

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL



Facultad de Ingeniería en Electricidad y Computación

“SIMULACIÓN Y OPTIMIZACIÓN DE MIMO SYSTEMS USANDO
BEAMFORMING A TRAVES DEL SINGULAR VALUE DECOMPOSITION ”

TESINA DE SEMINARIO

Previa a la obtención del título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Luis Alfredo Zambrano Vega

Carlos Alberto Castillo Arias

GUAYAQUIL – ECUADOR

2011

AGRADECIMIENTO

A todas las personas que han colaborado con la realización de este trabajo en especial a nuestras familias y a nuestro director del proyecto, PhD. Hernán Córdova por su continuo apoyo.

DEDICATORIA

A Dios por ser mi guía y siempre estar presente, a mi familia, los que se han ido, los que están y los que vinieron, a mi querida Madre, mi tía Lucky, mis hermanos, Jazmin, David, Maury, mis sobrinos, Jasmine, Gaby, David Enrique. Ustedes son el tesoro más grande que puedo tener, son mi fuerza, mi inspiración y motivación constante para superarme cada vez más.

A mi segunda familia Baca Izquierdo y a todos mis amigos por su apoyo, preocupación y siempre creer en mí.

Luis Alfredo

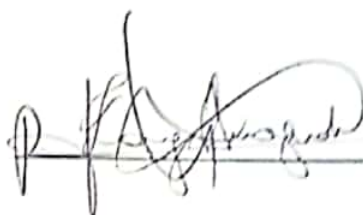
DEDICATORIA

A mis padres, quienes nunca dejaron de apoyarme y fueron siempre mi principal fortaleza para seguir adelante. Me han dado todo lo que soy como persona, mis valores, mis principios, mi perseverancia y mi empeño, y todo ello con una gran dosis de amor. Son los mejores y todos los días agradezco a Dios que los haya puesto en mi vida.

A mis amigos, ha sido un honor haber recorrido este duro camino con ustedes. Gracias por todos esos buenos momentos y espero que esta amistad dure por siempre.


Carlos

TRIBUNAL DE SUSTENTACIÓN



PhD. Hernán Córdova

PROFESOR DE SEMINARIO



MSc. Juan Carlos Avilés

PROFESOR DELEGADO DEL DECANO

DECLARACIÓN EXPRESA

“La responsabilidad del contenido de esta Tesina de Seminario, nos corresponde exclusivamente a nosotros; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL”

(Reglamento de Graduación de la ESPOL)

Luis Alfredo Zambrano Vega

Carlos Alberto Castillo Arias

RESUMEN

CAPÍTULO 1

El marco teórico del proyecto, las definiciones fundamentales y funciones básicas de los Sistemas MIMO que poseen varias antenas de transmisión y recepción. Se explica claramente lo que son y sus bondades, también presentamos a la matriz del canal, la diversidad de antena y la técnica de OFDM que permite reducir el ancho de banda y ayuda a combatir con la interferencia entre símbolos.

CAPÍTULO 2

Tratamos las interferencias que se presentan en los sistemas de transmisión inalámbricas MIMO y los que se tratarán de combatir en este estudio, mitigando dichas interferencias con la ayuda de varios métodos, algoritmos y artificios matemáticos. Entre los problemas que se presentan está el ISI (interferencia Intersimbólica), ICI (interferencia intercanal) y la interferencia co-canal o el crosstalk que se produce entre distintos transmisores que operan a una misma frecuencia. También se explican los análisis y comparaciones que se esperan desarrollar en este estudio, que son gráficas de rendimiento como BER vs SNR.

CAPÍTULO 3

Nos permite analizar las soluciones que se aplicaran para combatir problemas de interferencias y optimizar nuestro sistema.

Se realiza el modelamiento del Sistema MIMO estableciendo las condiciones y parámetros a usar en este estudio, se establece que la información del canal o CSI es totalmente conocida, para así poder aplicar el beamforming (Precodificación) con el objetivo de diagonalizar el canal permitiendo que

este se descomponga en varios canales SISO paralelos entre sí, eliminando interferencias ICI y facilitando la carga adaptiva de bits por cada subportadora. También se realiza un análisis matemático de este modelamiento, del beamforming, el SVD y explicamos cómo añadiendo un intervalo de guarda permitimos que los símbolos OFDM no se superpongan e interfieran entre sí.

También se detalla la capacidad del sistema en el sistema MIMO, que es la velocidad de transmisión máxima alcanzable, un parámetro fundamental para analizar el rendimiento de nuestro sistema.

Y damos un preámbulo de lo que es la carga adaptiva que nos ayudará para alcanzar un mayor velocidad de transmisión que se basa en el conocimiento de ganancia de los subcanales.

CAPÍTULO 4

En este capítulo se detalla el proceso de la simulación. Nuestros datos a transmitir y el canal a utilizar son creados de forma aleatoria. Se simulan todos los procesos de modulación y demodulación, y los procedimientos para la optimización y transformaciones para el mejor manejo de la información. También se calcula información que nos dará características del desempeño del sistema utilizado. La simulación se realiza ingresando las características del sistema MIMO que se desea simular por medio de una interfaz grafica.

Finalmente analizamos las graficas de rendimiento obtenidas, BER vs SNR con diferentes esquemas y distintas combinaciones de antenas, los efectos multicaminos en la transmisión y la capacidad del canal para determinar las máximas velocidades que puede alcanzar nuestro sistema con las soluciones previamente planteadas.

INDICE GENERAL

Resumen	VII
Índice General	IX
Abreviaturas	XII
Introducción	XIV

CAPÍTULO 1

1. Marco Teórico	
1.1. Definición de Sistemas MIMO	1
1.2. Matriz del Canal en Sistemas MIMO	2
1.3. Diversidad Espacial en MIMO	4
1.4. OFDM	5
1.4.1. Funcionamiento de OFDM	7

CAPÍTULO 2

2. Descripción del Problema	
2.1. Interferencias presentes en el Sistema	9
2.2. BER y SNR	10
2.3. Objetivo del Proyecto	11

CAPÍTULO 3

3. Comprensión de Soluciones	
3.1. Modelamiento del Sistema	12
3.1.1. Modelamiento Matemático del Canal MIMO.....	13
3.1.2. Señal Recibida en Sistema MIMO	16
3.2. Intervalo de Guarda - OFDM	17
3.3. Beamforming.....	19
3.4. Capacidad del Canal	20
3.5. Singular Value Decomposition.....	22
3.6. Adaptive Loading	25

CAPÍTULO 4

4. Simulación del Sistema	
4.1. Modelamiento del canal.....	27
4.2. Modulación y Demodulación.....	30
4.3. Carga Adaptiva	34
4.4. Operaciones de Transformada de Fourier y su Inversa en OFDM	43
4.5. Interfaz Gráfica GUI	48
4.6. Análisis de Resultados.....	49

CONCLUSIONES Y RECOMENDACIONES

ANEXOS

BIBLIOGRAFÍA

ABREVIATURAS

WLAN	Wireless Local Area Network
LAN	Local Area Network
WIFI	Wireless Fidelity
WIMAX	Interoperabilidad mundial para acceso por microondas
IEEE	Institute of Electrical and Electronics Engineers
MIMO	Multiple Input Multiple Output
SISO	Single Input Single Output
LOS	Línea de vista
NLOS	Sin línea de vista
OFDM	Orthogonal Frequency Division Multiplexing
ADSL	Asymmetric Digital Subscriber Line
BPSK	Binary Phase Shift Keying
QAM	Quadrature amplitude modulation
ASK	Amplitude-shift keying
IFFT	Transformada rápida de Fourier inversa
FFT	Transformada rápida de Fourier
ISI	Interferencia Inter-simbólica
ICI	Interferencia Inter-subcanal
CCI	interferencia Co-canal
SNR	Relación Señal Ruido
BER	Tasa de error de bit
dB	Decibelio
E_b	Energía por bit
N₀	Densidad espectral de ruido
SVD	Singular value decomposition
GI	Intervalo de Guarda

AWGN	Ruido Blanco Aditivo Gausiano
ZF	ZeroForcing
Mbps	Megabits por segundo
Hz	Hertzio
kHz	kilohertz
Mhz	megahertz

INTRODUCCIÓN

En nuestra vida cotidiana, tecnologías como Wireless Local Area Network (WLAN) ha ganado mucha importancia gracias al desarrollo en gran escala de diversos dispositivos portátiles, como también por la libertad de poder estar conectado sin cables. Las redes inalámbricas han evolucionado tanto que hoy en día muchos usuarios o empresas no podrían vivir sin sus bondades.

Compartir datos, voz, video es el principal propósito dentro de una red inalámbrica. En la actualidad existe alto contenido digital de calidad, que generalmente posee un mayor tamaño de archivo, lo cual provocaría una limitación en el intercambio de información si se posee una conexión lenta. Razón por la cual los consumidores siempre piden más, mayor velocidad y mayor cobertura, por lo que siempre se está apuntando al desarrollo de nuevas normas y mejoras al estándar IEEE 802.11¹, para la tecnología WLAN.

Una de las mejoras obligatorias en la capa física es el uso de la tecnología con múltiples antenas de transmisión y recepción, o mejor conocido por sus siglas en ingles Multiple Input-Multiple Output (MIMO).

¹ El estándar IEEE 802.11 define el uso de los dos niveles inferiores de la arquitectura OSI (capas física y de enlace de datos), especificando sus normas de funcionamiento en una WLAN. IEEE 802.11n es una norma que mejora los últimos estándares 802.11 añadiendo múltiples antenas de entrada y salida (MIMO).

