REPÚBLICA DEL ECUADOR
Escuela Politécnica Nacional
"E SCIENTIA HOMINIS SALUS"

La versión digital de esta tesis está protegida por la Ley de Derechos de Autor del Ecuador.

Los derechos de autor han sido entregados a la "ESCUELA POLITÉCNICA NACIONAL" bajo el libre consentimiento del (los) autor(es).

Al consultar esta tesis deberá acatar con las disposiciones de la Ley y las siguientes condiciones de uso:

- Cualquier uso que haga de estos documentos o imágenes deben ser sólo para efectos de investigación o estudio académico, y usted no puede ponerlos a disposición de otra persona.

- Usted deberá reconocer el derecho del autor a ser identificado y citado como el autor de esta tesis.

- No se podrá obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de licencia que el trabajo original.

El Libre Acceso a la información, promueve el reconocimiento de la originalidad de las ideas de los demás, respetando las normas de presentación y de citación de autores con el fin de no incurrir en actos ilegítimos de copiar y hacer pasar como propias las creaciones de terceras personas.

Respeto hacia sí mismo y hacia los demás.

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

SIMULACIÓN DE LAS TÉCNICAS DE DETECCIÓN ZF, MMSE Y MLD PARA UN SISTEMA MIMO – OFDM

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN ELECTRÓNICA Y TELECOMUNICACIONES

ORDÓÑEZ ALARCÓN AMANDA VALERIA

DIRECTOR: DR. DIEGO JAVIER REINOSO CHISAGUANO

Quito, julio 2019

AVAL

Certifico que el presente trabajo fue desarrollado por Amanda Valeria Ordóñez Alarcón, bajo mi supervisión.

Dr. Diego Javier Reinoso Chisaguano
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Amanda Valeria Ordóñez Alarcón, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

Amanda Valeria Ordóñez Alarcón

DEDICATORIA

A mis padres que a pesar de la distancia creyeron y confiaron en mí, y siempre han sido un ejemplo de lucha y persistencia ante las adversidades que se han presentado.

A mis hermanos que siempre han estado junto a mí brindándome su apoyo y comprensión.

AGRADECIMIENTO

A mis padres y hermanos por la confianza y el apoyo incondicional que han depositado en mí; gracias por nunca dejar que me rinda y creer en mí dándome fortaleza y confianza para asumir mis éxitos y fracasos.

A mi familia y amigos que me han estado junto a mí brindándome su apoyo desinteresado; gracias por el cariño y las experiencias compartidas.

Agradezco al Dr. Diego Reinoso por estar dispuesto a brindarme su ayuda, por su paciencia y total apoyo en el desarrollo del presente trabajo de titulación.

A la Escuela Politécnica Nacional por todas las enseñanzas impartidas a través de sus colaboradores, los profesores, que han sido de ayuda a lo largo de la vida universitaria y en el desarrollo del presente proyecto.

ÍNDICE DE CONTENIDO

AVAL I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO V
RESUMEN VII
ABSTRACT VIII
1. INTRODUCCIÓN 1
1.1 OBJETIVOS 2
1.2 ALCANCE 2
1.3 MARCO TEÓRICO 3
1.3.1. MIMO 3
1.3.2. OFDM 9
1.3.3. MIMO – OFDM 12
1.3.4. TÉCNICAS DE DETECCIÓN PARA SISTEMAS SM MIMO - OFDM 15
1.3.5. ANÁLISIS DE LA COMPLEJIDAD 19
2. METODOLOGÍA 23
2.1. IMPLEMENTACIÓN EN MATLAB 24
2.1.1. PROGRAMA PRINCIPAL – DETECTORES2 24
2.1.2 FUNCIÓN MIMOOFDMSINGLETEST3 34
2.1.3 DETECTORES 48
2.1.4 FUNCIÓN GRAFICARDETECTORES 54
3. RESULTADOS Y DISCUSIÓN 60
3.1. ESCENARIO DE SIMULACIÓN 60
3.2. SISTEMA MIMO – OFDM CON ARREGLO DE ANTENAS 2 X 2 Y MODULACIONES QPSK, 16QAM Y 64QAM. 61
3.3. SISTEMA MIMO – OFDM CON ARREGLO DE ANTENAS 2 X 4 Y MODULACIONES QPSK, 16QAM Y 64QAM. 64
3.4. SISTEMA MIMO – OFDM CON ARREGLO DE ANTENAS 4 X 4 Y MODULACIONES QPSK, 16QAM Y 64QAM. 67
3.5. TIEMPOS DE SIMULACIÓN DE LOS DETECTORES PARA ARREGLOS DE ANTENAS Y MODULACIONES. 72
4. CONCLUSIONES 75
4.1 CONCLUSIONES 75

4.2 RECOMENDACIONES 77
5. REFERENCIAS BIBLIOGRÁFICAS 78
6. ANEXOS 80
ANEXO A 81
ANEXO B 82

RESUMEN

En el presente proyecto de titulación se realiza la simulación de los algoritmos de detección ZF (*Zero-Forcing*), MMSE (*Minimum Mean Squared Error*) y MLD (*Maximum Likelihood Detection*) para un sistema MIMO – OFDM (*Multiple-input Multiple-output - Orthogonal Frequency-Division Multiple Access*). Con la finalidad de obtener el análisis del algoritmo con mejor rendimiento, los resultados obtenidos se presentan en gráficas de BER (*Bit Error Rate*) vs. Eb/No (*Energy Per-Bit To Noise Power Spectral Density Ratio*) y tiempos de simulación.

En el capítulo 1 se presenta la información, funcionamiento y principales características de los sistemas MIMO – OFDM, ZF, MMSE y MLD. También, en este capítulo se tiene un análisis teórico de la complejidad de los tres algoritmos de detección.

En el capítulo 2 se tienen diagramas de flujo que indican la estructura del programa realizado, ya sea para la parte de transmisión, recepción y detección. En este capítulo también se describe simultáneamente con los diagramas de flujo las funciones y comandos de Matlab (*MATrix LABoratory*) utilizados.

El capítulo 3 muestra los resultados de la implementación de las simulaciones realizadas para las distintas configuraciones y modulaciones. Aquí se tienen los gráficos del BER vs Eb/No, así como el gráfico de los tiempos de simulación.

Finalmente, en el capítulo 4 se presentan las conclusiones y recomendaciones obtenidas durante todo el proyecto.

PALABRAS CLAVE: MIMO, OFDM, ZF, MMSE, MLD

ABSTRACT

This project presents the simulation of the detection algorithms ZF (*Zero-Forcing*), MMSE (*Minimum Mean Squared Error*) and MLD (*Maximum Likelihood Detection*), for a MIMO – OFDM (*Multiple-input Multiple-output - Orthogonal Frequency-Division Multiple Access*) system. With the purpose of obtaining the analysis of the algorithm with better performance, the results are presented in graphs of BER (*Bit Error Rate*) vs. Eb/No (*Energy Per-Bit To Noise Power Spectral Density Ratio*) and simulation times.

Chapter 1 presents the information, operation and main characteristics of the MIMO-OFDM, ZF, MMSE and MLD systems. Also, this chapter includes a theoretical analysis of the complexity of the three detection algorithms.

In chapter 2 there are flowcharts that indicate the structure of the program for the transmission, reception and detection part. In this chapter, the functions and commands of Matlab (*MATrix LABoratory*) used are also described simultaneously with the are flowcharts.

Chapter 3 shows the results of the implementation of the simulations carried out for the different configurations and modulations. The graphs of the BER vs Eb / No are presented, as well as the graph of the simulation times.

Finally, chapter 4 presents the conclusions and recommendations obtained throughout the project.

KEYWORDS: MIMO, OFDM, ZF, MMSE, MLD

1. INTRODUCCIÓN

Los avances tecnológicos no paran, la demanda del canal inalámbrico en los últimos años está creciendo constantemente y es un hecho que la cantidad de espectro electromagnético disponible nunca aumentará. En la actualidad, los sistemas de comunicación inalámbrica son los más solicitados por su facilidad en la movilidad del usuario, lo cual ha provocado, que estos sistemas exploren diferentes soluciones para satisfacer a los mismos, como es la búsqueda de mayores velocidades para solventar dichas demandas.

El uso de la tecnología MIMO (*Multiple-input Multiple-output*) con la intervención de la técnica OFDM (*Orthogonal Frequency-Division Multiple Access*) resultan una combinación que permite el incremento de la tasa de transmisión de datos en los sistemas de comunicación. La tecnología de múltiples antenas (MIMO) para las comunicaciones inalámbricas, básicamente habla sobre, cuantas más antenas disponga el transmisor / receptor, más posibles serán las rutas de señal y mejor será el rendimiento en términos de velocidad de datos y confiabilidad del enlace. La técnica OFDM que acompaña a MIMO permite altas tasas de transmisión y alta eficiencia espectral, por ser una técnica de modulación que divide el ancho de banda en muchas subportadoras paralelas, incrementando así la duración del símbolo, lo que reduce la interferencia entre símbolos provocada por las trayectorias múltiples [1].

El sistema MIMO – OFDM ofrece mayor robustez y tasas de transmisión altas, sin necesidad de aumento en el ancho de banda o potencia transmitida, por las múltiples señales que se distribuyen simultáneamente a la misma frecuencia y por el mismo canal. En la recepción, la detección de la señal es un proceso complejo y obligatorio, debido a las múltiples señales recibidas y el ruido siempre presente en los sistemas. Existen varios algoritmos o técnicas que permiten la detección de las señales enviadas por el transmisor, como son ZF (*Zero-Forcing*), MMSE (*Minimum Mean Squared Error*) y MLD (*Maximum Likelihood Detection*) [2].

Los algoritmos ZF y MMSE basan su funcionamiento en la inversión del canal o cálculo de la inversa, diferenciándose el MMSE porque se puede obtener un mejor rendimiento minimizando el error cuadrático medio. La técnica MLD, es considerada como la solución óptima para la detección, pero presenta un gran reto debido a su complejidad [2].

El presente proyecto estará constituido por la simulación de las diferentes técnicas de detección presentadas anteriormente (ZF, MMSE y MLD) para un sistema MIMO – OFDM,

con la utilización del software MATLAB (*MATrix LABoratory*), teniendo en cuenta las diferentes modulaciones y arreglos de antenas. Con la información resultante de las simulaciones se obtendrán las gráficas del BER (*Bit Error Rate*) en función del Eb/No (*Energy Per-Bit To Noise Power Spectral Density Ratio*) y los tiempos de simulación de cada técnica para así deducir el que presente mejor rendimiento en base a su complejidad.

1.1 OBJETIVOS

El objetivo general de este Estudio Técnico es: Simular las técnicas de detección ZF, MMSE y MLD para un sistema MIMO – OFDM mediante la utilización de Matlab.

Los objetivos específicos de este Estudio Técnico son:

- Analizar la teoría necesaria para el desarrollo del proyecto técnico.
- Implementar el sistema MIMO – OFDM en el software de simulación.
- Realizar la implementación de los métodos de detección.
- Efectuar pruebas de funcionamiento y comparación de resultados.

1.2 ALCANCE

En este proyecto se realizará la simulación de un sistema MIMO – OFDM de acuerdo con la Figura 1.1.

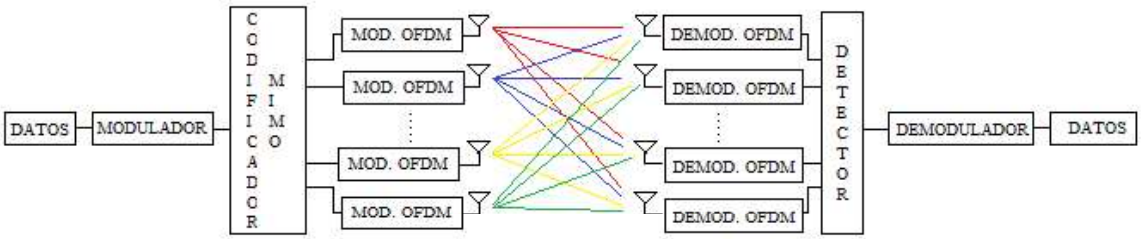

Figura 1.1 Sistema de comunicación MIMO – OFDM [1].

Se tiene en consideración para la simulación del sistema MIMO – OFDM los siguientes factores:

- No se considera la etapa de codificación FEC (*Forward Error Correction*) (Sistema no codificado).
- Las modulaciones empleadas son QPSK (*Quadrature Phase Shift Keying*), 16QAM (*Quadrature Amplitude Modulation)* y 64QAM.
- Se considera sincronización perfecta.
- Se considera que se conoce el canal.
- Se considera un canal AWGN (*Additive White Gaussian Noise*), desvanecimiento Rayleigh y propagación multitrayectoria.

En la Figura 1.1 se observa el sistema MIMO – OFDM el cual inicia con los datos a transmitir, los cuales ingresan a la etapa de modulación, las modulaciones a implementar son QPSK, 16QAM y 64QAM, para cada caso. Posteriormente los datos ya modulados ingresan al codificador MIMO y cada flujo de símbolos se modula usando OFDM para luego ser transmitidos por las antenas hacia el canal. En la etapa de recepción las antenas reciben las señales, las cuales son demoduladas gracias a la FFT (*Fast Fourier Transform*). Como parte final se realiza la detección y la demodulación de la señal [1].

Para la etapa de detección, la cual es de estudio en este proyecto, se implementará la simulación de tres algoritmos como: ZF, MMSE, MLD, y cada uno será analizado con diferentes configuraciones de antenas (2x2, 2x4 y 4x4) y modulaciones (QPSK, MMSE Y ML).

La simulación arrojará gráficas de BER vs. Eb/No y de los tiempos de simulación de cada uno de los detectores y sus diferentes configuraciones. La gráfica BER vs. Eb/N0 presentará la tasa de errores obtenida para los tres algoritmos implementados y por cada una de las modulaciones y configuraciones de antenas. También se realizará un análisis teórico de la complejidad de los tres algoritmos de detección.

1.3 MARCO TEÓRICO

1.3.1. MIMO

La tecnología MIMO se ha venido investigando desde varios años atrás. A continuación, en la Tabla 1.1 se observa la evolución de esta tecnología con el paso de los años desde la última década del siglo XX.

Tabla 1.1. Evolución de la tecnología MIMO [3].

Año	Autores	Reseña
1984	Jack Winters	Patente sobre comunicaciones inalámbricas con múltiples antenas en Bell Laboratories.
1987	Jack Winters	Estudio de los límites fundamentales en la velocidad de datos de múltiples sistemas de antenas en un entorno de desvanecimiento de Rayleigh.
1993	Arogyaswami Paulraj Thomas Kailath	Patente sobre la multiplexación espacial enfatizando en las aplicaciones a la radiodifusión inalámbrica. Se propuso el concepto de multiplexación espacial utilizando MIMO.
1995	Emre Teletar	Artículo "Capacidad de canales gaussiano multi antena".
1996	Greg Raleigh Gerard Joseph Foschini	Inventaron nuevos enfoques que implican técnicas de codificación de tiempo espacial.
1998	Raleigh & John	Probaron los enfoques que implican técnicas de codificación de tiempo espacial para aumentar la eficiencia espectral de los sistemas MIMO.
1999	Thomas Marzetta Bertrand Hochwald	Artículo que proporciona un estudio riguroso sobre el enlace MIMO Rayleigh teniendo en cuenta los aspectos de la teoría de la información.
2001	Iospan Wireless Inc.	El primer sistema MIMO comercial fue desarrollado.
2006 -	Broadcom e Intel	Mejoran los sistemas inalámbricos de red de área local (WLAN) con MIMO. El nuevo estándar de los sistemas de WLAN se denomina IEEE 802.11 n.

Con la información de la Tabla 1.1 se tiene claro que la tecnología MIMO ha tenido un avance constante en los sistemas de comunicación inalámbricos y está en el centro de varios de estos. MIMO con el paso de los años ha ido evolucionando y con eso se han presentado varios sistemas como: sistemas MIMO de acceso múltiple, MIMO ad-hoc, MIMO cooperativo y MIMO cooperativo en redes de sensores, etc. [4].

1.3.1.1. ¿Qué es MIMO?

El uso de múltiples antenas en el transmisor y receptor en sistemas inalámbricos, conocido como tecnología MIMO, ha ganado popularidad en la última década debido a sus capacidades de mejora del rendimiento, haciendo la señal más fuerte y confiable, que alcance mayores distancias, mayor capacidad de usuarios. Lo más importante, estas características vienen sin aumento en el ancho de banda o la potencia transmitida, que son recursos escasos; más bien, vienen a costa de complejidad en el sistema [3].

Las comunicaciones por el canal inalámbrico presentan importantes desafíos, como son el desvanecimiento por multi-path (múltiples caminos), que es la llegada de la señal transmitida a un receptor a través de diferentes ángulos, diferentes retardos de tiempo y/o diferencias de frecuencia debido a la dispersión de ondas electromagnéticas en el medio ambiente. Por lo tanto, la potencia de la señal recibida podría oscilar en el espacio, frecuencia y/o tiempo a través de la superposición aleatoria de los componentes de múltiples trayectos que impactan. La fluctuación aleatoria de la señal, desvanecimiento, puede afectar la calidad y confiabilidad de la comunicación inalámbrica [4].

MIMO, al ser un avance en las comunicaciones inalámbricas, ofrece múltiples beneficios que ayudan a superar los retos presentes en el canal inalámbrico y deficiencias del sistema. Los sistemas MIMO exploran de forma constructiva la propagación de múltiples trayectorias, utilizando las diferentes rutas proporcionadas por las múltiples antenas en el transmisor y el receptor (dimensión espacial).

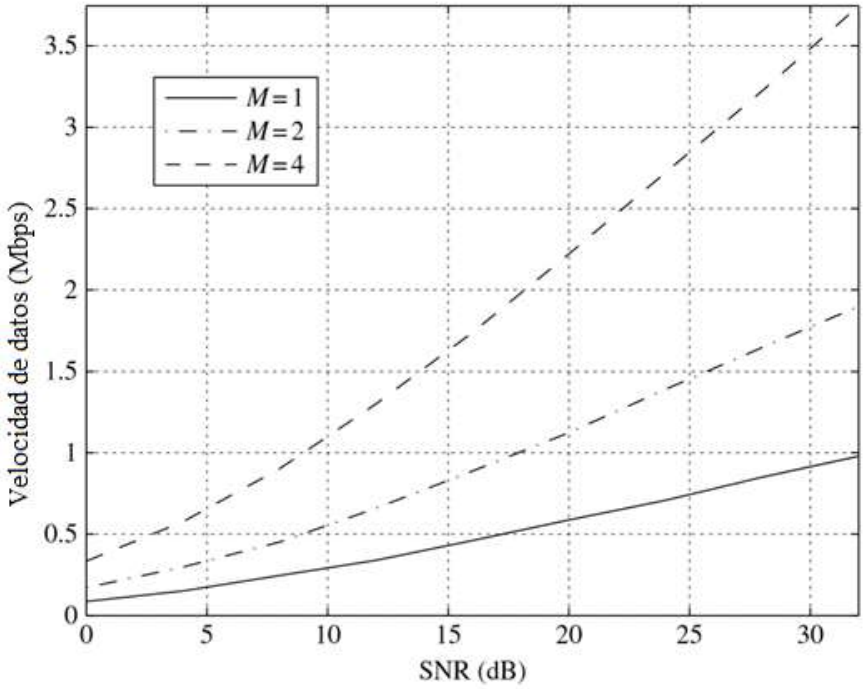

Figura 1.2 Velocidad de datos vs. SNR para diferentes configuraciones de antena [4].

En la Figura 1.2 se presenta un ejemplo de los tipos de ganancias que se esperan del uso de MIMO, con una gráfica de Velocidad de datos vs. SNR en un canal de 100 kHz, para un

arreglo de antenas M x M (M antenas transmisoras y M antenas receptoras) con M = 1, 2, 4. La respuesta del canal se asume constante en este ejemplo [4].

Asumiendo que el receptor reciba un SNR (*Signal-to-Noise Ratio*) de 30 decibelios (dB), para un sistema M = 1, 1x1, se tiene una velocidad de datos de aproximadamente 0.9 Mbps. Con los sistemas de M = 2 y 4 (sistemas MIMO), se obtiene velocidades de 1.8 y 3.5 Mbps respectivamente. Este aumento en la velocidad de datos se realiza sin elevar potencia o ancho de banda respecto a un sistema con M = 1. El sistema 1 x 1 puede alcanzar una velocidad de 3.5 Mbps con un SNR en la recepción de 30 dB si se aumenta el ancho de banda a 400 kHz o manteniendo el ancho de banda en 100 kHz en la recepción sería necesario aumentar el SNR a 116 dB.

Con el ejemplo antes presentado se puede apreciar las mejoras significativas del uso de múltiples antenas en transmisión y recepción, con respecto a sistemas de una sola entrada y salida, ya que, con los mismos recursos, los resultados obtenidos no necesitaron de aumento de ancho de banda, potencia o velocidad de datos.

Entonces, MIMO es una tecnología que permite altas tasas de velocidad, mayor capacidad de usuarios, más fiabilidad, sin necesidad de aumento de ancho de banda o potencia transmitida, gracias a las múltiples señales que viajan por el canal inalámbrico simultáneamente, aprovechando así las ventajas de la propagación multitrayectoria [3] [4].

1.3.1.2. Componentes del sistema MIMO

El sistema de comunicación MIMO está constituido por varios bloques, los cuales se observan en la Figura 1.3, donde el sistema inicia con los bits de información, los cuales para ser trasmitidos ingresan en el bloque de codificación, como, por ejemplo, un codificador convolucional, binario, etc., y se intercalan. La palabra código intercalada se asigna a símbolos de datos, como las modulaciones QPSK o QAM. Estos símbolos de datos son introducidos en un codificador de espacio-tiempo que genera una o más secuencias de datos espaciales. Estas secuencias se transmiten a través de las antenas de transmisión y se propagan a lo largo del canal y llegan a las antenas de recepción. El receptor recoge las señales en la salida de cada elemento de sus antenas e invierte las operaciones del transmisor para decodificar los datos: decodificación espacio-tiempo, demapeo del símbolo QPSK o QAM, desentrelazado y decodificación [4].

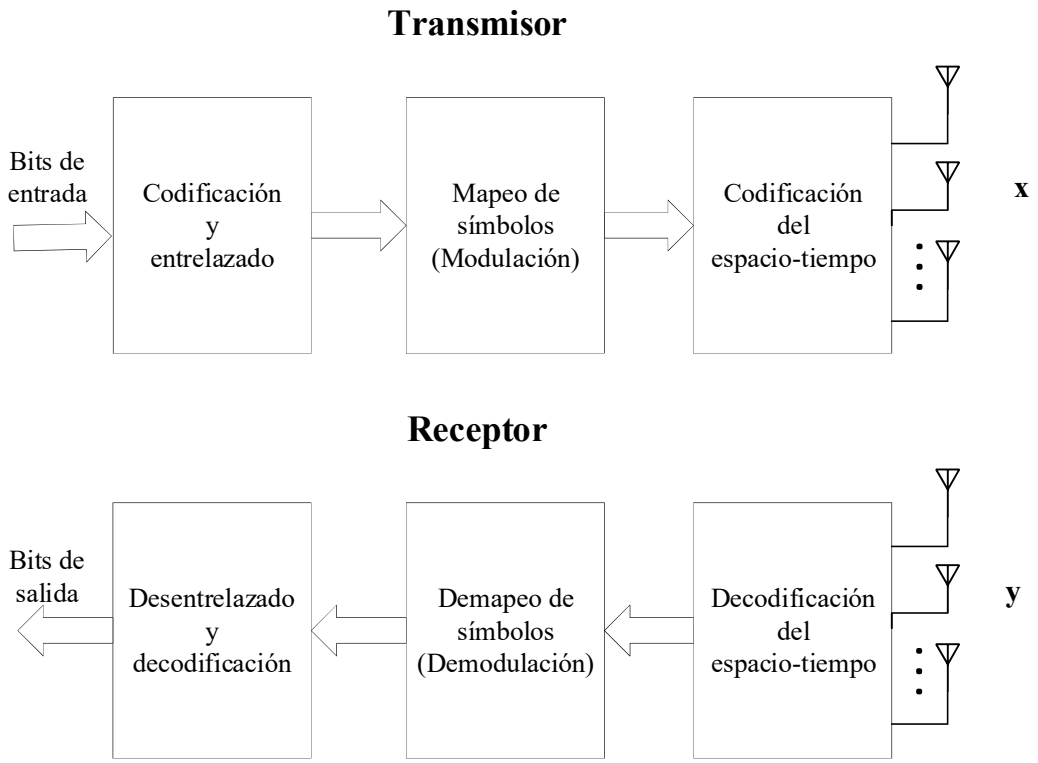

Figura 1.3 Diagrama de un sistema MIMO [4].

