqueda en grilla *Grid Search* y la optimización basada en heurísticas. En resumen, la investigación en algoritmos metaheurísticos es un campo en constante evolución y se están desarrollando continuamente nuevas técnicas y enfoques para mejorar la eficacia de estos algoritmos [27].

De acuerdo con el teorema "No hay almuerzo gratis" *(no free lunch theorems for optimization)* no es posible encontrar un algoritmo que sea mejor para todos los problemas o universalmente robusto o un conjunto de parámetros que pueda resolver todos los problemas. Este teorema plantea que, si un algoritmo A tiene un mejor desempeño que el algoritmo B en ciertas funciones, entonces B tendrá un mejor desempeño que A en otras funciones. En general se puede concluir que no hay un algoritmo universal y cada problema puntual tiene que tratarse de forma especial con sintonización de parámetros que sean adecuados a las métricas y al problema [25].

De acuerdo con lo anterior se definieron ciertos parámetros de sintonización para las funciones de prueba considerando los recomendados en los antecedentes [25], donde más que buscar los algoritmos más eficientes, se desea conocer la correcta implementación de los algoritmos para que luego sean usados en los distintos experimentos. Cabe resaltar que en los experimentos es obligatorio definir un nuevo conjunto de parámetros que más se ajuste al problema del Beamforming en los experimentos numéricos.

2.3.1. Particle Swarm Optimization (PSO)

Particle Swarm Optimization PSO o método de optimización de enjambre de partículas se define para funciones continuas no lineales. creado por Kennedy y Eberhart en el año de 1995, El método establece sus bases sobre modelos sociales de interacción como la cría de ciertas aves (*"bird flocking"*), escolarización de ciertos peces (*"fish schooling"*) y la teoría de enjambres en general, donde se observan aspectos inspirados en el comportamiento conocido como inteligencia de enjambre, por ejemplo, realizan movimientos físicos para evitar predadores, buscar comida y parejas, optimizar parámetros ambientales como puede ser la temperatura, etc. [12].

Para la implementación, este algoritmo explora el espacio de la función objetivo realizando una métrica para ajustar las trayectorias de las partículas bajo dos componentes principales: una componente determinística donde cada partícula es atraída hacia la mejor posición por iteración x_i^* y consecuentemente hacia la mejor solución global g^* , y la componente estocástica que sucede al mismo tiempo con tendencia a moverse aleatoriamente. Entonces, cuando una partícula encuentra una posición mejor a x_i^* , automáticamente se actualiza como la mejor posición por iteración, y

se contrasta con la mejor solución global hasta el momento g^*, todo esto hasta que el objetivo no mejore o el criterio de parada así lo defina. A nivel computacional, se implementa en pocas líneas de código utilizando el producto Hadamard y buscando encontrar la mejor posición histórica de la función objetivo.

Para el planteamiento del algoritmo (véase: pseudocódigo 2.4) consideramos x_i como el vector posición de la partícula i uniformemente distribuida sobre la superficie, v_i el vector velocidad de la partícula i. Definimos la posición (véase ecuación 2.42) como:

$$x_i^{t+1} = x_i^t + v_i^{t+1}. \tag{2.42}$$

Para el cálculo de v_i^{t+1} (véase ecuación 2.43), el algoritmo considera dos aspectos: El primero considera posiciones iniciales uniformemente distribuidas, buscando explorar mejor la superficie, lo cual es importante para problemas multimodales y velocidad inicial cero. El segundo, considera la mejor posición histórica x_i^* para la partícula i para aumentar la diversidad en la calidad de las soluciones. Las nuevas posiciones se actualizan como:

$$v_i^{t+1} = v_i^t + \alpha\varepsilon_1 \odot \left[g^* - x_i^t\right] + \beta\varepsilon_2 \odot \left[x_i^* - x_i^t\right], \tag{2.43}$$

donde x_i^* es la mejor posición por iteración de la partícula i, la mejor solución global g^*, $\odot$ es el producto de Hadamard $u \odot v = u_{ij}v_{ij}$, ε_1, ε_2 son vectores aleatorios donde $\varepsilon_1, \varepsilon_2 \in [0, 1]$ con distribución $N(0, 1)$. $\alpha = \beta = 2$ son las constantes de aceleración o parámetros de aprendizaje de acuerdo con [25] y $v_i^{t=0} = 0$, la velocidad inicial de la partícula i. Para mejorar la estabilidad y convergencia una estrategia estándar es usar la función de inercia θ en la función velocidad v_i^{t+1} (véase ecuación 2.44), reemplazando v_i^t por θv_i^t, obteniendo la siguiente expresión:

$$v_i^{t+1} = \theta v_i^t + \alpha\varepsilon_1 \odot \left[g^* - x_i^t\right] + \beta\varepsilon_2 \odot \left[x_i^* - x_i^t\right], \tag{2.44}$$

donde θ es la inercia de la partícula i con valores típicos $\theta \in [0{,}5 \sim 0{,}9]$ de acuerdo con [25]. La inercia agrega una masa virtual que busca estabilizar el movimiento de las partículas para que el AM converja más rápido.

2.3.2. Bat Algorithm (BA)

En el 2010 el profesor Xin-She Yang de la Universidad de Cambridge, plantea el algoritmo del Murciélago, su inspiración biológica se basa en el comportamiento de ecolocalización de los murciélagos [24]. Estos son los únicos mamíferos con alas y con una gran capacidad de ecolocalización, la cual sirve para detectar presas, evitar obstáculos y encontrar sus grietas en la oscuridad. Los murciélagos emiten un sonido muy fuerte hasta 110 decibeles en la región de ultrasonido, y escuchan el eco que rebota sobre los objetos que lo rodean; estos pulsos varían en sus propiedades,

Algorithm 2.4 Pseudocódigo de *Particle Swarm Optimization* (PSO)

Require: Función Objetivo $f(x)$, $x = (x_i, ..., x_p)^T$
Ensure: F_best, Pos_best
  Definir población de n partículas
  Inicializar posición x_i^t con Caminata Aleatoria
  Inicializar velocidad v_i^t con Caminata Aleatoria
  while Criterio de parada: Tolerancia **do**
    Generar nueva velocidad v_i^{t+1} basados en Ec. (2.44)
    Generar nueva población x_i^{t+1} basados en Ec. (2.42)
    Determinar evaluación P_{best} iteración, G_{best} histórico
    if $Pos_{best} <= G_{best}$ **then**
      $g^* = x^*$
    end if
    $t = t + 1$ (contador de iteraciones)
  end while
  Procesar la información y Visualizar

estrategias de caza, armónicos, ancho de banda o de las mismas especies. La mayoría usa señales cortas moduladas en frecuencia, a diferencia de otros que usan señales de frecuencia constante[24]. El rango de frecuencias típicas de los murciélagos se encuentra entre los 25 KHz hasta los 150 KHz, con ráfagas de pulsos que duran entre 5 y 20 ms, si consideramos la velocidad del sonido en el aire como $v = 340m/s$.

Las longitudes de onda son entre $2mm$ hasta los $14mm$ que corresponden al tamaño de sus presas. Se estima que hay más de 1000 especies en una gama amplia de pesos y tamaños. Por ejemplo, en el caso del murciélago abejorro (*bumblebee bat*), tiene un peso entre 1.5 y 2 gramos, y una longitud de alas entre 2.2 centímetros y 11 centímetros. Por otro lado, los murciélagos más grandes pueden pesar hasta 2 kilogramos, y sus alas pueden medir hasta 2 metros de largo.

Todos los murciélagos usan la ecolocalización, pero los grandes no suelen hacerlo, a diferencia de los micro murciélagos que lo hacen de forma extensa, siendo capaces incluso, de evitar obstáculos tan pequeños como el cabello humano.

Entonces, basados en la inspiración biológica del murciélago se definieron los aspectos básicos del algoritmo (véase: pseudocódigo 2.5), por lo cual se estableció una población de n murciélagos que utilizan la ecolocalización para detectar la distancia; los murciélagos vuelan de forma aleatoria con velocidad v_i y posición x_i con $i = 1, 2, \cdots, n$ y cierta frecuencia mínima definida inicialmente de forma aleatoria entre $[f_{min}, f_{max}]$ (véase ecuación 2.45). Para casos prácticos se considera $[0, f_{max}]$, que se establecerá de acuerdo con el dominio del problema, la intensidad A_i definida inicialmente de forma aleatoria, con distribución uniforme que disminuye conforme se acerca a la presa y la tasa

de emisión de pulsos $r_i \in [0,1]$, donde 0 significa que no se emiten pulsos y 1 significa la máxima emisión de pulsos. Se define como se van a mover los murciélagos, por ello se considera la frecuencia f_i, posición x_i (véase ecuación 2.47) y velocidad v_i(véase ecuación 2.46):

$$f_i = f_{min} + (f_{max} - fmin)\beta, \tag{2.45}$$
$$v_i^t = \theta v_i^{t-1} + \alpha\varepsilon(x_i^t - g^*)f_i, \tag{2.46}$$
$$x_i^t = x_i^{t-1} + v_i^t. \tag{2.47}$$

En el caso de la velocidad consideraremos la velocidad de (APSO) Donde θ es la inercia de la partícula i con valores típicos $\theta \in [0{,}5 \sim 0{,}9]$ de acuerdo con [25]. $\beta, \varepsilon \in [0,1]$, son vectores aleatorios con distribución uniforme, y la constante de aceleración $\alpha = 0{,}2$ y g^* es la mejor posición global por iteración que se obtiene comparando todas las soluciones de los n murciélagos. El algoritmo realiza una búsqueda local donde se selecciona una posición x_i (véase ecuación 2.48) en términos de la tasa de emisión de pulsos y se establece una nueva posición para el murciélago generada localmente usando caminatas aleatorias.

$$x_i = g^* + \varepsilon A^t, \tag{2.48}$$

donde $\varepsilon \in [-1,1]$ es un número aleatorio y $< A^t >$ es el promedio de todas las intensidades de los murciélagos. Se establece la tasa de emisión de pulsos r_i definida inicialmente de forma aleatoria (véase ecuación 2.50), con distribución uniforme que aumenta de acuerdo con la proximidad del objetivo en el rango $r_i \in [0,1]$, donde 0 significa que no hay pulsos, y 1 que es la tasa de emisión de pulsos máxima. Los valores de A_i (véase ecuación 2.49) se puede escoger a conveniencia por simplicidad definimos $A_i \in [1,2]$ donde $A_{min} = 1$ significa que el murciélago encontró su presa y temporalmente dejó de emitir algún sonido [29].

$$A_i^{t+1} = \alpha A_i^t, \tag{2.49}$$
$$r_i^{t+1} = r_i^0[1 - e^{-\gamma t}], \tag{2.50}$$

donde $\alpha \in [0,1]$ y $\gamma > 0$ son constantes, α hace las veces de esquema de enfriamiento en el algoritmo SA[22]. En general, se considera que $\alpha = \gamma = 0{,}9$ [24]. La intensidad y la tasa de emisión de pulsos solo se actualizará si las nuevas posiciones mejoran, lo cual indica que los murciélagos se están moviendo hacia el óptimo.

2.3.3. Cuckoo Search by Levy Flights (CKLF)

En el año 2009 el profesor *Xin-She Yang* y su colega *Suash Deb* publican el artículo sobre el algoritmo *Cuckoo Search via Lévy Flights*, el cual describe un algoritmo metaheurístico llamado *Cuckoo Search* o búsqueda del pájaro Cuckoo para resolver problemas de optimización. Este algoritmo se inspira en el comportamiento reproductivo parasitario de ciertas especies de pájaros

Algorithm 2.5 Pseudocódigo de *Bat Algorithm* (BA)

Require: *target*: Minimizar función objetivo
  Función Objetivo $f(x), x = (x_i, ..., x_p)^T$
  Definir f_i, r_i y A_i
  Inicializar x_i y v_i con Caminata aleatoria
  while Criterio de parada: Tolerancia **do**
    Nuevas soluciones ajustando frecuencia
    Actualizar x_i y v_i Ec. (2.45) hasta (2.47)
    for $i = 1 : n$ **do**
      if $rand < r_i$ **then**
        Generar una solución local Ec. (2.48)
      end if
      Generar una nueva solución volando de forma aleatoria
      if $rand > A_i$ & $f(X_i) < f(g^*)$ **then**
        Acepte las nuevas soluciones
        Aumente r_i y disminuya A_i Ecuación (2.49) y (2.50)
      end if
    end for
    Organizar los murciélagos y encontrar g^*
  end while
  Procesar la información y Visualizar

Cuckoos en combinación con cierto comportamiento de vuelos de Lévy detectados en moscas de frutas y ciertas aves [23].

Los pájaros Cuckoos tienen una agresiva estrategia de reproducción, algunas especies de Cuckoos dejan sus huevos en nidos comunales y otros eliminan los huevos huéspedes para aumentar la probabilidad de eclosión de estos. Existen tres tipos de parasitismo de cría: Intraespecífico, Reproducción cooperativa y Toma de Nido. Este parasitismo puede generar conflictos con ciertas especies, por ejemplo, si el parasitismo lo descubre, el ave huésped puede tirar los huevos o simplemente abandona el nido [23].

Otro momento importante es cuando el Cuckoo pone los huevos, que usualmente es en nidos donde el ave huésped apenas acaba de poner los suyos. Los huevos del Cuckoos demoran menos tiempo en eclosionar, lo cual garantiza que, al nacer, los polluelos, de forma instintiva busquen desalojar los huevos huéspedes expulsándolos ciegamente fuera del nido, esto garantiza una mayor porción de alimento por parte del ave huésped, el cual hace pensar a sus anfitriones que el polluelo Cuckoo es su hijo [23].

Por otra parte, se analiza el comportamiento de los vuelos de Lévy, donde ciertas moscas frutales exploran la superficie realizando vuelos en trayectorias rectas con giros repentinos de 90 grados; estos vuelos son usados como caminatas aleatorias que definen el tamaño de paso de los cuckoos y puede ser utilizados en la optimización [23] metaheurística.

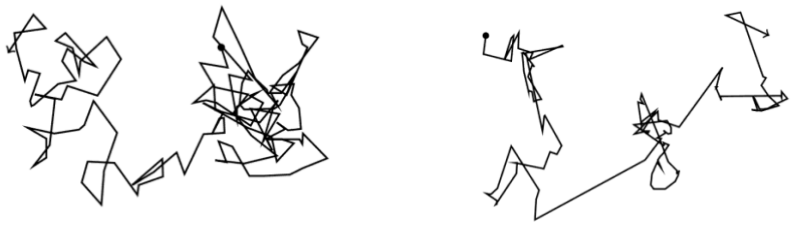


Figura 2.11: Movimiento: Izq. Browniano. Der. Vuelo de Lévy Inicio(•). Fuente: *[28]*.

Los vuelos de Lévy son caminatas aleatorias en cual su tamaño de paso está definido por la distribución probabilidad de Lévy, usualmente definida como $L(s) \sim |s|^{-1-\beta}$, con $0 < \beta \leq 2$ (véase ecuación 2.51):

$$L(s,\gamma,\mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} \exp\left[-\frac{\gamma}{2(s-\mu)}\right] \frac{1}{(s-\mu)^{3/2}} & 0 < \mu < s < \infty \\ 0 & \text{en otro caso} \end{cases} \tag{2.51}$$

donde $\mu > 0$ es el paso mínimo y γ es un parámetro de escalamiento. Para una implementación

practica y de forma más general, se puede estimar la distribución de Lévy cuando s es grande, $s \to \infty$ de modo que:

$$L(s) = \frac{\alpha\beta\Gamma(\beta)\sin(\pi\beta/2)}{\pi\,|s|^{1+\beta}}. \tag{2.52}$$

El valor es $\beta = 3/2$ y la función gamma es:

$$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt. \tag{2.53}$$

Los vuelos de Lévy son más eficientes que las caminatas aleatorias brownianas explorando la superficie y en espacios de búsqueda a gran escala porque la varianza del vuelo de Lévy aumenta mucho más rápido que por ejemplo la varianza de las caminatas aleatorias Brownianas. La mejor forma de implementar de forma eficiente el vuelo de Lévy en el código del algoritmo del Cuckoo es a través del algoritmo de Mantegma (véase ecuación 2.54); este determina la distribución simétrica estable de Lévy, es decir que los pasos de la caminata aleatoria pueden ser positivos y negativos. Entonces, la longitud del paso se define como:

$$s = \frac{u}{|v|^{1/\beta}}, \tag{2.54}$$

donde u, v provienen de distribuciones normales tales que $u \sim \mathcal{N}(0, \sigma_u^2)$ y $v \sim \mathcal{N}(0, \sigma_v^2)$ (véase ecuaciones 2.55 y 2.56), las varianzas se definen como:

$$\sigma_u^2 = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\beta\Gamma\left[(1+\beta)/2\right]2^{(\beta-1)/2}} \right\}^{1/\beta}, \tag{2.55}$$

$$\sigma_v^2 = 1. \tag{2.56}$$

Por todo lo anterior, la ventaja adicional que tiene el algoritmo (CKLF) es que su búsqueda global utiliza los vuelos de Lévy en sus caminatas aleatorias, esto lleva al algoritmo a explorar la superficie de búsqueda de forma más eficiente que aquellos algoritmos que utilizan la caminata aleatoria estándar con distribución gaussiana. Esta ventaja combinada con una búsqueda local y convergencia global garantizada hacen del Cuckoo un algoritmo muy eficiente [26].

Para la implementación del algoritmo del Cuckoo se consideran tres reglas generales: la primera, cada Cuckoo pone un huevo (solución) a la vez en un nido al azar; la segunda, los mejores nidos con huevos de alta calidad se mantienen hasta la siguiente generación (próxima iteración); la tercera, la cantidad de nidos huéspedes n es fijo. El ave huésped descubre los huevos del Cuckoo con una probabilidad de $p_a \in [0, 1]$ y en este caso el ave huésped simplemente abandona el nido y construye

uno nuevo [26]. También, se llamará solución a todo nido huésped que tenga un huevo de un Cuckoo.

Por otra parte, la población inicial donde los Cuckoos dejan sus primeros huevos en los nidos (véase ecuación 2.57) se define a través de una caminata aleatoria con distribución normal, la misma utilizada en los algoritmos metaheurísticos:

$$x = \varepsilon * (UB - LB) + LB., \tag{2.57}$$

donde LB y UB son las cotas inferiores y superiores del dominio del problema y x el vector de la población inicial de Cuckoo, el Algoritmo (CKLF) tiene dos características importantes, relacionadas con las caminatas aleatorias, una local y la otra global, para la local se define como:

$$x_i^{t+1} = x_i^t + \alpha\varepsilon \odot H(p_a - \varepsilon) \odot (g^* - x_k^t), \tag{2.58}$$

donde g^* es la mejor solución por iteración y x_k^t es una solución seleccionada de forma aleatoria a través de permutación aleatoria, α el factor de escalamiento, H la función escalón de Heaviside que analiza la diferencia entre la probabilidad de abandono de un nido con un huevo Cuckoo y valor aleatorio ε uniformemente distribuido; el producto de Hadamard ($\odot$) se define como: $u \odot v = u_{ij} v_{ij}$. La caminata aleatoria global se define a través del vuelo de Lévy.

$$x_i^{t+1} = x_i^t + \alpha L(s, \beta). \tag{2.59}$$

El vuelo de Lévy se define con la ecuación 2.52 (véase: pseudocódigo 2.6), α es el factor de escalamiento. El vuelo de Lévy garantizará que las soluciones no queden atrapadas en óptimos locales. Si se considera $Q = \alpha s \odot H(p_a - \varepsilon)$ en la caminata aleatoria local con $Q > 0$ se obtiene la mejor actualización de la Evolución Diferencial (*Differential Evolution* DE) o como una variante acelerada (APSO) sin su componente histórico. La caminata aleatoria global es, de hecho, la forma fundamental del algoritmo (SA), donde el esquema de enfriamiento es controlado por la probabilidad de abandono de los nidos p_a.

2.4. Métricas

Para las funciones de prueba se plantearon ciertas métricas ampliamente usadas en la literatura y distintos *frameworks* de *Machine Learning*. Como problema de optimización se consideró realizar 100 ejecuciones de cada algoritmo y obtener el promedio del tiempo que toma el algoritmo en realizar una ejecución, la cantidad de iteraciones por ejecución y la desviación estándar [22].