CAPÍTULO 1

1 MARCO TEÓRICO

1.1 Definición de Sistemas MIMO

MIMO es un sistema de transmisión en el cual el transmisor y receptor poseen varias antenas físicamente separadas. En el formato de transmisión inalámbrica tradicional la señal se ve afectada por reflexiones, lo que ocasiona degradación o corrupción de la misma y por lo tanto ocurre pérdida de datos, los sistemas MIMO son una solución a este problema ya que estos aprovechan la propagación multicamino², con la finalidad de incrementar la habilidad del receptor para recobrar los mensajes de la señal modulada, aumentando la cobertura y fiabilidad del enlace, mejorando la velocidad de transmisión y reduciendo la tasa de error.

² Se refiere al fenómeno por el cual el receptor recibe la señal transmitida de dos o más trayectorias y distancias distintas (incluyendo ecos o imágenes de la señal original). Estas rutas pueden ser el resultado de las reflexiones en edificios, montañas u otras superficies reflectantes como el agua, etc...

Este tipo de sistemas es ampliamente utilizado en las comunicaciones inalámbricas como por ejemplo en los routers (con estándar IEEE 802.11n) que usamos a diario. MIMO es también usado en telefonía celular como una solución para aumentar el throughput o el ancho de banda efectivo por usuario por ejemplo utilizar sistemas 2x2 (dos en el receptor, dos en el emisor) como en la Figura 1.1, permite transferencias de hasta 300Mbps, y en 4x4 hasta 600 Mbps.

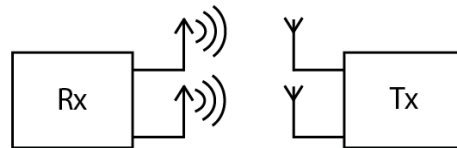


Figura 1.1 Sistema Mimo de 2x2

1.2 Matriz del Canal en Sistemas MIMO

En un sistema MIMO de M antenas transmisoras y N antenas receptoras, sistema $M \times N$, se generan MN subcanales, la respuesta del subcanal desde cualquier j , con $j = 1, 2, \dots, M$, a cualquier antenna i , con $i = 1, 2, \dots, N$, será la suma de reflexiones a lo largo de los múltiples caminos de propagación.

El canal MIMO es representado en forma matricial como se detalla a continuación:

$$H = \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1M} \\ h_{21} & h_{22} & \dots & h_{2M} \\ \vdots & \vdots & \dots & \vdots \\ h_{N1} & h_{N2} & \dots & h_{NM} \end{bmatrix} \quad (1)$$

Donde H es la matriz del canal y cada elemento h_{ij} representa el subcanal generado entre la antena transmisora j y la antena receptora i .

El canal MIMO descrito por la expresión (1) se representa en la Figura 1.2 en donde se puede apreciar las M antenas transmisoras, las N antenas receptoras y como se generan $M \times N$ subcanales.

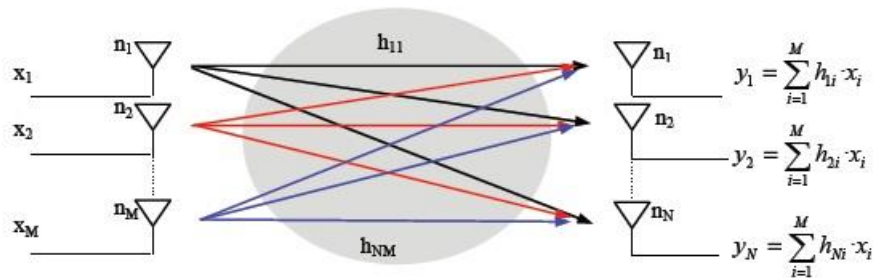


Figura 1.2 Esquema del canal MIMO $M \times N$.

Una manera sencilla de representar a un canal MIMO como se describe anteriormente es con una matriz del canal H de $i \times j$ elementos, cada elemento h_{ij} representa el canal generado entre la antena transmisora j y la antena receptora i , donde cada una de las antenas transmisoras envía distintas tramas de datos (color negro, rojo, azul) simultáneamente en la misma frecuencia para luego recombinar la información en la parte receptora.

Gracias a la propagación multicamino la información puede ser decodificada en el receptor ya que las señales no toman un camino directo sino que estas llegan al receptor por medio de diferentes trayectorias en diferentes intervalos de tiempo.

El total de la señal recibida por cada antena receptora será la suma de las señales de todas las antenas de transmisión.

1.3 Diversidad Espacial en MIMO

La diversidad espacial en sistemas MIMO mejora la calidad y la fiabilidad de un enlace inalámbrico. A menudo, especialmente en entornos urbanos y de interior, no hay clara línea de vista (LOS) entre el transmisor y el receptor. En estos casos la señal se refleja a lo largo de múltiples caminos antes de ser finalmente recibidos. Cada uno de estos rebotes puede introducir cambios de fase, retrasos, atenuaciones, e incluso las distorsiones que pueden interferir destructivamente entre sí al llegar a la antena receptora. La diversidad espacial es especialmente efectiva a mitigar estas situaciones multicaminos. Esto se debe a que múltiples antenas ofrecen un receptor de varias observaciones de la misma señal. Cada antena experimentará un entorno de interferencia diferente. Así, si una antena está experimentando un desvanecimiento profundo, es probable que otra tenga señal suficiente. Fundamentalmente, la diversidad de codificación provee de redundancia al sistema ya que envía varias copias a través de múltiples antenas de transmisión, a fin de mejorar la fiabilidad de la recepción de datos. Si una de las antenas receptoras no llega a recibir, las otras se utilizan para la decodificación de datos.

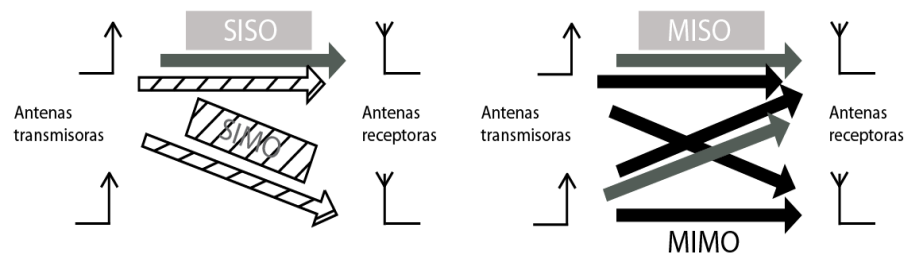


Figura 1.3 Variación del canal de comunicación con diferente combinación de antenas.

En los sistemas MIMO, la transmisión de un extremo a otro se conecta por medio de canales. En la Figura 1.3, un sistema SISO contiene un solo canal que conecta el transmisor y el receptor, mientras que un sistema MIMO de 2x2 puede tener hasta cuatro conexiones entre sí. En caso de que un canal se viera afectado por la interferencia, cualquier otro se encarga de que los datos puedan ser transmitidos de forma segura, por lo que cada canal se encarga de aumentar la relación señal ruido (SNR).

1.4 OFDM

Orthogonal Frequency Division Multiplexing (OFDM) es una técnica robusta que se ha aplicado en muchos sistemas de comunicación digital de alta velocidad que requieren un gran ancho de banda, incluyendo transmisión a través de cables de cobre, o de transmisión inalámbrica a través de canales con desvanecimiento de frecuencia selectiva tales como ADSL, televisión digital, radiodifusión, etc....

Esta técnica lo que hace es particionar la señal banda base en varias sub-bandas, y en vez de transmitir la señal mediante la modulación de una única portadora, la señal se transmite bajo múltiples portadoras ortogonales en frecuencia para que no se superpongan entre sí, lo que permite reducir el ancho de banda total requerido.

La Figura 1.4 muestra la comparación entre un técnica multiportadora convencional y la técnica OFDM, donde claramente se aprecia el ahorro de ancho de banda al transmitir con múltiples portadoras ortogonales.

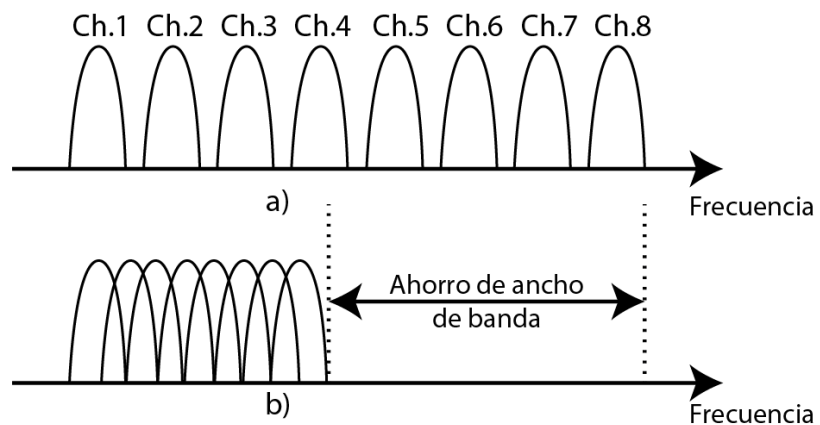


Figura 1.4 a) Técnica Multiportadora,
b) Modulación con portadoras ortogonales convencionales.

1.4.1 Funcionamiento de OFDM

Para generar una señal OFDM con éxito, cada espaciamento entre las subportadoras debe ser cuidadosamente separado para mantener la ortogonalidad.