Al no ser un modelo preestablecido o fijo, una serie de variaciones se pueden realizar en la colocación relativa de los bloques, la funcionalidad y las interacciones entre los mismos dependiendo de los requerimientos del sistema.

En el sistema de comunicación MIMO la característica que le da su popularidad y enorme crecimiento es el canal inalámbrico, conocido como canal **H**. En la Figura 1.4, se observa una representación del canal para un sistema MIMO, en el que para antenas múltiples en el receptor y en el transmisor, se considera una matriz de antenas con n_t elementos en el

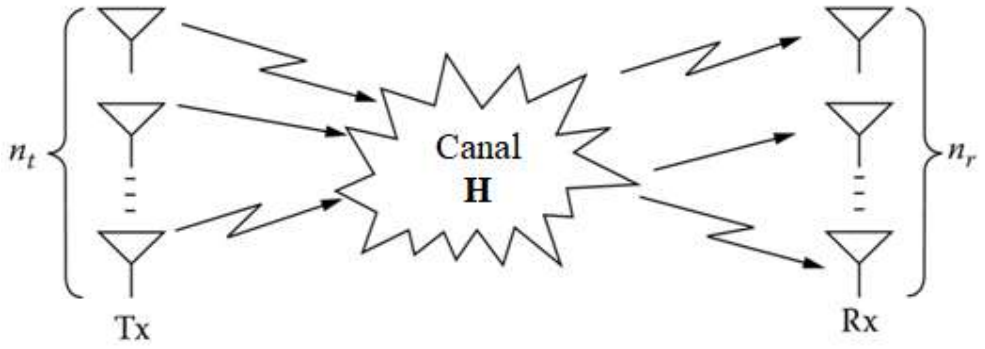

Figura 1.4 Canal MIMO [5].

transmisor y un matriz de antenas con n_r elementos en el receptor. La respuesta impulsiva del canal entre el j-ésimo transmisor y i-ésimo receptor se denota como $h_{ij}(\tau, t)$. El canal MIMO puede ser descrito entonces por la matriz n_t × n_r $\boldsymbol{H}(\tau, t)$, de la Ecuación 1.1:

$$\mathbf{H}(\tau, t) = \begin{bmatrix} h_{1,1}(\tau, t) & h_{1,2}(\tau, t) & \cdots & h_{1,n_t}(\tau, t) \\ h_{2,1}(\tau, t) & h_{2,2}(\tau, t) & \dots & h_{2,n_t}(\tau, t) \\ \vdots & \vdots & \ddots & \vdots \\ h_{n_r,1}(\tau, t) & h_{n_r,2}(\tau, t) & \cdots & h_{n_r,n_t}(\tau, t) \end{bmatrix} \tag{1.1}$$

los elementos de la matriz son números complejos que corresponden a la atenuación y al desplazamiento de fase que el canal inalámbrico introduce a la señal que llega al receptor con retardo τ. La notación de entrada-salida del sistema MIMO se puede expresar ahora mediante la siguiente Ecuación 1.2:

$$\boldsymbol{y}(t) = \mathbf{H}(\tau, t) \otimes \mathbf{s}(t) + \mathbf{u}(t) \tag{1.2}$$

donde $\otimes$ denota convolución, **s (t)** es un vector n_t × 1 correspondiente a las señales transmitidas de n_t, **y (t)** es un vector n_r × 1 correspondiente a las señales de n_r recibidas y **u (t)** es el ruido blanco aditivo [5].

1.3.1.3. MIMO con Multiplexación Espacial (SM – MIMO)

El uso de múltiples antenas en un sistema significa la existencia de diferentes rutas de propagación. Existen varios métodos para visualizar las diferentes mejoras o ventajas que proporciona un sistema MIMO, como, por ejemplo, para mejorar la confiabilidad del sistema se tiene la diversidad espacial que opta por enviar los mismos datos a través de las diferentes rutas de propagación (espaciales) y con el objetivo de mejorar la velocidad de datos del sistema, se puede colocar diferentes partes de los datos en diferentes trayectorias de propagación (multiplexación espacial). En la Figura 1.5 se observa la diferencia entre diversidad y multiplexación espacial. En la diversidad, los mismos datos de información se envían por los diferentes canales espaciales colocándolos en tres diferentes antenas de transmisión, en cambio, en la multiplexación espacial, cada bit de información se multiplexa entre canales diferentes, aumentando así la velocidad de datos [6].

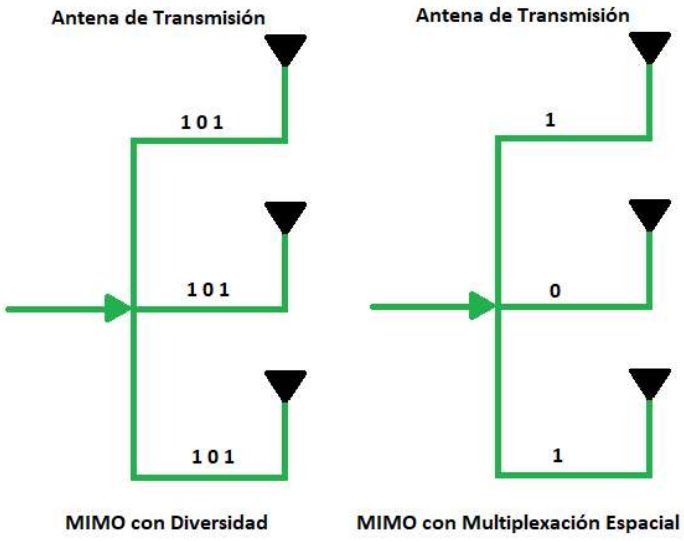

Figura 1.5. MIMO con Diversidad y Multiplexación Espacial [6].

Para el presente caso de estudio (MIMO – OFDM), se utilizará MIMO con multiplexación espacial.

1.3.2. OFDM

1.3.2.1. Fundamentos básicos de OFDM

Una señal OFDM es una suma de subportadoras que se modula individualmente mediante el uso de M – PSK o M – QAM. La señal trasmitida $s(t)$ se representa como la Ecuación 1.3 [9]:

$$s(t) = \sum_{i=-\infty}^{\infty} \sum_{k=1}^{N} c_{ki} s_k(t - iT_s) \tag{1.3}$$

Donde:

c_{ki} símbolo de información *i* asociado a la subportadora *k*.

s_k forma de onda de la subportadora *k*.

T_s periodo de símbolo.

N subflujos de datos.

- **ORTOGONALIDAD**

OFDM aprovecha la ortogonalidad entre las subportadoras para comprimir el espectro y hacerlo más eficiente. Como principio básico OFDM divide el flujo de datos en flujos de tasas más pequeñas que se transmiten simultáneamente a través de un número de subportadoras. Las formas de onda de las subportadoras deben tener espectros de

transmisión solapados. Deben ser ortogonales para permitir la separación simple de estas subportadoras superpuestas en el receptor. La palabra "ortogonal" indica que hay una relación matemática precisa entre las frecuencias de las subportadoras en el sistema. En un sistema FDM normal (Figura 1.6. (a)), muchas subportadoras están separadas de tal manera que las señales pueden ser recibidas usando filtros convencionales y demoduladores. Las bandas de guarda se introducen entre los canales en el dominio de frecuencia, lo que resulta en una disminución de la eficiencia del espectro. Sin embargo, es posible organizar las subportadoras en una señal OFDM para que las bandas laterales se solapen y las señales todavía se reciban sin interferencia de subportadora adyacente (Figura 1.6 (b)). Para ello, las subportadoras deben ser matemáticamente ortogonales [7] [8].

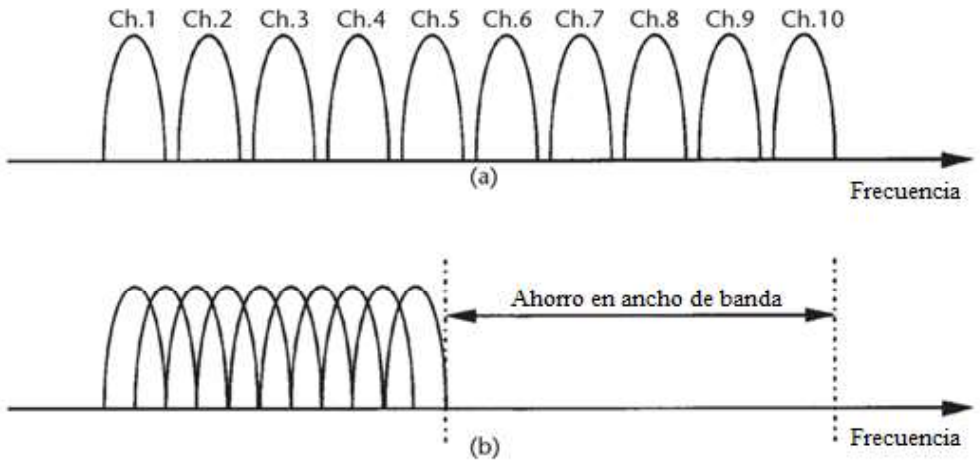

Figura 1.6 Concepto de la señal OFDM: (a) Técnica multi-portadora convencional y (b) Técnica de modulación multi-portadora ortogonal [7].

La Figura 1.6 ilustra la diferencia entre la técnica multi-portadora convencional y la técnica de modulación multi-portadora ortogonal. Mediante el uso de la técnica de modulación multi-portadora ortogonal, se ahorra aproximadamente un 50% de ancho de banda [7].

En la Figura 1.7 se observa el espectro de una señal OFDM con 6 subportadoras. Como se puede apreciar, el punto máximo de cada subportadora coincide con el cruce por cero de las otras subportadoras. En consecuencia, a pesar de estar sobrelapadas, las subportadoras no se interfieren. Por esta característica OFDM es una transmisión de subportadoras "ortogonales".

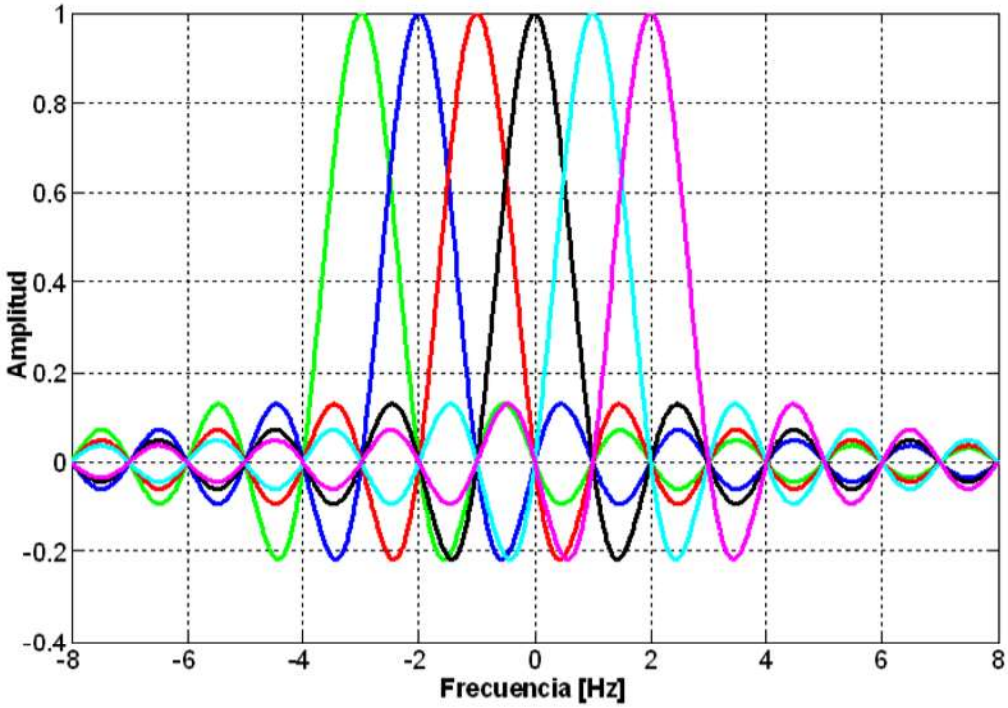

Figura 1.7 Espectro de una señal OFDM con 6 subportadoras [13].

- **TRANSFORMADA DE FOURIER**

El estudio realizado por Weinstein y Ebert permitió que la transformada discreta de Fourier (*Discrete Fourier Transform, DFT*) y la transformada inversa discreta de Fourier (*Inverse Discrete Fourier Transform, IDFT*) sean utilizadas en los procesos de modulación y demodulación OFDM. Los cálculos de la DFT e IDFT se pueden implementar de manera más eficiente con el uso de la transformada rápida de Fourier (*Fast Fourier Transform, FFT*) y la transformada rápida inversa de Fourier (*Inverse Fast Fourier Transform, IFFT*), las cuales se derivan de la DFT, debido a que en la DFT el cálculo de los puntos se realiza uno por uno, mientras que para FFT/IFFT, el cálculo se realiza simultáneamente, lo que implica ahorro de tiempo. Las ecuaciones para la función FFT/IFFT se pueden derivar de la ecuación general de DFT [7] [10].

De la Ecuación 1.3, suponiendo que se muestrea $s(t)$ para cada intervalo T_s/N, se tiene la señal muestreada s_m en la Ecuación 1.4:

$$s_m = \sum_{k=1}^{N} c_k e^{j2\pi f_k \frac{(m-1)T_s}{N}} \tag{1.4}$$

Reemplazando $f_k = \frac{m-1}{N}$ en la Ecuación 1.4, se tiene la IDFT en la Ecuación 1.5:

$$s_m{}' = \sum_{k=1}^{N} c_k e^{j2\pi\frac{(m-1)(k-1)}{N}} = IDFT(c_k) \quad (1.5)$$

En el intervalo $m \in [1, N]$. La señal por trasmitir se puede generar usando la IDFT, y puede ser recuperada en la recepción mediante el uso de la DFT, cuya ecuación se muestra en la Ecuación 1.6 [9]. Siendo r_m la señal recibida muestreada.

$$DFT(r_m) = \sum_{k=1}^{N} r_m e^{-j2\pi\frac{(m-1)(k-1)}{N}} \quad (1.6)$$

- **PREFIJO CÍCLICO**

El paso de las señales por el canal multitrayecto causa interferencia inter símbolo (*Inter-Symbol Interference, ISI*). En un sistema OFDM, esto hace que la ortogonalidad de las subportadoras se pierda, obteniendo interferencia inter portadora (*Inter-Carrier Interference, ICI*). Para superar estos problemas causados por el canal, se introduce el uso del prefijo cíclico (*Cyclic Prefix, CP*), el cual es una extensión cíclica del símbolo OFDM. El prefijo cíclico se construye asignándole una copia de la última parte del símbolo OFDM al inicio de este, quedando el símbolo OFDM final compuesto por el prefijo cíclico y el símbolo OFDM inicial, como se muestra en la Figura 1.8. El uso del CP brinda beneficios como, evitar ISI porque actúa como un espacio de protección entre símbolos sucesivos y evitar ICI ya que cualquier réplica de la señal que llegue atrasada seguirá teniendo un número entero de ciclos, manteniendo así a las subportadoras ortogonales [8] [9].

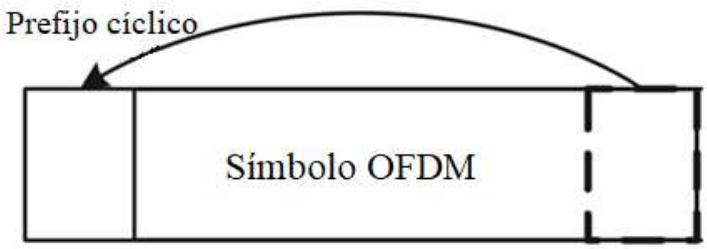

Figura 1.8 Símbolo OFDM con prefijo cíclico [2].

1.3.3. MIMO – OFDM

En varias tecnologías, OFDM se utiliza con varias combinaciones y técnicas como MIMO. Esta combinación ayuda en gran medida a mitigar el desvanecimiento multitrayectoria y alcanzar altas velocidades de datos. Con MIMO-OFDM se reduce complejidad y manipulación del receptor a medida que la información es distribuida por las múltiples

subportadoras y transmitidas en diferentes niveles de frecuencia. MIMO-OFDM envía flujo de información de datos independientes para aumentar la velocidad espacial sobre diferentes antenas. Lipson Wireless introdujo por primera vez el esquema MIMO-OFDM [14].

Algunas características de MIMO – OFDM [14]:

- La utilización de múltiples antenas en transmisión y recepción con OFDM, reduce el requerimiento de más ancho de banda para la transmisión de la señal, pero incrementando la velocidad de los datos.
- MIMO-OFDM mejora la estimación de canal y la detección de señales.
- MIMO-OFDM es una técnica prometedora para la transmisión de banda ancha con alta eficiencia espectral y se puede utilizar eficazmente en los sistemas inalámbricos de alta velocidad de datos.

1.3.3.1. Modelo del sistema MIMO - OFDM

El sistema MIMO – OFDM tiene varios componentes los cuales se encuentran interactuando entre sí en el transmisor y receptor. Como se explica a continuación.

- **TRANSMISOR**

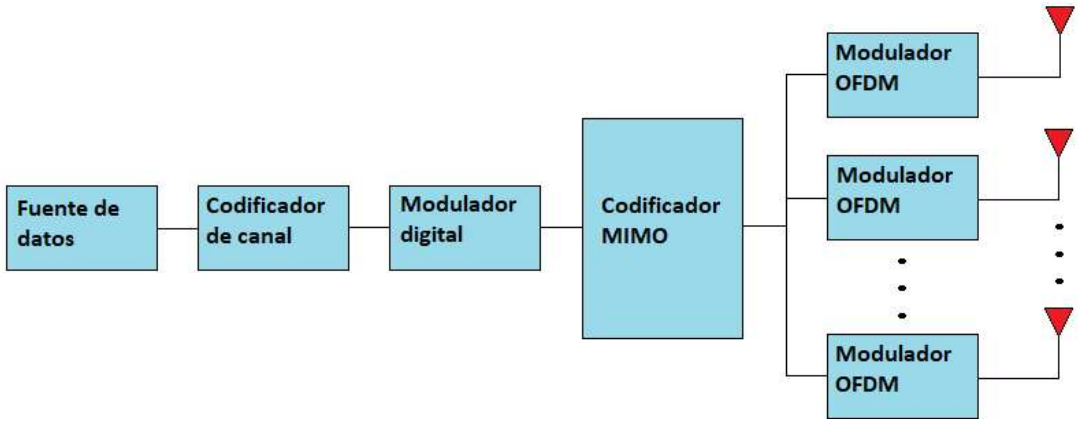

Figura 1.9 Diagrama de un transmisor MIMO – OFDM [12].

La Figura 1.9 representa un diagrama de bloques para un transmisor MIMO – OFDM. Inicialmente consta de una fuente de datos (*bitstream*), donde los datos que se desea enviar salen e ingresan al codificador de canal, en donde el bitstream es codificado utilizando técnicas como FEC (*Forward Error Correction*), al codificar el bitstream, este ingresa en el bloque de modulación, aquí es convertido en una constelación, dependiendo

del tipo de modulador empleado como, M – QPSK o M – QAM. Posteriormente los datos modulados ingresan al codificador MIMO, donde cada salida corresponde a un flujo de símbolos, y cada uno de los flujos de símbolos son modulados utilizando OFDM para posteriormente ser transmitidos por las correspondientes antenas. En el modulador OFDM se agrega el prefijo cíclico y preámbulo a los símbolos OFDM, y finalmente la trama de datos es enviada al canal inalámbrico [13].

- **RECEPTOR**

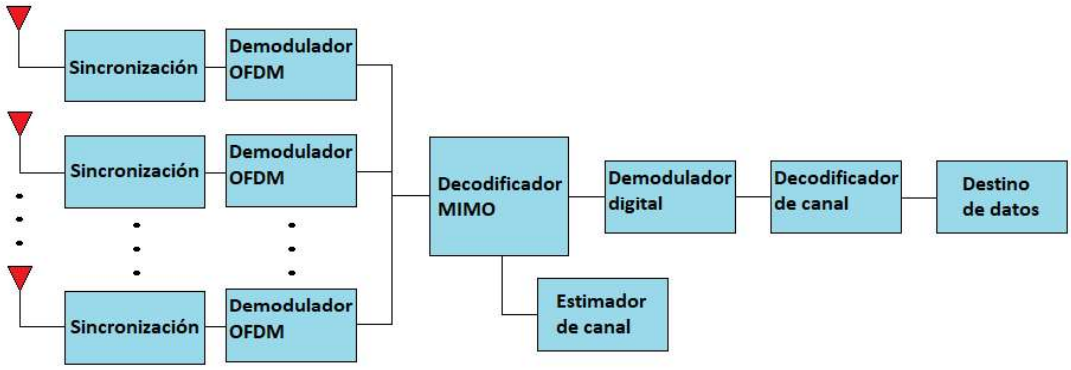

Figura 1.10 Diagrama de un receptor MIMO – OFDM [12].

En la Figura 1.10 se observa el diagrama de bloques de un receptor MIMO – OFDM. Los símbolos que fueron transmitidos son recibidos en las antenas, como primer paso estos símbolos son sincronizados en frecuencia y tiempo gracias al preámbulo. Una vez extraído el preámbulo y el prefijo cíclico de los símbolos OFDM, estos pasan al demodulador OFDM, donde son demodulados gracias a la FFT. El estimador de canal permite determinar la matriz del canal basándose en los valores recibidos en las subportadoras piloto. La matriz del canal (**H**) permite al decodificador MIMO, determinar los símbolos OFDM que fueron transmitidos, los cuales son luego demodulados y decodificados. Finalmente, el bitstream llega a su destino [13].

Estos modelos de transmisor y receptor para MIMO – OFDM, pueden variar dependiendo de los requerimientos de cada sistema. Para el presente caso de estudio, como se mencionó anteriormente, no se considera la etapa de codificación FEC (sistema no codificado), se utilizará modulaciones QPSK, 16QAM y 64QAM. Para el receptor, se considera que se conoce la matriz del canal y sincronización perfecta. El sistema MIMO – OFDM planteado según los requerimientos previamente mencionados se observa en la Figura 1.11 para el transmisor y la Figura 1.12 para el receptor.

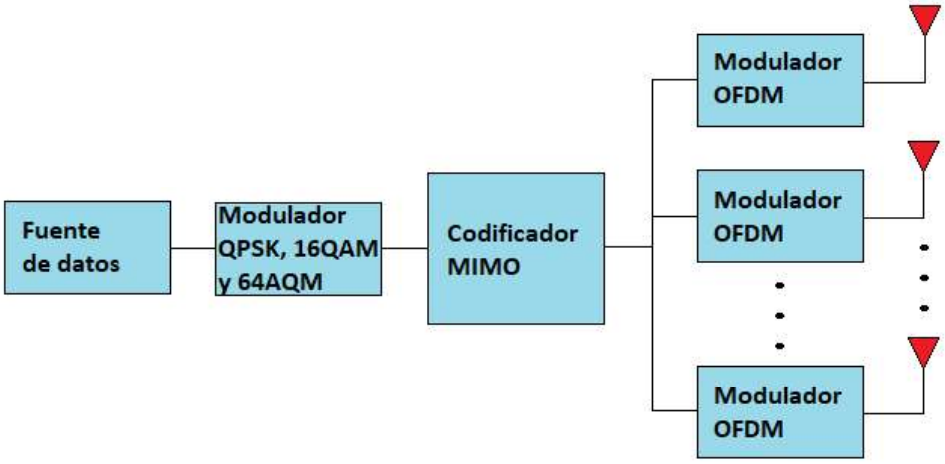

Figura 1.11 Diagrama de transmisor MIMO – OFDM usado en este estudio.

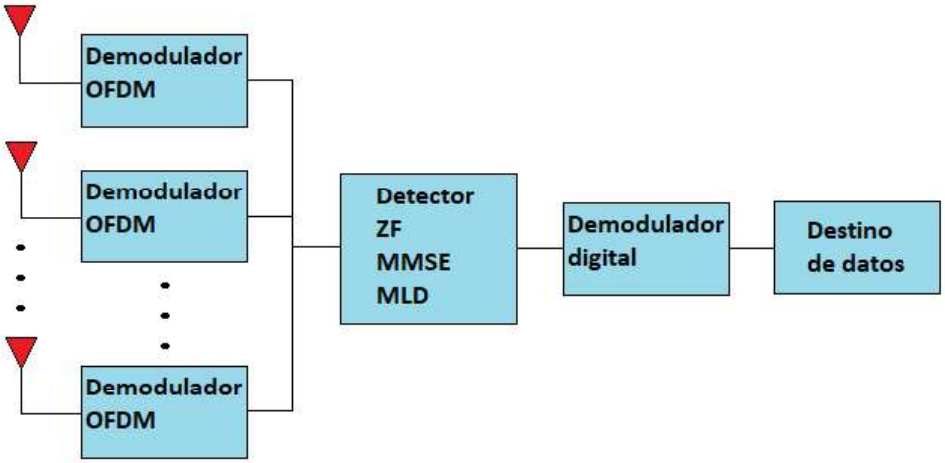

Figura 1.12 Diagrama de receptor MIMO – OFDM usado en este estudio.