En el contexto del *Machine Learning*, las métricas de desempeño son ampliamente utilizadas para evaluar la calidad de los modelos en problemas de optimización [4]. Estas métricas han sido estudiadas en diversos campos y permiten una comparación objetiva entre diferentes modelos y sus

Algorithm 2.6 Pseudocódigo de *Cuckoo Search via Lévy Flights* (CKLF)

Require: *target*: Minimizar función objetivo
1: Función Objetivo $f(x)$, $x = (x_i, ..., x_p)^T$
2: Definir n nidos huéspedes con huevos
3: Crear población inicial x_i Ec. (2.57)
4: Evaluar soluciones iniciales
5: **while** Criterio de parada: Tolerancia **do**
6: Generar población con vuelo de Lévy $x_i^{t+1} = x_i^t + \alpha L(s, \beta)$ Ec. (2.59)
7: Evaluar soluciones
8: Una fracción p_a de los peores nidos son abandonados
9: Nuevas soluciones son generadas por: $x_i^{t+1} = x_i^t + \alpha s \odot H(p_a - \varepsilon) \odot (g^* - x_k^t)$ Ec. (2.58)
10: Mantener las mejores soluciones
11: Ordenar las soluciones encontrar la mejor solución por iteración g^*
12: $t = t + 1$ (contador de iteraciones)
13: **end while**
14: Procesar la información y Visualizar

predicciones con respecto a los datos reales o no vistos. En particular, en el caso de la optimización, estas métricas se utilizan para comparar la calidad de las soluciones propuestas por diferentes modelos, como SDG, NMS, PSO, Cuckoo Search, entre otros, y la solución teórica $\hat{y}$ de las funciones de prueba y señales de referencia en el caso del *Beamforming Adaptativo*.

Las métricas de error se definen como una forma genérica de medir la diferencia entre las predicciones del modelo y los datos reales o no vistos. En la literatura se han propuesto diversas métricas de desempeño, como la precisión, el error cuadrático medio, el coeficiente de determinación, entre otras, que permiten una evaluación adecuada del fenómeno.

$$\mathbb{M} = \underset{j=1,n}{\mathbb{G}^Z} \left\{ \mathbb{N}^Z \left[\mathbb{D}^Z \left(A_j, P_j \right) \right] \right\}, \tag{2.60}$$

donde A_j corresponde a la j-ésima solución proporcionada por el algoritmo dentro del problema de optimización, P_j la solución teórica suministrada por la función de prueba el problema, el valor Z indica el índice de la métrica utilizada no confundir con en símbolo de potenciación, n la cantidad de soluciones proporcionadas por el algoritmo, el cual depende en gran medida si el algoritmo es de solución única (SDG,CGM) o de población (PSO, BAT, CKLF) [4].

- **MSE:** métrica conocida como *Mean Squared Error* o error medio cuadrático, es una métrica con características tipológicas $\mathbb{G}1\left\{\mathbb{N}1\left[\mathbb{D}3\left(A_j, P\right)\right]\right\}$, dada por la siguiente ecuación:

Distancia $\mathbb{D}$	**Normalización** $\mathbb{N}$	**Agregación** $\mathbb{G}$
Error (Magnitud del error) $\mathbb{D}1 = A_j - P_j$	Normalización unitaria $\mathbb{N}1 = 1$	Agregación con media aritmética $\mathbb{G}1$
Error Absoluto $\mathbb{D}2 = \lvert A_j - P_j \rvert$	Normalización actual $\mathbb{N}2 = A_j^{-c}$	Agregación con mediana $\mathbb{G}2$
Error Cuadrático $\mathbb{D}3 = (A_j - P_j)^2$	Normalización de variabilidad actual $\mathbb{N}3 = \left(A_j - \bar{A}\right)^{-c}$	Agregación con media geométrica $\mathbb{G}3$
Error de cociente logarítmico $\mathbb{D}4 = \log\left(\frac{A_j - 1}{P_j - 1}\right)$		Agregación con suma $\mathbb{G}4$

Cuadro 2.1: Tabla de tipología de las métricas.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2 . \tag{2.61}$$

- **RMSE:** métrica conocida como *Root Mean Squared Error* o error medio cuadrático de la raíz, es una métrica con características tipológicas $\mathbb{G}1\{\mathbb{N}1[\mathbb{D}3(A_j, P)]\}$, dada por la siguiente ecuación:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2} = \sqrt{MSE}. \tag{2.62}$$

- **MAE:** métrica conocida como *Mean Absolute Error* o error medio absoluto, es una métrica con características tipológicas $\mathbb{G}1\{\mathbb{N}1[\mathbb{D}2(A_j, P)]\}$, dada por la siguiente ecuación:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\lvert y_i - \hat{y}\rvert . \tag{2.63}$$

- **MSPE:** métrica conocida como *Mean Squared Percentage Error* o error medio porcentual, es una métrica con características tipológicas $100\mathbb{G}1\{\mathbb{N}2[\mathbb{D}3(A_j, P)]\}$ con $c = 2$, dada por la siguiente ecuación:

$$MSPE = \frac{100\,\%}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{y}}{y_i}\right)^2 . \tag{2.64}$$

- **MAPE:** métrica conocida como *Mean Absolute Percentage Error* o error absoluto porcentual, es una métrica con características tipológicas $100\mathbb{G}1\{\mathbb{N}2[\mathbb{D}2(A_j, P)]\}$ con $c = 1$, dada por la siguiente ecuación:

$$MAPE = \frac{100\,\%}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}}{y_i} \right|. \tag{2.65}$$

- **RMSLE:** métrica conocida como *Root Mean Squared Logarithmic Error* o error medio cuadrático logarítmico, es una métrica con características tipológicas $\mathbb{G}1\left\{\mathbb{N}1\left[\mathbb{D}4\left(A_j, P\right)\right]\right\}$ con $c = 1$, dada por la siguiente ecuación:

$$RMSLE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left[\log\left(y_i - 1\right) - \log\left(\hat{y} - 1\right)\right]^2}. \tag{2.66}$$

2.5. Criterio de parada

Para la construcción de los algoritmos se establecieron dos criterios de parada, El primer criterio considera dos bloques con dos bloques cada uno, estos bloques están determinados por las características de corto circuito (Short Circuit) de los operadores lógicos AND (&&) y OR (||). El primer bloque se utiliza para continuar la ejecución del algoritmo y se activa cuando la perturbación es mayor a la tolerancia, lo que indica que el algoritmo aún está explorando la superficie sin encontrar el óptimo.

El segundo bloque considera evaluar un posible estancamiento en las soluciones y se activa cuando el error medio cuadrático es el mismo que en la iteración anterior (este elemento esta en el segundo bloque por superficies que son planas esto significa que el algoritmo esta explorando pero el error en la iteración anterior es el mismo que la iteración actual) y el otro elemento se activa cuando se ha alcanzado el número máximo de iteraciones.

$$(perturb \geq tol \parallel mse(t) \geq tol) \;\&\&\; (mse_{t-1} == mse_t \;\&\&\; t \leq max_{iter}). \tag{2.67}$$

El secundario o criterio de éxito (*Successful rate*), considera 3 elementos: El primero si el error medio cuadrático de la iteración actual es menor a la tolerancia indicando que la solución o las soluciones se están acercando al valor teórico de la función de prueba o señal de referencia, el segundo, si en promedio todas las soluciones menos el valor teórico de la función de prueba o señal de referencia son más pequeñas que la tolerancia esto define una vecindad sobre el valor teórico y nos indica que las soluciones están convergiendo al valor teórico y la tercera si la perturbación es menor a la tolerancia. En el caso de los algoritmos de gradiente se agrega la norma del gradiente menor a la iteración.

$$mse_t < tol \;\&\&\; \bar{y} - f(x^*) < tol \;\&\&\; perturb < tol. \tag{2.68}$$

Si se cumple esta condición entonces se incrementa el contador de éxito y termina la ejecución del algoritmo. La implementación de estos criterios de parada se hace en el software MATLAB en configuración corto circuito *(short circuit)* y dependiendo del algoritmo se pueden agregar o quitar ciertos elementos del criterio de éxito, las configuraciones en corto circuito consideradas son:

2.6. Orden de Convergencia

Planteamos la tasa de convergencia basados en la definición de [21] y [5] para tener una métrica que describa el desempeño de los algoritmos, para esto considere el vector $\mathbf{x}$ de longitud n y la sucesión $\{x_k\}_{k=0}^{\infty}$ de n vectores, $\|\cdot\|$ la norma de n vectores. Entonces $\{x_k\}_{k=0}^{\infty}$ converge a $\mathbf{x}$ con $\mathbf{x}_k \neq \mathbf{x}$ si y solo si $\forall \varepsilon > 0$ existe un entero positivo K tal que $\forall k > K$ tenemos:

$$\|\mathbf{x}_k - \mathbf{x}\| \leq \varepsilon. \tag{2.69}$$

Entonces de forma constructiva se puede decir que la sucesión $\{x_k\}_{k=0}^{\infty}$ es linealmente convergente a $\mathbf{x}$ si y solo si existe un entero $K > 0$ y un factor de reducción del error $M \in [0, 1)$ tal que $\forall k > K$ tenemos:

$$\|\mathbf{x}_{k+1} - \mathbf{x}\| \leq M \|\mathbf{x}_k - \mathbf{x}\|. \tag{2.70}$$

De forma general podemos expresar que la sucesión tiene una tasa de convergencia p (orden de convergencia p) a $\mathbf{x}$ si y solo si hay un escalar $p \geq 1$, un entero $K \geq 0$ y un escalar $M \geq 0$ tal que $\forall k > K$:

$$\|\mathbf{x}_{k+1} - \mathbf{x}\| \leq M \|\mathbf{x}_k - \mathbf{x}\|^p, \tag{2.71}$$

a través de la idea de limite podemos expresar que:

$$\lim_{k \to \infty} \frac{|\mathbf{x}_{k+1} - \mathbf{x}|}{|\mathbf{x}_k - \mathbf{x}|^p} = M. \tag{2.72}$$

Para el caso del error vamos a considerar la expresión $e_k = \|y_k - \hat{y}\|_p$, donde y_k es el vector de salida del algoritmo en la iteración k y $\hat{y}$ es el valor teórico o de referencia, basados en la ecuación 2.71 y sabiendo que la sucesión del error converge a cero tenemos:

$$e_{k+1} = M e_k^p. \tag{2.73}$$

para e_k tenemos:

$$\|e_k\| = M \|e_{k-1}\|^p , \tag{2.74}$$

dividiendo (2.73) entre (2.74)

$$\frac{e_{k+1}}{e_k} = \frac{e_k^p}{e_{k-1}^p}, \tag{2.75}$$

luego aplicando las propiedades de los logaritmos tenemos que:

$$p = \frac{\ln\left(\frac{e_{k+1}}{e_k}\right)}{\ln\left(\frac{e_k}{e_{k-1}}\right)}, \tag{2.76}$$

de (2.73) obtenemos M

$$M = \frac{e_{k+1}}{e_k^p}. \tag{2.77}$$

Para evaluar el orden de convergencia de los algoritmos analizamos las curvas de error (*RMSE Root Mean Squared Error*), estas curvas tienen características estocásticas dada la naturaleza de sus caminatas aleatorias en el caso de los AM, en la selección del x_0 de los métodos de gradiente y en el caso de Nealder-Mead la selección inicial de los vértices del poliedro. Con esta idea buscamos analizar el error de los métodos como una serie de tiempo estocástica

Para la construcción de la curva final de error *RMSE* consideramos de las 100 ejecuciones aquellas que fueron exitosas de cada algoritmo haciendo un promedio dinámico del comportamiento por cada iteración, este comportamiento está asociado a que cada ejecución exitosa puede tener más o menos iteraciones, a esta curva final se le calcula la desviación estándar por iteración y se construye una banda que define el área del error.

Para tener una mejor visualización del error se consideran las iteraciones efectivas, es decir, aquellas que son mayores o iguales a la tolerancia 1×10^{-6}, esto es necesario porque el criterio de parada considera varios elementos y es posible que el algoritmo por satisfacer la condición de corto circuito tenga errores más pequeños y aumente sustancialmente la cantidad de iteraciones y en ocasiones teniendo la condición $e_k = e_{k-1}$. Esta condición en especial genera un error en la ecuación (2.76) que hace el denominador igual a cero.

Para encontrar el orden de convergencia en esta serie de tiempo es necesario hacer la descomposición de la serie calculando su tendencia, para esto consideramos a la serie Y y asignamos el

esquema de agregación aditivo como estándar para todas las series de tiempo de los errores, donde $Y = (T+C)+E+I$, y sus componentes aditivos son la tendencia (T), ciclo (C), estacionalidad (E) y su componente irregular (I). La tendencia la calculamos a través del algoritmo Análisis del espectro singular (*singular spectrum analysis (SSA)*) utilizando la función de MATLAB® *trenddecomp*$(\cdot)$, este algoritmo es muy útil cuando se desconocen los períodos de las tendencias estacionales.

Con esta tendencia de la serie hacemos los cálculos para determinar el orden de convergencia p con la ecuación 2.76 y la constante M con la ecuación 2.77 en cada iteración y al final establecemos el promedio, esta información en el caso de $\mathbb{R}^2$ se agrega a dos gráficas de errores: error versus cantidad de iteraciones y $10\log(RMSE)$ versus número de iteraciones.

2.7. Experimento Beamforming Adaptativo

El experimento realizado se basó en el anteproyecto y en la sección "*Beamforming* adaptativo" (ver subsección 2.1.3). Se utilizó un arreglo rectangular uniforme de 64 elementos isotrópicos *(URA: Uniform Rectangular Array)* con un tamaño de 8×8, y una distancia entre elementos de 0,5 longitudes de onda. La configuración permitió la formación de lóbulos de rejilla. El ángulo de cobertura de la antena abarcó desde 0 hasta 180 grados en el eje azimutal (ángulo θ) y de 0 hasta 360 grados en el eje de elevación (ángulo ϕ). Se consideraron 3 señales que llegaron a la antena: la señal deseada o de interés *(SOI: Signal of Interest)* y dos señales interferentes *(SNOI: Signal Not of Interest)*. Estas señales presentaban 3 características: potencia y ángulos de llegada en (θ, ϕ).

En la literatura se reportan distintos tipos de funciones objetivo, varias relacionan el error como es el caso de *Mean Squared Error (MSE)*, *Root Mean Square Error (RMSE)*, *Mean Absolute Error (MAE)* y *Signal-to-Interference-plus-Noise Ratio (SINR)* ampliamente utilizadas con algoritmos adaptativos clásicos como el *Least Mean Squares (LMS)* y filtros de Kalman [16]. Para el caso de CGM se consideró la función objetivo $J(w_{mn}) = \frac{1}{2}w^H Aw - s^H \bar{w}$ que tiene como gradiente a $\nabla J(w_{mn}) = Aw - s$ con A la matriz dirección del arreglo [16].

Para los demás algoritmos se planteó una función objetivo para encontrar los pesos complejos adecuados con el fin de ubicar el lóbulo principal en la dirección de la señal deseada y minimizar la ganancia hacia las señales interferentes. Para lograr esto, en la ecuación de la fase progresiva (véase 2.3 - 2.4) se consideraron $\beta_x = \beta_y = 0$ y por cada señal que entraba en la antena, se definió una función objetivo [3]. La primera función objetivo estaba relacionada con la magnitud del factor de arreglo en la dirección deseada $|AF(\theta_{s0}, \phi_{s0})|$ y las otras dos funciones objetivo $|AF(\theta_{i1}, \phi_{i1})|$, $|AF(\theta_{i2}, \phi_{i2})|$ estaban relacionadas con las señales interferentes. La intención de la función objetivo era maximizar la ganancia del factor de arreglo hacia la señal deseada y minimizar la ganancia hacia las señales interferentes.

$$\min_{b_{m,n}} - \left| AF(\theta_{i1}, \phi_{i1}) - [AF(\theta_{i1}, \phi_{i1}) + AF(\theta_{i2}, \phi_{i2})] \right|, \tag{2.78}$$

$$W_{mn} = \sum_{m=1}^{M} \sum_{n=1}^{N} e^{jb_{m,n}}, \tag{2.79}$$

basados en la ecuación de arreglo (véase 2.9) tenemos:

$$AF(\theta_{s0}, \phi_{s0}) = \sum_{m=1}^{M} \sum_{n=1}^{N} e^{jb_{m,n}} e^{j[(m-1)(kd_x \sin\theta_{s0} \cos\phi_{s0}) + (n-1)(kd_y \sin\theta_{s0} \sin\phi_{s0})]}, \tag{2.80}$$

$$AF(\theta_{i1}, \phi_{i1}) = \sum_{m=1}^{M} \sum_{n=1}^{N} e^{jb_{m,n}} e^{j[(m-1)(kd_x \sin\theta_{i1} \cos\phi_{i1}) + (n-1)(kd_y \sin\theta_{i1} \sin\phi_{i1})]}, \tag{2.81}$$

$$AF(\theta_{i2}, \phi_{i2}) = \sum_{m=1}^{M} \sum_{n=1}^{N} e^{jb_{m,n}} e^{j[(m-1)(kd_x \sin\theta_{i2} \cos\phi_{i2}) + (n-1)(kd_y \sin\theta_{i2} \sin\phi_{i2})]}. \tag{2.82}$$

Para el vector de peso de desplazamiento de fase $b_{m,n}$ en el elemento (m, n), consideramos un rango de valores entre -2π y 2π, es decir, $b_{m,n} \in [-2\pi, 2\pi]$. Estos valores son necesarios para establecer las posiciones iniciales x_0 en los métodos de gradiente (CGD, SGD), definir el poliedro para el NMS y establecer las poblaciones iniciales en los algoritmos metaheurísticos (PSO, BA, CKLF).

Aunque se utilizaron los criterios de parada de las funciones de prueba, se encontró que no eran adecuados para el problema del *Beamforming* debido a su complejidad, dimensionalidad y función objetivo. En lugar de estos criterios, se utilizó el criterio de parada ampliamente reportado en la literatura del *Beamforming*: "cantidad máxima de iteraciones". Cada algoritmo se ejecutó durante 1000 iteraciones. No se consideró el porcentaje de error y la tolerancia como criterios de parada en los algoritmos metaheurísticos debido a la complejidad del problema del Beamforming, la naturaleza de la función objetivo seleccionada, los parámetros de sintonización de los algoritmos y los posibles estancamientos que impiden establecer estos criterios de parada como puntos de comparación futura.

Para validar los resultados obtenidos por los algoritmos se consideró el mejor vector de pesos W_{mn} y su diagrama de radiación en el ángulo ϕ, identificando la señal deseada, las señales interferentes, el ancho de banda de la potencia media. Por efectos de simetría del factor de arreglo y en general de la función objetivo, se buscaron valores de (θ, ϕ) que no castigaran a los algoritmos, por lo cual se establecieron valores simétricos con valores 20°, 40° y 60° para θ y 70°, 90° y 110° para ϕ.

La información de los ángulos de llegada *(AOA Angle of Arrival)* es ciega para los algoritmos, lo cual significa que dentro del problema de optimización los algoritmos no saben las direcciones a

donde concentrar la energía. Se consideraron 6 algoritmos 3 con un enfoque clásico y 3 metaheurísticos, los cuales realizaron la búsqueda exploratoria sobre la superficie de la función objetivo y se determinó la matriz de pesos complejos del arreglo que maximiza la potencia sobre la señal deseada y la minimiza sobre las señales interferentes.

AL final se consideraron los siguientes parámetros para los algoritmos clásicos: para CGM se consideró β con la forma conjugado descendiente, x_0 aleatorio. Para SGD se consideró una taza de aprendizaje o tamaño de paso de $\alpha = 0{,}1$, x_0 aleatorio. Para NMS se consideró la reflexión $\rho = 1$, la expansión $\chi = 2$, la contracción $\psi = 0{,}5$ y la compresión $\sigma = 0{,}5$. Para los AM se consideró una población de $n = 40$ y los siguientes ajustes en los parámetros: En el caso de PSO se definieron las constantes de aceleración $\alpha = 0{,}3$, $\beta_{PSO} = 0{,}5$, y la inercia $\theta = 0{,}95$. En el caso de BA, las constantes de Volumen y emisión de pulsos $\alpha = 0,6$, $\gamma = 0,50$, Constante de aceleración $\beta = 0{,}3$ y la inercia $\theta = 0,95$, el volumen $A \in [1, 2]$ y la frecuencia $f \in [0, 2]$. Por último, el algoritmo Cuckoo con vuelo de Lévy se define una probabilidad de rechazo de nidos $pa = 0,25$, un factor de escalamiento $\alpha = 0,45$ y la constante $\beta = 3/2$ de la función Gamma.