Luego cada subportadora es asignada los bits de datos necesarios para transmitir, un distinto número de bits se puede modular en cada subportadora dependiendo de la modulación, por ejemplo, Binary Phase Shift Keying (BPSK), QAM, ASK, etc.

La Figura 1.5 muestra los 3 componentes importantes en la transmisión OFDM tanto para el transmisor como el receptor.

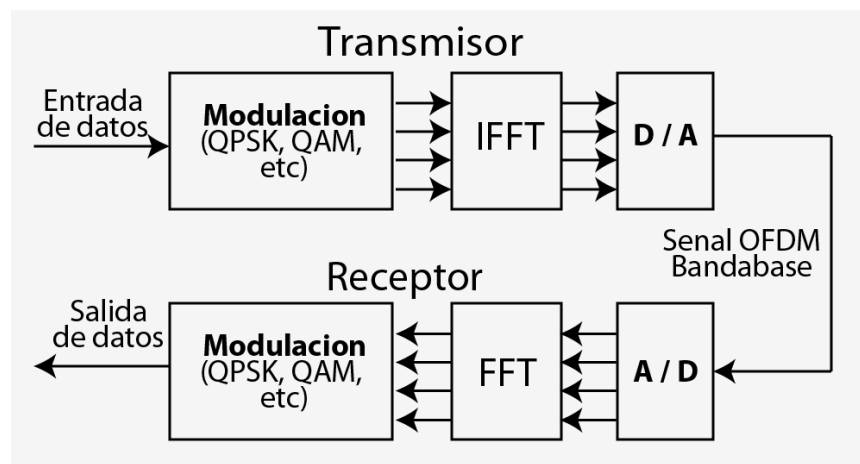


Figura1.5 Diagrama de flujo que muestra la transmisión OFDM

Después de que cada subportadora es modulada, y como dicha modulación de los bits de datos se realiza en el dominio de la frecuencia, es necesario transformar este espectro de nuevo en el dominio del tiempo para la transmisión práctica de la señal. Para este proceso utilizamos la transformada rápida de Fourier

inversa (IFFT), un método muy eficiente que puede garantizar la ortogonalidad entre cada subportadora.

Después que la IFFT genera una señal para ser transmitida en formato digital, de tiempo y discreta, la convertiremos en su representación analógica con la ayuda de un convertidor digital analógico (D / A), para luego proceder con la transmisión.

El receptor, es sólo una operación inversa del transmisor, como se muestra en la Figura 1.4 anterior, la antena en el receptor posee un convertidor analógico digital (A / D), para poder reconstruir la señal analógica recibida en su respectiva representación digital, y en lugar de la operación IFFT, se utiliza la transformada rápida de Fourier (FFT) para encontrar el equivalente del espectro de frecuencias de la señal en el dominio del tiempo.

IFFT se define como:

$$x(n) = \sum_{k=0}^{N-1} x(k) \cdot e^{-\frac{j2\pi kn}{N}}$$

Mientras que FFT está expresado como:

$$x(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-\frac{-j2\pi kn}{N}}$$

Donde,

N es el periodo de la señal.

$x(n)$ es la señal con N muestras periódicas que se obtiene en dominio del tiempo.

$x(k)$ es la señal con N muestras periódicas que se obtiene en dominio frecuencial.

CAPÍTULO 2

2. DESCRIPCIÓN DEL PROBLEMA

2.1 Interferencias presentes en el Sistema

Uno de los principales problemas que se presentan en el sistema de transmisión inalámbrica MIMO, es la interferencia entre símbolos o Inter-simbólica (ISI), la interferencia Inter-subcanal (ICI) y la interferencia Co-canal (CCI).

El ISI es una forma de distorsión de la señal, se produce en un solo canal cuando un símbolo es interferido por cualquiera de su predecesor o su sucesor. Esto es un fenómeno no deseado ya que su presencia en el sistema introduce errores al receptor provocando una comunicación menos confiable.

Mientras el ICI se produce cuando flujos de datos independientes en dos o más canales distintos interfieren entre sí, en los sistemas MIMO los múltiples subcanales que se generan por cada antena en el transmisor y receptor, son los responsables de dicha interferencia.

Otro tipo de fenómeno no deseable presente en los sistemas MIMO es la interferencia co-canal o también conocido como el crosstalk³ entre distintos transmisores que operan a una misma frecuencia.

Este tipo de interferencia ocurre cuando la misma frecuencia de portadora de distintos transmisores, llega a un mismo receptor en el mismo tiempo, limitando la capacidad de la red y aumentando el BER.

2.2 BER y SNR

La Tasa de Error de Bit (BER) es una de las técnicas más comunes para analizar el rendimiento de un sistema de telecomunicaciones. Una representación matemática simple se muestra a continuación:

$$BER = \frac{\text{Total de bits erróneos recibidos}}{\text{Total de bits originalmente transmitidos}} \quad (2)$$

Por lo tanto cuanto más pequeño sea el BER, mejor será el rendimiento, y a su vez el peor valor posible sería si $BER = 1$.

$$SNR = \frac{E_b}{N_o} \quad (3)$$

E_b/N_o es una medida de relación entre la energía por bit con respecto a la densidad espectral del ruido, un parámetro

³ Fenómeno en que una señal adyacente genera un efecto indeseable en el canal.

importante en comunicaciones digitales y transmisión de datos que normalmente es expresada en decibelios [dB], mientras mayor sea dicha relación, mayor cantidad de datos podrán ser transmitidos en la señal y ser recibidos con éxito en el receptor.

2.3 Objetivo del Proyecto

El objetivo principal de este proyecto es demostrar que mediante diferentes métodos incluyendo la técnica de beamforming y su precodificación basada en SVD, nuestro sistema MIMO será capaz de obtener mejoras de capacidad y SNR, a su vez que se logra disminuir el BER, optimizando el canal y cancelando interferencias presentes en el sistema como las detalladas anteriormente.

Se usará una plataforma de software para simular la transmisión inalámbrica y procederemos a discutir los resultados haciendo una comparación con otras técnicas existentes como Zero Forcing, así analizaremos su rendimiento y mejoras.

CAPÍTULO 3

3. COMPRENSIÓN DE SOLUCIONES

3.1 Modelamiento del Sistema

Para la realización de nuestro proyecto, nos planteamos el siguiente escenario:

En este estudio usaremos un sistema MIMO $M \times N$, con igual número de antenas receptoras y transmisoras, nos enfocamos en el downstream y asumiremos que la Información del Estado de Canal (CSI) se encuentra disponible en ambos extremos y la matriz beamforming es perfectamente conocida en el transmisor con delay respectivo. En dicho caso, los vectores de beamforming son obtenidos a través de la descomposición en valores singulares (SVD) del canal que se analizará posteriormente en la sección 3.5.

La Información del Estado de Canal (CSI) es muy importante por lo que se considerará un Canal FEEDBACK ideal entre estación base y usuario.

También consideraremos un canal plano tipo Rayleigh, el cual nos indica que no existe línea de vista (NLOS) entre el transmisor y el receptor. El ser plano implica que todas las bandas de frecuencias sufren la misma magnitud de fading o atenuación. Además, consideraremos que nuestro canal variará en el tiempo durante la transmisión de un símbolo hasta la llegada al receptor, por lo que nuestro modelo funcionará de forma adaptiva.

3.1.1 Modelamiento Matemático del Canal MIMO

Para una mejor comprensión en este caso procederemos a analizar un sistema MIMO con una configuración de antenas de 2x2.

Se modela matemáticamente nuestro canal MIMO basándonos en la ecuación:

$$y_n = \sum_{l=0}^{L-1} H_l x_{n-1} + n_n \quad (4)$$

Donde,

y_n , x_n , n_n son salida, entrada y ruido respectivamente.

H_l es la ganancia del canal

L es el periodo de símbolo de retraso de propagación

Modelaremos la matriz del canal como la suma de una constante con una matriz LOS, y una variable con una matriz NLOS. Este modelo escrito por Thomas y Tokunbo [11], contempla la caída exponencial de potencia por causa de los retardos. El modelamiento lo realizamos con la siguiente ecuación:

$$H_l = \sqrt{P_L} \left(\sqrt{\frac{K}{K+1}} \begin{bmatrix} e^{j\phi_{11}} & e^{j\phi_{12}} \\ e^{j\phi_{21}} & e^{j\phi_{22}} \end{bmatrix} + \sqrt{\frac{1}{K+1}} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \right) \quad (5)$$

Donde,

P_L es la potencia total del canal l

K es el factor Rician

$e^{j\phi_{ij}}$ son los elementos fijos de la matriz LOS

X_{ij} son los coeficientes de una matriz Rayleigh NLOS

Esta fórmula la demostramos generando una función para un desvanecimiento tipo Rician. Tomamos el coeficiente h de un canal con desvanecimiento tipo Rician, de frecuencia no selectiva con potencia P y factor Rician K .

$$h = \sqrt{\frac{P}{K+1}} \cdot (\sqrt{K} + \alpha) \quad (6)$$

En la suma constan las partes reales e imaginarias, en nuestro ejemplo representan los casos con LOS y NLOS respectivamente. La parte NLOS está representada por α . Despejando (6) obtenemos:

$$h = \sqrt{P} \left(\sqrt{\frac{K}{K+1}} + \sqrt{\frac{1}{K+1}} \cdot \alpha \right) \quad (7)$$

Las partes $\sqrt{\frac{K}{K+1}}$ y $\sqrt{\frac{1}{K+1}}$ son conocidas como parámetros de la distribución Rician. El factor K representa la razón de potencia entre los caminos directos e indirectos. Por último, en este estudio al trabajar con un sistema multicaminos, las partes que viajan con LOS y NLOS vienen representadas de forma matricial, generando la expresión (5) que queríamos demostrar:

$$H_l = \sqrt{P_L} \left(\sqrt{\frac{K}{K+1}} \begin{bmatrix} e^{j\phi_{11}} & e^{j\phi_{12}} \\ e^{j\phi_{21}} & e^{j\phi_{22}} \end{bmatrix} + \sqrt{\frac{1}{K+1}} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \right) \quad (5)$$

Debido a que en nuestro escenario asumimos NLOS, asignamos al factor Rician K un valor igual a 0.