1.3.4. TÉCNICAS DE DETECCIÓN PARA SISTEMAS SM MIMO - OFDM

Los sistemas SM MIMO (*Spatial Multiplexing Multiple-input Multiple-output*) permiten mayores velocidades de datos. Sin embargo, en la parte de recepción se presentan inconvenientes debido a que la detección de la señal se vuelve una tarea difícil. Aquí se revisarán técnicas de detección para sistemas SM MIMO [2]. Cabe mencionar que las técnicas de detección para sistemas SM MIMO se pueden utilizar también en un sistema SM MIMO-OFDM (*Spatial Multiplexing Multiple-input Multiple-output - Orthogonal Frequency-Division Multiple Access)*, la diferencia radica en que para un sistema SM MIMO-OFDM el proceso de detección se realiza para cada una de las subportadoras de

datos. La Figura 1.13 presenta el modelo de señal de un sistema SM MIMO con N_T transmisores y N_R receptores.

Donde:

$\boldsymbol{H}$ Matriz de canal.

h_{ji} Estado del canal entre el i-ésimo transmisor y j-ésimo receptor.

x_i Señal de transmisión de la i-ésima antena de transmisión.

y_j Señal de recepción de la j-ésima antena de recepción.

z_j Ruido blanco gaussiano con varianza σ_z^2 en la j-ésima antena de recepción.

$j = 1,2,3, \ldots, N_R$

$i = 1,2,3, \ldots, N_T$

$\boldsymbol{x} = [x_1, x_2, x_3, \ldots, x_{N_T}]^T$ Vector de datos de usuarios multiplexados espacialmente.

$\boldsymbol{y} = [y_1, y_2, y_3, \ldots, y_{N_R}]^T$ Vector de las señales recibidas.

$\boldsymbol{z} = [z_1, z_2, z_3, \ldots, z_{N_R}]^T$ Vector de ruido.

Entonces, el sistema SM – MIMO $N_T \; x \; N_R$ es representado por la Ecuación 1.7:

$$\boldsymbol{y} = \boldsymbol{H} * \boldsymbol{x} + \boldsymbol{z} \tag{1.7}$$

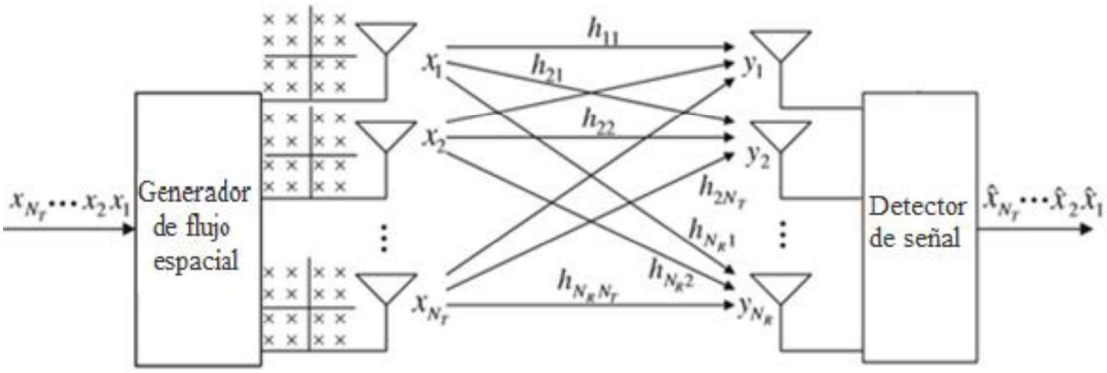

Figura 1.13 Sistema SM MIMO [2].

1.3.4.1. Detección lineal de la señal.

Para la reducción de la complejidad, se utilizan detectores lineales. Este método trata todas las señales trasmitidas como interferencias exceptuando el flujo deseado de la antena de transmisión objetivo. Las señales restantes se minimizan o anulan durante la detección de la señal deseada. Para detectar las señales de cada antena, el efecto del canal se invierte

por una matriz de pesos **W** (Ecuación 1.8), tal que, la detección de cada símbolo se da por una combinación lineal de las señales recibidas [2].

$$\tilde{\boldsymbol{x}} = \left[\tilde{x}_1 \tilde{x}_2 \cdots \tilde{x}_{N_T}\right]^T = \boldsymbol{W}\boldsymbol{y} \tag{1.8}$$

Entre los métodos que se utilizan en la detección lineal se tiene el ZF y MMSE, que se revisarán a continuación.

1.3.4.1.1. Zero-Forcing (ZF)

El detector ZF induce menor complejidad en comparación con otros detectores como el error medio cuadrado mínimo (*Minimum Mean Squared Error, MMSE*) o el de máxima probabilidad (*Maximum Likelihood, ML*). La técnica de detección ZF elimina la interferencia provocada por el resto de los símbolos o capas [2] [15].
El detector Zero-Forcing anula la interferencia con la siguiente matriz de pesos presentada en la Ecuación 1.9:

$$\boldsymbol{W}_{ZF} = (\boldsymbol{H}^H \cdot \boldsymbol{H})^{-1}\boldsymbol{H}^H \tag{1.9}$$

Donde $(\cdot)^H$ denota la operación de transposición Hermitiana. Para obtener el vector de símbolos transmitidos se multiplica la matriz de pesos por el vector de símbolos recibidos como se muestra en la Ecuación 1.10:

$$\tilde{\boldsymbol{x}}_{\boldsymbol{ZF}} = \boldsymbol{W}_{\boldsymbol{ZF}}\boldsymbol{y} \tag{1.10}$$

1.3.4.1.2. Minimum Mean Squared Error (MMSE)

El detector MMSE presenta las mismas características que el detector ZF, con la diferencia que éste utiliza la varianza de ruido. MMSE ayuda a reducir el impacto del ruido de fondo. Para maximizar la relación señal/interferencia & ruido (SINR) después de la detección, la matriz de pesos MMSE se da según la Ecuación 1.11:

$$\boldsymbol{W}_{MMSE} = (\boldsymbol{H}^H \cdot \boldsymbol{H} + \sigma_z^2 \boldsymbol{I})^{-1}\boldsymbol{H}^H \tag{1.11}$$

Para aplicar el detector MMSE requiere información estadística de la varianza de ruido, σ^2. Usando la matriz de pesos del detector MMSE se puede obtener la señal transmitida usando la Ecuación 1.12 [2] [15]:

$$\tilde{\boldsymbol{x}}_{MMSE} = \boldsymbol{W}_{MMSE}\boldsymbol{y} \tag{1.12}$$

1.3.4.2. Maximum Likelihood (ML)

El detector ML según [8] es el cálculo de la distancia entre el vector de la señal recibida y el producto de todos los vectores de señal transmitidos posibles con el canal **H** dado, y encuentra el que tiene la distancia mínima.

La detección con ML ayuda a minimizar la probabilidad de error, mediante la búsqueda de todos los vectores candidatos y selección del máximo probable con la menor probabilidad de error [16]. ML determina la estimación del vector de señal transmitida $\boldsymbol{x}$ como se muestra en la Ecuación 1.13 [2]:

$$\tilde{\boldsymbol{x}}_{ML} = arg\ min_{x \in C^{N_T}} \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\|^2 \tag{1.13}$$

Donde:

C	Denotan un conjunto de puntos posibles de la constelación.
N_T	Número de antenas de transmisión.
$\|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\|^2$	Corresponde a la métrica ML.

Dado que se lleva a cabo una búsqueda exhaustiva para encontrar el vector de ML y el número de vectores candidatos para $\boldsymbol{x}$ es $\boldsymbol{C}^{\boldsymbol{N_T}}$, la complejidad crece exponencialmente con el número de antenas de transmisión. El método ML logra el rendimiento óptimo cuando todos los vectores transmitidos son igualmente probables. Sin embargo, su complejidad aumenta exponencialmente con el orden de modulación y/o el número de antenas de transmisión. Para la detección ML, aunque su rendimiento es óptimo, la complejidad extremadamente alta hace que sea poco realista para ser empleado en sistemas prácticos con gran número de antenas [2] [16].

Por ejemplo, se considera un sistema MIMO 2 x 2 con modulación 4QAM. Sin tener en cuenta la energía de la señal, los vectores candidatos de señal de transmisión se presentan en la Tabla 1.2. Se tiene los siguientes valores para la matriz de canal $\boldsymbol{H} = \begin{bmatrix} 0.54 + 0.32i & -2{,}26 - 0.43i \\ 1.82 - 1.31i & 0.86 + 0.34i \end{bmatrix}$ y el vector de ruido $\boldsymbol{z} = \begin{bmatrix} 0.04 + 0.04i \\ -0.03 - 0.01i \end{bmatrix}$. Suponiendo que se transmite el octavo vector candidato en la Tabla 1.2 y es $\boldsymbol{x} = \begin{bmatrix} -1 + i \\ 1 + i \end{bmatrix}$, la señal recibida se obtiene como: $y = H * x + z = \begin{bmatrix} -2{,}65 - 2{,}43i \\ -0.03 + 4.33i \end{bmatrix}$. Sustituyendo todos los vectores candidatos en la ecuación $d_{ml} = \|y - Hx\|^2$, se obtienen 16 valores d_{ml} que se presentan en la Tabla 1.3. Como se puede observar, el d_{ml} más bajo es 0.0042 que corresponde al símbolo

$x_{ML} = \begin{bmatrix} -1+i \\ 1+i \end{bmatrix}$ en la Tabla 1.2. Por lo tanto, el vector de símbolo se ha detectado correctamente con la detección ML [16].

Tabla 1.2. Vectores candidatos para un sistema MIMO 2 x 2 con 4QAM [16].

s	
1	-1-i, -1-i
2	-1-i, -1+i
3	-1-i, 1-i
4	-1-i, 1+i
5	-1+i, -1-i
6	-1+i, -1+i
7	-1+i, 1-i
8	-1+i, 1+i
9	1-i, -1-i
10	1-i, -1+i
11	1-i, 1-i
12	1-i, 1+i
13	1+i, -1-i
14	1+i, -1+i
15	1+i, 1-i
16	1+i, 1+i

Tabla 1.3. d_{ml} correspondiente a **x** según la Tabla 1.2 [16].

$\boldsymbol{d_{ml}}$	
1	86.0746
2	63.6070
3	44.1126
4	21.6450
5	48.3522
6	24.0478
7	24.3086
8	0.0042
9	127.6954
10	87.3094
11	83.8966
12	43.5106
13	89.9730
14	47.7502
15	64.0926
16	21.8698

1.3.5. ANÁLISIS DE LA COMPLEJIDAD

El siguiente análisis de complejidad se realiza en términos de operaciones de punto flotante complejas (flops) $\mathcal{F}$. Por simplicidad, se considera cada suma compleja escalar como un flop y cada multiplicación compleja escalar como tres flops. Para los cálculos de

complejidad se utiliza la Tabla de operaciones 1.4. El número de cálculos depende del número de antenas M [18].

Tabla 1.4. Número de flops para algunas operaciones vector y matriz [18].

Operación	Número de flops $\mathcal{F}$
$\boldsymbol{A} + \boldsymbol{B}$	nm
$\boldsymbol{A} \cdot \boldsymbol{C}$	$4nmo + no$
$\boldsymbol{A} \cdot \boldsymbol{A}^H$	$\left(\frac{1}{2}n^2 + \frac{1}{2}n\right)(4m - 1)$
$\boldsymbol{D}^{-1}$	$4m^3 + m^2 + m$
$\boldsymbol{A} \cdot \boldsymbol{b}$	$4nm - n$
$\boldsymbol{a} + \boldsymbol{b}$	m
$\boldsymbol{a}^H \cdot \boldsymbol{b}$	$4m - 1$
$\lVert \boldsymbol{a} \rVert^2$	$\frac{3}{2}m - \frac{1}{2}$
$\boldsymbol{Q}(\boldsymbol{a})$	m

- **Complejidad del detector ML**

  De acuerdo con la Ecuación 1.13, el número de flops necesarios para el cálculo del vector ML se tiene en la Ecuación 1.14 [18]:

$$\mathcal{F}_{ML} = |C|^M (4\,M^2 + 1.5\,M + 0.5) \tag{1.14}$$

- **Complejidad del detector MMSE**

  El ecualizador MMSE, dado por la Ecuación 1.11 y según la Tabla 1.4, presenta el siguiente esfuerzo computacional mostrado en la Ecuación 1.15:

Tabla 1.5. Número de flops para operaciones vector y matriz del algoritmo MMSE.

Operación	Número de Flops
$\boldsymbol{A} \cdot \boldsymbol{A}^H$	$\left(\frac{1}{2}n^2 + \frac{1}{2}n\right)(4n - 1)$
$\sigma_z^2 \cdot \boldsymbol{I}$	$\frac{n^2}{2}$
$\boldsymbol{A} + \boldsymbol{B}$	n^2
$\boldsymbol{A}^{-1}$	$4n^3 + n^2 + n$
$\boldsymbol{A} \cdot \boldsymbol{C}$	$4n^3 + n^2$

Cabe recalcar que se asume que una multiplicación escalar es igual a medio flop.

$$\mathcal{F}_{MMSE} = \left(\frac{1}{2}n^2 + \frac{1}{2}n\right)(4n-1) + \frac{n^2}{2} + n^2 + 4n^3 + n^2 + n + 4n^3 \ + \ n^2$$

$$\mathcal{F}_{MMSE} = 10\ M^3 + 5\ \ M^2 + 0.5\ M \tag{1.15}$$

- **Complejidad del detector ZF**

  De acuerdo con la Ecuación 1.9 y la Tabla 1.4, el número de flops necesarios utilizando el detector ZF se presenta en el Ecuación 1.16:

Tabla 1.6. Número de flops para operaciones vector y matriz del algoritmo ZF.

Operación	Número de Flops
$\boldsymbol{A} \cdot \boldsymbol{A}^H$	$\left(\frac{1}{2}n^2 + \frac{1}{2}n\right)(4n-1)$
$\boldsymbol{A}^{-1}$	$4n^3 + n^2 + n$
$\boldsymbol{A} \ \cdot \ \boldsymbol{C}$	$4n^3 \ + \ n^2$

$$\mathcal{F}_{ZF} = \left(\frac{1}{2}n^2 + \frac{1}{2}n\right)(4n-1) + 4n^3 + n^2 + n + 4n^3 \ + \ n^2$$

$$\mathcal{F}_{ZF} = \ 10M^3 + 3.5M^2 + 0.5M \tag{1.16}$$

En la Tabla 1.4 se tienen algunos cálculos de complejidad de operaciones de matriz útiles. Las operaciones de la matriz se pueden describir en sumas y multiplicaciones escalares básicas. Para matrices complejas $\boldsymbol{A} \in \mathbb{C}^{n\times m}, \boldsymbol{B} \in \mathbb{C}^{n\times m}, \boldsymbol{C} \in \mathbb{C}^{m\times o},\ \boldsymbol{D} \in \mathbb{C}^{m\times m}$ y vectores complejos $\boldsymbol{a} \in \mathbb{C}^{m\times 1}, \boldsymbol{b} \in \mathbb{C}^{m\times 1}$, el número de flops para algunos vectores útiles, y las operaciones de matriz, se dan en esta Tabla 1.4. Q (.) denota el operador de cuantización [18].

Para apreciar la complejidad computacional de los detectores, en la Tabla 1.7, se presenta un ejemplo de aplicación de la complejidad computacional de los algoritmos de detección, para los diversos números de antenas M y modulaciones C.

Tabla 1.7. Esfuerzo computacional de algoritmos de detección.

	4QAM		**16QAM**		**64QAM**	
	M = 2	M = 4	M = 2	M=4	M=2	M=4
DETECTOR ZF	95	698	95	698	95	698
DETECTOR MMSE	105	734	105	734	105	734
DETECTOR MLD	312	18048	4992	4620288	79872	1182793728

Al examinar la Tabla 1.7 se puede apreciar fácilmente que la relación del ZF y MMSE respecto al MLD presentan menos complejidad o esfuerzo, esto es debido a que los algoritmos ZF y MMSE son lineales. Se observa que el vector ZF es el de menor complejidad computacional respecto al vector MMSE, esto debido a que el detector MMSE presenta ciertas variaciones al ZF como es la inclusión de la varianza del ruido en su algoritmo, lo que se ve reflejado en su complejidad. También es importante destacar que para el vector MLD la complejidad crece exponencialmente con las diferentes modulaciones y antenas.

2. METODOLOGÍA

En el proyecto se utilizó algunos tipos o métodos de investigación como el método deductivo ya que se ha puesto énfasis en la teoría, modelos, explicación y abstracción de la información requerida. Según la naturaleza de los objetivos se ha utilizado la investigación explicativa ya que se ha descrito en la parte teórica de este proyecto los componentes principales, se ha acercado al problema y se han buscado las causas. También se utiliza la investigación experimental ya que se ha logrado la simulación requerida, controlando así el fenómeno a estudiar. La metodología empleada según el tipo de datos es la cuantitativa ya que el objeto del estudio busca la máxima objetividad y encontrar generalidades para cada sujeto (detectores) [20].

En el presente capítulo se pormenoriza el programa desarrollado en Matlab para la simulación de las técnicas de detección, el cual consta principalmente de: sistema MIMO – OFDM, canal de Rayleigh, detectores, gráficas y algunos otros que ayudan a la obtención de los resultados. Las gráficas resultantes de las simulaciones y tiempos de simulación están presentes en el capítulo 3 con su respectivo análisis.

El programa principal implementado en Matlab presenta cinco procesos que se repiten finitas veces, iniciando con el proceso de escoger el arreglo de antenas (2x2, 2x4, 4x4). El siguiente bloque es la selección del detector, estando disponibles los tres detectores que se planteó analizar ZF, MMSE y MLD. Posteriormente se ingresa al proceso de escoger modulación ya sea QPSK, 16QAM y 64QAM. Todas las combinaciones posibles de arreglo de antenas, detector y modulación se aprecian en Tabla 2.1. Terminado el proceso anterior se asigna un Eb/No y una vez establecidos estos parámetros se procede al ingreso de la función mimoOfdmSingleTest3, en donde se realiza la transmisión, recepción y detección de los datos, al igual que el cálculo de errores y el cálculo de los tiempos de simulación dependiendo del número de iteraciones establecido. Con los resultados obtenidos se ingresa al bloque de la función graficarDetectores, aquí se grafica los resultados obtenidos para cada arreglo de antenas, detector y modulación simulado, así como la gráfica de los tiempos de simulación para cada detector y arreglo de antena.

Tabla 2.1. Combinaciones posibles de las variables en el sistema MIMO – OFDM.

DETECTOR	CONFIGURACIÓN DE ANTENAS	MODULACIÓN
ZF	2 x 2	QPSK
		16QAM
		64QAM
	2 x 4	QPSK
		16QAM
		64QAM
	4 x 4	QPSK
		16QAM
		64QAM
MMSE	2 x 2	QPSK
		16QAM
		64QAM
	2 x 4	QPSK
		16QAM
		64QAM
	4 x 4	QPSK
		16QAM
		64QAM
ML	2 x 2	QPSK
		16QAM
		64QAM
	2 x 4	QPSK
		16QAM
		64QAM
	4 x 4	QPSK
		16QAM
		64QAM

2.1. IMPLEMENTACIÓN EN MATLAB

2.1.1. PROGRAMA PRINCIPAL – DETECTORES2

El programa principal toma el nombre de Detectores2.m aquí se realiza la selección de datos como número de antenas, modulación, detectores, Eb/No, los cuales ingresarán a una función donde se realizará la transmisión, recepción y detección. En la Figura 2.1 se presenta el diagrama de flujo a detalle del programa principal, cada bloque se explica a continuación.

El Segmento de código 2.1 muestra las variables y valores principales para todas las posibles combinaciones utilizadas a lo largo de toda la simulación, se especifica los tipos de detectores, modulaciones, antenas transmisoras, antenas receptoras, número de iteraciones a realizar y el Eb/No.

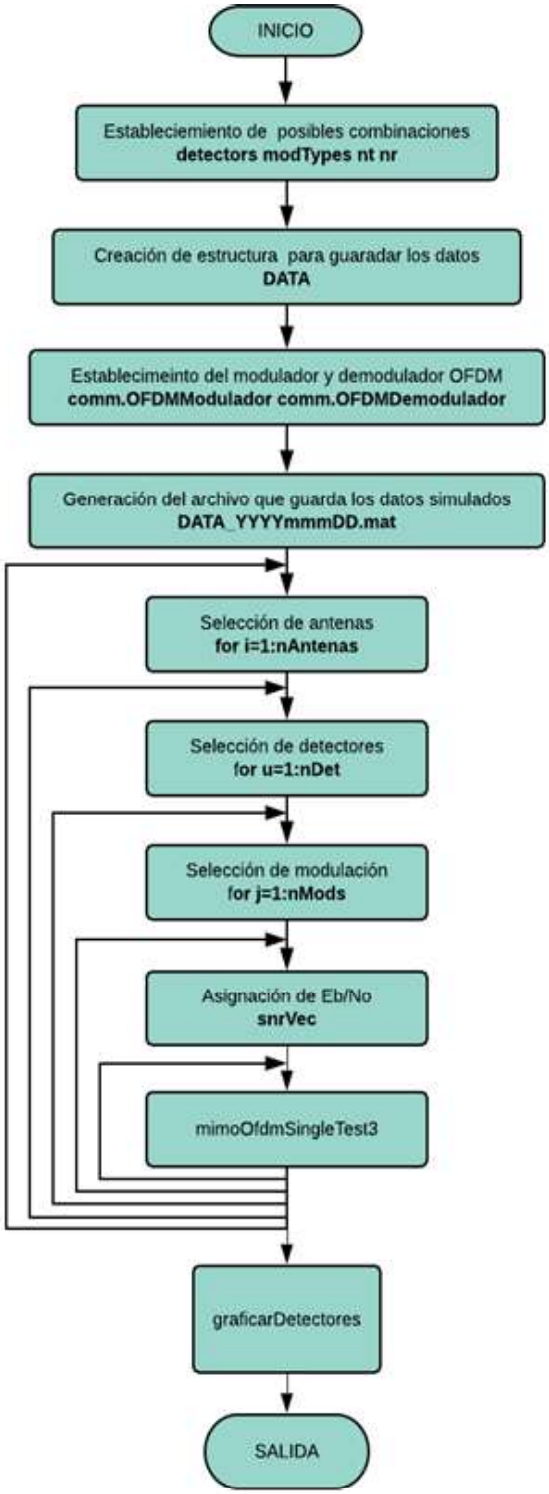

Figura 2.1 Diagrama de flujo del programa principal Detectores2.

```
%% POSIBLES COMBINACIONES
detectors = { 'ZF','MMSE','ML'}; % algoritmos detectores a usar
modTypes = { 'QPSK', 'QAM16','QAM64' }; % modulaciones a realizar
nt = [ 2 2 4];  % antenas transmisoras
nr = [ 2 4 4];  % antenas receptoras
nIter1 = 10000; % cantidad de iteraciones
nIter2 = 100;   % cantidad de iteraciones ( NT = 4, ML, 64QAM )
EbNoVec = 0:5:20; % vector de ruido
```

Segmento de código 2.1 Variables y valores de las posibles combinaciones.

Siguiendo con el código del programa en el Segmento de código 2.2 se guardan los datos de las variables anteriores en una estructura de nombre *DATA*.

```
%ESTRUCTURA DE DATOS
DATA = struct();
DATA.detectors = detectors;
DATA.modTypes = modTypes;
DATA.nt = nt;
DATA.nr = nr;
DATA.nIter = nIter1;
DATA.EbNoVec = EbNoVec;
```

Segmento de código 2.2 Estructura DATA.

Para la realización de la modulación y demodulación en OFDM de los datos, se define tanto el modulador como el demodulador con los comandos *comm.OFDMModulator y comm.OFDMDemulator* respectivamente. Se tiene en cuenta los parámetros de acuerdo con el estándar IEEE 802.11a el cual funciona con conexiones de hasta 54 Mbps, opera en la banda de 5 GHz y utiliza OFDM en la capa física. En la Tabla 2.2 se observan los parámetros básicos para el estándar 802.11a los cuales serán tomados en cuenta para la simulación [21].

En el Segmento de código 2.3 se exponen los parámetros de número de portadoras, prefijo cíclico, se establece el modulador y demodulador OFDM los cuales quedan definidos en las variables *ofdmMod* y *ofdmDemod,* respectivamente. En la estructura *ofdmDism* se guardan el tamaño de entrada de los datos y el tamaño de salida (*DataInputSize* y *OutputSize*).

Los comandos *comm.OFDMModulator y comm.OFDMDemulator,* presentan algunos argumentos o propiedades, los cuales se explican en la Tabla 2.3.