2.8. Glosario

Se establecen las definiciones fundamentales para entender la metodología del trabajo de grado.

- **MIMO Masivo:** *Multiple Input, Multiple Output* Es una tecnología de múltiples entradas y múltiples salidas, con ventajas como permitir múltiple transmisión de datos para una alta eficiencia espectral, mejorar la calidad del enlace y la adaptación a los patrones de radiación para la ganancia de la señal y reducción de la interferencia a través de *Beamforming* adaptativo usando arreglo de antenas [17].

- **Beamforming:** Conformación del haz de la onda electromagnética, en la cual se busca dirigir el haz generado por el arreglo y aumentando la potencia a los usuarios. El *Beamforming* ayuda a reducir la interferencia interceldas aumentando la calidad del servicio.

- **Algoritmos Metaheurísticos:** Son algoritmos inspirados mayoritariamente en la naturaleza y ciertas abstracciones de esta. Estos algoritmos tienen dos componentes: la selección de las mejores soluciones asegura que el algoritmo converja a la optimalidad, mientras que la aleatoriedad garantiza que las soluciones no queden atrapadas en óptimos locales. Otra forma de clasificas los AM son basados en su trayectoria como el caso de SA que es una solución que se mueve sobre la superficie o basados en su población como PSO que es un conjunto de soluciones (partículas) que se mueven sobre la superficie.

- **Arreglos planos:** Arreglo de antenas que se ubican sobre una geometría rectangular/plana. Los arreglos planos proveen variables adicionales las cuales pueden ser usadas para controlar y modificar la forma del patrón de radiación del arreglo. Adicionalmente los arreglos planos son muy versátiles con lóbulos laterales bajos [2].

- **Usuarios:** Para efectos prácticos consideramos los Usuarios (k) como los receptores de la onda electromagnética distribuidos de la siguiente forma: Usuario estático, aquel que su posición no varía en el tiempo, como puede ser el caso de la telefonía fija celular, Usuario nómada, aquel que es temporalmente estático y el usuario móvil aquel usuario que entra al área de cobertura de celda, pero presenta una caminata aleatoria sobre el área de cobertura.

- **Eficiencia Espectral:** Corresponde a la cantidad información que se puede transmitir de cierto ancho de banda, indicando que tan eficiente es el ancho de banda utilizado para trasmitir información de un lado a otro. Se define como $E = R/B$, donde R corresponde a la cantidad de $bits/s$ y el ancho de banda B en la cual se va a transmitir la información en Hz. La eficiencia espectral se expresa en términos de $(bits/s)/Hz$.

- **Main Lobe o Major Lobe:** Es el lóbulo principal que contiene la mayor potencia (ver Figura 2.12).

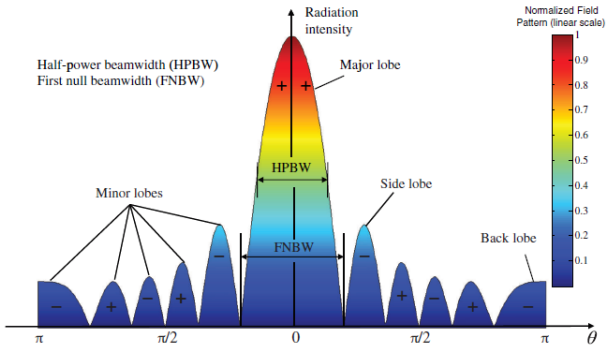


Figura 2.12: Patrón de radiación en escala lineal *fuente: [2]*.

- **Minor lobes:** Son todos los lóbulos distintos del lóbulo principal y usualmente representa la radiación hacia direcciones no deseadas (ver Figura 2.13).

- **Side Lobe:** Son los lóbulos laterales y adyacentes al lóbulo principal (ver Figura 2.12), su dirección es cualquiera excepto la del lóbulo deseado, se encuentran en el hemisferio donde se encuentra el lóbulo principal, son los lóbulos de mayor tamaño de los *Minor lobes*, el nivel de los lóbulos menores usualmente se expresan como una relación de la densidad de potencia entre el lóbulo menor y el lóbulo mayor, esta relación suele llamarse relación de lóbulo lateral (Side Lobe Ratio SLR) o nivel de lóbulo lateral (Side Lobe Level SLL) [2].

- **Back Lobe:** Es el lóbulo de radiación que se encuentra en la parte trasera y forma un ángulo aproximado de 180 grados con el lóbulo principal, también se dice que es el lóbulo que se encuentra en el hemisferio opuesto del lóbulo principal. (ver Figura 2.12).

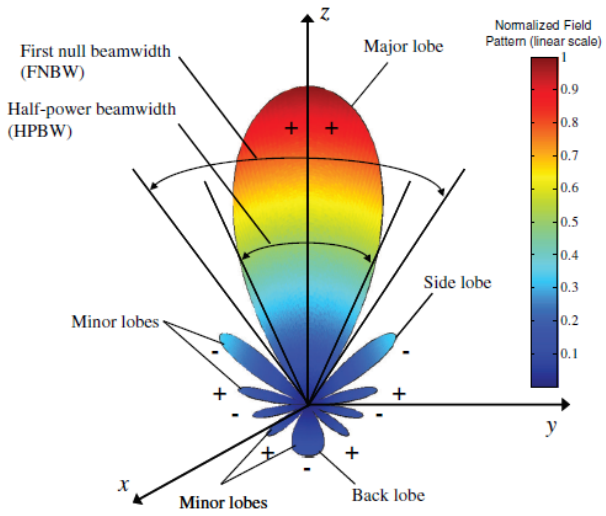


Figura 2.13: Patrón de radiación en forma polar *fuente: [2].*

- **Nulls:** Corresponde a la separación de los lóbulos, es donde se encuentra la menor potencia (ver Figura 2.12). estos valores suelen ser muy pequeños por eso se hace una representación logarítmica para tener una mejor comprensión del fenómeno.
- **HPBW:** HPBW o Half Power Beam Witdth corresponde al ancho de banda o separación angular del lóbulo principal cuando la potencia cae a la mitad de su valor máximo, en escala logarítmica (decibeles) corresponde cuando decae a -3 dB's, esta medida generalmente se encuentra en grados o radianes. Se suele visualizar en dos esquemas de radiación en función te θ para ciertos valores de ϕ y viceversa. (ver Figura 2.14).

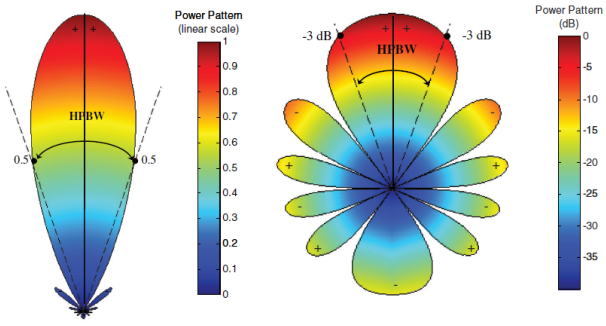


Figura 2.14: Patrón de radiación de un arreglo lineal de 10 elementos y $d = 0{,}25\lambda$. Izquierda: Escala Lineal, Derecha: Escala en dB's *fuente: [2].*

- **FNBW:** FNBW o First Null Beam Witdth corresponde al ancho de banda del lóbulo principal

entre sus primeros nulos esta medida generalmente se encuentra en grados o radianes (ver Figura 2.12).

Capítulo 3

Resultados y discusión

3.1. Resultados funciones prueba

Se realizó la comparación los algoritmos clásicos (CGM), (SGD) y (NMS) y metaheurísticos (PSO), (BA) y (CKLF) en las distintas funciones de prueba; esta implementación se realizó en MATLAB® R2022B. Los algoritmos se ejecutaron 100 veces para tener información suficiente para el análisis estadístico. De las 100 ejecuciones por función se obtiene el tiempo promedio de ejecución, la cantidad promedio de iteraciones, su desviación estándar, tasa de éxito (cuántas soluciones llegan de forma efectiva al óptimo) y las distintas métricas de errores con una tolerancia fija de 10^{-6}. Para (CGM) consideramos un máximo de iteraciones de 10000 en los demás solo se consideró 1000.

En el caso de CGM *(véase tablas 3.1, 3.4)* se puede ver el comportamiento del algoritmo en las 4 funciones de prueba en $\mathbb{R}^2$ y $\mathbb{R}^{10}$. El algoritmo presentó un mal desempeño en general con una tasa de éxito del 88 % en la función 1 *(véase figura 4.1)* con (1227 ± 3256) cantidad de iteraciones. En las demás funciones presenta un desempeño pobre con una taza de éxito máxima de 26 % en la segunda función y en la cuarta función solo 2 ejecuciones alcanzan el óptimo. *(véase figura 4.4)*.

Fun	Tiempo (s)	μ_{iter}	$\pm\sigma$	SR (%)
Fun. (2.16)	13.61	1227	3256	88
Fun. (2.17)	34.77	8273	3170	26
Fun. (2.18)	32.74	9098	2720	10
Fun. (2.19)	29.48	9809	1343	2

Cuadro 3.1: Desempeño CGM: Métricas estadísticas del algoritmo en 2D

Para el análisis de los errores, consideramos los errores de las ejecuciones exitosas, para esto se consideró el ultimo valor del error (ultima iteración) que posteriormente se promedia con todos los errores luego se multiplica por la tolerancia 1×10^{-6} (aproximadamente $-138{,}1551 dB$) para tener una sensibilidad del error y poderlo visualizar, los errores que aparecen como 0,00 no son cero, esto se da al redondeo que se hace para poder visualizar el error. En los errores *MSPE* (Error de porcentaje cuadrático medio) y *MAPE* (Error porcentual absoluto medio) todos los algoritmos tienen el mismo valor de 1, esto se debe a que el valor teórico de las funciones de prueba $f(x^*) = 0$, de ahora en adelante vamos a omitir los resultados de estos errores.

Fun	MSE (1×10^{-6})	RMSE (1×10^{-6})	MAE (1×10^{-6})	MSPE	MAPE	RMSLE (1×10^{-6})	p	M
Fun. (2.16)	0.00	0.00	0.00	1	1	0.00	1.07	8.72
Fun. (2.17)	0.00	0.08	0.08	1	1	0.00	1.00	1.01
Fun. (2.18)	0.01	111.33	111.33	1	1	0.01	0.97	5.57
Fun. (2.19)	0.00	0.00	0.00	1	1	0.00	1.00	1.16

Cuadro 3.2: Desempeño CGM: Métricas de errores del algoritmo en 2D

En el caso de CGM solo presenta un valor grande de 111.33 en *RMSE* y *MAE* en la función 3, estas dificultades están asociadas al tipo de función, donde el algoritmo tuvo una tasa de éxito del 10 %. Se evidenció que el algoritmo CGM tiene en promedio un orden lineal de convergencia $p = 1$ *(véase figura 3.1)* con una gran cantidad por encima de las 1200 iteraciones en las funciones de prueba. En el caso de la primera función el algoritmo en sus 88 ejecuciones exitosas se demora aproximadamente 70 iteraciones para alcanzar $-70dBs$ con leves características oscilatorias. En el caso de la función 2, el algoritmo en sus 26 ejecuciones exitosas realiza la búsqueda sobre la superficie con grandes dificultades pues necesita alrededor de 8000 iteraciones para explorar y aproximadamente 1000 iteraciones para encontrar el óptimo.

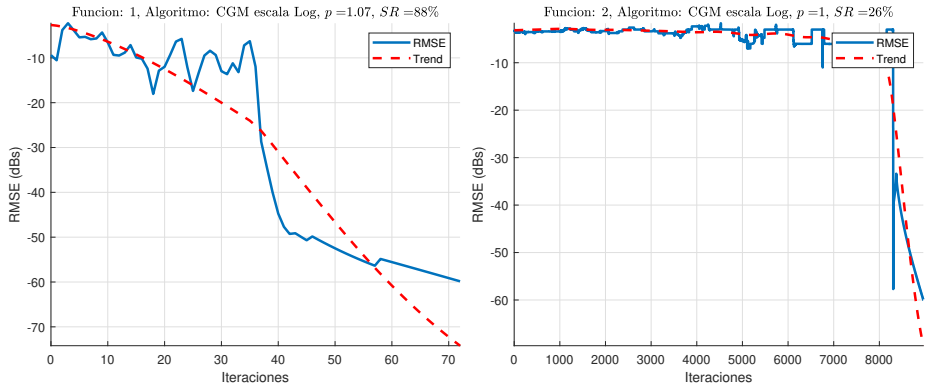


Figura 3.1: Error *RMSE* de CGM en escala Logarítmica. Fuente: *Autor*.

En el caso de las funciones en $\mathbb{R}^{10}$ CGM no tiene ningún problema con la función 1 con una tasa de éxito de 100 % con tan solo (42 ± 31) iteraciones. Las funciones restantes el algoritmo no alcanza el óptimo saliendo de la ejecución a través del criterio de parada máximo número de iteraciones, en el caso de CGM se consideró 1000 iteraciones.

Fun	MSE (1×10^{-6})	RMSE (1×10^{-6})	MAE (1×10^{-6})	MSPE	MAPE	RMSLE (1×10^{-6})	p	M
Fun. (2.16)	0.00	0.00	0.00	1	1	0.00	0.99	6.98

Cuadro 3.3: Desempeño CGM: Métricas de errores del algoritmo en 2D

En el caso de CGM se evidencia que el algoritmo tiene en promedio un orden lineal de convergencia $p = 0{,}97$ con un mejor desempeño en cantidad de iteraciones con la misma función 1 en $\mathbb{R}^2$. En general se evidencia que CGM tiene lineal de convergencia y a su vez presenta muchas dificultades para encontrar el óptimo en las demás funciones con resultados pobres o nulos.

Fun	Tiempo (s)	μ_{iter}	$\pm\sigma$	SR (%)
Fun. (2.16)	28.14	42	31	100
Fun. (2.17)	189.28	1000	0	0
Fun. (2.18)	215.50	1000	0	0
Fun. (2.19)	89.28	1000	0	0

Cuadro 3.4: Desempeño CGM: Métricas estadísticas del algoritmo en 10D

En el caso de SGD *(véase tablas 3.5, 3.7)* se puede ver el comportamiento del algoritmo en las 4 funciones de prueba en $\mathbb{R}^2$ y $\mathbb{R}^{10}$. El algoritmo presentó un mal desempeño en general con una tasa de éxito del 100 % en la función 1 con (100 ± 10) iteraciones *(véase figura 4.5)*, en las demás funciones no alcanza el óptimo saliendo de la ejecución a través del criterio de parada máximo número de iteraciones. En las demás funciones el algoritmo se estanca en los valles de la función 2 *(véase figura 4.6)* y en las curvas de nivel de la función 3 *(véase figura 4.7)*.

Fun	Tiempo (s)	μ_{iter}	$\pm\sigma$	SR (%)
Fun. (2.16)	8.06	100	10	100
Fun. (2.17)	16.78	1000	0	0
Fun. (2.18)	12.47	1000	0	0
Fun. (2.19)	11.96	1000	0	0

Cuadro 3.5: Desempeño SGD: Métricas estadísticas del algoritmo en 2D

En el caso de SGD presentó dificultad en las funciones 1,2 y 3 asociadas a la naturaleza de cada función siendo imposible encontrar el óptimo. Se evidenció que el algoritmo tiene en promedio un orden lineal de convergencia $p = 1$ *(véase figura 3.2)*, con el clásico comportamiento reportado en la literatura de los métodos de gradiente donde las 100 ejecuciones exitosas convergen al óptimo en aproximadamente 80 iteraciones $(-70dBs)$.

Fun	MSE (1×10^{-6})	RMSE (1×10^{-6})	MAE (1×10^{-6})	MSPE	MAPE	RMSLE (1×10^{-6})	p	M
Fun. (2.16)	0.00	0.02	0.02	1	1	0.00	1.00	0.86

Cuadro 3.6: Desempeño SGD: Métricas de errores del algoritmo en 2D

En el caso de las funciones en $\mathbb{R}^{10}$ SGD no tiene ningún problema con la función 1 con una tasa de éxito de 100 % con tan solo (209 ± 24) iteraciones. Las funciones restantes el algoritmo

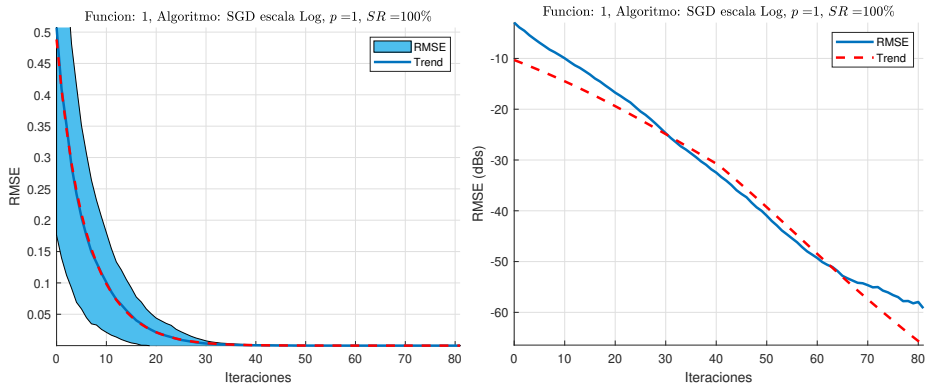


Figura 3.2: Error *RMSE* de SGD en función 1. Fuente: *Autor*.

no alcanza el óptimo saliendo de la ejecución a través del criterio de parada máximo número de iteraciones, este comportamiento es muy parecido al obtenido con el algoritmo CGM. El orden de convergencia en $\mathbb{R}^{10}$ es lineal, al igual que en $\mathbb{R}^2$.

Fun	Tiempo (s)	μ_{iter}	$\pm\sigma$	SR (%)
Fun. (2.16)	27.46	209	24	100
Fun. (2.17)	89.74	1000	0	0
Fun. (2.18)	51.61	1000	0	0
Fun. (2.19)	28.18	1000	0	0

Cuadro 3.7: Desempeño SGD: Métricas estadísticas del algoritmo en 10D

Fun	MSE (1×10^{-6})	RMSE (1×10^{-6})	MAE (1×10^{-6})	MSPE	MAPE	RMSLE (1×10^{-6})	p	M
Fun. (2.16)	0.20	322.10	322.10	1	1	0.20	1.00	0.95

Cuadro 3.8: Desempeño SGD: Métricas de errores del algoritmo en 10D

En el caso de NMS este algoritmo no derivativo presenta mejor desempeño comparado con CGM y SGD, en el caso de $\mathbb{R}^2$ las dos primeras funciones presentan una tasa de éxito excepcional con (48 ± 2) iteraciones para la primera y (112 ± 92) para la segunda, siendo estas las iteraciones más bajas alcanzadas por un algoritmo clásico *(véase tabla 3.9)*. Los valores de 0 en las distintas métricas de error indican que esos valores están por debajo del valor de tolerancia establecido de 10^{-6}. El algoritmo presentó un mal desempeño en la función 3 y 4 general con estancamiento en curvas de nivel y óptimos locales *(véase figuras 4.11, 4.12)*.

Fun	Tiempo (s)	μ_{iter}	$\pm\sigma$	SR (%)
Fun. (2.16)	9.12	48	2	100
Fun. (2.17)	16.71	112	92	99
Fun. (2.18)	14.23	1000	0	0
Fun. (2.19)	10.83	991	94	1

Cuadro 3.9: Desempeño NMS: Métricas estadísticas del algoritmo en 2D

El orden de convergencia del algoritmo en las 3 funciones 1,2 y 4 es lineal con un valor cercano a 1, La función 1 no presenta dificultad *(véase figura 3.3)* y el comportamiento es parecido al obtenido por SGD, en la segunda función el algoritmo tiene características exploradoras de la superficie con 120 iteraciones y características explotadoras, es decir, el algoritmo escapa de óptimos locales, aunque eso implique aumentar error (0.5).