Por ende la expresión que forma nuestro canal solo usará los componentes NLOS, por lo que la expresión (5) queda simplificada como se muestra a continuación:

$$H_l = \sqrt{P_l} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \quad (8)$$

3.1.2 Señal recibida en Sistema MIMO

La señal que recibe la antena receptora 1 es:

$$y_1 = h_{11}x_1 + h_{12}x_2 + n_1$$

$$y_1 = [h_{11} \quad h_{12}] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + n_1$$

La señal que recibe la antena receptora 2 es:

$$y_2 = h_{21}x_1 + h_{22}x_2 + n_2$$

$$y_2 = [h_{21} \quad h_{22}] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + n_2$$

Por lo tanto la señal total recibida es:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}$$

Y matricialmente la podemos representar como:

$$y = Hx + n \quad (9)$$

Donde y es la señal recibida, x la señal enviada, H es la matriz del canal, representa los distintos caminos de propagación que son creados por la cantidad de antenas existentes tanto en el transmisor como en el receptor y n es el ruido blanco aditivo Gaussiano (AWGN) y varianza σ^2 .

3.2 Intervalo de Guarda - OFDM

Una de las ventajas del uso de la técnica multiportadora OFDM, es su capacidad para combatir el ISI, causado por la propagación multicamino, básicamente se trata de una naturaleza de la modulación multiportadora, donde por lo general poseen un periodo más largo entre cada símbolo de datos, mitigando eficazmente el problema de la interferencia entre símbolos. Y este puede mejorarse aún más mediante la introducción de un período adicional entre dos símbolos adyacentes con el uso de un intervalo de guarda (GI: Guard Interval) como se muestra en la Figura 3.2.

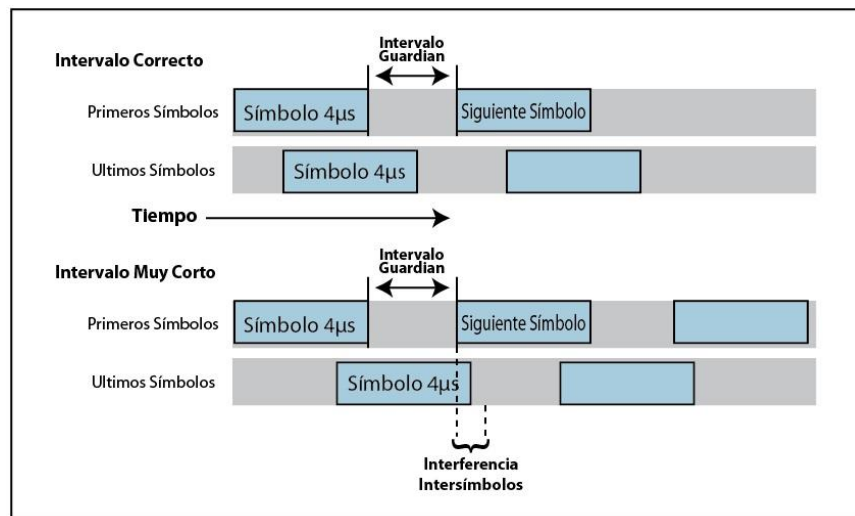


Figura 3.2 Implementación de Intervalo de Guardia en OFDM.

El GI permite que las múltiples copias de la señal que llegaron al receptor en un momento posterior (a través de diferentes trayectorias), se extingan antes de que el siguiente símbolo de datos sea recolectado.

En la Figura 3.2, podemos apreciar que si un intervalo de tiempo es elegido correctamente, el símbolo de datos que llegó más tarde todavía puede permanecer en la “zona segura” del GI, mientras que para un GI corto, el símbolo que llegó tarde no tiene tiempo suficiente para desaparecer antes de que llegue el siguiente símbolo de datos, por lo tanto estos símbolos interferirían entre sí provocando ISI, reduciendo la relación SNR.

Por esto el GI tiene que ser correctamente seleccionado para asegurar que los ecos de la señal, presentes debido a los delays en las trayectorias multicaminos, permanezcan dentro de dicho intervalo de tiempo GI.

3.3 Beamforming

Esta técnica consiste en la creación de arreglos que direccionen la señal en la transmisión y recepción, en otras palabras beamforming concentra la energía de la antena transmisora combinando las señales y enviándolas en una sola dirección. Por lo que una transmisión direccionada a un ángulo y dirección específica en comparación con una transmisión omnidireccional tendrá una importante ganancia de relación de señal con respecto al ruido (SNR).

Cuando se tiene Información del Estado del Canal (CSI) tanto en el transmisor como en el receptor, la técnica de beamforming es usada para maximizar la SNR en el receptor, este es implementado multiplicando el símbolo (s) por el vector de máxima ganancia perteneciente a la matriz beamforming V , esta matriz se la obtiene mediante Singular Value Decomposition (SVD) que se analizará posteriormente.

3.4 Capacidad del Canal

De acuerdo al teorema de Shannon, la capacidad del canal (velocidad de transmisión máxima alcanzable) en un sistema SISO se estima de la siguiente manera:

$$C = B \cdot \log_2 \left(1 + \frac{E_s}{N_o} \right) \quad (10)$$

Donde,

C = la capacidad del canal (bits/segundos)

B = es el ancho de banda ocupado

$$\frac{E_s}{N_o} = SNR$$

Como se puede apreciar, la capacidad del canal es proporcional al ancho de banda ocupado y SNR, lo que significa que si deseamos aumentar la capacidad, se puede aumentar cualquiera de estos dos parámetros. Pero el uso de espectro normalmente es asignado y fijado por autoridades regulatorias, por lo que alterar el ancho de banda estaría prohibido. Mientras el aumento de SNR se puede lograr aplicando un esquema de modulación más complejo, lo que aumentaría el costo de fabricación, sin embargo la capacidad solo aumentaría logarítmicamente.

Asumimos valores singulares iguales de la matriz de canal H , $\sigma_i^2 = \frac{NM}{M} = N, \forall i$. Debido a la igualdad de los valores singulares, la asignación de energía es claramente uniforme.

Y para difundir la energía de manera uniforme entre todos los transmisores se divide la energía de la señal para N .

A la energía de la señal la multiplicamos por su respectivo valor singular que fue obtenido mediante el proceso de SVD que será analizado en la sección 3.5.

$$C = \sum_{i=1}^N B \cdot \log_2 \left[1 + \frac{E_s \sigma_i^2(H)}{NN_o} \right] = \sum_{i=1}^N B \cdot \log_2 \left[1 + \frac{E_s}{N_o} \right] \quad (11)$$

σ = son los valores singulares de la matriz del canal H

Por ende como se detalla y desarrollando la sumatoria de la expresión (11), la capacidad del canal de un sistema MIMO se puede estimar de la siguiente manera:

$$C = N \cdot B \cdot \log_2 \left[1 + \frac{E_s}{N_o} \right] \quad (12)$$

N = es el número de parejas de antenas transmisor – receptor

La expresión (12) muestra que la capacidad del canal es proporcional al número de pares de antenas transmisor-receptor N que aumentan de forma lineal. Por lo tanto, pares de antenas pueden ser agregados para aumentar la capacidad del canal y la velocidad de transmisión de datos sin la necesidad de cambiar el ancho de banda del canal o el de usar una modulación compleja. Esto es una manera más práctica y económica para mejorar la red de transmisión inalámbrica.

3.5 Singular Value Decomposition

Recordemos que para que un sistema MIMO pueda aplicar adecuadamente el beamforming, este necesita conocer las propiedades del canal a usar. En consecuencia, el receptor determina la respuesta del canal H y luego lo ofrece como una retroalimentación al transmisor para determinar las propiedades de dicho canal.

En este estudio asumimos un conocimiento perfecto del canal, lo que significa que la respuesta del canal H se encuentra disponible tanto en el transmisor como en el receptor, por lo tanto dicho canal se puede descomponer en canales paralelos SISO⁴ que no interfieran entre sí, mediante Singular Value Decomposition (SVD).

El canal quasi-estatico plano tipo Rayleigh MIMO de $M \times N$ lo denotaremos como H , donde M es el número de antenas transmisoras y N es el número de antenas receptoras.

Entonces el SVD del canal H se puede escribir como:

$$H = UDV^T = [u_1 \ u_2 \ \dots \ u_n]D [v_1 \ v_2 \ \dots \ v_n] \quad (13)$$

⁴ Single Input Single Output, un sistema con una antena transmisora y una receptora.