Tabla 2.2. Parámetros para el estándar IEEE 802.11a [21].

Modulaciones	QPSK	16QAM	64QAM
Normalización	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{10}}$	$\frac{1}{\sqrt{42}}$
Subportadoras totales	64		
Prefijo cíclico	0.8 µs		
Intervalo de guarda	1.6 µs		
Frecuencia de muestreo	20 MHz		

```
%PARÁMETROS PARA EL OFDM
numSC = 64;    % Number of OFDM subcarriers
cpLen = 16;    % OFDM cyclic prefix length

ofdmMod = comm.OFDMModulator('FFTLength',numSC,...
    'CyclicPrefixLength',cpLen,'NumGuardBandCarriers',[0;0]);
ofdmDemod = comm.OFDMDemodulator('FFTLength',numSC,...
    'CyclicPrefixLength',cpLen,'NumGuardBandCarriers',[0;0]);

ofdmDims = info(ofdmMod);          %tamaño de los datos
numDC = ofdmDims.DataInputSize(1);%tamaño de datos de entrada
frameSize = [numDC 1];
```

Segmento de código 2.3 Parámetros para OFDM.

Tabla 2.3. Parámetros para las funciones de modulación y demodulación OFDM.

ARGUMENTOS	**DESCRIPCIÓN**
FFTLength	Longitud de la FFT o número de subportadoras utilizadas en el proceso de modulación
CyclicPrefixLength	Especifica la longitud del prefijo cíclico, es ¼ del número de subportadoras. Si es escalar la longitud del CP es la misma para todos los símbolos a través de todas las antenas.
NumGuardBandCarriers	El número de subportadoras asignadas a las bandas de guardia de izquierda y derecha. Se establece en un vector columna 2 por 1, por ejemplo: [0;0].

El Segmento de código 2.4 muestra cómo se asigna el nombre del archivo que contiene todos los datos necesarios para mostrar los resultados. El archivo generado toma por nombre *DATA* más la fecha actual con el formato Año/Mes/Día y la extensión .mat, si se siguen realizando pruebas el mismo día al final del nombre se colocará un número entre paréntesis de acuerdo a las veces que se realice las simulación, como se observa en la Figura 2.2.

```
%% NOMBRE DEL ARCHIVO
date = datestr( datetime("today"),"YYYYmmmDD" );
name = "DATA";
extension = ".mat";
index = 0;
while true
    fullName = strcat( name, '_', date );
    if index > 0
     fullName = strcat( fullName, '(',num2str(index), ')' );
    end
    fullName = strcat( fullName, extension );

    if isfile(fullName)
        index = index + 1;
    else
        break;
    end
end
```

Segmento de código 2.4 Nombre del archivo.

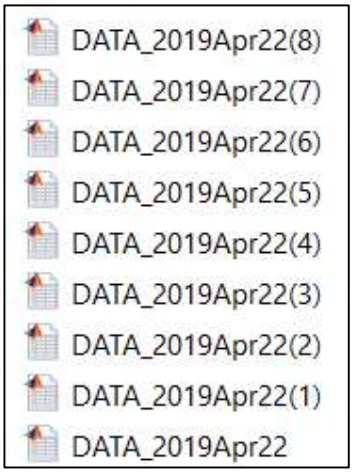

Figura 2.2 Asignación de nombres a archivos.

El lazo sobre las posibles combinaciones está compuesto por varios lazos *for*, uno para las antenas, uno para detectores y uno para modulaciones. Iniciando con el lazo *for* para antenas, se tiene el Segmento de código 2.5, donde antes de iniciar el lazo se ha colocado un temporizador el cual informará el tiempo de duración de todo el programa (*timerVal*). El lazo *for* inicia en 1 hasta *nAntenas* (variable con la longitud de las antenas transmisoras). *NT* y *NR* son los valores que van a ir variando. Posteriormente se establece el despliegue en pantalla de la antena actual en ejecución con la variable *antena,* como en la Figura 2.3.

```
%% LAZO SOBRE LAS POSIBLES COMBINACIONES
timerVal = tic; %inicio del temporizador para el programa
for i = 1:nAntenas
    nIter = nIter1;
    NT = nt(i);
    NR = nr(i);
    antena = sprintf( 'ANT_%ux%u', NT, NR );
    disp(antena)
```

Segmento de código 2.5 Combinaciones de antenas.

```
ANT_2x2
    ZF
        QPSK
        QAM16
        QAM64
    MMSE
        QPSK
        QAM16
        QAM64
    ML
        QPSK
        QAM16
        QAM64
```

Figura 2.3 Despliegue en pantalla de antenas, detectores y modulaciones.

En el Segmento de código 2.6 se presenta el lazo *for* para los detectores, el cual inicia en 1 hasta *nDet* (variable con el número de detectores), aquí se va a ir desplegando en pantalla el detector en ejecución como en la Figura 2.3. En este segmento la variable *det,* contiene el detector asignado, el cual ingresa en un *switch* o interruptor, aquí se evalúa dicho detector y de acuerdo con este parámetro se asigna la función correspondiente, ya sea ZF_detector, MMSE_detector y/o ML_detector, cada caso hace la llamada a su respectiva función. Cada función será explicada en la sección 2.1.3.

```
for u = 1:nDet
    det = detectors{u};
    fprintf('\t%s\n', det );
    switch det
        case 'ZF'
            detector = @ZF_detector;
        case 'MMSE'
            detector = @MMSE_detector;
        case 'ML'
            detector = @ML_detector5;
    end
```

Segmento de código 2.6 Lazo de detectores.

```
for j = 1:nMods
    modulador = modTypes{j};
    fprintf('\t\t%s\n', modulador );
    switch modulador
        case 'QPSK'
            M = 4;
            modFun = comm.QPSKModulator;
            demFun = comm.QPSKDemodulator;
            energy = 1;
        case 'QAM16'
            M = 16;
            modFun = @(x) qammod(x,M,'UnitAveragePower',true);
            demFun = @(y) qamdemod(y,M,'UnitAveragePower',true);
            energy = 1;
        case 'QAM64'
            M = 64;
            modFun = @(x) qammod(x,M,'UnitAveragePower', true);
            demFun = @(y) qamdemod(y,M,'UnitAveragePower',true);
            energy = 1;
            if NT == 4 && strcmpi( det, 'ML' )
                nIter = nIter2;
            end
    end
```

Segmento de código 2.7 Lazo de modulaciones.

La selección de las modulaciones se realiza en el tercer y último *for*, aquí también se despliega en pantalla la modulación en ejecución como en la Figura 2.3. El *for* inicia en 1 y va hasta nMods (variable con el número de modulaciones), para luego ingresar a un interruptor en el que cada caso contiene una modulación específica ya sea QPSK, 16QAM o 64QAM. Dependiendo de la modulación actual asignada, se ejecutará el caso correspondiente. En cada caso se establece el orden de la modulación *M*, la modulación y demodulación correspondiente, así como la energía la cual será utilizada para la simulación del algoritmo de detección MMSE. En el caso de 64QAM se tiene un *if – end* que permite cambiar el número de iteraciones cuando el número de antenas transmisoras sea igual a 4, debido a que para este número de antenas el proceso de detección con el algoritmo ML requiere un costo computacional demasiado alto. Este cambio permite agilizar un poco el proceso de detección, como se aprecia en el Segmento de código 2.7. Para las modulaciones se aplican diferentes funciones como:

- QPSK

  modFun = comm.QPSKModulator

  demFun = comm.QPSKDemodulator

En este caso *modFun* y *demFun* crean un modulador y demodulador System object[1], este objeto de sistema modula la señal de entrada utilizando el método de codificación por desplazamiento de fase en cuadratura (QPSK, *Quadrature Phase Shift Keying*). Una vez ejecutada la modulación se obtiene la señal de constelación con los datos normalizados, como se observa en la Figura 2.4 [22].

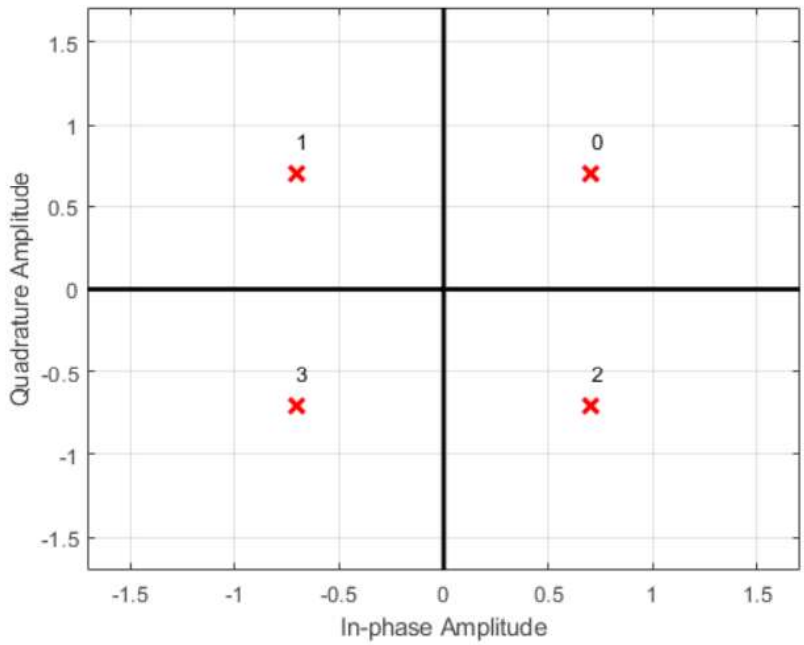

Figura 2.4 Constelación de la modulación QPSK [22].

- 16QAM

modFun = @(x) qammod(x,M,'UnitAveragePower', true)

demFun = @(y) qamdemod(y,M,'UnitAveragePower', true)

Para la modulación 16QAM *modFun* y *demFun* contienen la modulación y demodulación de amplitud en cuadratura para la señal de entrada en el caso de la modulación y para la señal de salida en el caso de la demodulación, según el orden de modulación que es M=16. El argumento de entrada *'UnitAveragePower',* es la bandera de unidad de potencia media, esta necesita un escalar lógico en este caso *true* lo que permite a la función escalar la constelación a una potencia media de 1 Watt referenciada a 1 Ohm, es decir normaliza la constelación, como en la Figura 2.5 [22].

[1] System object: Los objetos del sistema están diseñados para implementar y simular sistemas dinámicos con entradas que cambian con el tiempo como sistemas de procesamiento de señales, comunicaciones y controles dinámicos [22].

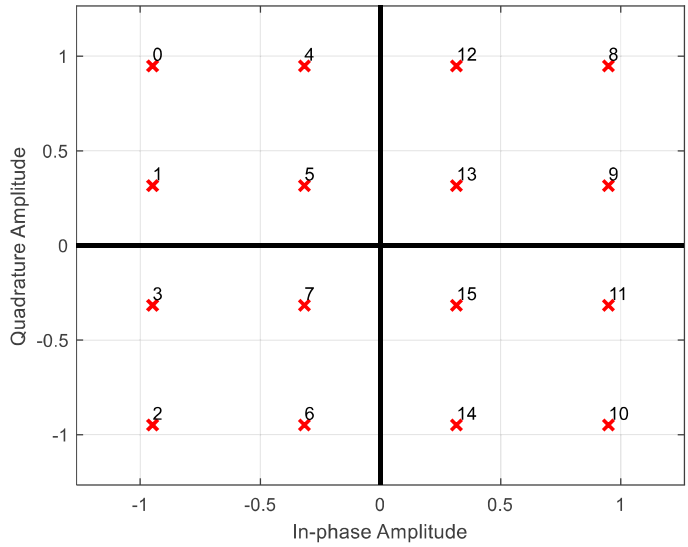

Figura 2.5 Constelación de la modulación 16QAM.

- 64QAM

```
modFun = @(x) qammod(x,M,'UnitAveragePower', true)
demFun = @(y) qamdemod(y,M,'UnitAveragePower', true)
```

La modulación y demodulación 64QAM sigue el mismo modelo de 16QAM, aquí cambia el orden de modulación y demodulación, es decir M=64. En la Figura 2.6 se observa la constelación normalizada para 64QAM, la normalización asegura la misma potencia promedio en cada una de las modulaciones.

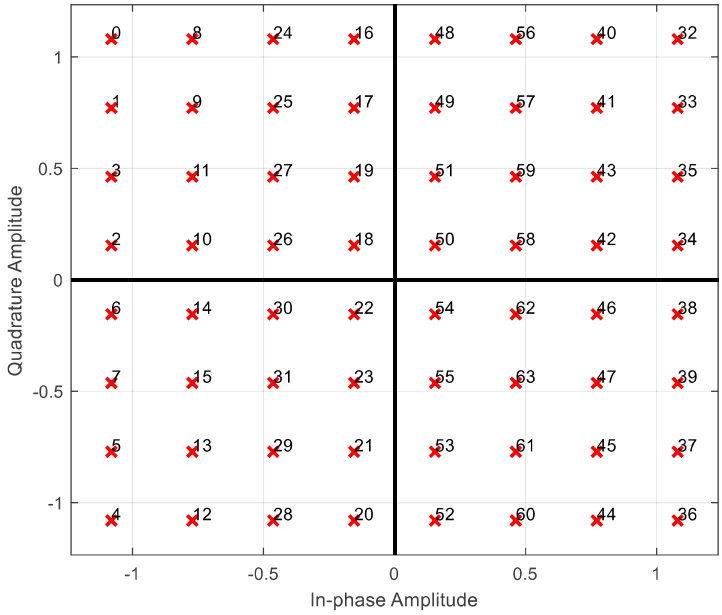

Figura 2.6 Constelación de la modulación 64QAM.

En el último *for* también se realizan algunos procesos como la obtención de los bits por símbolo, *k,* utilizados en la obtención del vector snrVec. Se realiza la creación de otra estructura de datos *model* la cual se utiliza en la función *mimoOfdmSingleTest3*. Se crea también el *snrVec* el cual es un vector que se expresa en dB y nos indica la relación señal a ruido, todo esto se presenta en el Segmento de código 2.8. En esta sección también se guardan datos en la estructura DATA, que se presenta en el Segmento de código 2.9.

La función *mimoOfdmSingleTest3* realiza los procesos de transmisión, recepción y detección los cuales se explican en el siguiente punto.

```
k = log2(M);            % Bits/symbol
model = struct();
model.M = M;
model.NR = NR;
model.NT = NT;
model.energy = energy;
model.detector = detector;
model.modFun = modFun;
model.demFun = demFun;
model.ofdmMod = ofdmMod;
model.ofdmDemod = ofdmDemod;
model.ofdmDims = ofdmDims;

snrVec = EbNoVec(:) + 10*log10(k) + 10*log10(numDC/numSC);
%Llamado a la función mimoOfdmSingleTest3
[ BER, T ] = mimoOfdmSingleTest3( model, snrVec, nIter );
```

Segmento de código 2.8 Valores necesarios para la función mimoOfdmSingleTest3.

```
DATA.(antena).(det).(modulador).model = model;
DATA.(antena).(det).(modulador).BER = BER;
DATA.(antena).(det).(modulador).T = T;
```

Segmento de código 2.9 Estructura de datos DATA.

```
toc(timerVal)
%% Guardado de Datos
save( fullName, 'DATA' )
%% Grafico de Datos
graficarDetectores( DATA )
clearvars -except DATA
```

Segmento de código 2.10 Proceso final del programa Detectores2.

Una vez terminado el proceso se finaliza el temporizador con *toc*, posteriormente se guardan todos los datos de la estructura *DATA*, y luego se llama a la función de graficarDetectores, esta función se explica en la sección 2.1.4. Finalmente, se limpian las variables utilizadas con el comando *clearvars* exceptuando DATA. Todo este proceso se observa en el Segmento de código 2.10.

2.1.2 FUNCIÓN MIMOOFDMSINGLETEST3

En esta sección se explica la función *mimoOfdmSingleTest3.* La llamada a esta función se encuentra en el Segmento de código 2.8, en la Tabla 2.4 se tienen los argumentos de entrada y salida que requiere la función y en la Figura 2.7 se presenta el diagrama de flujo de esta. La función comienza con la inicialización de la barra de progreso, como se observa en el Segmento de código 2.11. La barra de progreso ayuda al usuario a saber el avance de cada modulación, para cada una de las detecciones y arreglo de antenas, como en la Figura 2.8. Los comandos *parallel.pool.DataQueue* y *afterEach* especifica la función que se ejecutará cada vez que la cola reciba datos. En este ejemplo se llama a una subfunción que actualiza la barra de espera [22].

Tabla 2.4. Parámetros de la función *mimoOfdmSingleTest3.*

Parámetros	Tipo de Datos	Descripción
model	Entrada	Estructura de datos con modulaciones, demodulaciones, antenas, detectores, energía y orden de modulación
snrVec	Entrada	Vector de señal a ruido
nIter	Entrada	Número de iteraciones
BER	salida	Bits errados
T	salida	Tiempo de simulación de la detección

```
%% Inicializaciones de la barra de progreso
    D = parallel.pool.DataQueue;
    h = waitbar(0, 'Please wait ...');
    afterEach( D, @nUpdateWaitbar );
    p = 1;
    N = length(snrVec) * nIter;
```

Segmento de código 2.11 Inicialización de la barra de estado.

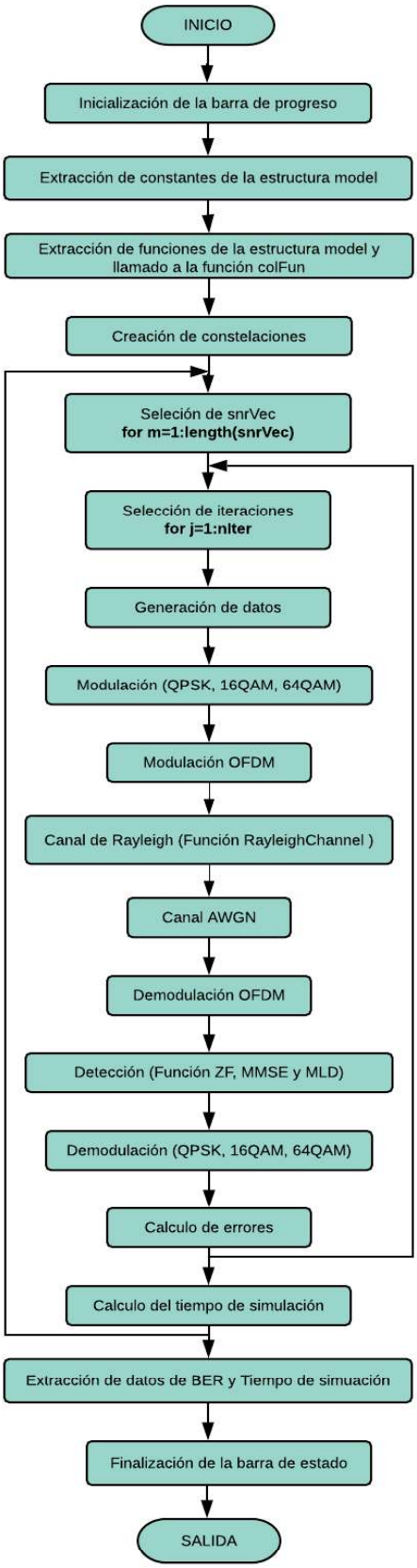

Figura 2.7 Diagrama de flujo de la función mimoOfdmSingleTest3.

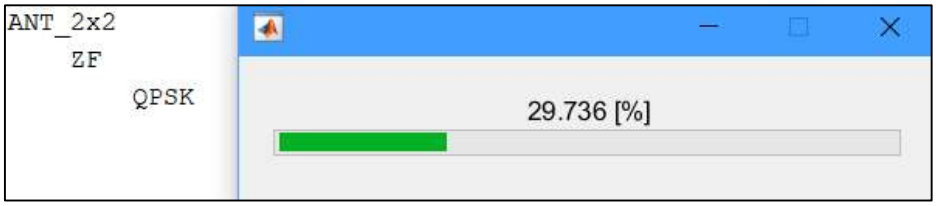

Figura 2.8 Barra de progreso para las distintas combinaciones.

En el Segmento de código 2.12, se extrae las constantes de la estructura de datos *model*, como *M* para obtener los bits/símbolo, NT, NR y datos de entrada y salida del modulador OFDM. Posteriormente en el Segmento de código 2.13 se extrae las funciones de la estructura como energía y detectores, todos estos datos extraídos se utilizan directa e indirectamente para los procesos de generación de datos a transmitir, modulaciones, canal inalámbrico, demodulaciones, detecciones y cálculo de errores

Las funciones que también se extraen, en funciones anónimas[2], son las de modulación, demodulación, modulación OFDM y demodulación OFDM, las cuales son argumentos de entrada de la función *colFun*.

```
%% CONSTANTES
M = model.M;
k = log2(M);
NT = model.NT;
NR = model.NR;
inputSize = model.ofdmDims.DataInputSize(1);
outputSize = model.ofdmDims.OutputSize(1);
```

Segmento de código 2.12 Extracción de constantes de la estructura model.

```
%% FUNCIONES
    energy   = model.energy;
    detector = model.detector;
```

Segmento de código 2.13 Extracción de funciones de la estructura model.

```
modFun    = @(x) colFun( model.modFun,    [inputSize, NT],  x );
ofdmMod   = @(x) colFun( model.ofdmMod,   [outputSize, NT], x );
ofdmDemod = @(x) colFun( model.ofdmDemod, [inputSize, NR],  x );
demFun    = @(x) colFun( model.demFun,    [inputSize, NT],  x );
```

Segmento de código 2.14 Llamada a la función *colFun* para las modulaciones y demodulaciones.

[2] Función Anónima: Una función anónima es una función que no se almacena en un archivo de programa, sino que está asociada a una variable cuyo tipo de datos es function_handle. Las funciones anónimas pueden aceptar entradas y salidas de retorno, tal como lo hacen las funciones estándar. Sin embargo, solo pueden contener una sola instrucción ejecutable [22].

```
function out = colFun( fun, matrixSize, data )
    out = zeros( matrixSize );
    for u = 1:matrixSize(2)
        out(:,u) = fun( data(:,u) );
    end
end
```

Segmento de código 2.15 Función *colFun*.

En el Segmento de código 2.14 se llama a la función *colFun* la que está compuesta de varios parámetros de entrada y un parámetro de salida. En la tabla 2.5 se explica cada parámetro que interviene en esta función, *colFun* ejecuta una función para cada columna, repite una función por columnas. En el Segmento de código 2.15 se presenta el código de la función *colFun.*

Tabla 2.5. Parámetros de la función *colFun.*

Parámetros	Tipo de Datos	Descripción
fun	Entrada	Función extraída de modulaciones o demodulaciones. model.modFun model.ofdmMod model.ofdmDemod model.demFun
matrixSize	Entrada	Matriz de tamaño de datos de entrada y salida por número de antenas transmisoras y receptoras. [inputSize, NT] Datos antes de la modulación OFDM [outputSize, NT] Datos después de la modulación OFDM [inputSize, NR] Datos antes de la demodulación OFDM [inputSize, NT] Datos después de la demodulación OFDM
data	Entrada	Variable de datos de la función
out	salida	Modulaciones y demodulaciones modFun ofdmMod ofdmDemod demFun

```
A = model.modFun( (0:M-1)' ); %constelación
ratio = zeros( length(snrVec), 1 );
T = 0;
```

Segmento de código 2.16 Creación del vector de datos de las constelaciones.

Siguiendo el diagrama de flujo de la figura 2.7, en el cuarto proceso se obtienen los valores de los datos de las diferentes constelaciones. En el Segmento de código 2.16 se inicializan algunas variables y se crea un vector de datos que contiene cada uno de los símbolos posibles dependiendo del orden de modulación, por ejemplo:

- M = 4 (Figura 2.9)

```
A: 4x1 complex double =

   0.7071 + 0.7071i
  -0.7071 + 0.7071i
   0.7071 - 0.7071i
  -0.7071 - 0.7071i
```

Figura 2.9 Vector de datos de la constelación con M =4.

- M = 16 (Figura 2.10)

```
A: 16x1 complex double =

  -0.9487 + 0.9487i
  -0.9487 + 0.3162i
  -0.9487 - 0.9487i
  -0.9487 - 0.3162i
  -0.3162 + 0.9487i
  -0.3162 + 0.3162i
  -0.3162 - 0.9487i
  -0.3162 - 0.3162i
   0.9487 + 0.9487i
   0.9487 + 0.3162i
   0.9487 - 0.9487i
   0.9487 - 0.3162i
   0.3162 + 0.9487i
   0.3162 + 0.3162i
   0.3162 - 0.9487i
   0.3162 - 0.3162i
```

Figura 2.10 Vector de datos de la constelación con M =16.