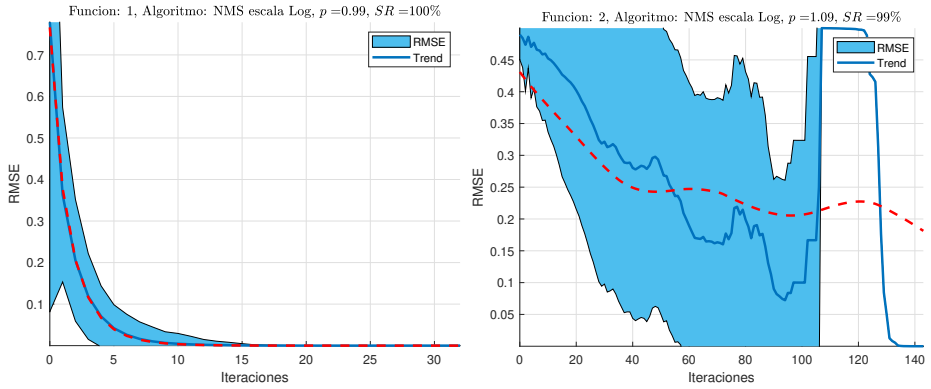


Figura 3.3: Error *RMSE* de NMS en función 1. Fuente: *Autor*.

Fun	MSE (1×10^{-6})	RMSE (1×10^{-6})	MAE (1×10^{-6})	MSPE	MAPE	RMSLE (1×10^{-6})	p	M
Fun. (2.16)	0.00	0.00	0.00	1	1	0.00	0.99	0.71
Fun. (2.17)	0.00	0.00	0.00	NaN	NaN	0.00	1.09	6223956.39
Fun. (2.19)	0.00	0.00	0.00	1	1	0.00	1.08	0.91

Cuadro 3.10: Desempeño NMS: Métricas de errores del algoritmo en 2D

En el caso de las funciones en $\mathbb{R}^{10}$ NMS no tiene ningún problema con la función 1 con una tasa de éxito de 78 % con (801 ± 150) iteraciones. Las funciones restantes el algoritmo no alcanza el óptimo saliendo de la ejecución a través del criterio de parada máximo número de iteraciones, este comportamiento es muy parecido al obtenido con el algoritmo CGM y SGD *(véase tabla 3.11)*. El algoritmo presenta un orden de convergencia lineal igual que en $\mathbb{R}^2$ con $p = 1$ y $M = 0{,}98$.

Fun	Tiempo (s)	μ_{iter}	$\pm\sigma$	SR (%)
$Fun.$ (2.16)	40.81	801	150	78
$Fun.$ (2.17)	93.59	1000	0	0
$Fun.$ (2.18)	64.07	1000	0	0
$Fun.$ (2.19)	27.36	1000	0	0

Cuadro 3.11: Desempeño NMS: Métricas estadísticas s del algoritmo en 10D

En el caso de PSO *(véase tabla 3.12)*, se puede ver el comportamiento del algoritmo en las 4 funciones de prueba. Como se puede ver en la función 1 *(véase figura 4.13)* tenemos una tasa de éxito de 100 % con (589 ± 59) iteraciones, y resultados pobres en la tasa de éxito de las demás funciones, llama la atención que el algoritmo si identifica el óptimo global *(véase figuras 4.15, 4.16)* pero el algoritmo no es capaz de satisfacer el criterio de parada (criterio de salida para indicar que fue una ejecución exitosa.) terminando la ejecución con el máximo número de iteraciones. No presenta estancamientos en óptimos locales y su desempeño puede mejorar realizando una sintonización de los hiperparámetros de PSO como α, β y la inercia θ.

Fun	Tiempo (s)	μ_{iter}	$\pm\sigma$	SR (%)
$Fun.$ (2.16)	10.25	589	59	100
$Fun.$ (2.17)	16.85	921	106	40
$Fun.$ (2.18)	14.11	1000	0	0
$Fun.$ (2.19)	9.76	993	26	9

Cuadro 3.12: Desempeño PSO: Métricas estadísticas del algoritmo en 2D

El algoritmo PSO presenta orden de convergencia lineal $p = 1$ en las funciones 1,2 y 4. con valores $M = 1$, donde la función 1 presenta los valores más pequeños respecto a MSE, RMSE y MAE. En la función 1 el algoritmo presenta un comportamiento oscilatorio y luego un comportamiento descendiente hasta la iteración 600 donde otra vez realiza un proceso oscilatorio evitando caer en un óptimo local, esta característica exploratoria la habíamos visto en el algoritmo NMS sacrificando el error.

Cabe resaltar que el proceso oscilatorio está asociado a la inteligencia colectiva (posición histórica) que maneja el algoritmo y de la inteligencia individual (mejor posición por iteración) en la población esto hace que de vez en cuando se dé un intercambio entre la inteligencia colectiva por la individual convirtiéndola a su vez en inteligencia colectiva (véase ecuación 2.43), esto modifica el vuelo de las partículas (movimiento conocido como inteligencia de enjambre *intelligent swarm*).

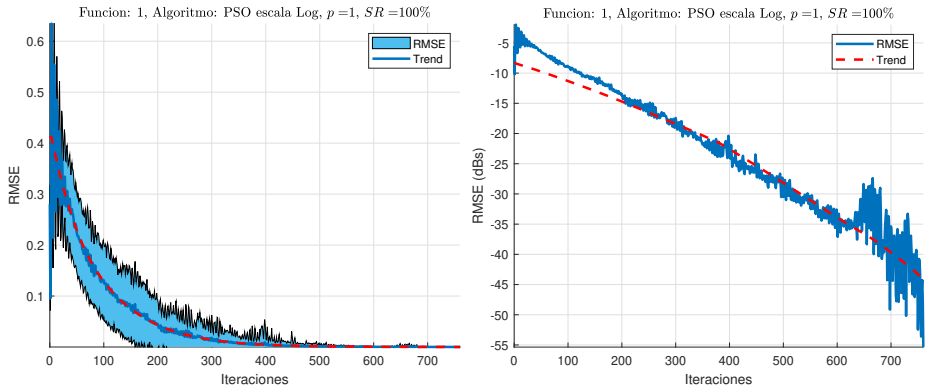


Figura 3.4: Error *RMSE* de PSO en función 1. Fuente: *Autor*.

Fun	MSE (1×10^{-6})	RMSE (1×10^{-6})	MAE (1×10^{-6})	MSPE	MAPE	RMSLE (1×10^{-6})	p	M
Fun. (2.16)	0.00	2.41	0.71	1	1	0.00	1.00	1.01
Fun. (2.17)	0.00	4.56	3.67	1	1	0.00	0.99	1.00
Fun. (2.19)	0.00	2.54	0.72	1	1	0.00	1.00	1.00

Cuadro 3.13: Desempeño PSO: Métricas de errores del algoritmo en 2D

En el caso de las funciones en $\mathbb{R}^{10}$ *(véase tabla 3.14)*, PSO no tiene ningún problema con la función 1 con una tasa de éxito de 100 % con (821 ± 55) iteraciones. Las funciones restantes el algoritmo no alcanza el óptimo saliendo de la ejecución a través del criterio de parada máximo número de iteraciones, al no tener la posibilidad de realizar una inspección visual se revisó el vector *y (fitness)* y presentó el mismo comportamiento de $\mathbb{R}^2$, es decir PSO identificó el óptimo global pero no cumplió las características del criterio de salida. El orden de convergencia también es lineal con $p = 1$ y $M = 0{,}99$.

Fun	Tiempo (s)	μ_{iter}	$\pm\sigma$	SR (%)
Fun. (2.16)	38.42	821	55	100
Fun. (2.17)	94.54	1000	0	0
Fun. (2.18)	60.38	1000	0	0
Fun. (2.19)	28.11	1000	0	0

Cuadro 3.14: Desempeño PSO: Métricas estadísticas del algoritmo en 10D

En el caso de BA *(véase tabla 3.15)*, se puede ver el comportamiento del algoritmo BA en las 4 funciones de prueba. Como se puede ver en la función 1 *(véase figura 4.13)* tenemos una tasa de éxito de 100 % con tan solo (66 ± 5) iteraciones, un buen desempeño en la segunda función con una tasa de éxito de 89 % con tan solo (66 ± 5) y un desempeño pobre en la función 3 con una tasa de

éxito de 1 % con tan solo (996 ± 51), al igual que el algoritmo PSO, el algoritmo identifica el óptimo global *(véase figuras 4.18, 4.19)* pero no es capaz de satisfacer el criterio de parada (criterio de salida para indicar que fue una ejecución exitosa.) terminando la ejecución con el máximo número de iteraciones.

Fun	Tiempo (s)	μ_{iter}	$\pm\sigma$	SR (%)
Fun. (2.16)	10.49	66	5	100
Fun. (2.17)	17.67	483	273	80
Fun. (2.18)	21.05	982	88	4
Fun. (2.19)	12.94	740	399	30

Cuadro 3.15: Desempeño BA: Métricas estadísticas del algoritmo en 2D

El algoritmo BA tiene orden de convergencia lineal con un valor aproximado de $p = 1$ como se puede observar en la tendencia de las imágenes con escala logarítmica *(véase figuras 3.5)*, en estas funciones el algoritmo presenta muy buenos desempeños y comportamiento generalizados de acuerdo con el área del error.

La forma oscilatoria de BA se debe a una búsqueda local que realiza el algoritmo en función de la mejor posición histórica y la tasa de emisión de pulsos promedio de los murciélagos (véase ecuación 2.48), por otra parte, realiza una búsqueda local inspirado en APSO (véase ecuación 2.46) repitiendo el esquema de intercambio entre colectivo e individual.

Fun	MSE (1×10^{-6})	RMSE (1×10^{-6})	MAE (1×10^{-6})	MSPE	MAPE	RMSLE (1×10^{-6})	p	M
Fun. (2.16)	0	1.47	0.90	1	1	0	0.99	376684.66
Fun. (2.17)	0	1.50	0.93	1	1	0	1.00	0.97
Fun. (2.18)	0	1.17	0.99	1	1	0	0.99	0.95
Fun. (2.19)	0	1.55	0.93	1	1	0	1.00	0.92

Cuadro 3.16: Desempeño BA: Métricas de errores del algoritmo en 2D

En el caso de la función 4 presenta un desempeño pobre con una tasa de éxito de 30 % y (740 ± 399) iteraciones y estancamientos en óptimos locales. El desempeño del algoritmo BA puede mejorar si se realiza un proceso de sintonización de los hiperparámetros considerando las constantes de Volumen α, emisión de pulsos γ, la constante de aceleración β y la inercia θ. Para las funciones en $\mathbb{R}^{10}$ *(véase tabla 3.17)*, el algoritmo presenta dificultad en todas las funciones teniendo un desempeño pobre.

Finalmente, en el caso de CKLF *(véase tabla resumen 3.18)*, se puede ver el comportamiento del algoritmo CKLF en las 4 funciones de prueba y su desempeño. El algoritmo tiene una tasa de éxito perfecta llegando al óptimo global en todas las funciones, solo en la función 2 presenta una

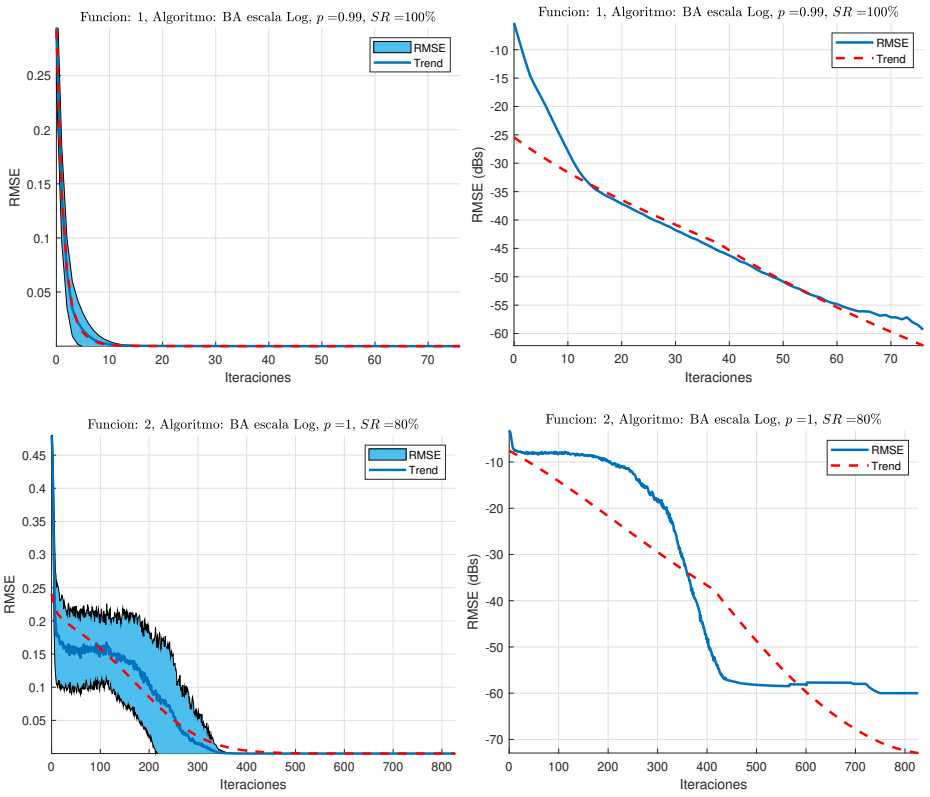


Figura 3.5: Error *RMSE* de BA en función 1. Fuente: *Autor*.

Fun	Tiempo (s)	μ_{iter}	$\pm\sigma$	SR (%)
Fun. (2.16)	88.47	977	130	3
Fun. (2.17)	103.09	1000	0	0
Fun. (2.18)	105.70	1000	0	0
Fun. (2.19)	37.86	1000	0	0

Cuadro 3.17: Desempeño BA: Métricas estadísticas del algoritmo en 10D

tasa de éxito de 98 % y (385 ± 192) iteraciones, con cantidad de iteraciones bajas muy similares a las reportadas por los demás algoritmos pero con tasas de éxito elevadas, con tiempos computacionales parecidos con diferencias mínimas de un par de segundos, esta situación se debe a las caminatas aleatorias a través de vuelos de Levy que le permite al algoritmo, escapar de óptimos locales y recorrer la superficie de búsqueda de forma óptima [22].

Fun	Tiempo (s)	μ_{iter}	$\pm\sigma$	SR (%)
Fun. (2.16)	8.61	29	5	100
Fun. (2.17)	19.40	385	192	98
Fun. (2.18)	17.58	342	68	100
Fun. (2.19)	10.18	113	43	100

Cuadro 3.18: Desempeño CKLF: Métricas estadísticas del algoritmo en 2D

En general podemos decir que el algoritmo que mejor se adaptó a las funciones de prueba fue el algoritmo CKLF con el mejor desempeño tanto en $\mathbb{R}^2$ como en $\mathbb{R}^{10}$, con tasas de éxito muy altas, un número de bajo de iteraciones, desviación estándar baja, tiempo computacional bajo y métricas de error en su gran mayoría por debajo del valor de tolerancia y ordenes de convergencia lineales.

Fun	MSE (1×10^{-6})	RMSE (1×10^{-6})	MAE (1×10^{-6})	MSPE	MAPE	RMSLE (1×10^{-6})	p	M
Fun. (2.16)	0.61	758.17	269.76	1	1	0.61	0.97	0.76
Fun. (2.17)	0.48	626.66	133.53	1	1	0.48	1.00	1.00
Fun. (2.18)	1.29	1019.04	790.48	1	1	1.28	1.00	0.97
Fun. (2.19)	394.43	5063.45	4574.49	1	1	362.09	0.97	2.45

Cuadro 3.19: Desempeño CKLF: Métricas de errores del algoritmo en 2D

Fun	Tiempo (s)	μ_{iter}	$\pm\sigma$	SR (%)
Fun. (2.16)	39.58	62	8	100
Fun. (2.17)	123.05	601	222	88
Fun. (2.18)	119.07	620	250	82
Fun. (2.19)	36.76	220	46	100

Cuadro 3.20: Desempeño CKLF: Métricas estadísticas s del algoritmo en 10D

Para la ejecución de los códigos en las distintas funciones de prueba, se consideraron los siguientes valores de sintonización de los algoritmos. Este proceso se llevó a cabo a través de ensayo y error, ejecutando el algoritmo hasta encontrar un buen desempeño en términos de las métricas consideradas. Inicialmente, se partió de las recomendaciones dadas por los distintos autores para las funciones prueba y se fueron sintonizando. Es importante destacar que un buen desempeño en

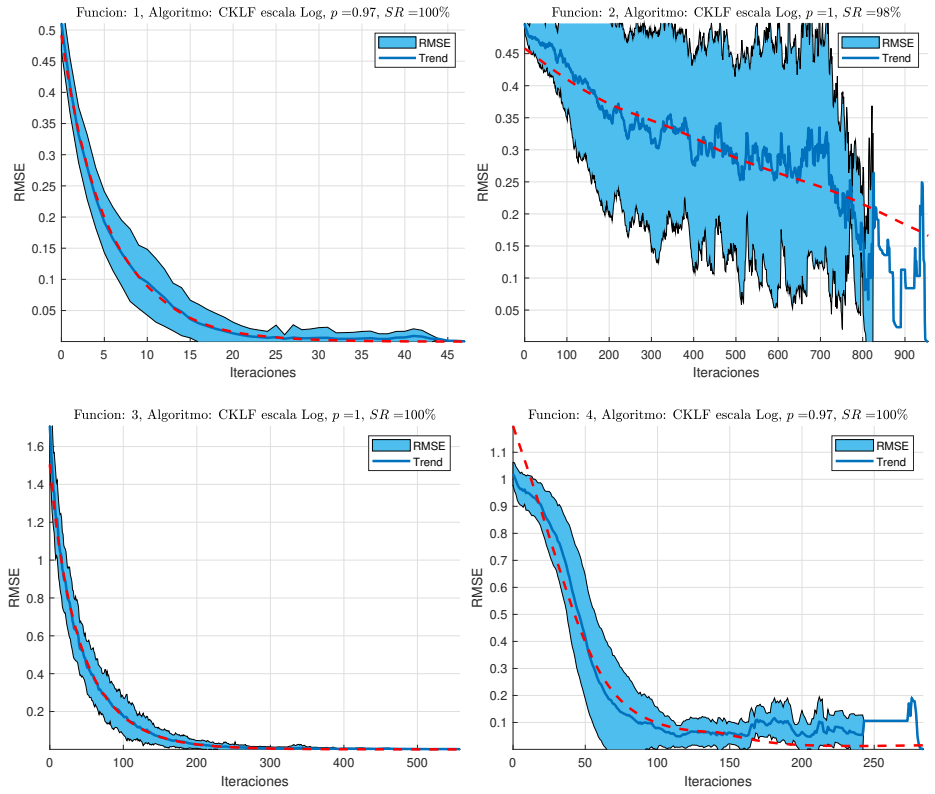


Figura 3.6: Error *RMSE* de CKLF. Fuente: *Autor*.