Donde D es una matriz diagonal de $M \times N$, con valores singulares $\{\sigma_i\}_{i=1}^{\min(M,N)}$ de H en orden decreciente en la diagonal principal. U es una matriz unitaria de $M \times M$ cuyas columnas son vectores propios de HH^T y V^T es la transpuesta conjugada de la matriz unitaria V de $N \times N$ cuyas columnas son vectores propios de $H^T H$.

De la expresión (13) se obtiene la matriz unitaria V , que será usada por el transmisor para llevar a cabo el beamforming. El objetivo de la precodificación es encontrar el equilibrio óptimo entre la ganancia de señal y a su vez limitar la interferencia. Este filtro se encarga de cancelar la matriz unitaria V^T .

SVD maximiza la capacidad del canal, la matriz del canal H es diagonalizada después de remover las dos matrices unitarias V^T y U mediante la pre/post-multiplicación (precoding y shaping) en el transmisor y receptor respectivamente. Luego una trama de datos (con el power loading apropiado) podrá ser transmitido por cada valor singular de H .

La técnica de SVD es necesaria para asignar de manera óptima la energía y locación de bits mediante el adaptive loading (sección 3.6), y se aplica también para el filtro de precodificación de la señal antes del proceso IFFT y para el shaping después del FFT.

Primero consideramos el beamforming/precoding a través de SVD, la señal transmitida es:

$$x = V.s \quad (14)$$

Donde V es la matriz beamforming unitaria obtenida por medio de SVD y el vector s contiene los símbolos de información s_i de la señal transmitida que se refieren a un determinado tipo de modulación ej. 16QAM. El receptor del sistema puede comportarse de la siguiente manera:

$$y = U^H(UDV^H)V.s + U^Hn \quad (15)$$

$$y = D.s + n \quad (16)$$

Reemplazamos (13) y (14) en la ecuación $y = Hx + n$, para detectar la información transmitida en el receptor, decodificamos la señal x y la multiplicamos por la matriz unitaria U^H lo que cancela a U para así obtener (15), como sabemos que $UU^H = I$ y que $VV^H = I$, simplificamos obteniendo la ecuación (16), donde n sigue siendo AWGN y D la matriz diagonal de valores singulares.

Queda demostrado que los símbolos transmitidos por cada subcanal son ponderados por los valores singulares respectivos del canal H . La diagonalización del canal H permite que este se descomponga en varios canales SISO paralelos entre sí, eliminando la interferencia inter-subcanal (ICI).

3.6 Adaptive Loading

Nuestro modelo funcionará de forma adaptiva, con un canal que varía en el tiempo. Y se posee la ventaja de que hay un conocimiento perfecto del canal tanto para el transmisor como para el receptor por lo que el adaptive loading o carga adaptiva es de gran utilidad.

Se aprovecha la subportadora con mejor rendimiento, asignando de forma dinámica diferente número de bits de datos de acuerdo a los distintos desempeños de las distintas subportadoras, por ejemplo una subportadora con mayor ganancia puede ser asignada un mayor número de bits de datos y viceversa.

El propósito del adaptive loading es alcanzar una mayor velocidad de transmisión y optimización de energía de la señal, basados en el conocimiento de ganancia de los subcanales previamente analizado mediante el método de SVD, esta técnica se la detalla a continuación y está se basa en el algoritmo desarrollado por P. Chow [13] explicado posteriormente paso a paso en el capítulo 4.

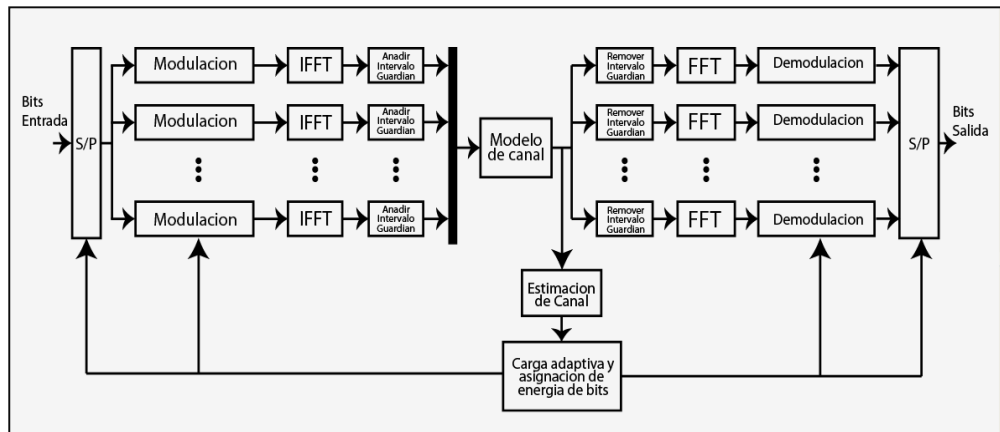
CAPÍTULO 4

4. SIMULACIÓN DEL SISTEMA

Los estudios y el desarrollo de esta simulación se basan generalmente en el código MATLAB desarrollado y publicado por Prateek Bansal y Andrés Brzezinski [10] del cual utilizamos un sistema MIMO y OFDM adaptivo. A este se le han agregado algunas modificaciones para poder simular con los diferentes esquemas de modulación. Además para optimización del sistema hay conocimiento de información entre el transmisor y receptor, lo que nos permite utilizar SVD para evitar interferencias entre subcanales, para una mejor asignación de bits de datos y energía cuando se comporte de manera adaptiva y además facilita las conversiones de Fourier y sus inversas al momento de la precodificación y armado de la señal.

El resultado de dicha simulación será una gráfica del BER vs SNR, con la que se podrá analizar el rendimiento del sistema con los diferentes parámetros y métodos.

Las técnicas y procesos de nuestro sistema se muestran en el diagrama de bloques a continuación:



4.1 Modelo a simular para la optimización de sistemas MIMO.

4.1 Modelamiento canal MIMO

Para la simulación de nuestro sistema nos basamos en la expresión (8) de la sección 3.1.1 detallada a continuación:

$$H_l = \sqrt{P_l} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \quad (8)$$

Más la información del número de antenas receptoras Rx y transmisoras Tx, y el perfil de retraso de potencia elegido, definimos el tamaño del canal:

```
H_int = 1/sqrt(2) * (randn(Mr*length(A),Mt) +
                    j*randn(Mr*length(A),Mt));
```

<create_channel.m>

Donde:

A: vector que contiene el perfil de retraso de potencia.

Mt: numero de antenas Tx.

Mr: numero de antenas Rx.

N: numero de vectores símbolo a ser enviados en una sola transmisión de Tx de símbolo OFDM.

Cada componente X_{ij} son coeficientes AWGN calculados aleatoriamente, correlacionados, con media cero y varianza unitaria para formar la matriz Rayleigh NLOS entre las antenas.

Cada elemento del canal debe ser independiente e idénticamente distribuido. Para generar estos valores aleatorios usamos el siguiente código:

```
H_int2=[];
for i = 1:length(A)
H_int2 = [H_int2;sqrt(A(i))*H_int((i-1)*Mr+1:i*Mr,:)];
end
```

<create_channel.m>

Para combatir el efecto de interferencia entre símbolos causada por los múltiples caminos, introducimos un intervalo de guarda al símbolo OFDM, tal como se explicó en la sección 3.2. De esta forma también se logra reducir la complejidad de la señal recibida en Rx. Este intervalo de guarda (GI) pasa a ser parte de cada canal de transmisión como se muestra en la siguiente línea de código:

```
H_int2 = [H_int2;zeros((N-length(A))*Mr,Mt)];
```

<create_channel.m>

Recordemos que los datos llegan en el dominio de la frecuencia un símbolo a la vez. Para pasarlos al dominio del tiempo aplicamos la función IFFT (Transformada Inversa de Fourier Rápida) de MATLAB, de esta manera los datos son transmitidos

de forma serial un símbolo a la vez. El receptor recibe estos datos en forma serial, los ordena en paralelo y luego aplica la función FFT (Transformada de Fourier Rápida) y obtiene los datos decodificados en el dominio de la frecuencia.

El canal representado de esta forma es también conocido como la respuesta de impulso del canal entre el transmisor y el receptor. El código con este procedimiento es:

```
H_f = zeros(Mr,Mt*(N-16));
for i = 1:Mt
    for jj = 1:Mr
        h_f = fft(H_int2(jj:Mr:(N-16-1)*Mr+jj,i));
        for k = 1:(N-16)
            H_f(jj,i+(k-1)*Mt) = h_f(k);
        end
    end
end
```

<create_channel.m>

Ahora que nuestro canal ya ha sido modelado y que sabemos que hay un conocimiento de información del canal entre el transmisor y receptor, podemos descomponer los canales MIMO a canales paralelos sin interferencia utilizando SVD. El modelo de la ecuación es igual al descrito anteriormente en el Capítulo 3, y para esto utilizamos la función SVD que ofrece MATLAB.

```
for i = 1:N
    [Utmp Stmp Vtmp] = svd(h_f(:,(i-1)*Mt+1:i*Mt));
    U=[U Utmp];
    V=[V Vtmp];
    S=[S Stmp];
end
```

<svd_decompose_channel.m>

4.2 Modulación y Demodulación

Para la modulación y demodulación utilizaremos el método MQAM (Modulación de Amplitud de Múltiple Cuadratura), en este proceso transformamos bits de datos individuales o en grupos en una constelación. Solo 6 niveles de constelación de señal MQAM serán utilizados y son 0, 2, 4, 16, 64, 256 QAMs. Estas constelaciones se encuentran en forma de archivos de matrices que deberán ser cargados al workspace de MATLAB en el momento de la simulación. El modulador toma los bits de entrada y sus valores de energía, basándose en el número de bits y se hace comparación con la tabla de constelación. A la salida del modulador se encuentra el símbolo en la constelación basándose en los bits de entrada y se calcula la energía apropiada para cada subportadora. Para cargar las constelaciones usamos:

```
load ENC2.mat  
load ENC4.mat  
load ENC16.mat  
load ENC64.mat  
load ENC256.mat
```

<ofdm_main.m>

Para reducir los errores en los bits, utilizamos el método de asignación de bits de Gray. Este método asegura que el símbolo adyacente pueda ser seleccionado aunque haya algún símbolo corrupto o con error. De esta forma solo tendremos error en un bit sin que se ponga en riesgo que todo el símbolo tenga error y deba ser retransmitido.