- M = 64 (Figura 2.11)

```
A: 64x1 complex double =

  -1.0801 + 1.0801i
  -1.0801 + 0.7715i
  -1.0801 + 0.1543i
  -1.0801 + 0.4629i
  -1.0801 - 1.0801i
  -1.0801 - 0.7715i
  -1.0801 - 0.1543i
  -1.0801 - 0.4629i
  -0.7715 + 1.0801i
  -0.7715 + 0.7715i
  -0.7715 + 0.1543i
  -0.7715 + 0.4629i
  -0.7715 - 1.0801i
  -0.7715 - 0.7715i
  -0.7715 - 0.1543i
  -0.7715 - 0.4629i
  -0.1543 + 1.0801i
  -0.1543 + 0.7715i
  -0.1543 + 0.1543i
  -0.1543 + 0.4629i
  -0.1543 - 1.0801i
```

Figura 2.11 Parte del vector de datos de la constelación con M =64.

Una vez obtenidos todos los datos necesarios, se ingresa al primer lazo de la función, el cual va desde 1 hasta la longitud de *snrVec*. Dentro de este lazo se inicializan variables para el ruido, cálculo de errores y tiempos de simulación (Segmento de código 2.17). También se presenta el segundo lazo de la función, en este lazo se realiza el proceso de transmisión, recepción y detección de la señal. El lazo *for* inicia en 1 y va hasta *nIter*, el cual fue declarado en el programa principal y representa el número de iteraciones a realizar.

```
for m = 1:length(snrVec)
    snr = snrVec(m);
    error = 0;
    tt = zeros( nIter, 1 );
    for j = 1:nIter % Iteraciones
```

Segmento de código 2.17 Primer y segundo lazo *for* e inicialización de variables.

- ## Sistema de transmisión

Los cálculos previos a este punto de la simulación permiten obtener los datos necesarios para realizar todo el proceso descrito en el alcance del proyecto. La Figura 2.12 explica el proceso de transmisión a realizar para el sistema MIMO – OFDM que se ha realizado, siguiendo el orden de la función especificada en la Figura 2.7.

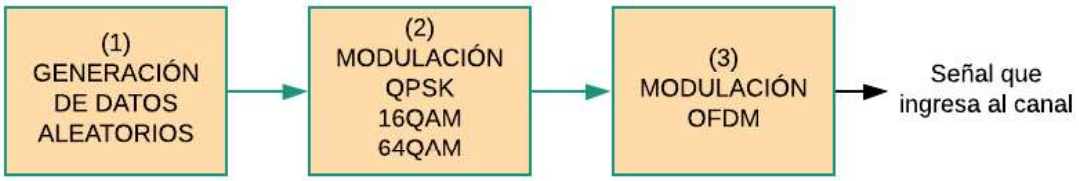

Figura 2.12 Diagrama de bloques para el sistema de transmisión.

1. Generación de datos aleatorios: Segmento de código 2.18, la generación de datos se da con el comando *randi,* el cual genera datos entre 0 y M-1 de tamaño *inputSize* por *NT*. Estos datos generados se guardan en la variable *dataIn.*

```
% GENERACION DE DATOS
dataIn = randi( [ 0, M-1 ], inputSize, NT );
```

Segmento de código 2.18 Generación de datos aleatorios.

2. Modulación QPSK – 16QAM – 64QAM: Segmento de código 2.19, la modulación a realizar depende de la función actual en la variable *modFun*, este valor es asignado en el programa principal. Hay que tener en cuenta que los datos generados aleatoriamente, *dataIn*, van a ser modulados y normalizados de acuerdo con el orden de modulación que se encuentre. La variable *modSig* contiene los datos complejos modulados y normalizados.

```
% MODULACION (QPSK,QAM)
modSig = modFun( dataIn );
```

Segmento de código 2.19 Modulaciones QPSK, 16QAM Y 64QAM.

3. Modulación OFDM: Segmento de código 2.20, aquí se realiza la modulación OFDM de los datos obtenidos por las modulaciones QPSK, 16QAM y 64QAM. La función *ofdmMod* realiza esta modulación, teniendo en cuenta el prefijo cíclico, intervalo de guarda y el número de subportadoras, como se explicó anteriormente. La variable *txSig* contiene los datos modulados en OFDM.

```
% MODULACION OFDM
txSig = ofdmMod( modSig );
```

Segmento de código 2.20 Modulación OFDM.

- ## Canal inalámbrico

Canal de Rayleigh y multitrayecto

```
%CANAL DE RAYLEIG Y MULTITRAYECTO
[H,h_c] = RayleighChannel( NT, NR, inputSize );%respt del canal
```

Segmento de código 2.21 Llamada a la función *RayleighChannel*.

Tabla 2.6. Parámetros de la función *RayleighChannel*.

Parámetros	Tipo de Datos	Descripción
NT	Entrada	Número de antenas transmisoras
NR	Entrada	Número de antenas receptoras
Nsubt	Entrada	Número de datos transmitidos
H	salida	Matriz de canal
h	salida	Respuesta impulsiva del canal

En el Segmento de código 2.21 se hace el llamado a la función *RayleighChannel* la cual está compuesta de algunos parámetros de entrada y salida, en la Tabla 2.6 se explica cada parámetro que interviene en esta función. Según las características definidas en el alcance del proyecto, se estableció un canal de Rayleigh y propagación multitrayectoria, para el desarrollo de estos requerimientos se tiene que, una característica del canal multitrayecto viene especificada por un PDP (*Power Delay Profile*), el cual define ciertos parámetros como taps, retardo relativo y potencia promedio. En la Tabla 2.7 se observa el PDP utilizado para esta simulación. Los retardos normalizados se obtienen de dividir el retardo relativo para 50 ns que sería la frecuencia de muestreo para el estándar utilizado (802.11a), estos retardos normalizados no deben exceder el intervalo de guarda que para el presente caso sería 16 [2] [21].

En la Figura 2.13 se presenta el diagrama de flujo para la función *RayleighChannel.* Los procesos que esta función cumple, se exponen en los siguientes segmentos de código. En el Segmento de código 2.22 se establecen los parámetros de creación del PDP para el canal de Rayleigh, como la potencia y retardos normalizados asignados en las variables *pm_n* y *retardos* respectivamente. Los valores de potencia y retardos se pueden variar para otros tipos de canales, no son valores fijos.

Tabla 2.7. Valores de PDP para simulación [2].

Tap	Retardo relativo (ns)	Retardos normalizados	Potencia promedio (dB)
1	0	0	0
2	110	2	-9.7
3	190	4	-19.2
4	410	8	-22.8

```
pot_rayos = [0, -3.6, -7.2, -10.8];%Potencia de rayos en dB
potencia = 10 .^ ( pot_rayos/10 ); %Potencia de rayos en veces
pm = potencia / sum( potencia );   %Potencia promedio
pm_n = sqrt( pm );                 %Potencia promedio normal
retardos = [ 0, 2, 4, 6 ];         %Retardo para cada rayo
N_rayos = length( retardos );      %Número de rayos por señal
```

Segmento de código 2.22 Parámetros de PDP para el canal de Rayleigh.

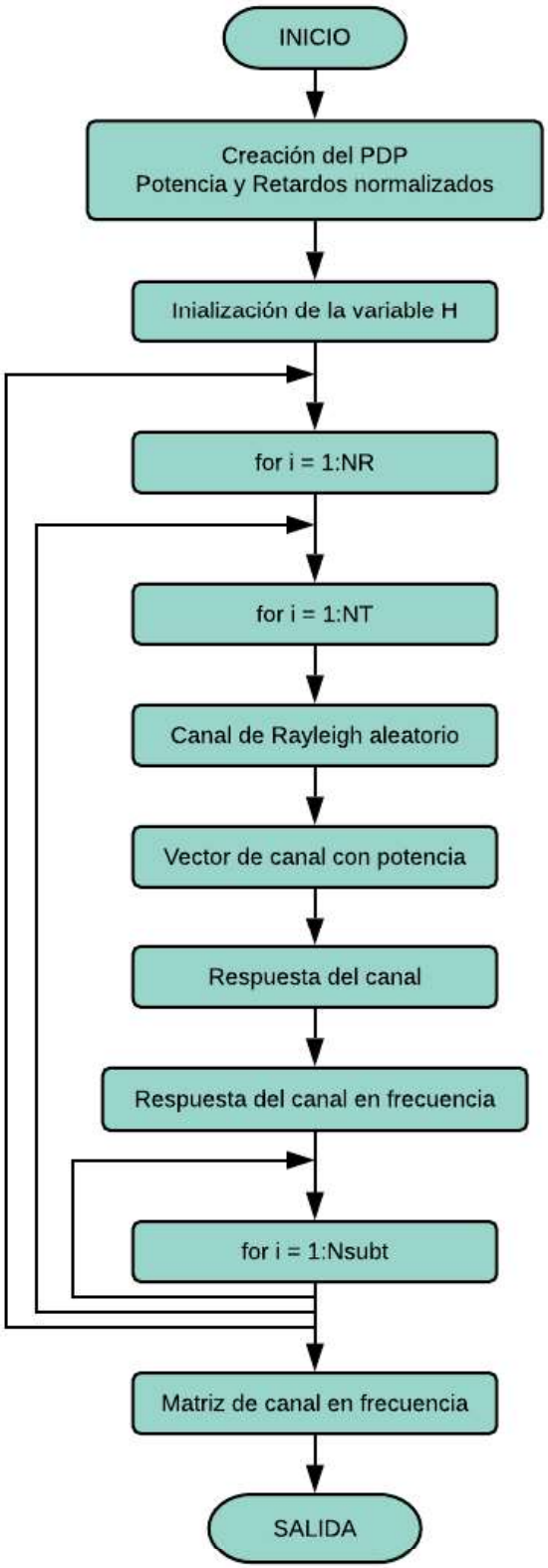

Figura 2.13 Diagrama de flujo de la función *RayleighChannel*.

Una vez establecidos los parámetros de PDP se procede a la construcción de la matriz de canal ***H***, como se observa en el Segmento de código 2.23. Esta matriz está hecha de tres dimensiones las cuales están formadas por el número de antenas trasmisoras, el número de antenas receptoras y el número de rayos transmitidos, por lo cual se utilizan varios lazos *for* para cada dimensión. En esta sección de código se genera el canal de Rayleigh mediante la generación de datos aleatorios guardados en el vector *canal.* Posteriormente, se aplica el PDP y se tiene la respuesta del canal en *h1*, que es un vector fila, en *h* se tiene la respuesta del canal en un vector columna. Mediante el uso de la FFT se obtiene la respuesta del canal en frecuencia, con la cual se genera finalmente la matriz de canal Rayleigh multitrayecto en el dominio de la frecuencia. En el Segmento de código 2.24, una vez obtenida la matriz y la respuesta del canal, se aplica la convolución entre los datos modulados en OFDM, *txSig*, y la respuesta del canal, *h_c,* para añadirle el multitrayecto al canal y finalmente obtener la señal recibida, *rx_chan*.

```
H = zeros( NR, NT, Nsubt ); %Matriz de canal de ceros
for i = 1:NR
   for j = 1:NT
       h1 = [];
       %Canal de Rayleigh
       canal = (randn(1,N_rayos) + 1i*randn(1,N_rayos))/sqrt(2);
       hy = canal .* pm_n;   %Vector de canal con potencia
       h1( retardos + 1 )=hy;%Respuesta del canal vector fila
       h(:,i,j)=h1;          %Respuesta del canal vector columna
       ffth = fft(h1, Nsubt);%Respuesta del canal en frecuencia
       for k = 1:Nsubt
          H(i,j,k) = ffth(k);%Matriz de canal en frecuencia
       end
   end
end
```

Segmento de código 2.23 Generación del canal Rayleigh.

```
rx_chan=zeros(80,NR);
for i=1:NT
  a =conv2(txSig(:,i),h_c(:,:,i));  %añade canal multipath
  rx_chan=rx_chan+a(1:80,:);        %señal recibida
end
```

Segmento de código 2.24 Obtención de la señal recibida.

Canal AWGN

La señal recibida se ve afectada por el ruido blanco Gaussiano, el cual siempre está presente en los sistemas de comunicación. El comando *awgn* agrega este ruido a la señal, requiriendo como parámetros: la señal recibida, *rx_chan,* el *snr* o la relación señal a ruido, y signalpower que para este caso es *'measured',* el cual permite que se mida el nivel de la señal de entrada para determinar el nivel de ruido apropiado basado en el valor de SNR. La señal resultante con ruido blanco Gaussiano se guarda en la variable *rxSignal.* El Segmento de código 2.25 muestra cómo se obtiene esta señal, además de la potencia y varianza del ruido.

```
% CANAL AWGN
for i=1:NR
    rxSignal(:,i) = awgn(rx_chan(:,i),snr,'measured');
end
powerDB = 10*log10(var(rxSignal(:,1))); % potencia del ruido
noiseVar = 10.^(0.1*(powerDB-snr));     % valor del ruido
```

Segmento de código 2.25 Canal AWGN.

- ## Sistema de recepción

Una vez que los datos se han generado, modulado y transmitido por el canal inalámbrico, llegan a la parte de recepción. En la Figura 2.14, se presenta un diagrama de bloques sobre el proceso que siguen los datos trasmitidos en la recepción hasta obtener el mensaje enviado.

Figura 2.14 Diagrama de bloques para el sistema de recepción.

En la recepción los parámetros utilizados son previamente definidos en la parte de transmisión, aquí se realiza la detección de la señal, objeto de estudio del proyecto, una vez demodulada en OFDM. La señal por tratar es *rxSignal,* ya que es la señal que se obtiene después del canal AWGN y llega a recepción. Después de que la señal pasa por cada bloque se realiza la comparación de datos enviados vs. recibidos para el cálculo de errores.

1. Demodulación OFDM: Segmento de código 2.26, aquí se realiza la demodulación OFDM de la señal que llega atravesando el canal inalámbrico. No se realiza sincronización, ya que se asume una sincronización perfecta. La función *ofdmDemod* realiza la demodulación, teniendo en cuenta las propiedades o argumentos declarados, ver Segmento de código 2.3. La variable *ofdmDemSig* contiene los datos demodulados OFDM.

```
% DEMODULACION OFDM
ofdmDemSig = ofdmDemod( rxSignal );
```

Segmento de código 2.26 Demodulación OFDM.

2. Detección ZF – MMSE – ML: Segmento de código 2.27, en esta parte de la simulación se realiza la detección. Se tiene en cuenta el número de datos de entrada para realizar el proceso de detección, *y* contiene la señal demodulada en OFDM, es decir la señal a ser analizada. *HH* representa la matriz de canal, *timerVal* es el temporizador utilizado para medir el tiempo que se demora cada subportadora en ser detectada, es decir el tiempo de detección. En *detSig* se guarda la señal detectada, la cual se obtiene mediante la combinación de los siguientes factores, según la Tabla 2.8:

Tabla 2.8. Combinación de variables para la detección.

Detector	ZF	MMSE	ML
y	x	x	x
HH	x	x	x
noiseVar (1)		x	
Energy (2)		x	
NT (3)		x	x
A (4)			x

Al llegar a este punto de la simulación ya se tiene asignada la variable *detector* por alguna de las siguientes funciones: ZF_detector, MMSE_detector y/o ML_detector. Entonces al llamar a la función actual o asignada, los valores de entrada que necesita la función para ejecutarse normalmente son los que están declarados para obtener *detSig,* estos valores son *y, HH, noiseVar, energy NT y A.* No todos estos valores son requeridos en las diferentes funciones, por lo que cada detector llama al argumento o dato que necesita. Hay que tener en cuenta que los algoritmos de detección van a necesitar siempre la señal a analizar, *y*, y la matriz de canal H, en este caso *HH.* Los datos siguientes a *HH* son los que dependen de los requerimientos del algoritmo por lo que se les asigna un número para ser llamados, este llamado se realiza dentro de la

función de detección con la variable *varargin.* La variable varargin permite pasar a una función un número o variable de argumentos. En la Tabla 2.8, se observan los datos requeridos para cada algoritmo de detección utilizado. Finalmente se detienen el temporizador, *toc(timerVal)*, ya que finaliza la detección, el cual está en un contador que depende del número de iteraciones.

```
% DETECCIÓN
detSig = zeros( inputSize, NT );
for u = 1:inputSize
   y = ofdmDemSig(u,:).';
   HH = H(:,:,u); %Matriz de canal
   timerVal=tic;  %Inicio del temporizador para la detección
   detSig(u,:)=feval(detector,y,HH,noiseVar,energy,NT,A ).';
   tt(j) = tt(j) + toc(timerVal); %Fin del temporizador
end
```

Segmento de código 2.27 Detección.

3. Demodulación QPSK – 16QAM – 64QAM: Segmento de código 2.28, la demodulación a realizar depende de la función actual en la variable *demFun,* parámetro ya establecido anteriormente. Aquí se demodula la señal detectada, *detSig*, y se guarda en *dataOut,* que será la señal final recibida por el usuario.

```
% DEMODULACION (QPSK,QAM)
dataOut = demFun( detSig );
```

Segmento de código 2.28 Demodulaciones QPSK, 16QAM Y 64QAM.

4. Cálculo de errores: Segmento de código 2.29, en este segmento se analiza la señal enviada con la señal recibida para obtener los errores que se han presentado con el comando *biterr.* Este comando permite obtener el número de bit errados entre los datos enviados y los datos recibidos, los cuales se van a ir sumando en el contador *error.*

```
% ERRORES
error = error + biterr( dataIn, dataOut );
```

Segmento de código 2.29 Cálculo de errores.

Aquí también se envían los datos para actualizar la barra de estado, y culmina el segundo lazo, como en el Segmento de código 2.30.

```
      send(D, m);
end
```

Segmento de código 2.30 Envío de datos para actualizar la barra de estado.

Para terminar con el primer lazo de la función, como se muestra en el Segmento de código 2.31, se obtiene el valor del BER a través de la variable *ratio*, dividiendo el número de bits errados por los datos de entrada por los bits/símbolo y por el número de iteraciones. También se tiene el tiempo de simulación en el contador *T.*

```
    ratio(m) = error / inputSize / k / nIter;
    T = T + sum(tt);
end
```

Segmento de código 2.31 Finalización del primer lazo de la función mimoOfdmSingleTest3.

Como penúltimo paso de la función, Segmento de código 2.32, se tienen los argumentos de salida, que son *BER* y *T.* El *BER* y el *T* se calcularon en el Segmento de código 2.32, el tiempo de simulación *T* se calcula dividiendo *T* que sale del primer lazo por el número de iteraciones y por el número de datos de entrada.

```
%% SALIDA
BER = ratio;
T = T / nIter / inputSize;
```

Segmento de código 2.32 Argumentos de salida de la función mimoOfdmSingleTest3.

Como último proceso a ejecutarse en la función *mimoOfdmSingleTest3*, se finaliza la barra de estado (Segmento de código 2.33). Esta barra finaliza en cada modulación realizada en la detección para los diferentes arreglos de antenas.

```
try
   close(h)
catch
end

function nUpdateWaitbar(~)
    txt = sprintf('%.3f [%%]', 100*p/N );
    try
       waitbar( p/N, h, txt );
    catch
       h = waitbar( p/N, txt );
    end
    p = p + 1;
end
```

Segmento de código 2.33 Finalización de la barra de estado.

2.1.3 DETECTORES

Los detectores son llamados en el Segmento de código 2.6 y se ejecutan en la función *mimoOfdmSingleTest3* en la parte de la detección (Segmento de código 2.27). Para poder ejecutarse de manera correcta se llaman a los datos requeridos como argumentos de entrada, como se mostró anteriormente en la Tabla 2.8, donde se pueden ver los valores requeridos por cada función de detección.

- **ZF_detector**

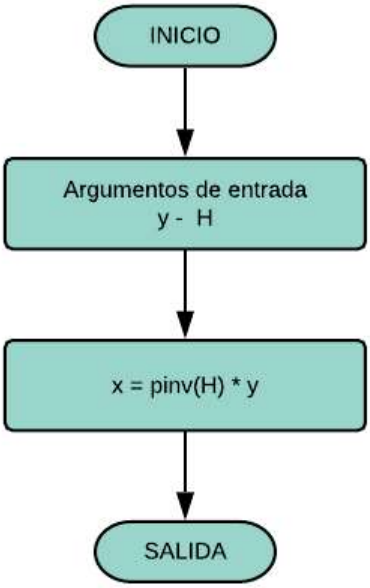

Figura 2.15 Diagrama flujo de la función *ZF_detector*.

```
function x = ZF_detector( y, HH, varargin )

    x = pinv(HH) * y;

end
```

Segmento de código 2.34 Detector ZF.

Tabla 2.9 Parámetros de la función *ZF_detector*.

Parámetros	Tipo de Datos	Descripción
y	Entrada	Señal para analizar
H	Entrada	Matriz de canal
x	Salida	Señal detectada con el algoritmo ZF

El diagrama de flujo presente en la Figura 2.15 contiene el proceso de detección de la señal del algoritmo ZF, esta función presenta dos parámetros de entrada y un parámetro de salida descritos en la Tabla 2.9.

En el Segmento de código 2.34, se tiene el algoritmo de detección ZF, el cual presenta como argumentos de entrada la señal a analizar y la matriz de canal. La señal detectada, *x,* se obtiene con el comando *pinv,* el cual nos entrega la pseudoinversa de la matriz H, esto multiplicado por la señal, *y*.

- **MMSE_detector**

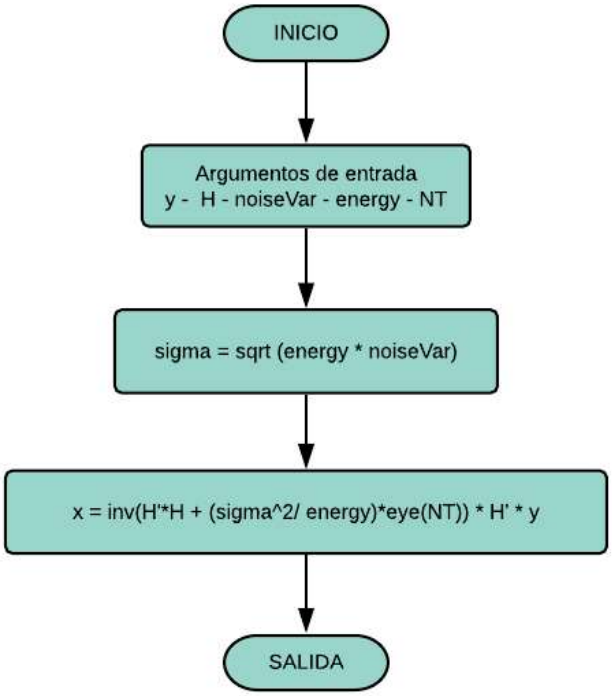

Figura 2.16 Diagrama flujo de la función *MMSE_detector*.

```
function x = MMSE_detector( y, HH, varargin )

   noiseVar = varargin{1};
   energy = varargin{2};
   NT = varargin{3};

   sigma = sqrt(  energy * noiseVar );
   x = inv( HH'*HH + (sigma^2/energy)*eye(NT) ) * HH' * y;

end
```

Segmento de código 2.35 Detector MMSE.

Tabla 2.10 Parámetros de la función *MMSE_detector*.

Parámetros	Tipo de Datos	Descripción
y	Entrada	Señal para analizar
H	Entrada	Matriz de canal
noiseVar	Entrada	Varianza de ruido
energy	Entrada	Energía
NT	Entrada	Número de antenas transmisoras
x	salida	Señal detectada con el algoritmo MMSE

El detector MMSE, como se muestra en el diagrama de flujo de la Figura 2.16, tiene como argumentos de entrada *y, H,* y otros que se llaman a través de la variable *varargin,* siguiendo la posición en la que fueron declarados, como se ve en el Segmento de código 2.35 y en la Tabla 2.10, aquí se observa la descripción de los parámetros de entrada y salida para la función. Posteriormente, se realiza el cálculo de *sigma*, que es la raíz cuadrada de la energía por la varianza de ruido. Con este dato, se puede calcular la señal detectada, ya que cumple con la Ecuación 1.11 que corresponde a la del detector MMSE.

- **ML_detector**

```
function x = ML_detector( y, HH, varargin )
    NT = varargin{3};
    A = varargin{4};

    d = repmat( {A}, 1, NT - 1 );

    xs = allcomb(d{:}).';

    M = length(A);
    N = size(xs,2);
    xss = zeros( NT, M );
    minnorm = zeros( M, 1 );

    for i = 1:M
        xsss = [ xs; repmat( A(i), 1, N ) ];
        norms = vecnorm( y - HH*xsss );
        [ minnorm(i), index ] = min(norms);
        xss( :, i ) = xsss( :, index );
    end

    [ ~, index ] = min(minnorm);
    x = xss( :, index );

end
```

Segmento de código 2.36 Detector ML.