Fun	MSE (1×10^{-6})	RMSE (1×10^{-6})	MAE (1×10^{-6})	MSPE	MAPE	RMSLE (1×10^{-6})	p	M
Fun. (2.16)	0.73	840.79	553.96	1	1	0.72	0.98	0.88
Fun. (2.17)	4629714.44	2068456.04	2068456.04	1	1	1255786.95	1.00	1.00
Fun. (2.18)	2406141.13	1506952.06	1506952.06	1	1	846613.93	1.00	1.00
Fun. (2.19)	85447352.40	9243359.75	9243359.75	1	1	5413109.08	0.99	1.02

Cuadro 3.21: Desempeño CKLF: Métricas de errores del algoritmo en 10D

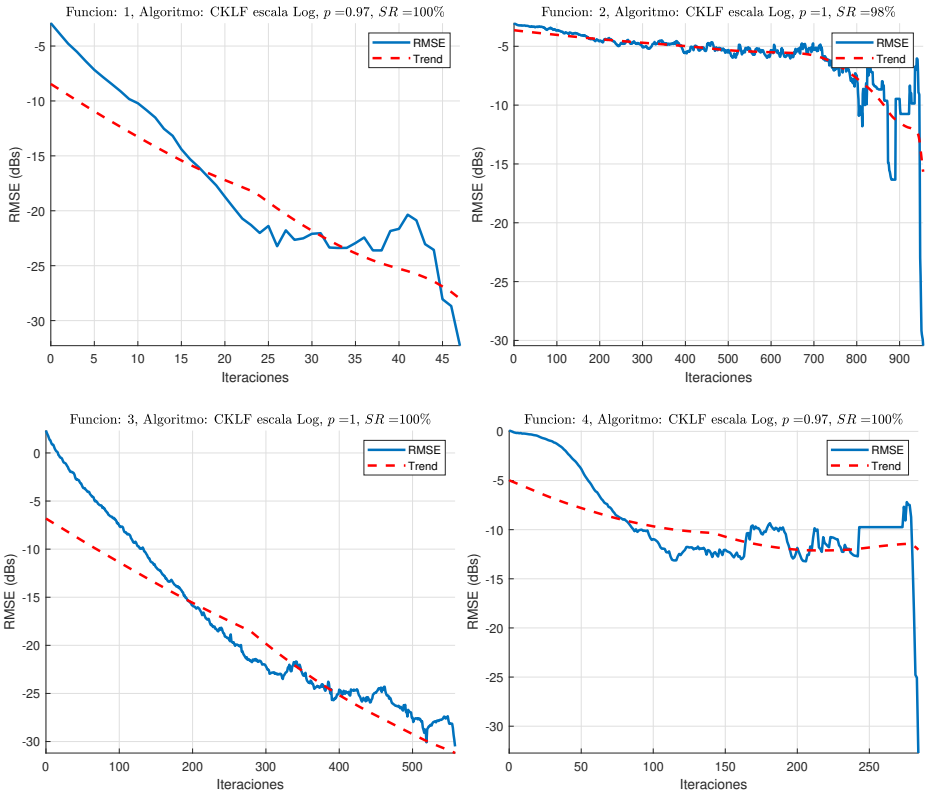


Figura 3.7: Error *RMSE* de CKLF. Fuente: *Autor*.

una función puede perjudicar o afectar el desempeño en otra función, de acuerdo con el teorema "No hay almuerzo gratis" (*No free lunch theorem*) [22, 27].

Para los algoritmos clásicos los procesos de sintonización fueron en el caso de CGM se consideró el β bajo la construcción *Hestenes-Stiefel*. Para SGD se consideró el parámetro de la tasa de aprendizaje $\alpha = 0{,}1$. Para NMS se consideró un poliedro que se construye de forma aleatoria y no la forma estática planteada en la literatura y un radio $r = 1$.

Para los algoritmos metaheurísticos los procesos de sintonización fueron en el caso de PSO una población $n = 40$ partículas, las constantes de aceleración $\alpha = 0{,}3$ y $\beta = 0{,}5$ y la inercia en $\theta = 0{,}95$. Para BA una población $n = 40$ murciélagos, las constantes de Volumen y emisión de pulsos $\alpha = 0{,}90$ y $\gamma = 0{,}90$ respectivamente, Constante de aceleración $\beta = 0{,}1$ y la inercia $\theta = 0{,}95$, el volumen $A \in [1, 2]$ y la frecuencia $f \in [0, 2]$ y finalmente para CKLF se consideró una población de $n = 40$ Cuckoos, trayectorias de vuelos de Lévy, una probabilidad de rechazo de nidos $p_a = 0{,}25$, un factor de escalamiento $\alpha = 0{,}45$ y la constante $\beta = 3/2$ de la función Gamma.

Las simulaciones se realizaron en un computador moderno usando el software MATLAB® R2022B, siendo este un portátil con procesador Intel(R) Core i7 8750H a 2.2 GHz con 6 núcleos, Memoria RAM 32 GB. Sistema Operativo Windows 10 Professional a 64 bits. Como se indicó anteriormente estos resultados se pueden haber mejorado haciendo una sintonización hiperparamétrica de los algoritmos, lo que pudiese haber implicado mejores resultados de cierto algoritmo comparado con otro.

3.2. Resultados *Beamforming* Adaptativo

Se realizó la comparación los algoritmos clásicos (CGM), (SGD) y (NMS) y metaheurísticos (PSO), (BA) y (CKLF) en el problema del Beamforming ; esta implementación se realizó en MATLAB® R2022B. Cada algoritmo tuvo una cantidad de 1000 iteraciones como criterio de parada y así obtener la información en decibeles de los nulos y el máximo valor de la señal deseada.

Nombre	1er Nulo (dB)	2do Nulo (dB)	Máximo (dB)	HPBW ($grados$)
CGM	-17.162	-17.492	0.000	20.01
SGD	-11.168	-6.957	-0.742	15.20
NMS	-6.229	-4.973	-0.066	30.41
PSO	-11.728	-11.854	0	21.21
BA	-6.375	-6.420	0	28.81
CKLF	-15.298	-23.368	0	19.2107

Cuadro 3.22: Resultados algoritmos: Nulos y máximos en diagrama de Radiación

En el caso de CGM *(véase tabla 3.22)* se puede ver el comportamiento del algoritmo en el dia-

grama de Radiación. El algoritmo presentó un excelente desempeño identificando los nulos sobre las señales interferentes y ubicando su el máximo del lóbulo principal sobre la señal deseada, cabe resaltar que aunque sus nulos están cercanos a los -17 decibeles el algoritmo profundiza los nulos hasta los -60 decibeles. Su valor en grados de la potencia media es de 20 grados lo que indica características de una antena altamente directiva. La ganancia máxima es igual a cero decibeles y esta se encuentra sobre la señal deseada y sus lóbulos laterales están por debajo de los -10 decibeles. En general podemos decir que el algoritmo presenta un buen comportamiento con posibilidades de mejora considerando otro tipo de función objetivo.

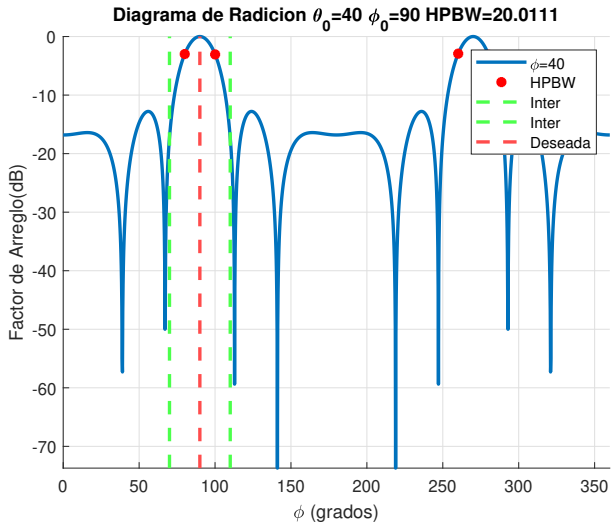


Figura 3.8: Diagrama de Radiación de CGM *fuente: Autor.*

En el caso de SGD *(véase tabla 3.22)* se puede ver el comportamiento del algoritmo en el diagrama de Radiación. El algoritmo un mal desempeño solo intentando identificar el primer nulo cercano a los -11 decibeles y el otro en -6 decibeles, intenta ubicar el máximo en el lóbulo principal pero presenta cierta dificultades con un valor de -0.74 decibeles, el algoritmo presenta dificultades en identificar las señales interferentes y minimizarlas para profundizar sus nulos. Su valor en grados de la potencia media es de 15.20 grados lo que indica características de una antena altamente directiva. Los lóbulos laterales no están bien identificados por la dificultad que tiene el algoritmo de ubicar los nulos sobre las interferentes. En general podemos decir que el algoritmo presenta un desempeño pobre con posibilidades de mejora considerando otro tipo de función objetivo, incrementando el número de iteraciones o revisando un valor de α que sea más adecuado.

En el caso de NMS *(véase tabla 3.22)* se puede ver el comportamiento del algoritmo en el diagrama de radiación. El algoritmo presentó un mal desempeño intentando identificar los nulos sobre las señales interferentes y ubicando el máximo del lóbulo principal sobre la señal deseada. Los ló-

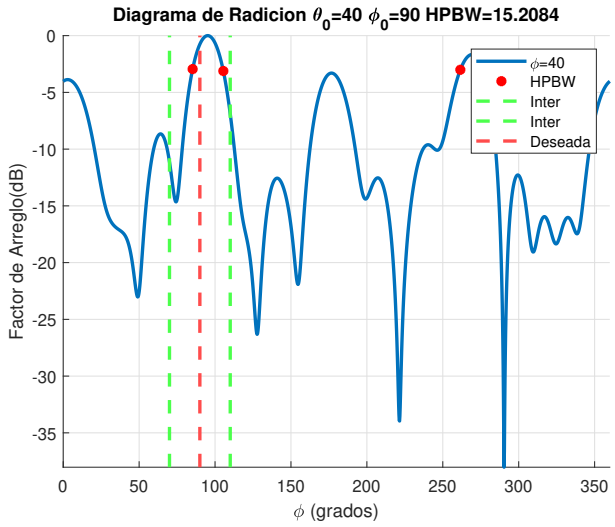


Figura 3.9: Diagrama de Radiación de SGD *fuente: Autor.*

bulos laterales son lóbulos de rejilla como se planteó en la sección de Beamforming adaptativo con longitud de onda igual a $\lambda/2$, cabe resaltar que aunque sus nulos están cercanos a los -5 decibeles el algoritmo profundiza los nulos hasta los -14 decibeles. Su valor en grados de la potencia media es de 30.41 grados lo que indica características de una antena directiva. La ganancia máxima es cercana a cero decibeles y esta se encuentra sobre la señal deseada. y sus lóbulos laterales están por debajo de los -10 decibeles. En general podemos decir que el algoritmo presenta un mal comportamiento con posibilidades de mejora considerando mejorar los parámetros reflexión, expansión, contracción y compresión.

En el caso de PSO *(véase tabla 3.22)* se puede ver el comportamiento del algoritmo en el diagrama de radiación. El algoritmo presentó un excelente desempeño identificando los nulos sobre las señales interferentes y ubicando su el máximo del lóbulo principal sobre la señal deseada, cabe resaltar que aunque sus nulos simétricos están cercanos a los -11 decibeles el algoritmo profundiza los nulos hasta los -14 decibeles. Su valor en grados de la potencia media es de 21.21 grados lo que indica características de una antena altamente directiva. La ganancia máxima es igual a cero decibeles y esta se encuentra sobre la señal deseada y sus lóbulos laterales están por debajo de los -12 decibeles. En general podemos decir que el algoritmo presenta un excelente comportamiento con posibilidades de mejora considerando otro tipo de función objetivo o haciendo un ajuste de hiperparámetros.

En el caso de BA *(véase tabla 3.22)* se puede ver el comportamiento del algoritmo en el diagrama de radiación. El algoritmo presentó un pobre desempeño con dificultades para encontrar los nulos sobre las señales interferentes y ubicando su el máximo del lóbulo principal sobre la señal

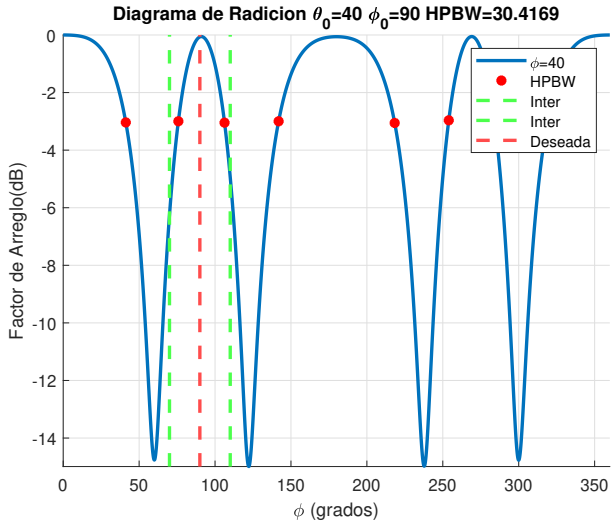


Figura 3.10: Diagrama de Radiación de NMS *fuente: Autor.*

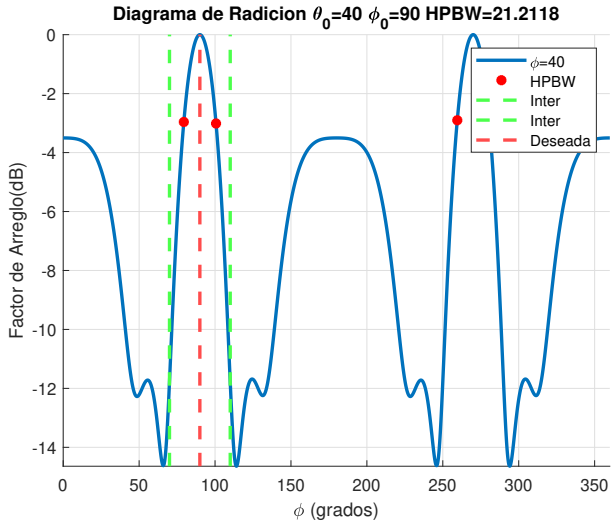


Figura 3.11: Diagrama de Radiación de PSO *fuente: Autor.*

deseada, cabe resaltar que sus nulos están cercanos a los -6 decibeles el algoritmo profundiza los nulos hasta los -55 decibeles. Su valor en grados de la potencia media es de 28.21 grados lo que indica características de una antena directiva. La ganancia máxima es igual a cero decibeles y esta se encuentra sobre la señal deseada y sus lóbulos laterales están por debajo de los -5 decibeles. En general podemos decir que el algoritmo presenta un pobre comportamiento con posibilidades de mejora considerando otro tipo de función objetivo que maximice la ganancia del lóbulo principal reduciendo su HPBW o haciendo un ajuste de hiperparámetros.

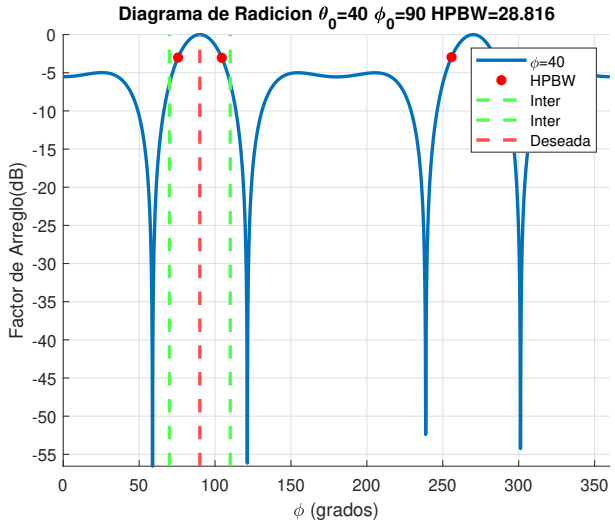


Figura 3.12: Diagrama de Radiación de BA *fuente: Autor*.

Por ultimo tenemos el caso de CKLF *(véase tabla 3.22)* se puede ver el comportamiento del algoritmo en el diagrama de radiación. El algoritmo presentó un excelente desempeño identificando los nulos sobre las señales interferentes y ubicando su ganancia máxima del lóbulo principal sobre la señal deseada con un valor de cero decibeles, cabe resaltar que aunque sus nulos no son simétricos el primero se encuentra a -15 decibeles y el segundo se encuentra a -23 decibeles siendo estos los mejores resultados obtenidos por un algoritmo comparables con los de CGM. Su valor en grados de la potencia media es de 19.21 grados lo que indica características de una antena altamente directiva. En general podemos decir que el algoritmo presenta un excelente comportamiento con posibilidades de mejora considerando otro tipo de función objetivo que busque reducir sus lóbulos laterales o haciendo un ajuste de hiperparámetros, este comportamiento es esperado por las características exploratorias que tiene el algoritmo gracias a sus vuelos de Lévy y por los resultados obtenidos en las funciones de prueba.

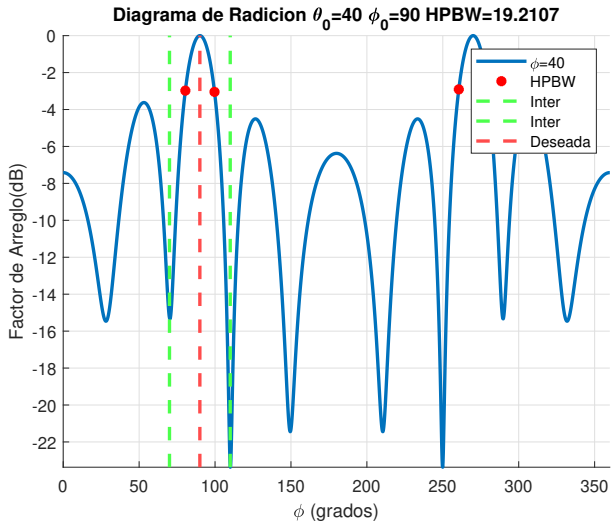


Figura 3.13: Diagrama de Radiación de CKLF *fuente: Autor.*

3.3. Posibles trabajos - Trabajos futuros

De acuerdo a los resultados obtenidos se pueden considerar los siguientes trabajos futuros que busquen dar una respuesta al problema del Beamforming utilizando un análisis comparativo entre algoritmos. Cabe aclacarar que el El *Beamforming Adaptativo* es una técnica utilizada en la tecnología de comunicaciones inalámbricas no necesariamente en el área de las telecomunicaciones, puede ser utilizado en la construcción de SONAR, Radares, GPS y distintos sistemas de comunicación. Al combinar esta técnica con algoritmos clásicos y metaheurísticos, se pueden realizar investigaciones futuras en varios ámbitos, algunos de ellos son:

- Mejora de la calidad de la transmisión de señales en entornos con alta interferencia y congestión de frecuencias, esto se puede hacer en un trabajo que busque aumentar el número de señales interferentes y viendo el desempeño que pueden tener los algoritmos en estos escenarios.

- Desarrollo de sistemas de *Beamforming* dinámicos que sean capaces de adaptarse en tiempo real a las condiciones cambiantes del canal, esto se puede hacer en un trabajo que considere señales deseadas e interferentes que tengan cierta característica de modulación o constelación además una función objetivo especial que considere objetivos móviles, es decir señales deseadas e interferentes que cambian respecto al tiempo. Se pueden realizar experimentos para ver cuales algoritmos tienen el mejor desempeño y cuales son aquellos que se adaptan más rápido, considerando efectos de latencia o retrasos en la señal, escenarios importantes para la quinta y sexta generación de telefonía móvil celular.

- Aplicación en la optimización de la cobertura y capacidad de redes inalámbricas, como redes

5G y WiFi, esto se puede hacer en un trabajo que relacione distintas celdas de antenas y como el Beamforming Adaptativo generado por los algoritmos clásicos y metaheurísticos puede mejorar métricas calidad del servicio como cobertura, capacidad y disponibilidad de la red comparándola a su vez con el sistema de telefonía actual.

- Desarrollo de sistemas de *Beamforming* múltiple para mejorar la eficiencia en la transmisión de datos en redes de sensores inalámbricos, como es el caso de dispositivos *(Internet of Things (IoT))*, para esto se pueden considerar varias señales deseadas y realizar un análisis comparativo con aquellas que proporcionen lóbulos de rejilla en la dirección de los dispositivos, la relevancia de este trabajo futuro está asociada por ejemplo con la agricultura de precisión que utiliza estos estos dispositivos y el objetivo sería encontrar aquel algoritmo que ofrezca la mejor cobertura de red.

- Aplicación en sistemas de comunicaciones acústicas tipo SONAR, como la transmisión de señales sonoras en ambientes submarinos, para este trabajo futuro se puede realizar un análisis comparativo de los algoritmos para determinar cuál es el más adecuado para el medio, en este caso el agua considerando por ejemplo varios objetivos y posibles interferencias multi trayectoria de señales interferentes, se pueden considerar otro tipo de función objetivo acorde a estos ambientes acuáticos como lo puede ser la relación señal a ruido *(signal-to-noise ratio (SNR))* o el método de capón utilizado en micrófonos, también se pueden considerar otros tipos de geométricas del arreglo, por ejemplo: Lineal, Circular y la cantidad de elementos.

3.4. Conclusiones

En conclusión, los experimentos realizados con las funciones de prueba y el *Beamforming* adaptativo son cruciales para determinar el desempeño y establecer un análisis comparativo entre la optimización clásica y metaheurística en arreglos planos URA MIMO masivo. Esta conclusión adquiere una importancia aún mayor en el contexto de la telefonía móvil de quinta y sexta generación, donde la optimización de la señal, la calidad del servicio *(Quality of Service (QoS))* y el aumento de la capacidad del sistema son factores clave para garantizar una transmisión de datos de alta velocidad y confiabilidad. Por ejemplo, en las nuevas redes 5G y 6G, el uso de técnicas de optimización metaheurísticas podría mejorar significativamente la eficiencia del espectro electromagnético con antenas altamente directivas y en la asignación de recursos y el desempeño del sistema de telefonía móvil en general.