Una vez que el algoritmo de carga adaptativa (detallado en sección 3.6) selecciona el número de bits de datos que cada

subportadora puede cargar, se procede a modular los datos y se preparan para la transmisión. Estos bits de datos son creados con una función aleatoria conteniendo solo dígitos binarios 1 y 0.

```
x = (randn(1,Btot)>0);
```

<ofdm_main.m>

Btot: número total de bits por símbolo de OFDM

La modulación de acuerdo al número de bits asignado se realiza mediante la selección de cualquiera de los casos de modulación mencionados anteriormente, asignándole un índice de constelación compleja a cada subportadora. También se multiplica la asignación de energía a cada subcanal con el índice complejo de constelación. Para el caso de que la subportadora este muy interferida y su SNR este debajo de algún margen predefinido, el sistema no asignaría ningún bit en dicha subportadora. El proceso de la modulación se lo muestra a continuación:

```
for i = 1:length(b)
switch b(i)
case {1}
y = [y s2(x(b2(i))+1)*sqrt(e(i))];
case {2}
y = [y s4(x(b2(i):(b2(i+1)-1))*[2;1]+1)
*sqrt(e(i))];
case {4}
y = [y s16(x(b2(i):(b2(i+1)-1))*[8;4;2;1]+1)
*sqrt(e(i))];
case {6}
y = [y s64(x(b2(i):(b2(i+1)-1))*[32;16;8;4;2;1]+1)
*sqrt(e(i))];
case {8}
y = [y s256(x(b2(i):(b2(i+1)-1))*[128;64;32;16;8;4;2;1]+1)
*sqrt(e(i))];
otherwise
```

<modulate.m>

y: Salida modulada en forma de un vector fila.

x: Vector bits de entrada para las subportadoras (vector fila).

b: Asignación de bits de subportadoras (matriz de 64 elementos, cada uno corresponde al número de bits asignados a la subportadora con el mismo índice).

e: Asignación de energía de subportadora (matriz de 64 elementos, cada uno corresponde a la energía asignadas a la subportadora con el mismo índice).

s: El codificador dado un lugar en la constelación.

El proceso de demodulación es muy similar a la modulación pero de manera inversa. Como hay información del canal tanto en el receptor y transmisor, se conoce la información de los bits y su asignación de energía. Podemos utilizar el método de Zero Forcing. Una vez encontrados los índices de constelación buscando el de menor distancia con el símbolo estimado, el dato binario puede ser recuperado usando las tablas de comparación de las constelaciones. La demodulación la realizamos con el siguiente código:

```

for i = 1:length(b)
    switch b(i)
    case {1}
        [tmp, index] = min(abs(s2-1/h(i)/sqrt(e(i))*x(i)));
        y = [y c2(index,:)];
    case {2}
        [tmp, index] = min(abs(s4-1/h(i)/sqrt(e(i))*x(i)));
        y = [y c4(index,:)];
    case {4}
        [tmp, index] = min(abs(s16-1/h(i)/sqrt(e(i))*x(i)));
        y = [y c16(index,:)];
    case {6}
        [tmp, index] = min(abs(s64-1/h(i)/sqrt(e(i))*x(i)));
        y = [y c64(index,:)];
    case {8}
        [tmp, index] = min(abs(s256-1/h(i)/sqrt(e(i))*x(i)));
        y = [y c256(index,:)];
    otherwise
        index = 0;
    end
end

```

<demodulate.m>

y: Salida modulada en forma de un vector fila.

x: Vector de símbolos de entrada para todas las subportadoras (vector fila).

h: Valor del canal (frecuencia) para todas las subportadoras (64 elementos).

b: Asignación de bits de subportadoras (matriz de 64 elementos, cada uno corresponde al número de bits asignados a la subportadora con el mismo índice).

e: Asignación de energía (matriz de 64 elementos)

s: El codificador dado un tamaño de constelación.

c: El código de palabra como un vector de bits para los índices.

4.3 Carga Adaptiva

Luego de que se terminó la demodulación y el dato ha sido recuperado, procederemos a realizar los cálculos para analizar el rendimiento de la transmisión.

Recordemos que estamos utilizando un sistema MIMO adaptivo. También recordemos que con OFDM tenemos la ventaja de que cada subcanal puede ser tomado como de banda estrecha y con flat-fading asumido, pero con la desventaja de que van a haber subcanales con baja ganancia dando como resultado un BER muy grande. Por lo tanto lo que se busca es sacar el provecho máximo de los subcanales que tienen mejor desempeño. Como tenemos canales variables en el tiempo hay un tiempo de decorrelación asociada a cada canal selectivo de frecuencia, por lo tanto debe ocurrir una adaptación con cada decorrelación del canal.

El esquema óptimo de la transmisión adaptiva para que alcance la capacidad de Shannon para un sistema de transmisión fija es el de la distribución waterfilling (llenado de agua) sobre el canal de frecuencia selectiva.

La distribución waterfilling como es explicada por Yu y Cioffi [12] consiste en asignar más poder a los mejores subcanales, es decir aquellos que tengan un alto SNR a fin de maximizar la suma de tasa de datos de todos los subcanales y en cada subcanal la velocidad de datos se relaciona a la asignación de potencia de la fórmula de capacidad de Gauss de Shannon. Aunque esta distribución nos daría una solución óptima, es muy difícil de calcular y asume una razón de cálculo contra

comunicación infinita en el tamaño de la constelación lo cual no es realizable.

La técnica de carga adaptiva utilizada para nuestro problema es la presentada por Bansal y Brzezinski, los cuales combinan los algoritmos de Chow [13] y Campello [14] para lograr una optimización de potencia y velocidad del sistema en base al conocimiento previo de la ganancia de los subcanales.

En el algoritmo de carga discreta de bits presentado por Campello, dadas N funciones convexas crecientes $e_n(b)$ que representan la cantidad de energía necesaria para transmitir b bits en el subcanal n a una probabilidad de error deseada. Se asume que $e_n(0) = 0$. El problema de asignación que vamos a utilizar es representado de la siguiente manera:

Problema de Minimización de Energía

Minimizar $\sum_{n=1}^N e_n(b_n)$

Sujeto a $\sum_{n=1}^N b_n = B$

$$b_n \in Z, b_n \geq 0, n = 1, 2, \dots, N$$

Para inicializar la asignación de bits usamos el algoritmo de Chow con el siguiente procedimiento:

Algoritmo de inicialización

1. Calcular el SNR del subcanal

En el caso de esta simulación lo que en realidad calculamos es un GNR lo cual no es más que el SNR dividido para el espacio o gap. Este espacio es solo un parámetro de sintonización. Diferentes valores del gap resulta en nuevo valores SNR para un

número B de bits que se deseen transmitir. Esto ocurre ya que el valor de gap afecta directamente la tabla de valores de cálculos de energía, así cambiar el valor del gap nos permite buscar un valor de BER de desempeño en nuestro sistema.

```
SNR = abs((subcar_gains.^2)./(noise*gap));
```

<ComputeSNR.m>

subcar_gains: ganancia subportadora

gap: brecha del sistema

Noise: potencia del ruido

2. Calcular el número de bits para el i -ésimo subportadora basados en la fórmula:

$$\hat{b}(i) = \log_2 \left(1 + \frac{SNR(i)}{gap} \right)$$

3. Redondear el valor de $\hat{b}(i)$ a $b(i)$.

Esto se logra fácilmente en MATLAB utilizando el comando *round*.