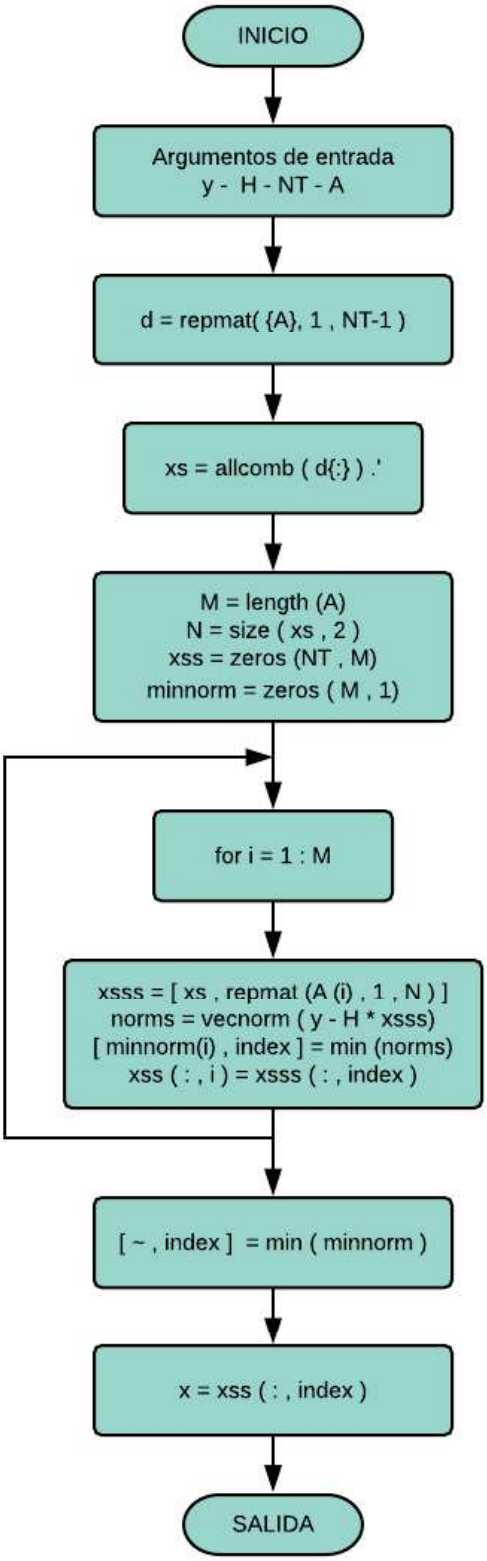

Figura 2.17 Diagrama flujo de la función *ML_detector*.

Tabla 2.11 Parámetros de la función *ML_detector*.

Parámetros	Tipo de Datos	Descripción
y	Entrada	Señal para analizar
H	Entrada	Matriz de canal
NT	Entrada	Número de antenas transmisoras
A	Entrada	Constelación
x	salida	Señal detectada con el algoritmo MMSE

El diagrama de flujo de la función *ML_detector* se presenta en la Figura 2.17. Los procesos realizados en esta función están descritos en el Segmento de código 2.36. La función presenta el algoritmo de detección ML, aquí también se llama a ciertos valores requeridos como argumentos de entrada (*NT* y *A*) con la variable *varargin,* Tabla 2.11. Con el comando *repmat*, se copia la matriz *A* y entrega una matriz *d* de copias de *A* de acuerdo con los valores declarados, aquí se declara que se copia 1 vez por *NT-1* veces. Esto se hace para reducir un poco la complejidad, ya que la misma aumenta exponencialmente con el número de antenas y modulación. Por ejemplo, al tener un M = 64 con un arreglo de antenas 4 x 4, la matriz es demasiado grande, incurriendo en mayor tiempo de detección.

Posteriormente se llama a la función que realiza todas las combinaciones posibles, *allcomb*, Segmento de código 2.37 [23]. Esta función entrega todas las combinaciones de los elementos declarados, pueden ser números o caracteres [23].

A continuación, se inicializan las variables que van a ser utilizadas dentro del *for*, el cual inicia en 1 y va hasta *M,* que especifica el orden de la modulación. Con *xsss* se añade el nuevo símbolo que corresponde a la antena que se quitó en *d,* entonces *xsss,* será la señal que va a ser detectada. Los siguientes pasos cumplen con la Ecuación 1.13 que corresponde a la ecuación del detector ML donde se obtiene la norma de la mínima distancia entre el vector de la señal recibida *y*, y el producto de todos los vectores de señal transmitidos con el canal, *H*xsss*. Se extrae de esos valores los mínimos (índices), terminado el lazo, se extrae el índice mínimo de todos los mínimos para obtener así la señal *x* con el mínimo valor.

```
function A = allcomb(varargin)
narginchk(1,Inf) ;
NC = nargin ;
% check if we should flip the order
if ischar(varargin{end}) && (strcmpi(varargin{end}, 'matlab') ||
strcmpi(varargin{end}, 'john'))
    % based on a suggestion by JD on the FEX
    NC = NC-1 ;
    ii = 1:NC ; % now first argument will change fastest
else
% default: enter arguments backwards, so last one (AN) is changing
fastest
    ii = NC:-1:1 ;
end
args = varargin(1:NC) ;
if any(cellfun('isempty', args)) % check for empty inputs
    warning('ALLCOMB:EmptyInput','One of more empty inputs result in
an empty output.') ;
    A = zeros(0, NC) ;
elseif NC == 0 % no inputs
    A = zeros(0,0) ;
elseif NC == 1 % a single input, nothing to combine
    A = args{1}(:) ;
else
    isCellInput = cellfun(@iscell, args) ;
    if any(isCellInput)
        if ~all(isCellInput)
            error('ALLCOMB:InvalidCellInput', ...
            'For cell input, all arguments should be cell arrays.') ;
        end
        % for cell input, we use to indices to get all combinations
        ix = cellfun(@(c) 1:numel(c), args, 'un', 0) ;

        % flip using ii if last column is changing fastest
        [ix{ii}] = ndgrid(ix{ii}) ;

        A = cell(numel(ix{1}), NC) ; % pre-allocate the output
        for k = 1:NC
            % combine
            A(:,k) = reshape(args{k}(ix{k}), [], 1) ;
        end
    else
        % non-cell input, assuming all numerical values or strings
        % flip using ii if last column is changing fastest
        [A{ii}] = ndgrid(args{ii}) ;
        % concatenate
        A = reshape(cat(NC+1,A{:}), [], NC) ;
    end
end
```

Segmento de código 2.37 Función *allcomb* [23].

2.1.4 FUNCIÓN GRAFICARDETECTORES

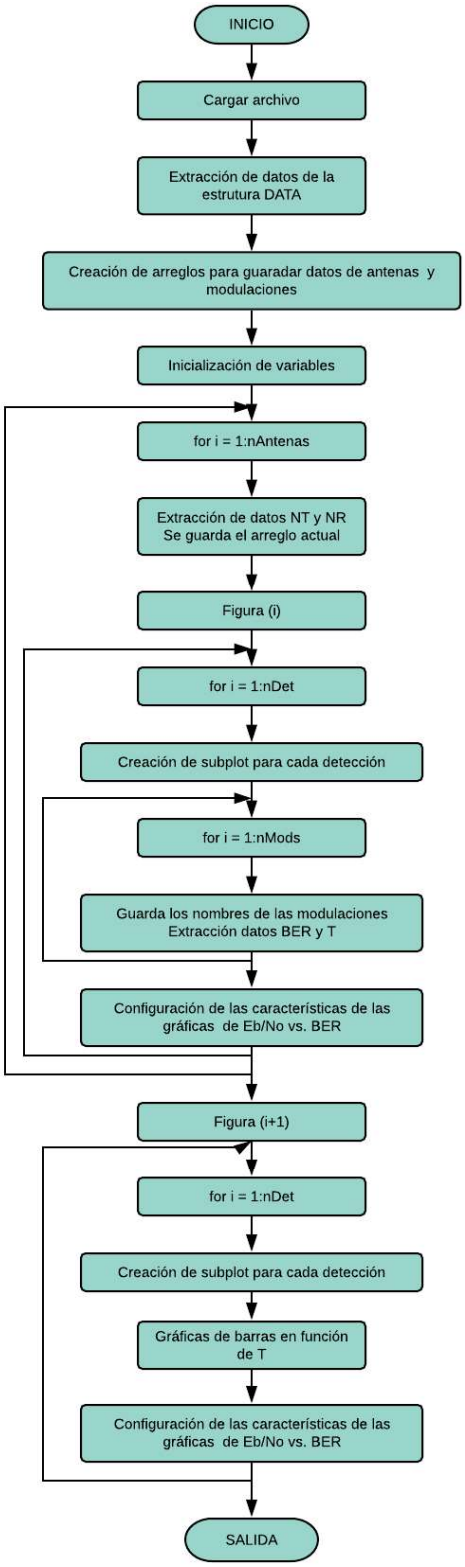

Figura 2.18 Diagrama flujo de la función *graficarDetectores*.

Para realizar las gráficas de BER vs. Eb/No y los tiempos de simulación, se utiliza la función graficarDetectores cuyo diagrama de flujo se encuentra en la Figura 2.18. Esta función permite cargar un archivo .mat generado con el programa principal, dando la facilidad de ver las simulaciones anteriores o pasadas. Cada archivo generado se guarda con un nombre específico, lo que no permite la sobre escritura y por ende evita perder simulaciones. En el Segmento de código 2.38 se realiza el proceso de cargar el archivo .mat que se va a graficar. La ventana emergente que permite cargar el archivo, Figura 2.19, no se cierra hasta que se seleccione un archivo, se introduzca la dirección en donde se encuentre el archivo (path) o se presione cancelar.

En el Segmento de código 2.39 se extrae los valores guardados en la estructura *DATA,* valores como detectores, modulaciones, antenas, Eb/No, BER, T, entre otros. Con la variable *names* y *antenas* se crean arreglos para guardar los nombres y antenas de las modulaciones. Con *BER* se crea un vector de ceros para guardar los *BER* de cada modulación y con *T* se crea otro vector de ceros, en este se guardará el tiempo de simulación por cada antena.

```
function graficarDetectores( varargin )

    if nargin == 0
        [file,path] = uigetfile('*.mat');

       if file == 0
            return
        else

           filepath = fullfile(path,file);
           load( filepath, 'DATA' );
        end
    else

        switch class( varargin{1} )
            case 'string'
                filepath = varargin{1};
                load( filepath, 'DATA' );
            case 'struct'
                DATA = varargin{1};
        end
    end
```

Segmento de código 2.38 Cargar archivo a graficar.

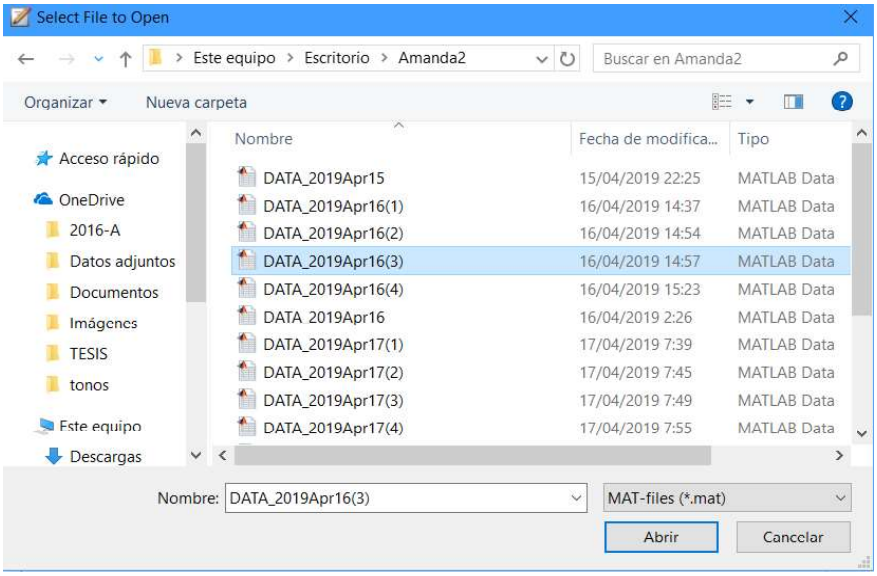

Figura 2.19 Ventana emergente para cargar el archivo a graficar.

```
detectors = DATA.detectors;
nDet = length( detectors );
modTypes = DATA.modTypes;
nMods = length( modTypes );
nAntenas = length( DATA.nt );
EbNo = DATA.EbNoVec;
names = cell( 1, nMods );
antenas = cell( 1, nAntenas );
BER = zeros( length(EbNo), nMods );
T = zeros( nAntenas, nMods, nDet );
```

Segmento de código 2.39 Extracción e inicialización de variables.

El Segmento de código 2.40 inicia con un lazo sobre las antenas, donde se extrae de la estructura *DATA* el número de antenas para transmisión como para recepción, *NT* y *NR.* Posteriormente, se escribe la configuración actual de antenas y se guardan los nombres en *antenas*. Con *figure(i)* se crea una nueva figura cada vez que se ejecuta el lazo. A continuación, sigue el lazo sobre los detectores, aquí se crean los *subplots* dependiendo del número de detecciones realizadas. En el lazo sobre las modulaciones se guardan datos y se extrae de la estructura los valores de *BER* y *T* para ser graficados. Las gráficas obtenidas son semilogarítmicas y se grafica en el eje *y* todos los BER extraídos y en el eje *x* el Eb/No. También se establece que en la leyenda se pongan todos los nombres de las modulaciones, y que se imprima el detector utilizado y el arreglo de antenas correspondiente a cada uno. Por último, se establecen los límites en los ejes. Un ejemplo de la gráfica obtenida se presenta en la Figura 2.20.

```
for i = 1:nAntenas

    NT = DATA.nt(i);
    NR = DATA.nr(i);
    antena = sprintf( 'ANT_%ux%u', NT, NR );
    antenas{i} = sprintf( '%ux%u', NT, NR );

    figure(i)

    for k = 1:nDet
        subplot( 1, nDet, k );
        det = detectors{k};
        for j = 1:nMods
            names{j} = modTypes{j};
            BER(:,j) = DATA.(antena).(det).(modTypes{j}).BER;
            T(i,j,k) = DATA.(antena).(det).(modTypes{j}).T;
        end
        semilogy( EbNo, BER )
        hold on
        grid on
        xlabel('Eb/No (dB)')
        ylabel('Bit Error Rate')
        legend( names )
        title( sprintf("%s (%ux%u)",det,NT,NR) )
        axis([0 20 1e-4 1e0])
    end
end
```

Segmento de código 2.40 Gráficas de errores.

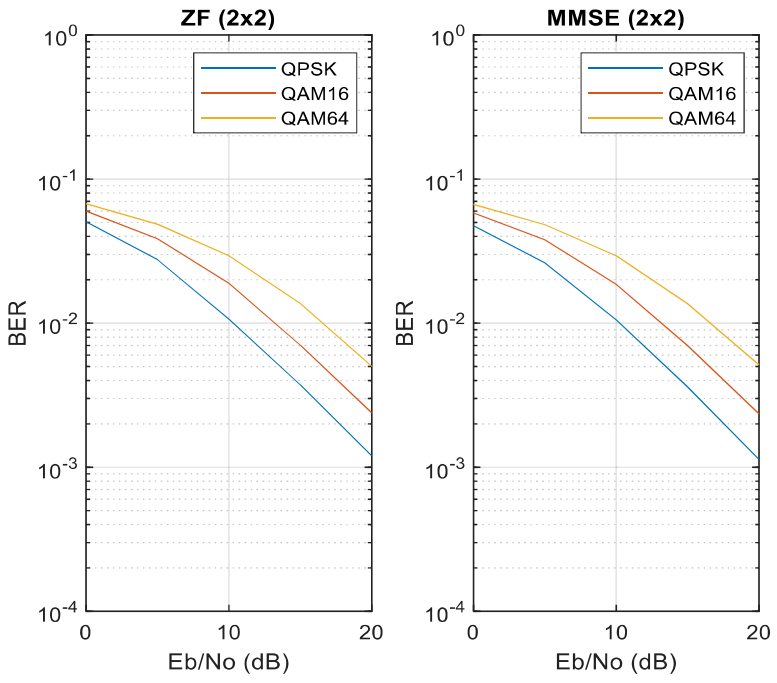

Figura 2.20 Figura de BER vs. Eb/No creada con la función graficarDetectores.

```
    figure(i+1)
    for k = 1:nDet
        subplot( 1, nDet, k );
        det = detectors{k};
        bar(T(:,:,k))
        grid on
        xticklabels(antenas)
        xlabel('Arreglo de Antenas N_TxN_R')
        ylabel('Tiempo [seg]')
        legend( names )
        title(det)
    end

end
```

Segmento de código 2.41 Gráficas de los tiempos de simulación.

En el Segmento de código 2.41 se crea una nueva figura, para apreciar los tiempos de simulación, solo del proceso de detección, por lo que se crea el lazo sobre las detecciones. Aquí se crean los *subplots* dependiendo del número de detecciones realizadas, posteriormente para ver los tiempos se grafican barras con los datos de *T*. En cada *subplot* que corresponde a una detección se grafican los tiempos de cada modulación para cada arreglo de antenas. En el eje *x* se tienen los arreglos de antenas y en el eje *y* el tiempo en segundos, la leyenda muestra cada modulación. Un ejemplo de la gráfica generada se presenta en la Figura 2.21.

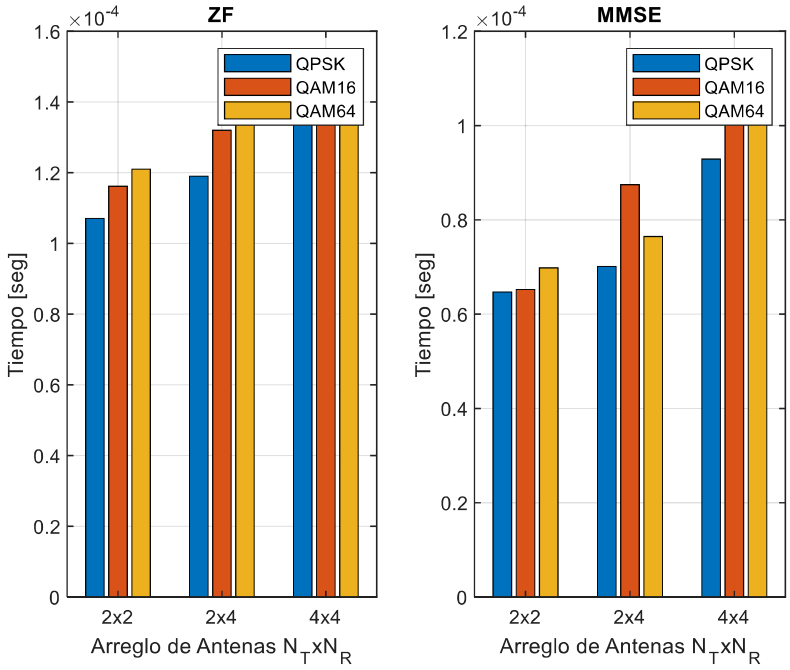

Figura 2.21 Figura de tiempos de detección creada con la función graficarDetectores.

El código completo del programa realizado con cada una de las funciones utilizadas en el software de Matlab se presenta en el Anexo I (Anexo digital). En el Anexo II se presenta una tabla con los comandos requeridos para la realización del programa, también se presenta la descripción de cada uno de ellos.

3. RESULTADOS Y DISCUSIÓN

En este capítulo se presentan y se analizan los resultados de las gráficas de Eb/No vs. BER de la simulación de los diferentes algoritmos detectores ZF, MMSE y ML para un sistema MIMO – OFDM con las diferentes configuraciones de antenas planteadas, así como las modulaciones. Además, se presentarán gráficas de los tiempos de simulación para cada algoritmo detector y así comprobar su funcionamiento y complejidad frente al análisis teórico.

3.1. ESCENARIO DE SIMULACIÓN

Los resultados de la simulación se obtienen mediante la simulación de un sistema MIMO – OFDM con la generación de funciones de Matlab, el escenario consiste en un sistema de transmisión y recepción, entre los dos sistemas existe un canal inalámbrico, este proceso inicia con la generación de datos aleatorios, los cuales son modulados en QPSK, 16QAM y 64QAM. Posteriormente los datos modulados digitalmente atraviesan el modulador OFDM. Los símbolos OFDM obtenidos, ingresan al canal multitrayecto con desvanecimiento de Rayleigh y se les agrega el ruido blanco Gaussiano, AWGN. La señal obtenida en la recepción es demodulada en OFDM e ingresa en la etapa de detección. Para la detección de la señal se implementará la simulación de tres algoritmos como ZF, MMSE y MLD, y cada uno es analizado con las tres configuraciones de antenas y teniendo en cuenta cada modulación. Posteriormente, la señal detectada es demodulada en QPSK, 16QAM y 64QAM, consiguiendo así los datos enviados para finalmente obtener el cálculo de errores y la realización de las gráficas.

Los resultados obtenidos son gráficas de Eb/No vs. BER y de tiempos de simulación de cada detección realizada.

Los parámetros por considerar son:

- Arreglo de antenas: 2 x 2, 2 x 4 y 4 x 4.
- Modulación: QPSK, 16QAM y 64QAM.
- Orden de modulación M: 4, 16 y 64
- Detector: ZF, MMSE y ML.
- Vector de Eb/No: [0:20]dB
- Número de iteraciones 1: 10000
- Número de iteraciones 2: 100 (Para 64QAM, ML y 4 x 4)

3.2. SISTEMA MIMO – OFDM CON ARREGLO DE ANTENAS 2 X 2 Y MODULACIONES QPSK, 16QAM Y 64QAM.

La discusión de los resultados se realiza en base a los detectores, para comparar y obtener el de mejor rendimiento. A continuación, se interpretan las gráficas para cada detección, y finalmente se realiza el análisis de los resultados para obtener el detector que proporciona un mejor rendimiento.

- Detector ZF

Para la obtención de la Figura 3.1 que corresponde a la gráfica de Eb/No vs BER de la simulación del sistema MIMO – OFDM con un arreglo de antenas de 2 x 2, algoritmo de detección ZF y las diferentes modulaciones, se utilizaron 10000 iteraciones por Eb/No para obtener gráficas mejor definidas. La Figura 3.1 muestra los resultados de Eb/No vs BER para cada modulación, descrita en la leyenda de la gráfica. La modulación 64QAM al cortar por ejemplo en 10 dB presenta un BER de 0.01469. La modulación 16QAM al cortar en 10 dB presenta un BER de 0.009451. Por último, la modulación QPSK al cortar igualmente en 10 dB muestra un valor de BER igual a 0.005337. De los resultados se obtiene que para un mismo Eb/No, QPSK presenta el menor valor de BER.

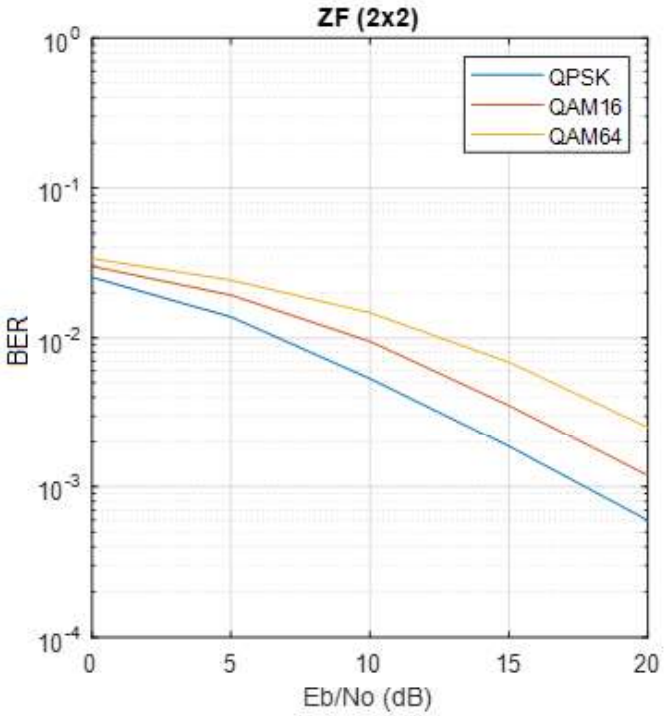

Figura 3. 1 Eb/No vs BER para arreglo de antenas 2 x 2, detector ZF y modulaciones QPSK, 16QAM Y 64QAM.