Bajo ciertas condiciones iniciales del vector de peso de desplazamiento de fase b_{mn} y del vector de pesos w es posible obtener diagramas de radiación adecuados que cumplen con las características del *Beamforming* Adaptativo como lo es ubicar el valor máximo del lóbulo principal sobre la dirección de la señal deseada y ubicar los nulos sobre las direcciones de las señales interferentes, esta dificultad como se mencionó a lo largo del documento está relacionada con la dificultad por sí mismo que presenta el problema del *Beamforming*, de las cuales se destacan las siguiente: La dimensión del problema, como es sabido $b_{mn} \in \mathbb{R}^{64}$ y $w_{mn} \in \mathbb{C}^{64}$, la complejidad del factor de

arreglo sobre ciertos valores (θ, ϕ) limitando la optimización clásica o alguna característica convexa en la función objetivo y la elección de una función objetivo que tenga mejores prestaciones como profundizar nulos *(Null steering NS)*, reducción de lóbulos laterales *(Sidelobe level SLL)* , etc.

El análisis realizado en las funciones de prueba ha permitido concluir que CKLF es el mejor algoritmo en las 4 funciones de prueba. Su desempeño se puede ver en la tabla resumen *(véase tabla resumen 3.18)*, donde se muestra una tasa de éxito perfecta en todas las funciones, excepto en la función 2 con una tasa de éxito de 98 % y una cantidad de iteraciones moderadas. La capacidad de CKLF para escapar de óptimos locales y optimizar la superficie de búsqueda gracias a las caminatas aleatorias a través de vuelos de Levy explica su alto rendimiento y tiempos computacionales similares a los demás algoritmos. Este análisis comparativo entre algoritmos clásicos y metaheurísticos es crucial para determinar cuál es el más adecuado para el problema del *Beamforming* Adaptativo.

El algoritmo CKLF demostró ser el mejor adaptado a las funciones de prueba, con tasas de éxito muy altas y un desempeño sobresaliente tanto en $\mathbb{R}^2$ como en $\mathbb{R}^{10}$, con un número bajo de iteraciones, desviación estándar baja, tiempo computacional bajo y métricas de error en su mayoría por debajo del valor de tolerancia y con una convergencia lineal. Este desempeño se debe en parte a la capacidad exploratoria efectiva del algoritmo gracias a sus vuelos de Levy que permiten al algoritmo escapar de óptimos locales. Los resultados obtenidos reafirman ciertos aspectos considerados en la teoría, como la presencia de aspectos heredados de PSO en la implementación de BA y CKLF.

La función multiobjetivo utilizada en este estudio es efectiva en ubicar los lóbulos principales en la dirección de las señales deseadas y nulos sobre las señales interferentes, objetivo logrado con todos los algoritmos algunos con alguna dificultad asociado a su naturaleza de programación. Sin embargo, es común que se presenten estancamientos en el proceso de optimización debido a la complejidad del problema, la condición inicial de las partículas, la sintonización de los parámetros y la convergencia de los algoritmos. Por lo tanto, es importante tener en cuenta estos factores para lograr una solución óptima con la función multiobjetivo.

De acuerdo a los resultados obtenidos una posible forma de obtener diagramas de radiación mejores es realizar procesos de sintonización que sean adecuados para el problema del *Beamforming* Adaptativo, se encontró que los valores de sintonización establecidos en las funciones de prueba no son adecuados y toco modificarlos bajo ensayo y error, una estrategia del *Machine Learning* puede ser empleada para la sintonización hiper paramétrica de los algoritmos considerando por ejemplo una búsqueda exhaustiva o búsqueda de rejilla, encontrando los valores precisos que se ajustan al problema.

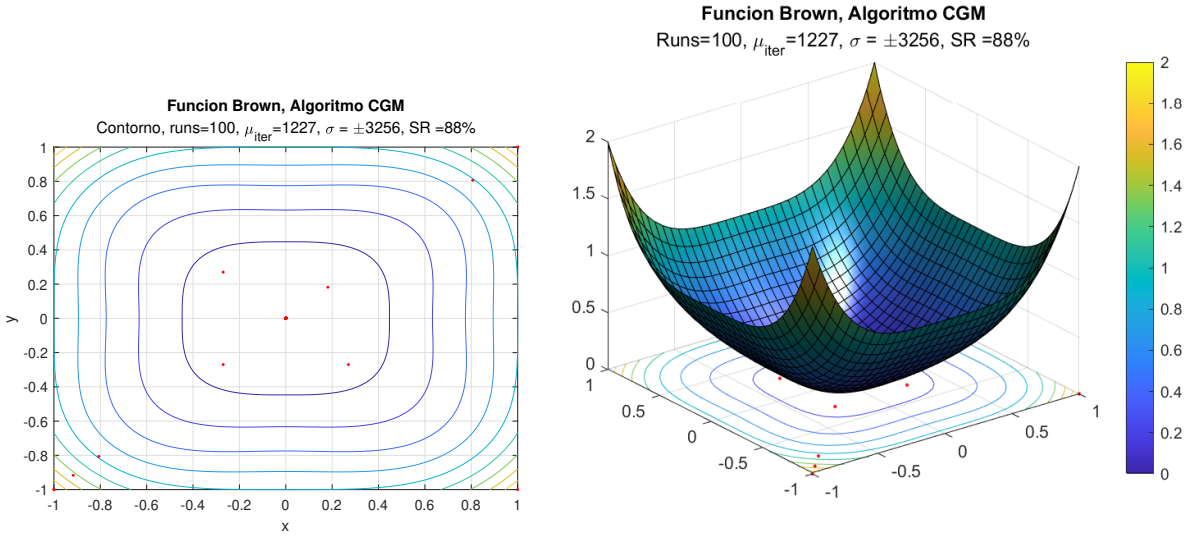
Capítulo 4

Anexos

4.1. Códigos de Matlab®

Para consultar los códigos de las funciones de prueba y el experimento de *Beamforming* adaptativo revisar el siguiente enlace *https://github.com/hamsomp3/pregrado_matematicas* en un repositorio de *GitHub* público, revisar el documento *README.md* para mas información.

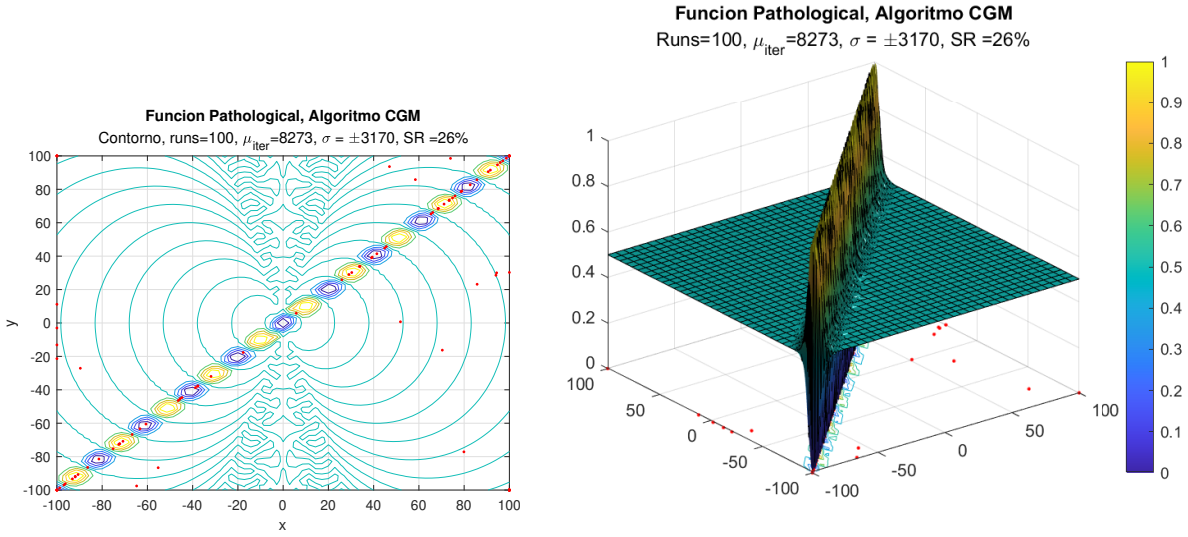
4.2. Imágenes funciones de prueba



(a) Contorno

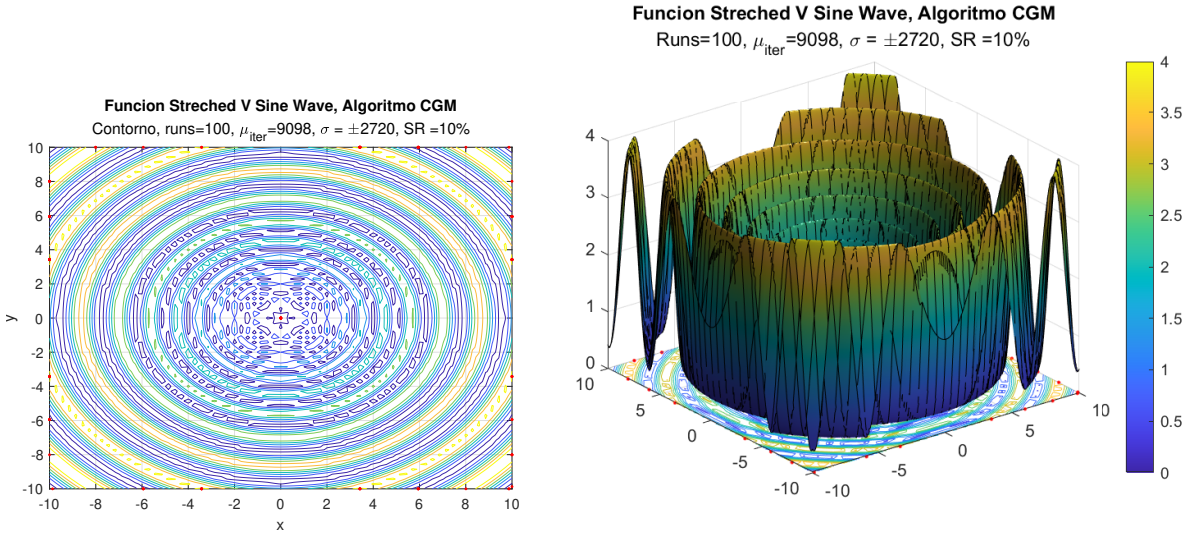
(b) Superficie

Figura 4.1: Función Brown del método CGM: *Autor*.



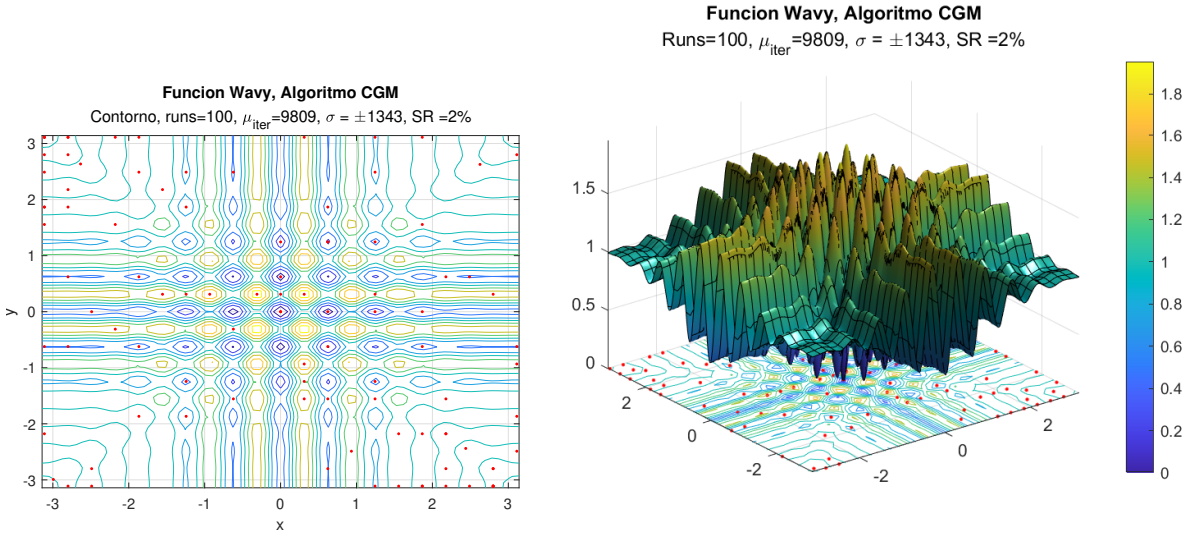
(a) Contorno (b) Superficie

Figura 4.2: Función Pathological del método CGM: *Autor*.



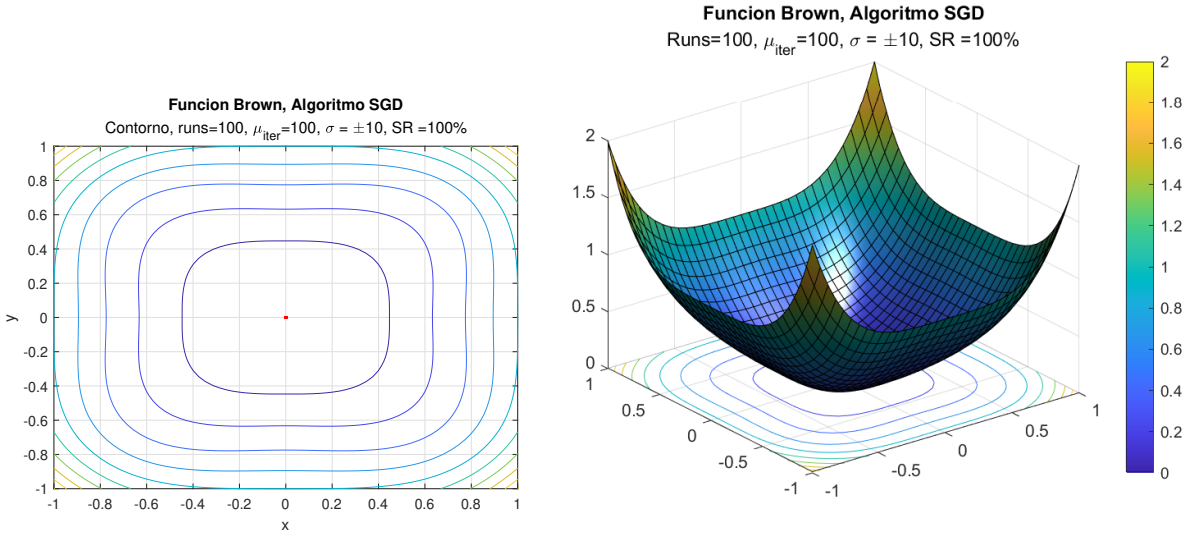
(a) Contorno (b) Superficie

Figura 4.3: Función Streched V Sine Wave del método CGM: *Autor*.



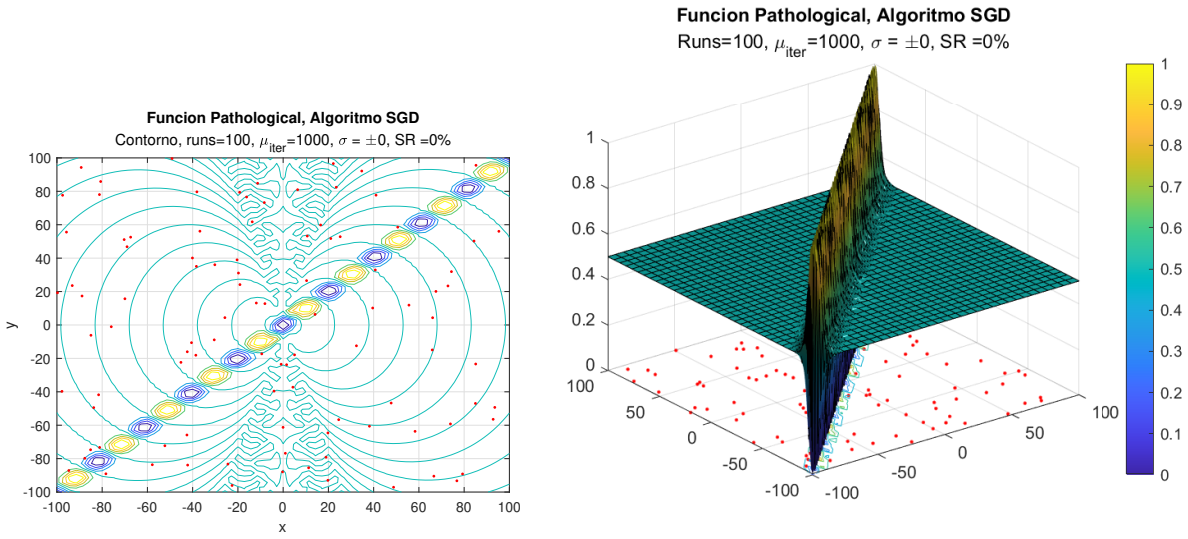
(a) Contorno (b) Superficie

Figura 4.4: Función Brown del método CGM: *Autor*.



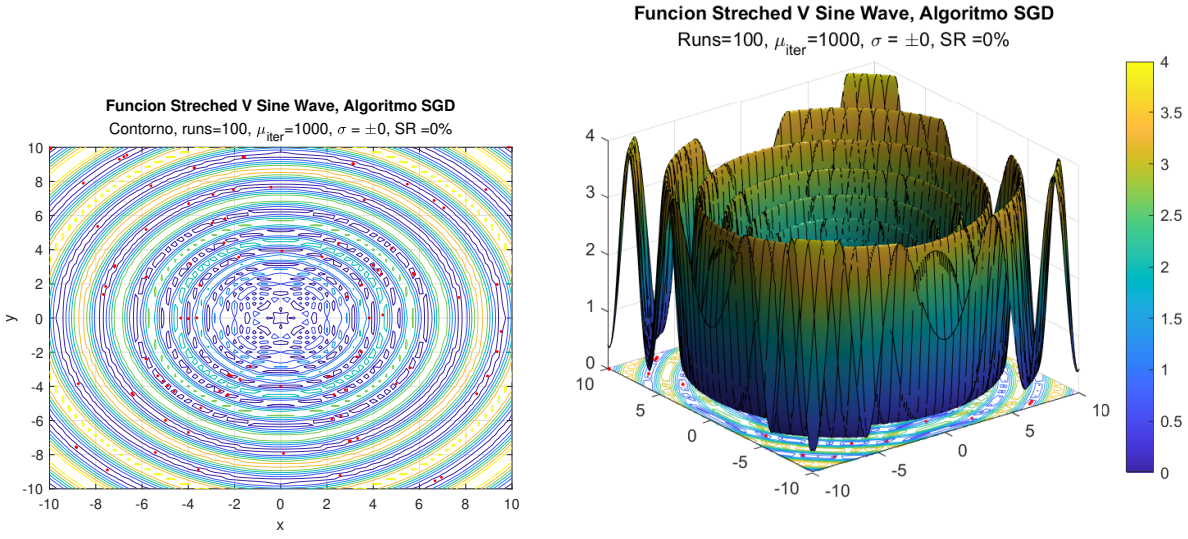
(a) Contorno (b) Superficie

Figura 4.5: Función Brown del método SGD: *Autor*.