4. Restringir el valor de $b(i)$ para que sean del orden de modulación MQAM (0,1,2,4,6,8)
5. Calcular la energía del i -ésimo subcanal basados en el numero de bits calculado en el paso 2 por la formula:

$$e_i(b(i)) = \frac{(2^{b(i)}-1)}{GNR(i)}, \text{ recordando que } GN R(i) = \frac{SNR(i)}{gap}$$

6. Crear una tabla de incrementos de energía para cada subcanal. Para el i -ésimo subcanal usamos la formula

$$\Delta e_i(b) = e_i(b) - e_i(b-1) = \frac{2^{b-1}}{GNR}$$

Este procedimiento queda resumido en la siguiente porción de código:

```

for i = 1:num_subc
    tempbits = log2(1 + abs(SNR(i)));
    roundtempbits = round(tempbits);

    if (roundtempbits > M)
        roundtempbits = M;
    end

    if (mod(roundtempbits,2)== 1 & roundtempbits ~= 1)
        roundtempbits = roundtempbits -1;
    end

    if roundtempbits > 0
        energy_alloc(i) = (2^roundtempbits-1)/SNR(i) ;
    else
        energy_alloc(i) = 0;
    end
    bits_alloc(i) = roundtempbits;
end

```

<chow_algo.m>

Consideremos un k-ésimo canal. Conocida la ganancia del canal y la densidad espectral de potencia del ruido, la tabla de incrementos de potencia nos va a proveer la energía requerida del subcanal para que pase de soportar 0 bits a 1 bit, de 1 bit a 2 bits y así sucesivamente. Como nuestro sistema soporta un máximo de 8 bits, el incremento a 9 lo fijamos a un valor muy alto. También como se requiere que los subcanales tengan solo 0, 1, 2, 4, 6 u 8 bits, los números impares después del 1 no son aceptados. Para esto utilizamos el siguiente método de promedio. Si en la tabla obtenemos que la energía para pasar de 2 a 3 bits es de 30 unidades, y para pasar de 3 a 4 bits es de 40 unidades. Promediamos ambas diferencias de energía y obtenemos 35. Con un incremento de 35 aseguramos que se excede la cantidad de energía mínima para llegar a 3 bits por lo

que la asignación se realizara en el bit siguiente. Esto se realiza en la porción de código siguiente:

```
energytable = abs((1./SNR)'.*(2.^([1:M+1]-1)));

energytable(:,M+1) = Inf*ones(size(energytable(:,M+1)));

for i = 3:2:M
    energytable(:,i) = (energytable(:,i)+energytable(:,i+1))/2;
    energytable(:,i+1) = energytable(:,i);
end
```

<EnergyTableInit.m>

M: tamaño máximo constelación

Esto funciona bien hasta que llegamos al final. Este problema lo solucionamos con el algoritmo de resolución del último bit propuesto por Campello, pero de este hablaremos más adelante. Primero utilizaremos lo propuesto por él para optimizar el proceso de asignación de bits que realizamos anteriormente.

Algoritmo B-Tighten (Compresión de bits)

Entradas:

b , asignación inicial del bit

B , numero total de bits asignados

Salida:

b , asignación de bit optimizada

Algoritmo:

```

 $B' \leftarrow 0$ 
for  $n = 1$  to  $N$ 
   $B' \leftarrow B' + b(n)$ 
while ( $B' \neq B$ )
  if ( $B' > B$ )
     $n = \arg \max_{1 \leq j \leq N} \Delta e_j(b_j)$ 
     $B \leftarrow B - 1$ 
     $b(n) \leftarrow b(n) - 1$ 
  else
     $m = \arg \min_{1 \leq i \leq N} \Delta e_i(b_i + 1)$ 
     $B \leftarrow B + 1$ 
     $b(n) \leftarrow b(n) + 1$ 

```

Este algoritmo se basa en la idea de la asignación incremental. Inicialmente, se asignan 0 bits a cada tono. Luego en cada iteración el algoritmo calcula la energía incremental necesaria para transmitir un bit más al tono. Este incremento de energía esta dado por la diferencia entre la energía que se necesitaría para transmitir un bit mas en ese tono y la energía actual asignada al tono. El algoritmo luego asigna el bit al tono que requiera el menor incremento de energía. Esto se repite con el algoritmo asignando bits uno por uno conservando las restricciones de potencia. Ahora mostramos como se desarrollo este algoritmo en código:

```

while (bt ~= total_bits)
  if (bt > total_bits)
    max_val = 0;
    max_ind = ceil(rand(1)*num_subc);

    for i = 1:num_subc
      if bits_alloc(i) ~= 0
        temp = energytable(i,bits_alloc(i)) ;
      else
        temp = 0;
      end
      if (temp > max_val)
        max_val = temp;
        max_ind = i;
      end
    end

    if (bits_alloc(max_ind) > 0)
      bits_alloc(max_ind) = bits_alloc(max_ind) -1;
      energy_alloc(max_ind) = energy_alloc(max_ind) - max_val;
      bt = bt-1;
    end
  else

    min_val = Inf;
    min_ind = ceil(rand(1)*num_subc);
    for i = 1:num_subc
      if bits_alloc(i) ~=0 & bits_alloc(i) < M+1
        temp = energytable(i,bits_alloc(i) + 1);
      else
        temp = Inf;
      end
      if (temp < min_val)
        min_val = temp;
        min_ind = i;
      end
    end

    if (bits_alloc(min_ind) < M)
      bits_alloc(min_ind) = bits_alloc(min_ind) +1;
      if (min_val==inf)
        min_val = energytable(min_ind,bits_alloc(min_ind));
      end
      energy_alloc(min_ind) = energy_alloc(min_ind) +min_val;
      bt = bt+1;
    end
  end
end
end

```

<campello_algo.m>

M: Numero de subcanales

Por último retomemos el problema de la asignación del último bit. Campello nos ofrece esta solución:

Algoritmo de Resolución del Ultimo Bit

1. Comprobar que la asignación de bits de entrada contiene al menos una violación de restricción de bits.
2. Si hay alguna violación digamos en el subcanal v , encontrar el bit entre la asignación de bits actual que tenga la mayor energía incremental que pueda ser usada para llenar el subcanal v .

$$E_1 = \Delta e_v(b(v)) - \Delta e_i(b(i))$$

3. Encuentre el bit que va a costar el menor incremento en los otros subcanales a los que les haya sido asignado 0 o 1 bits. Esto se debe a que solo entre 0 y 1 bit hay solo diferencia de 1 bit, en cambio en 2, 4, 6 y 8 una diferencia de solo 1 bit significaría una violación a la restricción.

$$E_2 = \Delta e_j(b(j) + 1) - \Delta e_v(b(v))$$

4. Realizar el cambio correspondiente al más pequeño de E_1 y E_2 .

```

max_val = 0;

for i = 1:num_subc
    if (i ~= index & bits_alloc(i) == 1)
        if bits_alloc(i) ~= 0
            temp = energytable(i,bits_alloc(i)) ;
        end

        if (temp > max_val)
            max_val = temp;
            max_ind = i;
        end
    end
end

min_val = Inf;
for i = 1:num_subc
    if (i ~= index & bits_alloc(i) == 1)
        if bits_alloc(i) ~= 0
            temp = energytable(i,bits_alloc(i) + 1);
        end
        if (temp < min_val)
            min_val = temp;
            min_ind = i;
        end
    end
end

if (min_val < max_val)
    bits_alloc(min_ind) = bits_alloc(min_ind) + 1;
    bits_alloc(index) = bits_alloc(index) - 1;
    energy_alloc(index) = energy_alloc(index) - min_val;
else
    bits_alloc(max_ind) = bits_alloc(max_ind) - 1;
    bits_alloc(index) = bits_alloc(index) + 1;
    energy_alloc(index) = energy_alloc(index) + max_val;
end

```

<ResolvetheLastBit.m>

```

SNR = ComputeSNR(subchan_gains,noise,gap);

[bits_alloc, energy_alloc] = chow_algo(SNR,num_subc,M);
energytable = EnergyTableInit(SNR,M);

[bits_alloc,energy_alloc] =
campello_algo(bits_alloc,energy_alloc,energytable
, total_bits,num_subc,M);

```

<BitLoad.m>

subchan_gains: Ganancias de subportadoras

total_bits: Número total de bits

num_subc: Numero de subportadoras

gap: Brecha del sistema

noise: Potencia ruido

M: Tamaño máximo de constelación

Ahora la subportadora está completamente optimizada y con asignación de bits y energía basada en condiciones selectivas del canal. Como resultado podemos esperar una transmisión inalámbrica con mejor desempeño.

4.4 Operaciones de Transformada de Fourier y su Inversa en OFDM

Al momento de usar SVD, en el canal de nuestro sistema la operación de la transformada inversa resulta más fácil.

Después que la información haya sido modulada hay que implementar IFFT para la transmisión de la señal en el dominio del tiempo. Justo antes de usar la inversa de Fourier debemos multiplicar la señal por el filtro precoding que se encarga de cancelar la matriz unitaria derecha V^T que se obtuvo por medio de SVD explicado anteriormente en sección 3.5.

```

x_pre = [];
for i = 1:N
    x_pre = [x_pre; [V(:, (i-1)*Mt+1:i*Mt)]
              *transpose(x_mod((i-1)*Mt+1:i*Mt))];
end

x_pre = transpose(x_pre);

```

<precode.m>

Mt: Número de antenas transmisoras

x_mod: Símbolo modulado para cada tono (los símbolos están agrupados en tonos, ej.: 3 subcanales MIMO por tono, tiene 3 símbolos x_mod consecutivos)

V: La matriz unitaria del canal, en orden de fila

N: Numero de subcanales

Lo contrario ocurre en el receptor, en el que un filtro multiplica el símbolo recibido por la matriz unitaria U, para que pueda ser vuelta a su composición original.

```

shaped_vals = [];
for i = 1:N
    shaped_vals =
        [shaped_vals; [U(:, (i-1)*Mr+1:i*Mr)]'
          *transpose(rec_symbol((i-1)*Mr+1:i*Mr))];
end
shaped_vals = transpose(shaped_vals);

```

<shape.m>

rec_symbol: Símbolo recibido después de realizar la transformada de Fourier inversa

Mr: Numero de antenas receptoras

U: Matriz de matrices unitarias multiplicando a cada subcanal MIMO

N: Numero de subcanales OFDM

Para realizar la transformada de Fourier inversa usamos la función IFFT y le añadimos el intervalo guardián.

```

ofdm_symbol = ifft(inp_symbol,num_subc);
if (guard_interval > length(ofdm_symbol))
    error('GI greater than ofdm symbol duration');
end

guard_symbol = ofdm_symbol(end-guard_interval+1:end);

ofdm_symbol = [guard_symbol ofdm_symbol];