- Detector MMSE

Para obtener la gráfica de Eb/No vs BER de la simulación del sistema MIMO – OFDM con un arreglo de antenas de 2 x 2, algoritmo de detección MMSE y las diferentes modulaciones, se utilizaron 10000 iteraciones por Eb/No. La Figura 3.2 muestra las gráficas de las modulaciones QPSK (de color azul), 16QAM (de color rojo) y 64QAM (de color amarillo). Entonces, la modulación 64QAM presenta un valor de BER = 0.01467 al cortar con Eb/No de 10 dB. La modulación 16QAM presenta un BER = 0.009293. Por último, la modulación QPSK con Eb/No = 10 dB muestra un valor de BER igual a 0.005266

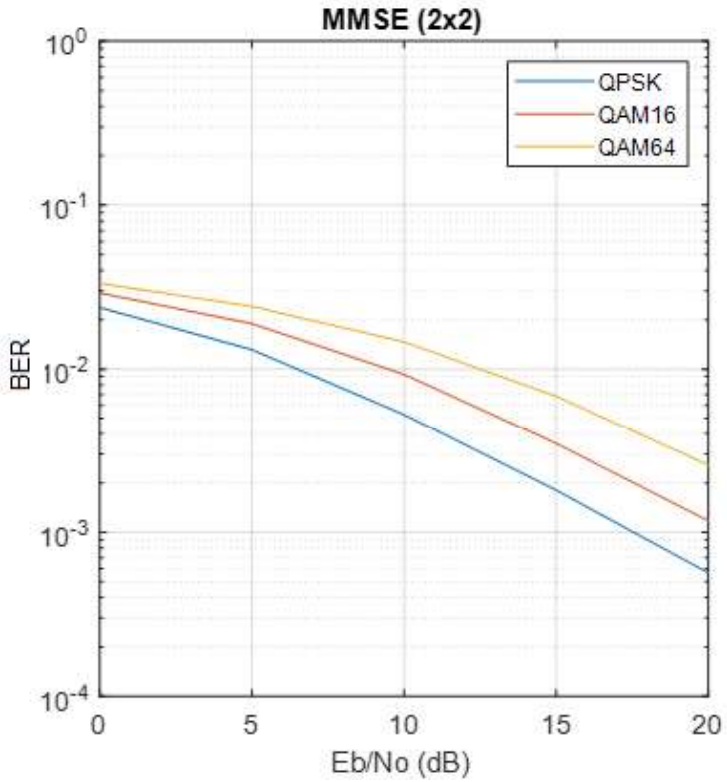

Figura 3. 2 Eb/No vs BER para arreglo de antenas 2 x 2, detector MMSE y modulaciones QPSK, 16QAM Y 64QAM.

- Detector MLD

En la Figura 3.3 se tiene la gráfica de Eb/No vs BER de la simulación del sistema MIMO – OFDM con un arreglo de antenas de 2 x 2, algoritmo de detección ML y las modulaciones QPSK, 16QAM Y 64QAM. Para la obtención de esta gráfica se utilizaron 10000 iteraciones por Eb/No. Los resultados obtenidos de Eb/No vs BER para la modulación 64QAM al cortar en 10 dB de Eb/No da un BER de 0.01428. La modulación 16QAM al cortar en 10 dB

presenta un BER de 0.007691. Por último, la modulación QPSK al cortar igualmente en 10 dB muestra un valor de BER igual a 0.002122.

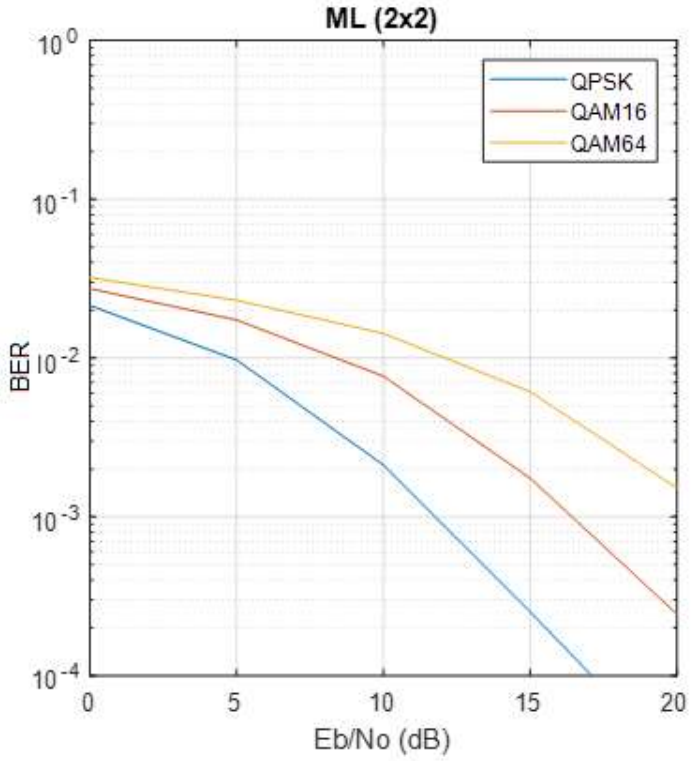

Figura 3. 3 Eb/No vs BER para arreglo de antenas 2 x 2, detector ML y modulaciones QPSK, 16QAM Y 64QAM.

Finalmente, para una mejor apreciación de los resultados se presenta la Tabla 3.1, que contiene los valores obtenidos de BER para un ejemplo de Eb/No = 10 dB. El valor de Eb/No a utilizar se escoge de forma arbitraria y teniendo en cuenta que debe ser el mismo valor para cada modulación en todos los detectores para que sea un análisis justo.

Tabla 3.1 Valores de BER para Eb/No = 10 [dB].

ARREGLO DE ANTENAS	Eb/No	MODULACIÓN	BER PARA LOS DETECTORES:		
			ZF	MMSE	ML
2 x 2	10 [dB]	QPSK	0.005337	0.005266	0.002122
		16QAM	0.009451	0.009293	0.007691
		64QAM	0.01469	0.01467	0.01428

Para los métodos de detección lineal (ZF y MMSE) los valores obtenidos presentan una variación mínima entre ambos. Los valores de BER que se obtienen del detector MMSE

son ligeramente mejores que los obtenidos por el detector ZF ya que, presentan un BER menor.

Para el detector ML, los valores obtenidos con un Eb/No = 10 dB son menores respecto a los obtenidos con MMSE, lo que implica un mejor BER, respecto de los detectores lineales y por ende una mejor detección.

3.3. SISTEMA MIMO – OFDM CON ARREGLO DE ANTENAS 2 X 4 Y MODULACIONES QPSK, 16QAM Y 64QAM.

Los resultados a continuación se obtienen de la comparación entre los detectores ZF, MMSE y ML. Hay que tener en cuenta que se ejecuta sobre un arreglo de antenas de 2 x 4 y modulaciones QPSK, 16QAM Y 64QAM. La obtención de cada una de las figuras se realiza con 10000 iteraciones por Eb/No, para así apreciar las gráficas mejor definidas. Para la interpretación y análisis de los resultados se toma arbitrariamente un Eb/No igual a 10 dB, como ejemplo. Cabe recalcar que para un análisis justo el Eb/No escogido debe ser igual para cada detector y modulación a analizar.

- Detector ZF

En la Figura 3.4 se tiene la gráfica de la simulación del sistema MIMO – OFDM con un arreglo de antenas de 2 x 4 con algoritmo de detección ZF y modulaciones QPSK, 16QAM y 64QAM. La Figura 3.4 muestra los resultados de Eb/No vs BER para cada modulación. La modulación 64QAM al cortar en 10 dB presenta un BER = 0.005519. La modulación 16QAM presenta un BER de 0.001523. Por último, la modulación QPSK al cortar igualmente en 10 dB muestra un valor de BER = 0.0002591.

- Detector MMSE

Los resultados de la simulación del detector MMSE para un sistema MIMO – OFDM con arreglo de antenas 2 x 4 y modulaciones QPSK, 16QAM y 64QAM se presentan en la Figura 3.5. Aquí cada modulación está representada por un color como se observa en la respectiva leyenda e indica los resultados de Eb/No vs BER para cada una de las mismas. Como se mencionó anteriormente el valor de Eb/No para el análisis será 10 dB. Para la modulación QPSK se tiene un valor de BER = 0.0002554. Con la modulación 16QAM el valor de BER es 0.001537. Como última modulación se tiene 64QAM que proporciona un BER de 0.005509.

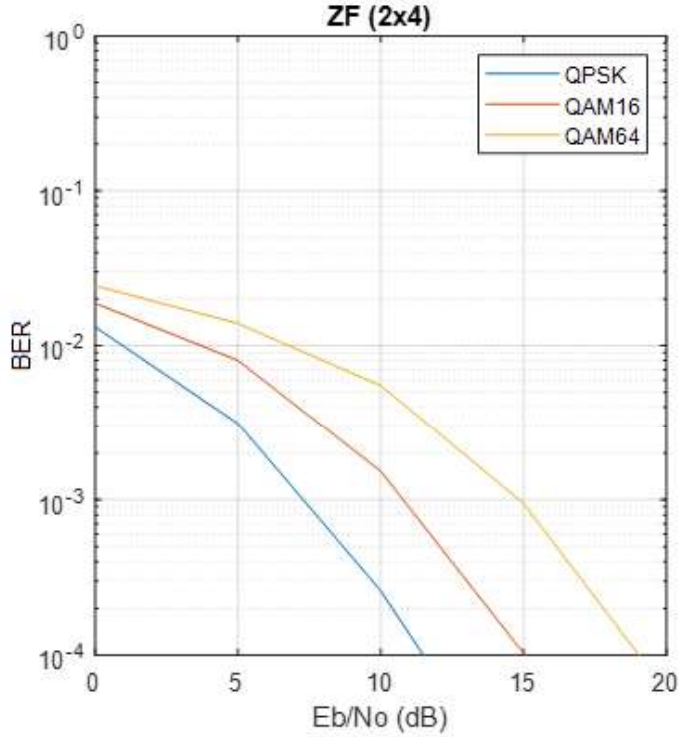

Figura 3. 4 Eb/No vs BER para arreglo de antenas 2 x 4, detector ZF y modulaciones QPSK, 16QAM Y 64QAM.

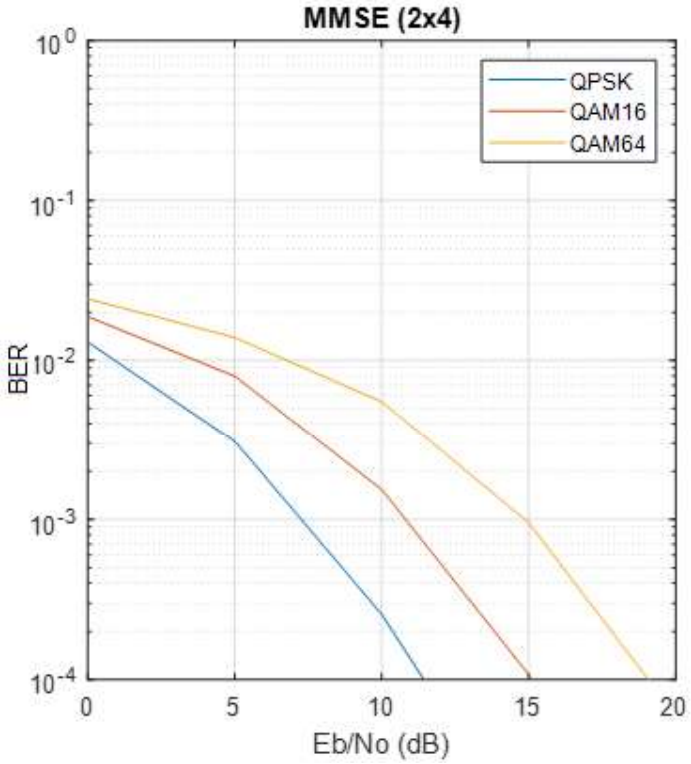

Figura 3. 5 Eb/No vs BER para arreglo de antenas 2 x 4, detector MMSE y modulaciones QPSK, 16QAM Y 64QAM.

- Detector ML

En la Figura 3.6 se presenta la simulación del sistema MIMO – OFDM con un arreglo de antenas de 2 x 4 con algoritmo de detección ML y modulaciones QPSK, 16QAM y 64QAM. Esta gráfica muestra los valores de Eb/No vs BER para cada modulación realizada, las mismas se distinguen una de otra por el color asignado en la leyenda, por ejemplo: QPSK esta representada por el color azul, 16QAM con el color rojo y 64QAM con el color amarillo. Los resultados obtenidos de Eb/No vs BER para la modulación 64QAM al cortar en 10 dB da un BER de 0.005465. La modulación 16QAM para un Eb/No = 10 dB presenta un BER de 0.001162. Finalmente, la modulación QPSK al cortar en 10 dB muestra un valor de BER igual a 9.844e-05.

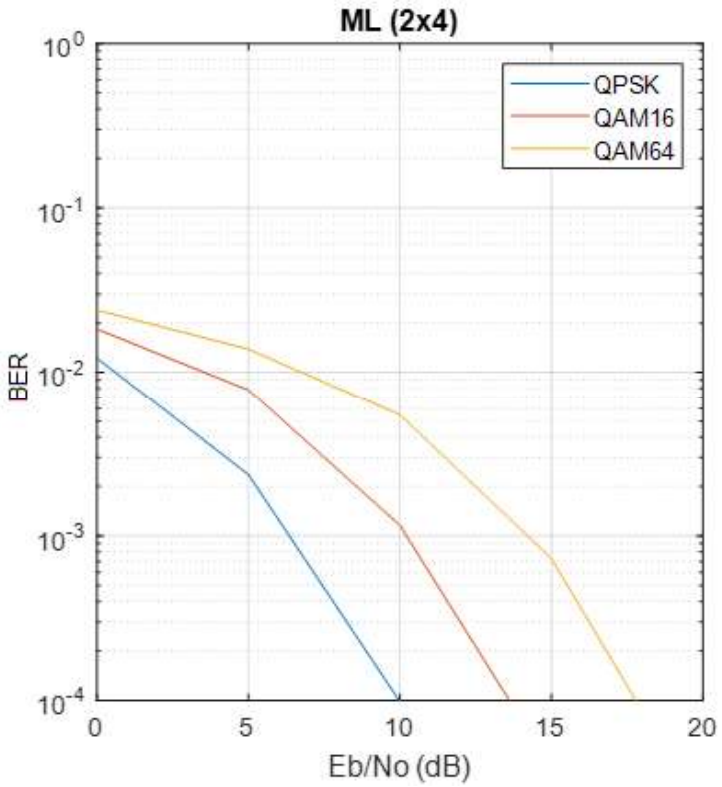

Figura 3. 6 Eb/No vs BER para arreglo de antenas 2 x 4, detector ML y modulaciones QPSK, 16QAM Y 64QAM.

En la Tabla 3.2 se presentan los valores de BER para un Eb/No = 10 dB. Según los datos obtenidos los detectores ZF y MMSE presentan valores de BER mayores a los del detector ML, lo que infiere una mejor detección respecto a los detectores lineales para el algoritmo de detección ML. Entre los detectores ZF y MMSE se observan los valores de BER más altos en ZF lo que implica una mejor detección para MMSE. Cabe recalcar que para la

modulación 16QAM el algoritmo ZF presenta un valor menor al del algoritmo MMSE, esto debido a la tendencia de la gráfica, lo que indica que se ha realizado una mejor detección con el algoritmo ZF.

Tabla 3.2 Valores de BER para Eb/No = 10 [dB].

ARREGLO DE ANTENAS	Eb/No	MODULACIÓN	BER PARA LOS DETECTORES:		
			ZF	MMSE	ML
2 x 4	10 [dB]	QPSK	0.0002591	0.0002554	9.844e-05
		16QAM	0.001523	0.001537	0.001162
		64QAM	0.005519	0.005509	0.005465

3.4. SISTEMA MIMO – OFDM CON ARREGLO DE ANTENAS 4 X 4 Y MODULACIONES QPSK, 16QAM Y 64QAM.

En esta sección los resultados que se obtienen provienen de la comparación entre los detectores ZF, MMSE y ML. Es importante tener en cuenta que se ejecutará cada detector sobre un arreglo de antenas 4 x 4 y modulaciones QPSK, 16QAM Y 64QAM. Para obtener gráficas mejor definidas se realiza la simulación con 10000 iteraciones por Eb/No para las modulaciones QPSK y 16QAM. Para la modulación 64QAM se realiza la simulación con 100 iteraciones por Eb/No. Estas variaciones del número de iteraciones se dan debido a que la complejidad y tiempo de simulación crece exponencialmente con el número de antenas y modulación. Para la interpretación y análisis de los resultados se toma un Eb/No igual a 10 dB. Cabe recalcar que para un análisis justo el Eb/No escogido debe ser igual para cada detector y modulación a analizar.

- Detector ZF

La Figura 3.7 corresponde a la simulación del sistema MIMO – OFDM con un arreglo de antenas de 4 x 4, algoritmo de detección ZF y modulaciones QPSK, 16QAM Y 64QAM. La gráfica muestra los resultados de Eb/No vs BER para cada modulación descrita en la respectiva leyenda. La modulación 64QAM presenta un BER de 0.02719 al cortar con un Eb/No de 10 dB. Para la modulación 16QAM el BER obtenido es de 0.02231. Por último, la modulación QPSK da un valor de BER igual a 0.01727.

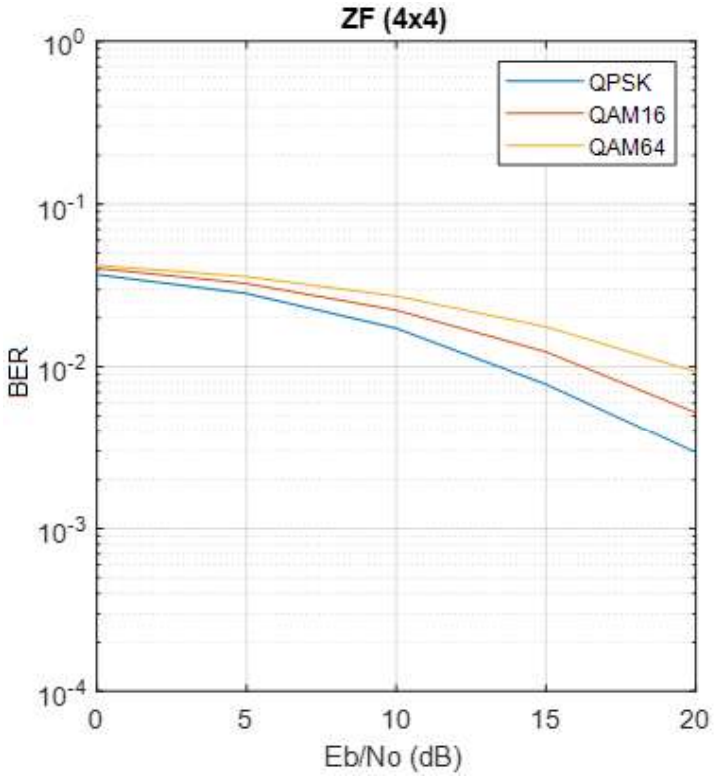

Figura 3. 7 Eb/No vs BER para arreglo de antenas 4 x 4, detector ZF y modulaciones QPSK, 16QAM Y 64QAM.

- Detector MMSE

Los resultados de la simulación del detector MMSE para un sistema MIMO – OFDM con arreglo de antenas 4 x 4 y modulaciones QPSK, 16QAM y 64QAM se presentan en la Figura 3.8. Aquí cada modulación está representada por un color como se observa en la respectiva leyenda e indica los resultados de Eb/No vs BER para cada una de las mismas. Como se mencionó anteriormente el valor de Eb/No para el análisis será 10 dB. Para la modulación QPSK se tiene un valor de BER = 0.0153. Con la modulación 16QAM el valor de BER es 0.02143. Como última modulación se tiene 64QAM que proporciona un BER de 0.02667.

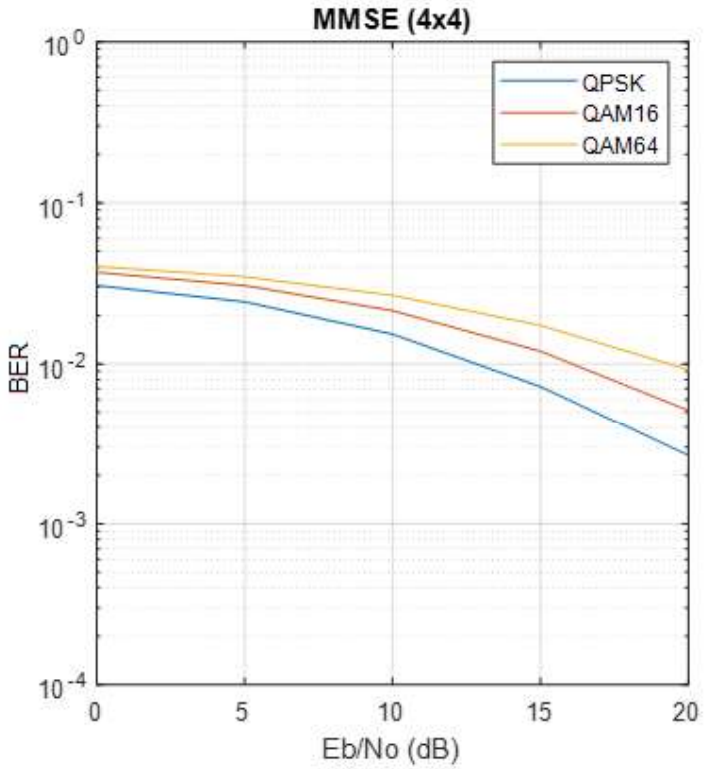

Figura 3. 8 Eb/No vs BER para arreglo de antenas 4 x 4, detector MMSE y modulaciones QPSK, 16QAM Y 64QAM.

- Detector MLD

En la Figura 3.9 se aprecia la simulación del sistema MIMO – OFDM con un arreglo de antenas de 4 x 4 con algoritmo de detección ML y modulaciones QPSK, 16QAM y 64QAM. Esta gráfica muestra los valores de Eb/No vs BER para cada modulación realizada, las mismas se distinguen una de otra por el color asignado en la leyenda. Los resultados obtenidos de Eb/No vs BER para la modulación 64QAM al cortar en 10 dB da un BER de 0.02324. La modulación 16QAM para un Eb/No = 10 dB presenta un BER de 0.0162. Finalmente, la modulación QPSK al cortar en 10 dB muestra un valor de BER igual a 0.00593.

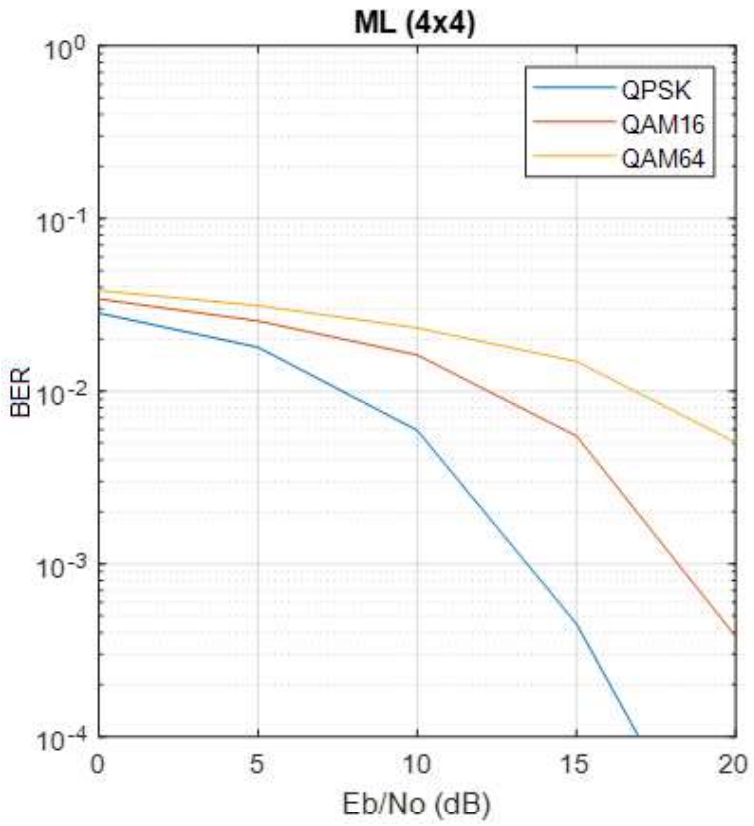

Figura 3. 9 Eb/No vs BER para arreglo de antenas 4 x 4, detector ML y modulaciones QPSK, 16QAM Y 64QAM.

Finalmente, para una mejor apreciación de los resultados se presenta la Tabla 3.3, que contiene los valores obtenidos de BER para un ejemplo de Eb/No = 10 dB.

Tabla 3.3 Valores de BER para Eb/No = 10 [dB].

ARREGLO DE ANTENAS	Eb/No	MODULACIÓN	BER PARA LOS DETECTORES:		
			ZF	MMSE	ML
4 x 4	10 [dB]	QPSK	0.01727	0.0153	0.00593
		16QAM	0.02231	0.02143	0.0162
		64QAM	0.02719	0.02667	0.02324

Para los métodos de detección lineal los valores obtenidos presentan una variación mínima entre ellos. Los valores de BER que se obtienen del detector MMSE son ligeramente mejores que los obtenidos por el detector ZF, ya que presentan un BER menor, implicando una mejor detección para MMSE con este arreglo de antenas.

Para el detector ML, los valores obtenidos con Eb/No = 10 dB son menores en comparación a los de MMSE, lo que implica un mejor BER, respecto de los detectores anteriores y por ende mejor detección.