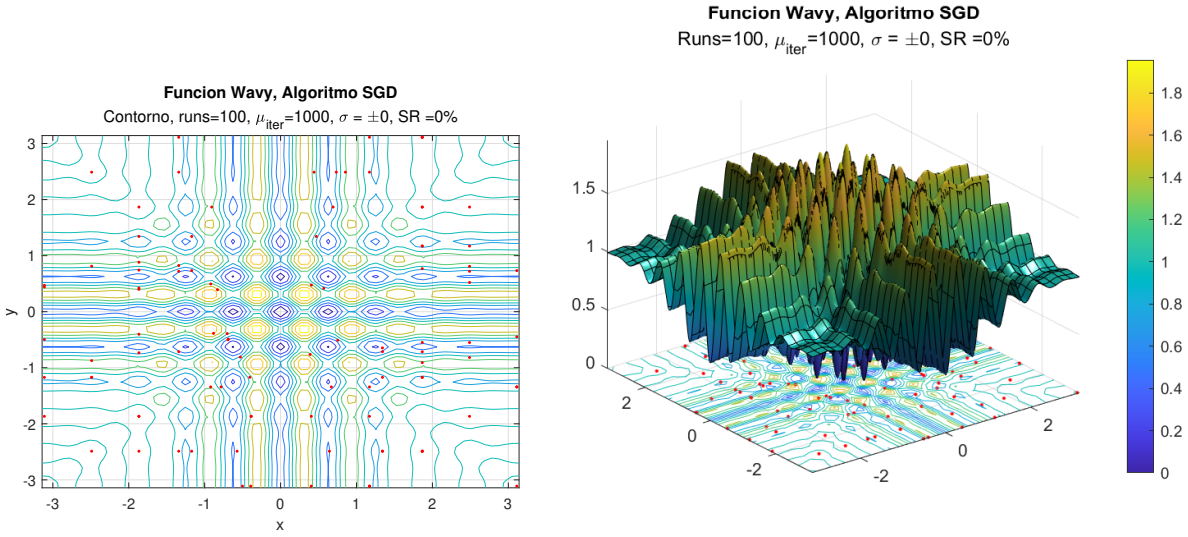
(a) Contorno (b) Superficie

Figura 4.6: Función Pathological del método SGD: *Autor*.



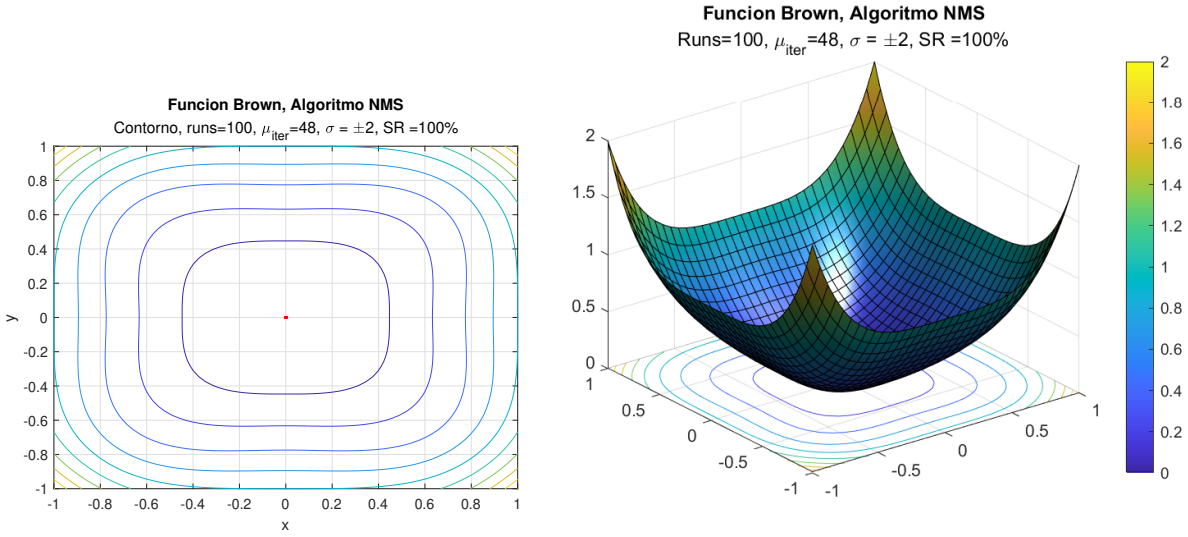
(a) Contorno (b) Superficie

Figura 4.7: Función Streched V Sine Wave del método SGD: *Autor*.



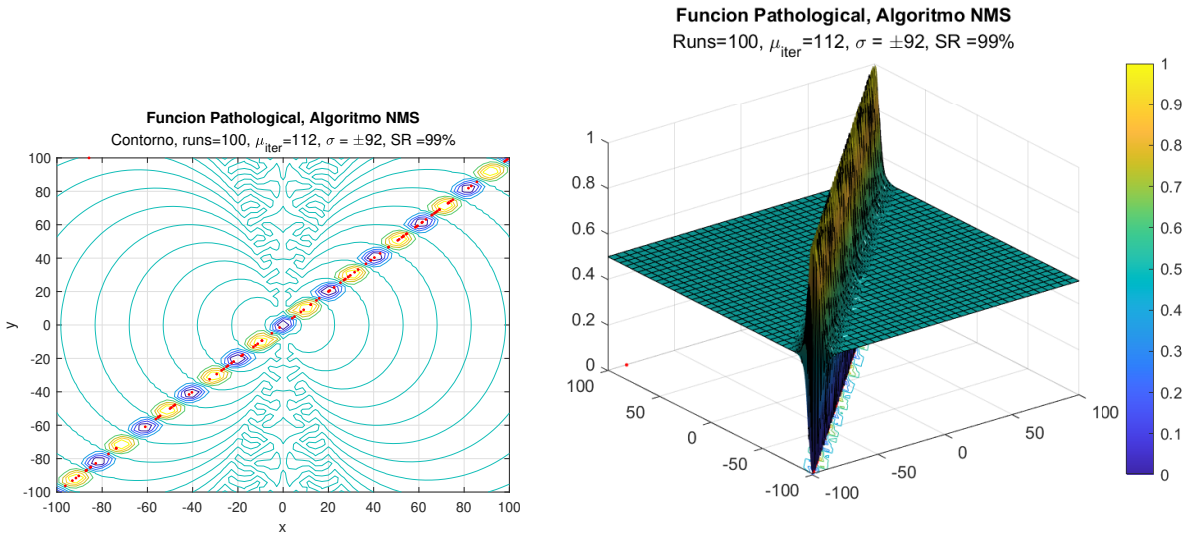
(a) Contorno (b) Superficie

Figura 4.8: Función Brown del método SGD: *Autor*.



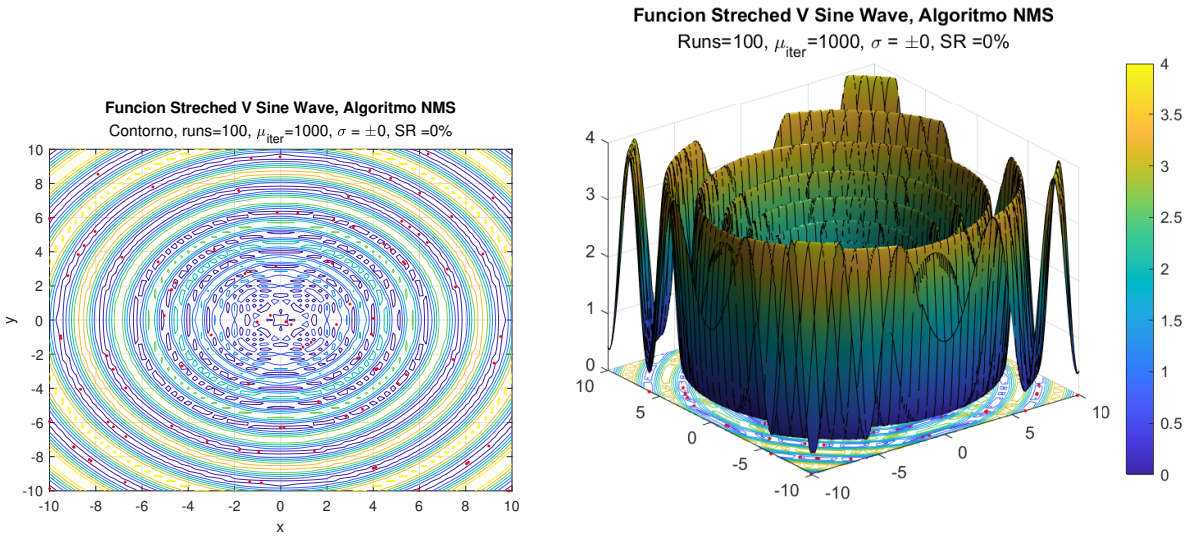
(a) Contorno (b) Superficie

Figura 4.9: Función Brown del método NMS: *Autor*.



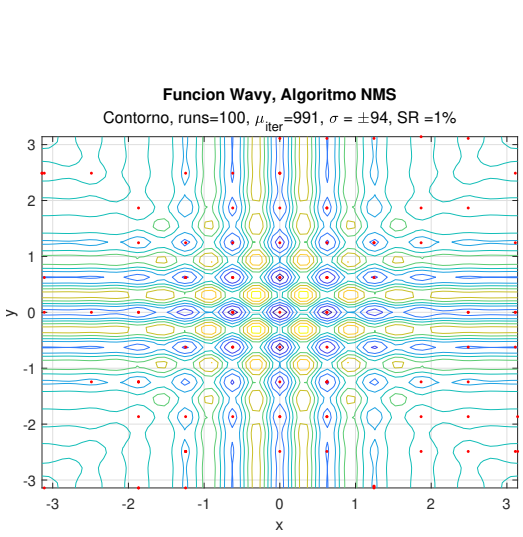
(a) Contorno (b) Superficie

Figura 4.10: Función Pathological del método NMS: *Autor*.

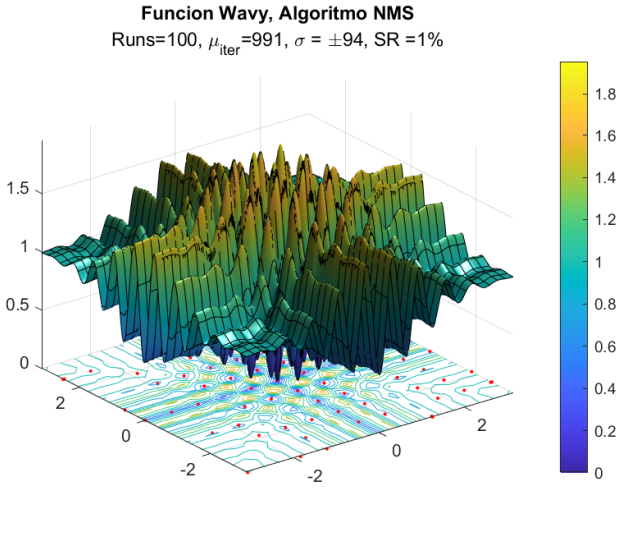


(a) Contorno (b) Superficie

Figura 4.11: Función Streched V Sine Wave del método NMS: *Autor*.

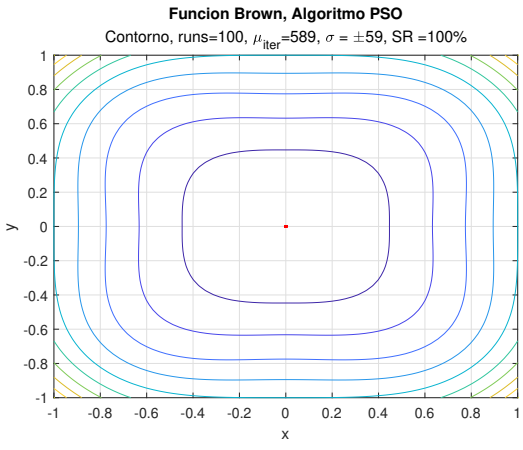


(a) Contorno

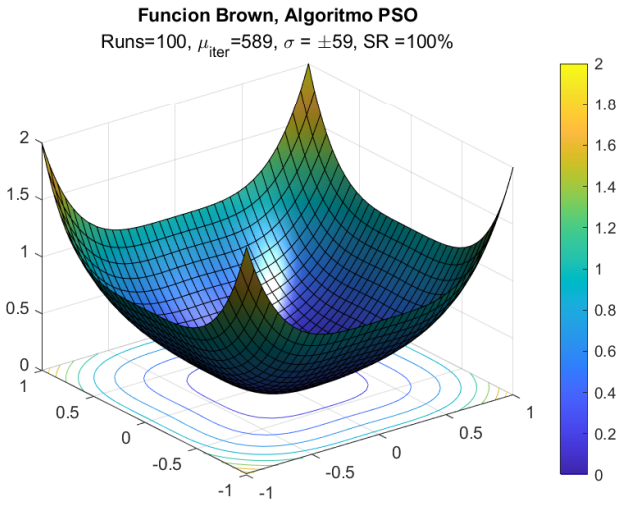


(b) Superficie

Figura 4.12: Función Brown del método NMS: *Autor*.

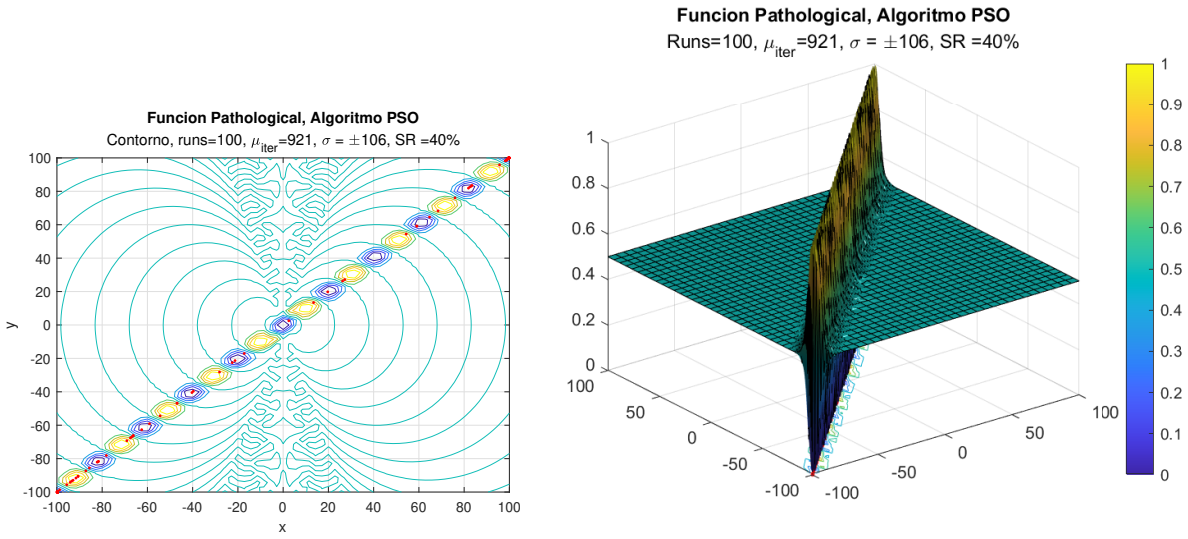


(a) Contorno



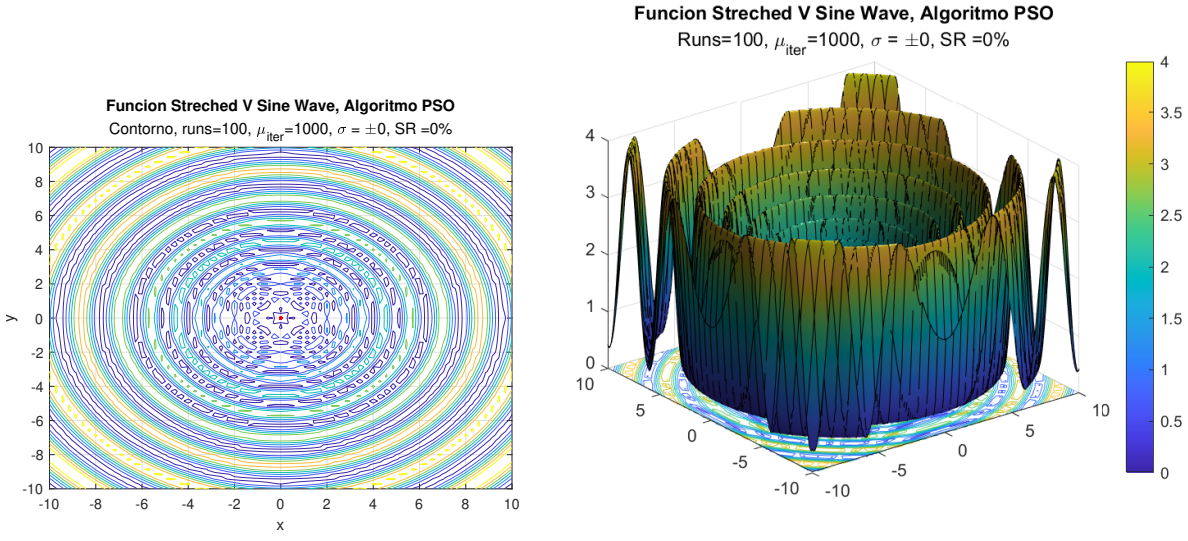
(b) Superficie

Figura 4.13: Función Brown del método PSO: *Autor*.



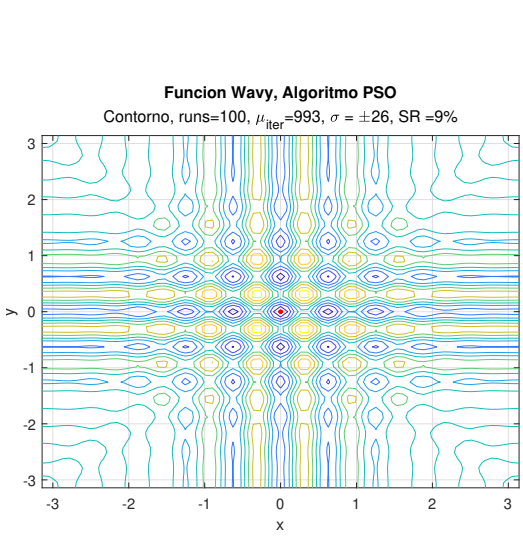
(a) Contorno (b) Superficie

Figura 4.14: Función Pathological del método PSO: *Autor*.

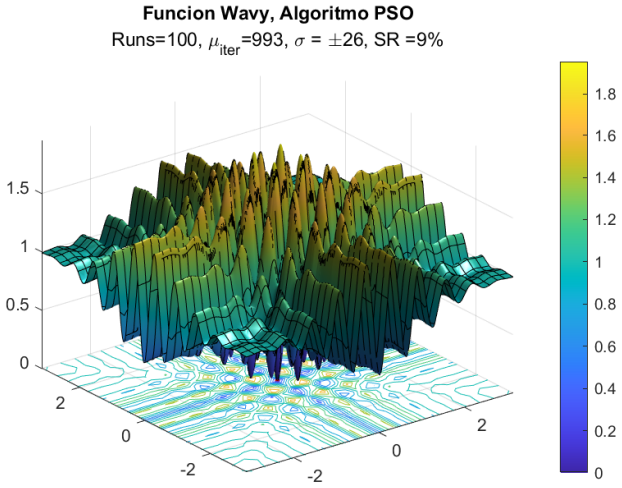


(a) Contorno (b) Superficie

Figura 4.15: Función Streched V Sine Wave del método PSO: *Autor*.

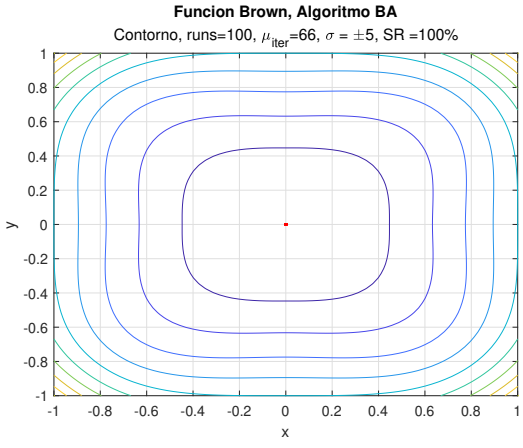


(a) Contorno

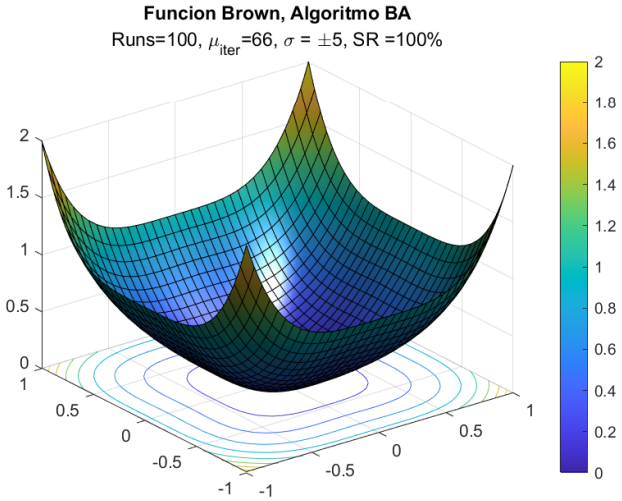


(b) Superficie

Figura 4.16: Función Brown del método PSO: *Autor*.