En la Tabla 3.4 se presentan los resultados del BER para cada modulación y algoritmo de detección analizado en Eb/No = 10.

Tabla 3.4 Valores de BER para Eb/No = 10 de los escenarios planteados.

Eb/No	MODULACIÓN	ARREGLO DE ANTENAS	BER PARA LOS DETECTORES:		
			ZF	MMSE	ML
10 [dB]	QPSK	2 x 2	0.005337	0.005266	0.002122
		2 x 4	0.0002591	0.0002554	9.844e-05
		4 x 4	0.01727	0.0153	0.00593
	16QAM	2 x 2	0.009451	0.009293	0.007691
		2 x 4	0.001523	0.001537	0.001162
		4 x 4	0.02231	0.02143	0.0162
	64QAM	2 x 2	0.01469	0.01467	0.01428
		2 x 4	0.005519	0.005509	0.005465
		4 x 4	0.02719	0.02667	0.02324

Con los datos que muestra la Tabla 3.4 se puede comparar los diferentes algoritmos de detección para cada modulación ejecutada con las diferentes configuraciones de antenas. Por ejemplo, para la modulación QPSK realizada con el algoritmo de detección MMSE se tiene que se obtuvo una mejor detección con el arreglo de antenas 2 x 4. También se aprecia que la detección del ML es mejor en comparación con ZF y MMSE debido a que presenta valores menores a estos dos algoritmos. Para el arreglo de antenas 2 x 4 con todos los algoritmos detectores se tienen valores de BER menores que los demás arreglos, independientemente de la modulación.

El valor tomado para el análisis realizado, Eb/No = 10, puede variar. Para la Tabla 3.5 se presentan valores de BER de todos los escenarios para un Eb/No = 15. Aquí se puede ver que los valores analizados para Eb/No = 15 presentan la misma tendencia que para Eb/No = 10. Estos valores de BER indican varias características de la detección. Por ejemplo, aquí también se observa que para la modulación 16QAM y 64QAM con arreglo de antenas 2 x 4 se tienen valores de BER ligeramente menores para el algoritmo de detección ZF con respecto al algoritmo MMSE, lo que implica una mejor detección para ZF. Hay que tener en cuenta que para Eb/No = 10 esta variación solo se presenta en la modulación 16QAM.

Tabla 3.5 Valores de BER para Eb/No = 15 de los escenarios planteados.

Eb/No	MODULACIÓN	ARREGLO DE ANTENAS	BER PARA LOS DETECTORES:		
			ZF	MMSE	ML
15 [dB]	QPSK	2 x 2	0.001864	0.001806	0.0002524
		2 x 4	1.07e-05	9.727e-06	1.68e-06
		4 x 4	0.007827	0.007216	0.0004482
	16QAM	2 x 2	0.003519	0.003489	0.001753
		2 x 4	0.0001029	0.0001079	3.895e-05
		4 x 4	0.01237	0.01197	0.005493
	64QAM	2 x 2	0.006859	0.006822	0.006136
		2 x 4	0.0009449	0.0009531	0.0007289
		4 x 4	0.01765	0.01734	0.01487

Es importante tener en cuenta que al utilizar un arreglo de 4 x 4 para cada una de las combinaciones posibles se transmite el doble de flujos de información que con una antena 2 x 2, convirtiéndose en una ventaja ya que más información será envida en una transmisión. Según las tendencias de las Tablas 3.4 y 3.5 se puede apreciar que los valores de BER para estos arreglos de 4 x 4 son mayores que para los arreglos 2 x 2 debido a que se analiza un mayor flujo de datos.

3.5. TIEMPOS DE SIMULACIÓN DE LOS DETECTORES PARA ARREGLOS DE ANTENAS Y MODULACIONES.

En la Figura 3.10 se presentan los tiempos promedio de detección para cada detector, arreglo de antenas y modulación. Estos valores de tiempos de detección son menores en el detector ZF, aumentan para MMSE y con el detector ML los tiempos de simulación se elevan considerablemente. Para el detector ML el tiempo obtenido para la modulación 64QAM con un arreglo de antenas 4 x 4 es muy alto lo cual deja que los otros tiempos de detección, analizados para este detector, sean casi imperceptibles.

Para un mejor entendimiento de la Figura 3.10, se obtienen de la misma figura los valores de los tiempos para cada modulación, arreglo de antenas y detectores, los cuales se presentan en la Tabla 3.6. Aquí se puede apreciar que para el detector ZF los tiempos aumentan conforme se aumenta el número de antenas. También se tiene que para las diferentes modulaciones los tiempos son similares.

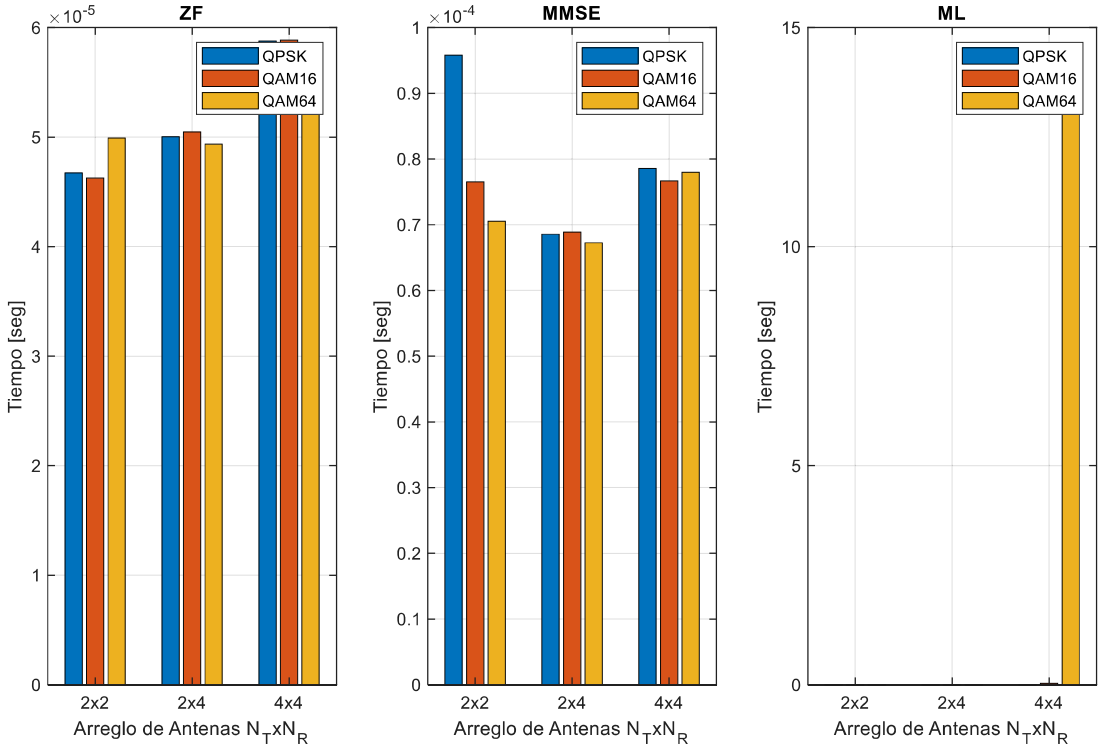

Figura 3. 10 Tiempos de detección para detector ZF, MMSE y ML, modulaciones QPSK, 16QAM y 64QAM, arreglos de 2 x 2, 2 x 4 y 4 x 4.

Para el detector MMSE se tienen tiempos ligeramente mayores para el arreglo de antenas de 4 x 4 con respecto al de 2 x 2. Aquí también los valores de tiempo para las diferentes modulaciones son bastantes similares, pero mayores en comparación al detector ZF.

Con ML los valores de tiempo son imperceptibles en la gráfica y no se obtiene ningún valor visible para los arreglos de 2 x 2 y 2 x 4. Los valores obtenidos para 16QAM y 64QAM con el arreglo de antenas 4 x 4 son de 14.2 [s] y 0.0331 [s] respectivamente. Estos tiempos con respecto a los de otros detectores y modulaciones son altos. Para el detector ML el tiempo de detección aumenta exponencialmente con el número de antenas y modulaciones, como se observa en la Tabla 3.6.

Tabla 3.6 Tiempos promedio de detección en segundos.

DETECTORES	MODULACIONES	2 x 2	2 x 4	4 x 4
ZF	QPSK	4.675e-05 [s]	5.004e-05 [s]	5.878e-05 [s]
	16QAM	4.627e-05 [s]	5.048e-05 [s]	5.885e-05 [s]
	64QAM	4.993e-05 [s]	4.935e-05 [s]	5.852e-05 [s]
MMSE	QPSK	9.582e-05 [s]	6.856e-05 [s]	7.859e-05 [s]
	16QAM	7.65e-05 [s]	6.889e-05 [s]	7.667e-05 [s]
	64QAM	7.054e-05 [s]	6.725e-05 [s]	7.8e-05 [s]
ML	QPSK	0.0004746 [s]	0.0004702 [s]	0.0008435 [s]
	16QAM	0.000876 [s]	0.0008479 [s]	0.0331 [s]
	64QAM	0.003352 [s]	0.003497 [s]	14.2 [s]

4. CONCLUSIONES

4.1 CONCLUSIONES

- La tecnología MIMO debido a sus múltiples antenas usadas tanto en la parte de transmisión como en la recepción logra varios beneficios para los sistemas de comunicación inalámbricos como mayor alcance de la comunicación, optimización de energía, mayores velocidades, mayor confiabilidad, etc.

- El limitado espectro radioeléctrico y las necesidades crecientes de comunicación inalámbrica hacen llamativo el uso de los sistemas OFDM. Este sistema se ha implementado en tecnologías antiguas y nuevas debido a sus múltiples prestaciones como tasas altas de transmisión, mayor eficiencia espectral, ortogonalidad e implementación de la FFT.

- MIMO – OFDM es un sistema robusto debido a la combinación de las características de cada tecnología ya que reduce el costo y complejidad de implementación, envía flujos de información independientes sobre cada una de las diferentes antenas (SM - MIMO) con la finalidad de mejorar la velocidad de datos. Además, la optimización del ancho de banda y detección de la señal.

- La detección de la señal es un proceso obligatorio en los sistemas SM – MIMO y por ende en los sistemas SM MIMO – OFDM. Este proceso de detección presenta complejidad ya sea computacional y de implementación. Los algoritmos implementados son: ZF, MMSE y ML; estos permiten la detección de la señal de datos enviada que atraviesan el canal multitrayecto y AWGN.

- El análisis teórico de la complejidad realizado para cada detector en el Capítulo 1 indica que, para los detectores lineales, el de menor complejidad es el ZF debido a que el detector MMSE presenta en su ecuación la varianza de ruido, lo que provoca más esfuerzo computacional. La complejidad de los detectores lineales es independiente del orden de modulación.

- El análisis de complejidad para el detector ML muestra que el esfuerzo computacional crece exponencialmente con el número de antenas y el orden de modulación. Los detectores lineales respecto al detector ML presentan bajo esfuerzo computacional desde un arreglo de antenas 2 x 2 y orden de modulación M = 4. Esto implica que el

detector ML es más complejo computacionalmente y por ende en la simulación de este para un número de antenas elevado.

- Para un Eb/N0 = 10, arreglo 4 x 4, modulación QPSK con el detector ML el BER obtenido es 0.00593, para el detector MMSE se tiene un BER de 0.0153 y con ZF el BER es igual a 0.01727. Estos valores de BER obtenidos indican que se tiene un menor BER para el detector ML, seguido por el detector MMSE y finalizando con el detector ZF, para todas las combinaciones realizadas se sigue la tendencia de los valores mencionados. El detector ML brinda una mejor detección de la señal que atraviesa el canal en un sistema MIMO – OFDM. Sin embargo, el costo computacional de este detector crece exponencialmente con el número de antenas y orden de modulación.

- Con la modulación QPSK se obtienen valores de BER menores a los de las otras modulaciones (16QAM y 64QAM) para un mismo arreglo de antenas y detector, esto se debe a que esta modulación presenta mayor robustez y menor susceptibilidad al ruido que se adhiere a la señal.

- Conforme se aumenta el número de antenas los valores de BER crecen, esto se da debido a que no existe redundancia en la transmisión de datos. Esto se debe a que se coloca diferentes partes de la señal en los diferentes flujos de información, es decir se multiplexa la señal, lo que garantiza una mejor velocidad, pero no mejora la confiabilidad del sistema (SM MIMO – OFDM).

- En la simulación de todos los detectores, modulaciones y arreglo de antenas, para la configuración 2 x 4, en todos los casos, presenta un mejor BER. Esto se debe a que existe redundancia de la información (diversidad) ya que envía dos flujos de datos por sus dos antenas transmisoras y recibe cuatro por sus cuatro antenas receptoras logrando mejorar la señal y por ende el BER disminuye.

- Para los detectores y modulaciones analizados, con arreglo de antenas 2 x 2 se presentan tiempos de detección menores a los realizados con arreglo de antenas 4 x 4. Por ejemplo, con el detector ZF, modulación QPSK se tiene para un arreglo de antenas 2 x 2 un tiempo de simulación de 4.675e-05 [s] y para un arreglo de antenas 4 x 4 el tiempo de simulación crece a 5.878e-05 [s], esta tendencia se cumple en las configuraciones analizadas.

- Con los detectores lineales se tiene que los tiempos de detección son cercanos para todas las modulaciones. Esto se debe a que sus ecuaciones son casi similares teniendo

en cuenta que estos detectores realizan la inversa de la matriz de canal y el detector MMSE añade la varianza de ruido a su ecuación. Por lo que se tiene que el detector ZF es el ligeramente más rápido de los detectores lineales.

- Para los detectores lineales el cálculo de la inversa de la matriz para un arreglo de 2 x 2 se resuelve más rápido que el cálculo de la inversa de una matriz para un arreglo de 4 x 4 en la detección. Esta rapidez del arreglo 2 x 2 frente al de 4 x 4 se aprecia en los tiempos de detección promedio, ya que sus valores de tiempo son menores.

- Para el detector ML el tiempo de detección crece exponencialmente. Si aumenta el número de antenas el tiempo también lo hace. Si aumenta el tamaño de la constelación a analizar, el tiempo también lo hace. Por ejemplo, para un arreglo 2 x 2, modulación QPSK el tiempo de detección es 0.0004746 [s] y para una modulación 64QAM el tiempo de detección aumenta a 0.003352 [s]. Para un arreglo de 4 x 4, modulación QPSK se tiene un tiempo de detección de 0.0008435 [s], con 16QAM el tiempo de detección es de 0.0331 [s] y con una modulación de 64QAM el tiempo de detección aumenta a 14.2 [s], mostrando un crecimiento exponencial en los tiempos de simulación.

4.2 RECOMENDACIONES

- El sistema MIMO – OFDM está compuesto de varios bloques para transmisión y recepción. Cada bloque cumple una función específica, y se instauran según los requerimientos del sistema a simular. Se recomienda establecer las necesidades básicas de la simulación y la adición o eliminación de los componentes innecesarios.

- Se recomienda para una mejor apreciación de las gráficas de Eb/No vs BER y resultados más fiables, un número de iteraciones alto, mayor a 10000.

- De acuerdo con el estándar 802.11a utilizado en este proyecto, se recomienda una elección de PDP adecuada, teniendo en cuenta que los retardos normalizados no deben exceder el intervalo de guarda porque ocasionaría la aparición de ISI.

- Se puede ampliar el proyecto con la adición de etapas de codificación FEC, sincronización, para así realizar la construcción del sistema y los detectores analizados en un ambiente de laboratorio y obtener resultados reales que verifiquen las respuestas obtenidas de las simulaciones realizadas.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] L. JIMÉNEZ, J. PARRADO, C. QUIZA y C. SUÁREZ, Modulación multiportadora OFDM, *Ingeniería,* vol. 6, nº 2, pp. 30-34, 2001

[2] J. Kim, C. G. Kang, W.-y. Yang y Y. S. Cho, MIMO-OFDM Wireless Communications with MATLAB, JOHN WILEY & SONS INC, 2010.

[3] H. K. Bizaki, MIMO Systems, Theory and Applications, Croatia: InTech, 2010.

[4] E. Biglieri, R. Calderbank, A. Constantinides, A. Goldsmith, A. Paulraj y H. V. P. Poor, MIMO Wireless Communications, New York : Cambridge, 2007.

[5] G. Tsoulos, Mimo System Technologyfor Wireless Communications, New York: Taylor & Francis Group , 2006.

[6] M. Viswanathan, «GaussianWaves Signal Processing Simplified,» GaussianWaves, Agosto 2014. [En línea]. Available: https://www.gaussianwaves.com/2014/08/mimo-diversity-and-spatial-multiplexing/. [Último acceso: Marzo 2019].

[7] R. Prasad, OFDM for Wireless Communications Systems, Boston: Artech House, Inc. , 2004.

[8] Wireless OFDM Systems, 2005.

[9] V. A. Masero Rubio, «Evaluación de la Modulación OFDM en Sistemas de Comunicaciones Ópticas» Universidad de Sevilla, Sevilla, 2014.

[10] A. D. Pelakar y P. V. Ingole, «Ofdm System Using FFT and LFFT,» *International Journal of Advanced Research in Computer Science and Software Engineering,* vol. 3, nº 12, p. 679, 2013.

[11] A. L. INTINI, «Orthogonal Frequency Division Multiplexing for Wireless Networks Standard IEEE 802.11a,» University of California , Santa Barbara, 2000.

[12] G. L. STÜBER, J. R. BARRY, S. W. MCLAUGHLIN, G. LI, M. A. INGRAM y T. G. PRATT, «Broadband MIMO-OFDM Wireless Communications,» *IEEE,* vol. 92, nº 2, p. 294, 2004.

[13] C. V. Vargas, W. E. Lopez y C. F. d. Rocha, «Sistemas de Comunicación Inalámbrica MIMO - OFDM,» *GPqCom - UFSC ,* vol. 3, nº 4, 2007.

[14] J. Garg, K. Gupta y P. K. Ghosh, «Performance Analysis of MIMO Wireless Communications over Fading Channels - A Review,» *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering ,* vol. 2, nº 4, 2013.

[15] I. Santamaría, «MIMO Detection,» UC, Santander.

[16] L. Bai y J. Choi, Low Complexity MIMO Detection, New York: Springer, 2012.

[17] X. Zhu y R. D. Murch, «Performance Analysis of Maximum Likelihood Detection in a MIMO Antenna System,» *IEEE,* vol. 2, nº 5, p. 191, 2002.

[18] M. Chouayakh, A. Knopp y B. Lankl, «Low Complexity Two Stage Detection Scheme for MIMO Systems,» *IEEE,* 2007.

[19] I. B. Collings, M. R. Butler y M. R. McKay, «Low Complexity Receiver Design for MIMO Bit-Interleaved Coded Modulation,» *IEEE,* p. 16, 2004.

[20] R. Hernández, C. Fernández y P. Bautista, Metodología de la investigación, Mexico: Mc Graw Hill, 2010.

[21] T. Cooklev, Wireless Communication Standards, New York: IEEE, 2004.

[22] MathWorks, [En linea]. Available: https://la.mathworks.com/help/. [Último acceso: Mayo 2019].

[23] Jos, «Mathworks,» Mathworks, Mayo 2019. [En línea]. Available: https://la.mathworks.com/matlabcentral/fileexchange/10064-allcomb-varargin?s_tid=srchtitle. [Último acceso: Mayo 2019].

6. ANEXOS

ANEXO A. Código del programa realizado en Matlab (Anexo digital).

ANEXO B. Tabla de comandos utilizados en el programa de Matlab

ANEXO A

Anexo digital.

ANEXO B

Tabla Anexo B. Comandos utilizados en el desarrollo del programa [22].

CÓDIGO	COMANDO	DESCRIPCIÓN	EJEMPLO DE USO
PROGRAMA Detectores2	clear	Elimina variables y funciones de la memoria del sistema.	clear all
	clc	Limpia la ventana de comandos.	clc
	close	Cierra figuras.	close all
	length	Obtiene el número de elementos de un vector.	length (nt)
	comm.OFDM Modulator	Modulador OFDM.	H = COMM. OFDMModulator (nombre,valor)
	comm.OFDM Demodulator	Demodulador OFDM.	H = COMM. OFDMDemodulator (nombre,valor)
	datestr	Convierte fecha y hora en formato de cadena.	datestr(t)
	strcat	Concatena cadenas horizontalmente.	strcat(s1,…,sN)
	datetime	Matrices que representan puntos en el tiempo.	datetime("today")
	tic	Iniciar cronometro.	tic
	sprintf	Escribe datos en un archivo de texto.	sprintf('ANT_ %ux%u', NT, NR)
	disp	Muestra el valor de una variable.	disp(x)
	fprintf	Da formato a los datos y muestra los resultados en la pantalla.	fprintf('\t%s\n', det)
	comm.QPSK Modulator	Modulador QPSK.	H = COMM. QPSKModulator
	comm.QPSK Demodulator	Demodulador QPSK.	H = COMM. QPSKDemodulator
	qammod	Modulación de amplitud en cuadratura.	qammod(x,M, 'UnitAveragePower', true)
	qamdemod	Demodulación de amplitud en cuadratura.	qamdemod(y,M, 'UnitAveragePower', true);
	strcmpi	Compara cadenas.	strcmp(s1, s2)
	log 2	Logaritmo base 2.	log2(M)
	toc	Tiempo transcurrido de lectura de cronómetro.	toc
	save	Guarda variables de trabajo en un archivo.	save(fullName, 'DATA')
	clearvars	Limpia variables de la memoria.	clearvars -except DATA
FUNCIÓN mimoOfdm-SingleTest3	waitbar	Crea o actualiza el cuadro de diálogo la barra de espera.	waitbar(0, 'Please wait ...')

	afterEach	Especifica una función para ejecutar cada vez que esta recibe nuevos datos.	afterEach(q, @disp)
	zeros	Crea una matriz de ceros.	zeros(n)
	randi	Genera valores aleatorios enteros (entre [min max]) para una matriz de tamaño M x N.	randi([0 12],M,N);
	conv2	Devuelve la convolución bidimensional de matrices.	conv2 (A, B)
	awgn	Ruido blanco Gaussiano.	awgn(in,snr)
	feval	Evalúa una función utilizando su nombre y argumentos.	feval(fun, x1,...,xM)
	biterr	Calcula el número de bit errados y la tasa de bit errados.	biterr(x,y)
FUNCIÓN RayleihChannel	randn	Genera una matriz M x N de números aleatorios normalmente distribuidos.	rand (M, N)
	fft	Obtiene la transformada rápida discreta de Fourier de la matriz x con N puntos.	fft(x,N)
	sqrt	Se obtiene la raíz cuadrada de x.	sqrt(X)
DETECTORES	pinv	Devuelve la Pseudoinversa de Moore-Penrose de la matriz.A.	pinv(A)
	inv	Calcula la inversa de la matriz cuadrada X.	inv(X)
	repmat	Devuelve una matriz que contiene n copias de A.	repmat(A,n)
	size	Devuelve la longitud de la dimensión dim de A.	size(A,2)
	vecnorm	Devuelve la norma o norma euclidiana de A.	vecnorm(A)
	min	Devuelve los elementos mínimos de una matriz.	min(A)
FUNCIÓN graficarDetectores	uigetfile	Abre un cuadro de diálogo que muestra los archivos en la carpeta actual.	uigetfile('*.m')
	fullfile	Construye el nombre completo del archivo a partir de partes.	fullfile(path,file)
	load	Cargar variables del archivo en el espacio de trabajo.	load(filename)
	varargin	Lista de argumentos de entrada de longitud variable.	varargin{1}
	cell	Devuelve una matriz n-by-n de matrices vacías.	cell(n)

	subplot	Divide la figura actual en una cuadrícula de m por n y crea ejes en la posición especificada por p.	`subplot(m,n,p)`
	semilogy	Crea una figura usando una escala logarítmica base 10 en el eje Y y una escala lineal en el eje X.	semilogy(Y)
	hold	Mantiene la figura actual sin eliminar las gráficas existentes.	hold on
	grid	Muestra las líneas de cuadrícula principales para los ejes o gráficos actuales.	grid on
	xlabel	Etiqueta el eje x.	xlabel(‘txt’)
	ylabel	Etiqueta el eje y.	ylabel(‘txt’)
	legend	Crea una leyenda con etiquetas descriptivas para cada serie de datos trazada.	legend(‘label1’,…,’labelN’)
	title	Agrega el título especificado a los ejes o el gráfico.	title(‘txt’)
	axis	Especifica los límites de los ejes actuales.	axis ([0 20 1e-4 1e0])
	figure	Crea una nueva ventana de figura mediante valores predeterminados.	figure (1)
	bar	Crea un gráfico de barras con una barra para cada elemento de y.	bar(y)
	xticklabels	Establece las etiquetas de marca del eje x para los ejes actuales.	xticklabels(labels)

ORDEN DE EMPASTADO