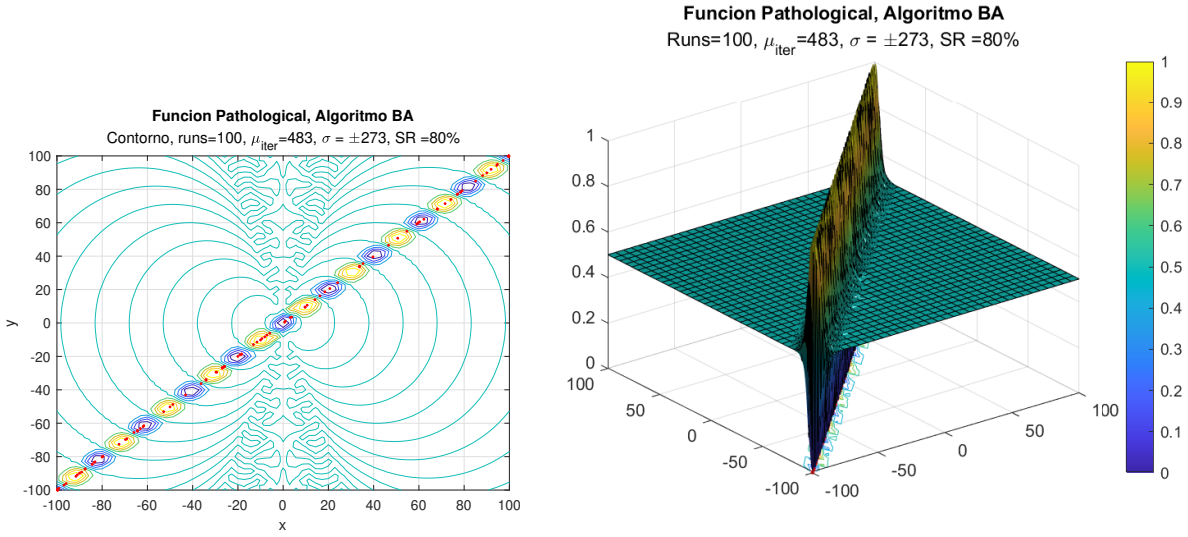


(a) Contorno



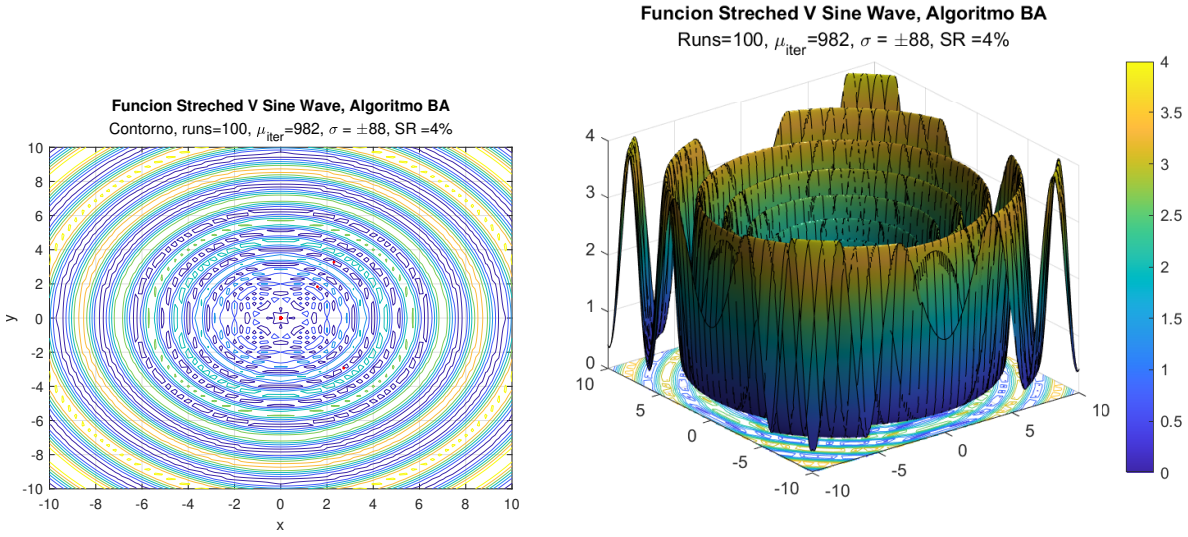
(b) Superficie

Figura 4.17: Función Brown del método BA: *Autor*.



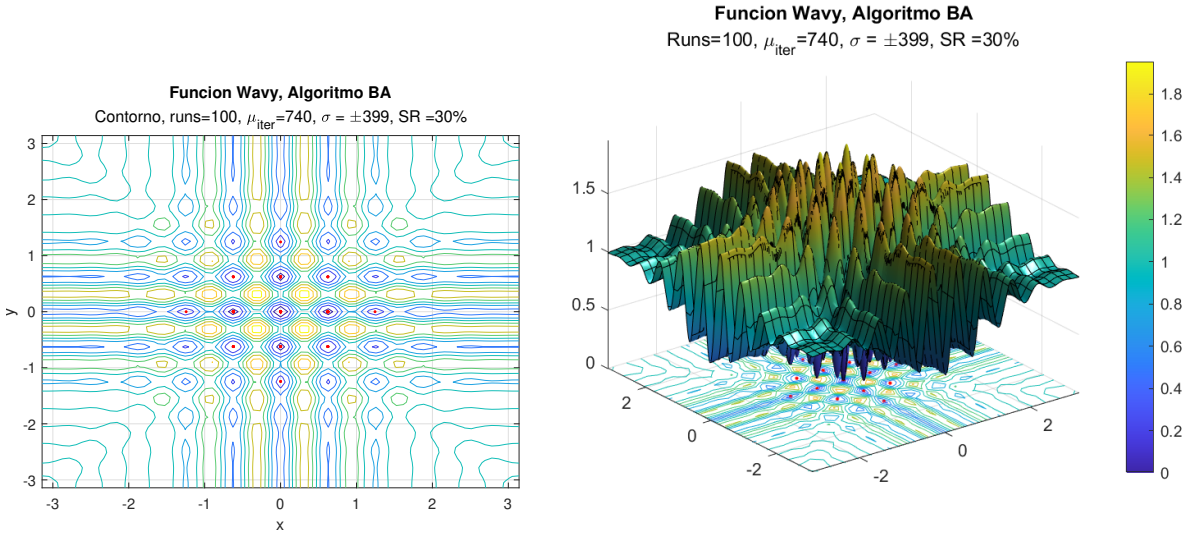
(a) Contorno (b) Superficie

Figura 4.18: Función Pathological del método BA: *Autor*.



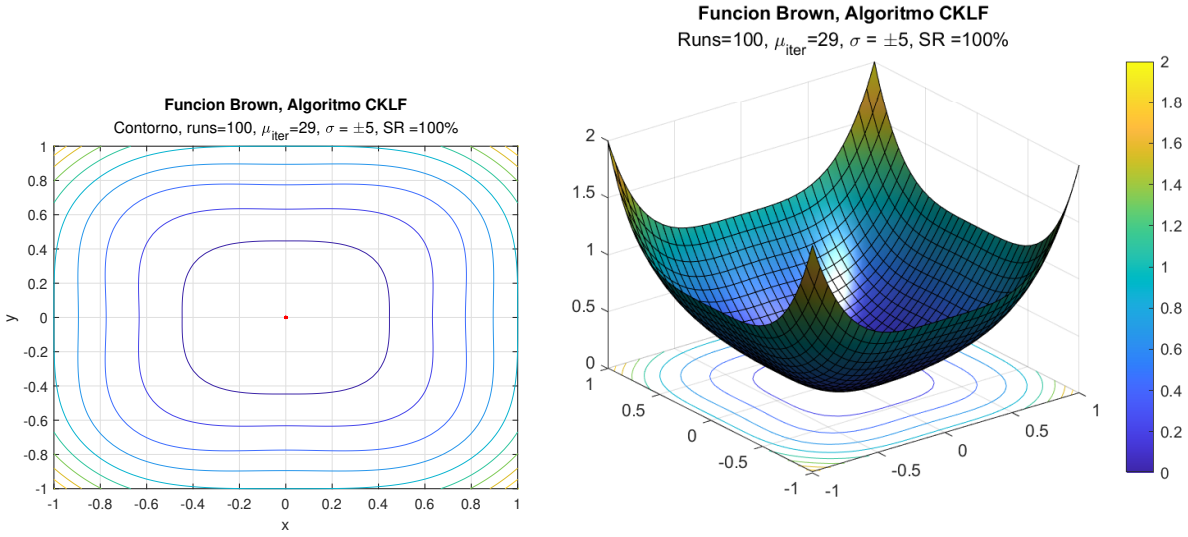
(a) Contorno (b) Superficie

Figura 4.19: Función Streched V Sine Wave del método BA: *Autor*.



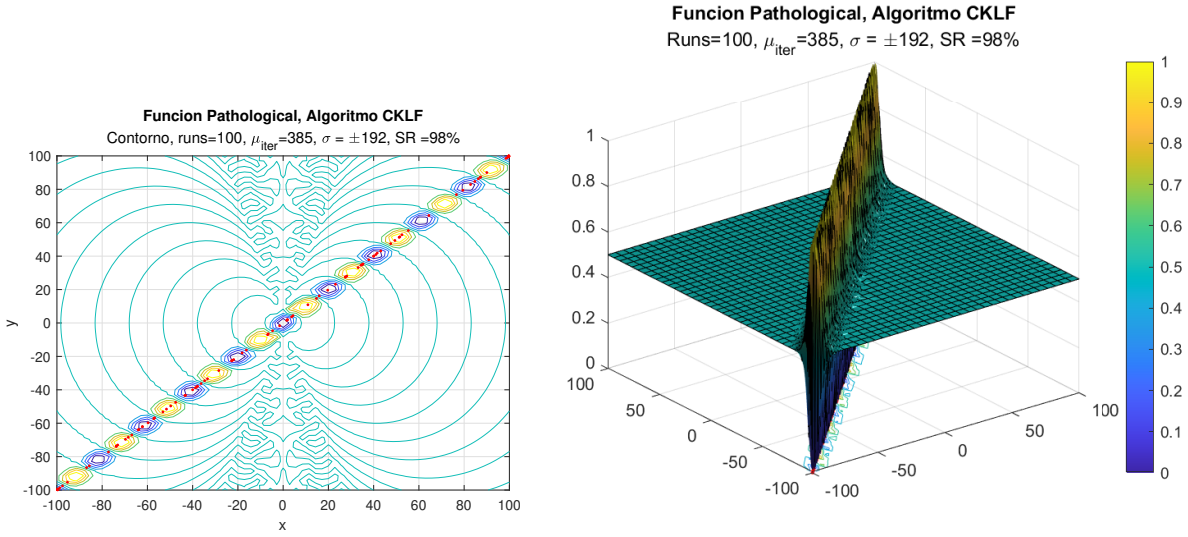
(a) Contorno (b) Superficie

Figura 4.20: Función Brown del método BA: *Autor*.



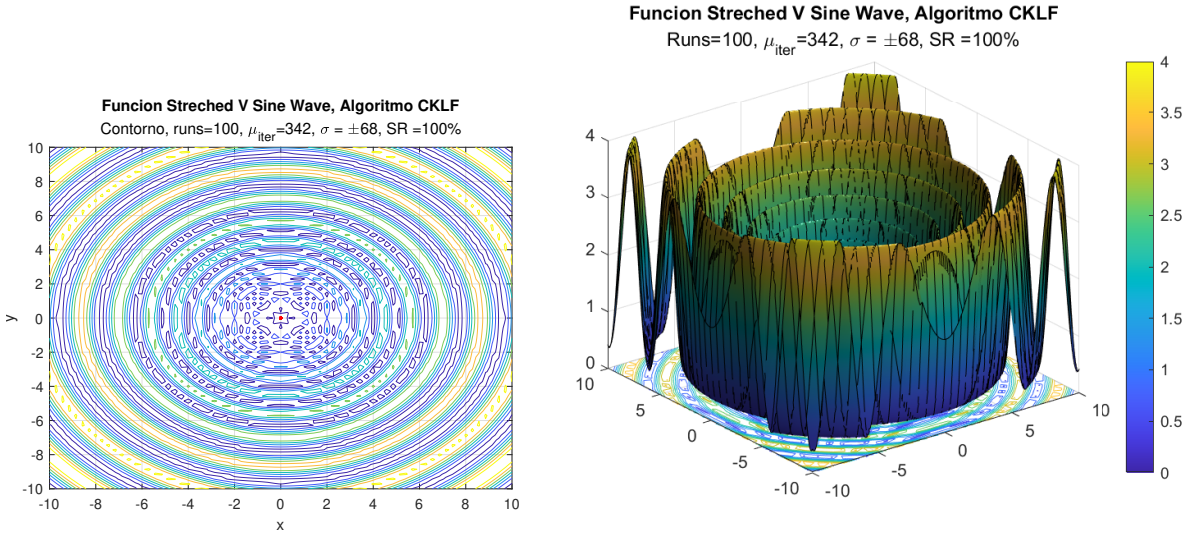
(a) Contorno (b) Superficie

Figura 4.21: Función Brown del método CKLF: *Autor*.



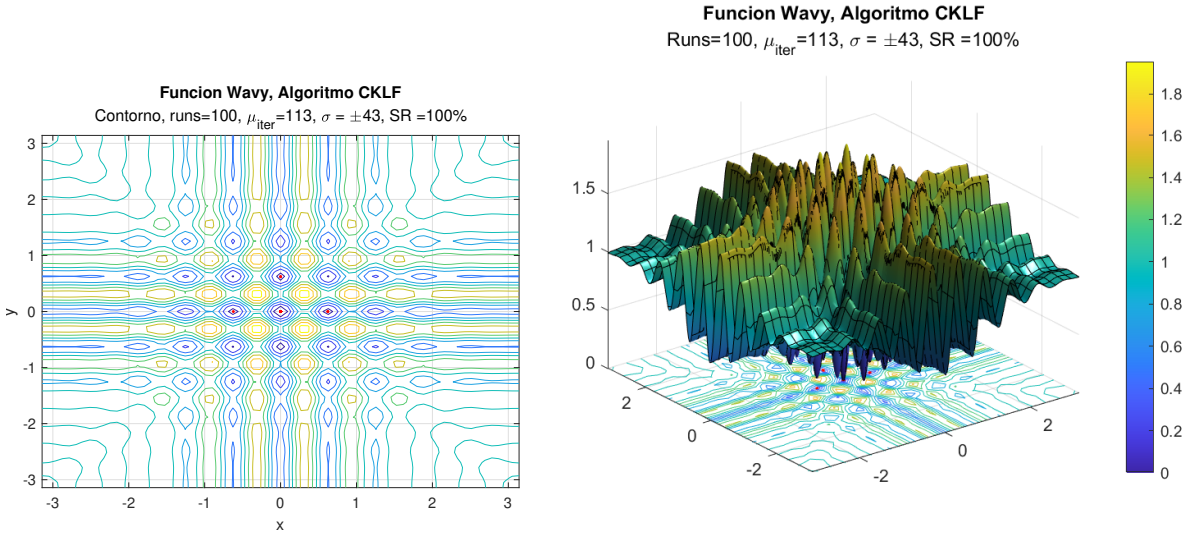
(a) Contorno (b) Superficie

Figura 4.22: Función Pathological del método CKLF: *Autor*.



(a) Contorno (b) Superficie

Figura 4.23: Función Streched V Sine Wave del método CKLF: *Autor*.



(a) Contorno

(b) Superficie

Figura 4.24: Función Brown del método CKLF: *Autor*.

Bibliografía

[1] Mohammad Abualhayja'a and Mousa Hussein. Comparative study of adaptive beamforming algorithms for smart antenna applications. In *2020 International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, pages 1–5. IEEE, 2021.

[2] Constantine A. Balanis. *Antenna Theory: Analysis and Design*. Wiley, 4 edition, 2016.

[3] Smita Banerjee and Ved Vyas Dwivedi. Performance analysis of adaptive beamforming using particle swarm optimization. In *2016 11th International Conference on Industrial and Information Systems (ICIIS)*, pages 242–246. IEEE, 2016.

[4] Alexei Botchkarev. Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. *arXiv preprint arXiv:1809.03006*, 2018.

[5] Richard L Burden, J Douglas Faires, and Annette M Burden. *Numerical analysis*. Cengage learning, 2015.

[6] Ministerio de Tecnologías de la Información y las Comunicaciones. *BOLETÍN TRIMESTRAL DE LAS TIC Cifras Primer Trimestre de 2021*. MinTic, Jul 2021.

[7] Magnus R Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving. *Journal of research of the National Bureau of Standards*, 49(6):409, 1952.

[8] Wei Hong, Zhi Hao Jiang, Chao Yu, Debin Hou, Haiming Wang, Chong Guo, Yun Hu, Le Kuai, Yingrui Yu, Zhengbo Jiang, Zhe Chen, Jixin Chen, Zhiqiang Yu, Jianfeng Zhai, Nianzu Zhang, Ling Tian, Fan Wu, Guangqi Yang, Zhang-Cheng Hao, and Jian Yi Zhou. The role of millimeter-wave technologies in 5g/6g wireless communications. *IEEE Journal of Microwaves*, 1(1):101–122, 2021.

[9] Vincenzo Inzillo, Floriano De Rango, and Alfonso Ariza Quintana. A low energy consumption smart antenna adaptive array system for mobile ad hoc networks. *International Journal of Computing*, 16(3):124–132, 2017.

[10] Momin Jamil and Xin-She Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013.

[11] Praneet Raj Jeripotula and B. Rajendra Naik. Performance analysis of adaptive beamforming algorithms. In *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET)*, pages 1–4, 2018.

[12] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, Nov 1995.

[13] Jeffrey C Lagarias, James A Reeds, Margaret H Wright, and Paul E Wright. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on optimization*, 9(1):112–147, 1998.

[14] Maulana Malik, Mustafa Mamat, Siti Sabariah Abas, Ibrahim Mohammed Sulaiman, Abdul Talib Bon, et al. Comparison of conjugate gradient method on solving unconstrained optimization problems. In *of theInternational Conference on Industrial Engineering and Operations Management, no*, 2020.

[15] Praneeth Netrapalli. Stochastic gradient descent and its variants in machine learning. *Journal of the Indian Institute of Science*, 99(2):201–213, 2019.

[16] Rajen Kumar Patra and Chinmay Kumar Nayak. A comparison between different adaptive beamforming techniques. In *2019 International Conference on Range Technology (ICORT)*, pages 1–4. IEEE, 2019.

[17] T. S. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. N. Wong, J. K. Schulz, M. Samimi, and F. Gutierrez. Millimeter wave mobile communications for 5g cellular: It will work! *IEEE Access*, 1:335–349, 2013.

[18] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[19] Timothy Sauer. *Numerical analysis*. Addison-Wesley Publishing Company, 2011.

[20] Rohde & Schwarz. *LTE Transmission Modes and Beamforming*. Bernhard Schulz, Jul 2015.

[21] John A Trangenstein. *Scientific Computing: Vol. II-Eigenvalues and Optimization*, volume 19. Springer, 2018.

[22] Jan Polanco Velasco. Beamforming de arreglos planos mimo masivo usando optimización metaheurística. *Trabajo de grado - Ingeniería Electrónica*, page 159, 2022.

[23] X. Yang and Suash Deb. Cuckoo search via levy flights. In *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, pages 210–214, 2009.

[24] X.-S. Yang. A new metaheuristic bat-inspired algorithm. *Studies in Computational Intelligence*, 284:65–74, 2010. cited By 2194.

[25] Xin-She Yang. *Engineering Optimization: An Introduction with Metaheuristic Applications*. Wiley, 1 edition, 2011.

[26] Xin-She Yang. *Cuckoo search and firefly algorithm: Theory and applications*, volume 516. Springer, 2013.

[27] Xin-She Yang. *Nature-inspired optimization algorithms*. Elsevier, 2014.

[28] Xin-She Yang. *Nature-inspired optimization algorithms*. Academic Press, 2020.

[29] Xin-She Yang and Amir Hossein Gandomi. Bat algorithm: a novel approach for global engineering optimization. *Engineering computations*, 2012.