```

<ifft_cp_tx_blk.m>

inp_symbol: Símbolo de entrada a todas las subportadoras

num_subc: Número de subportadoras

guard_interval: Intervalo guardián basando en el símbolo OFDM

Esta señal generada es modelada para ser transmitida por el medio inalámbrico, como se mencionó anteriormente la estimación de nuestro canal MIMO $y = Hx + n$.

```

noise = sqrt(sig2)*1/sqrt(2)*(randn(Mt*N,1)
                             + j*randn(Mt*N,1));
y = H*x + noise;

```

<channel.m>

sig2: Varianza de sonido

Mt: Numero de antenas Tx

Mr: Numero de antenas Rx

x: Vector de símbolos de entrada complejos (en MIMO es una matriz en la que cada columna es el valor de salida de antena en un solo instante de tiempo)

H: Canal de frecuencia selectiva

N: Numero de símbolos transmitidos en un marco OFDM

La señal y que llega al receptor, es trabajada con la función FFT para obtener el espectro de frecuencia codificado y recuperar el símbolo modulado originalmente. Antes de esto no nos debemos olvidar de remover el intervalo guardián.

```
if (guard_interval > length(ofdm_symbol))
    error('GI greater than ofdm symbol duration');
end

rec_time_symbol = ofdm_symbol(guard_interval+1:end);

rec_symbol = fft(rec_time_symbol,num_subc);
```

<fft_cp_rx_blk.m>

ofdm_symbol: Símbolo OFDM recibido por el canal

num_subc: Número de subportadoras

guard_interval: Intervalo de guardia basado en el símbolo OFDM

El error de cada transmisión puede ser calculado por medio de un xor comparando el dato recibido con el dato original transmitido.

```
totalErrors = totalErrors + sum(xor(y_demod,x));
```

<ofdm_main.m>

x: Bits a transmitir

y: Canal

El SNR normalizado puede ser calculado después de cada iteración de canal sumando la energía asignación a cada subportador y dividiéndola para el número total de bits transmitidos y varianza del ruido. El BER lo obtenemos dividiendo el error total de cada iteración de canal para el número total de bits de datos transmitidos y las iteraciones de canal y de transmisión.

```
EbNo = [EbNo sum(energy_alloc)/Btot/sig2];  
  
Errors =  
    [Errors totalErrors/Btot/ChannelIter/TransmitIter]  
TotEbNo = [TotEbNo mean(EbNo)]
```

<ofdm_main.m>

4.5 Interfaz gráfica GUI

Para un mejor manejo de la simulación se implementó una interfaz grafica GUI de MATLAB.

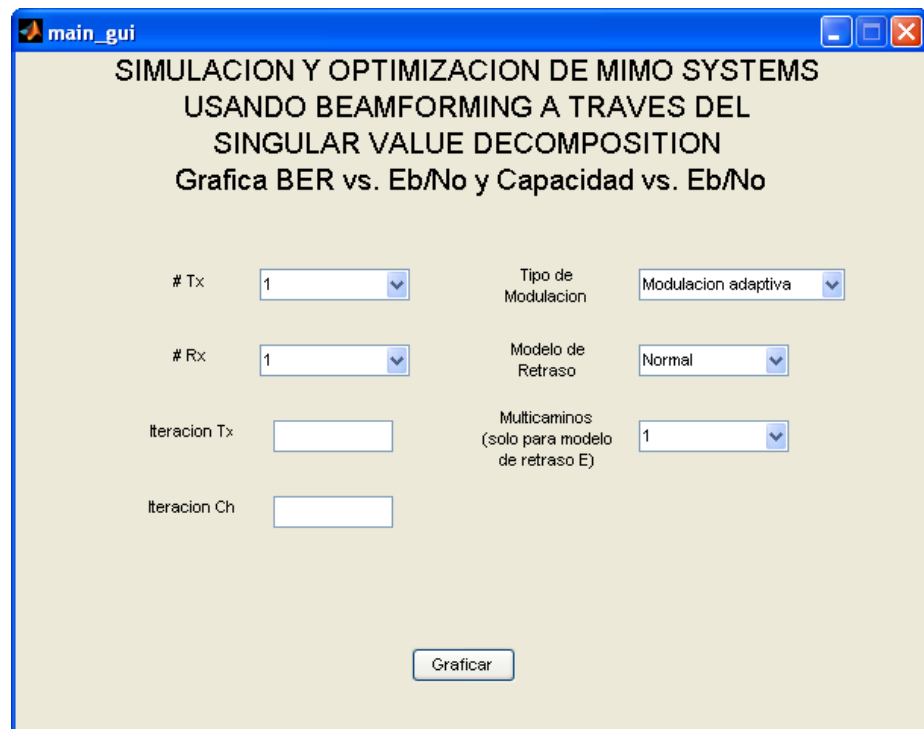


Figura 4.1 Interfaz Gráfica Matlab

Al realizar dicha interfaz, automáticamente se creó el archivo *main_gui.m*, el cual se tiene que compilar para abrirlo y así poder ingresar los datos para obtener nuestras gráficas a analizar.

4.6 Análisis de Resultados

Se ha realizado comparaciones para comprobar las diferencias en el rendimiento al variar la configuración en el sistema de transmisión inalámbrica.

La Figura 4.1 muestra la comparación entre tres diferentes condiciones, un MIMO – OFDM de 2x2 aplicando el método de SVD y con modulación adaptativa, un enlace SISO con modulación adaptativa, y un enlace SISO con modulación fija, ya que se asigna un número fijo de bits de datos para cada subportadora. Para una comparación válida, el número total de bits de datos en cada símbolo OFDM está establecido en una constante fija.

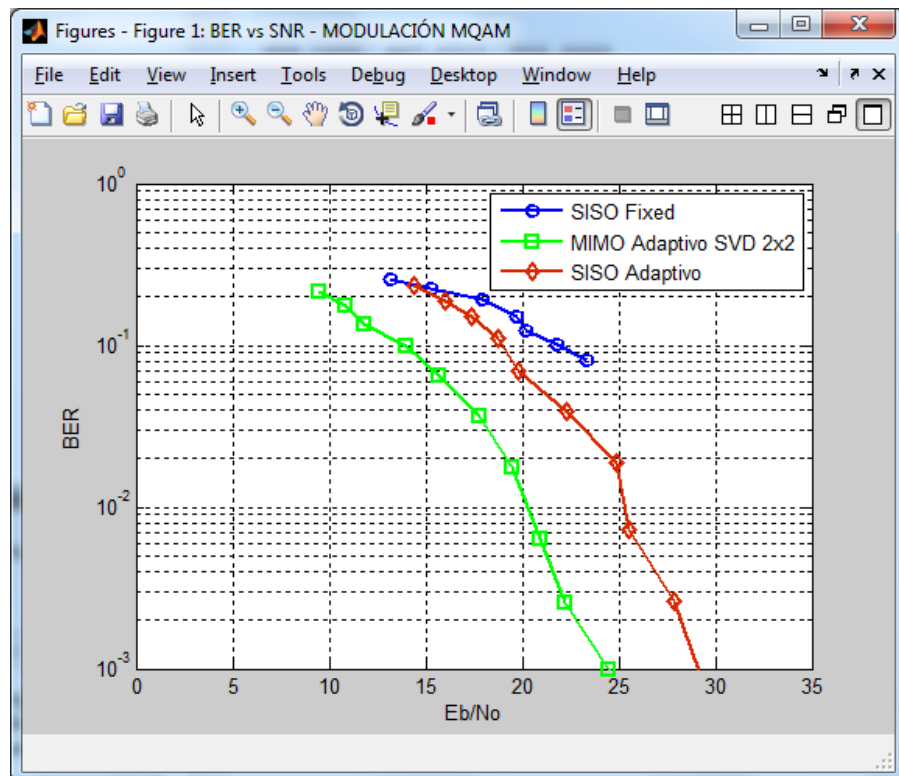


Figura 4.2. Desempeño del BER para diferentes esquemas.

Como se muestra en la Figura 4.2 el sistema MIMO de 2x2 con el método de SVD tiene el mejor rendimiento, al comparar el mismo valor de BER, el MIMO requiere un pequeño valor de E_b/N_0 [dB], mientras que el SISO adaptivo necesita un mayor E_b/N_0 para lograr el mismo valor de BER que tiene el MIMO.

También se puede apreciar que la gráfica del SISO Fixed muestra pocos puntos, esto se debe porque el valor variaba por un amplio margen que, en general no muestra información relevante, sobre todo porque dicha gráfica no es capaz de excederse más para valores mayores de E_b/N_0 , ya que el rendimiento del SISO con tasa fija no lo permite.

Se comprueba que el SISO fijo posee el peor rendimiento entre los tres. Por lo tanto el sistema MIMO supera claramente al SISO Adaptivo, de acuerdo a la Figura 4.2, MIMO posee una mejora de 3 - 4 dB de E_b/N_0 con respecto al SISO.

La sección 3.4 indica que pares de antenas pueden ser agregados para aumentar la capacidad del canal mejorando la red de transmisión inalámbrica. Por lo tanto a continuación analizaremos el sistema MIMO con diferentes combinaciones de pares antenas.

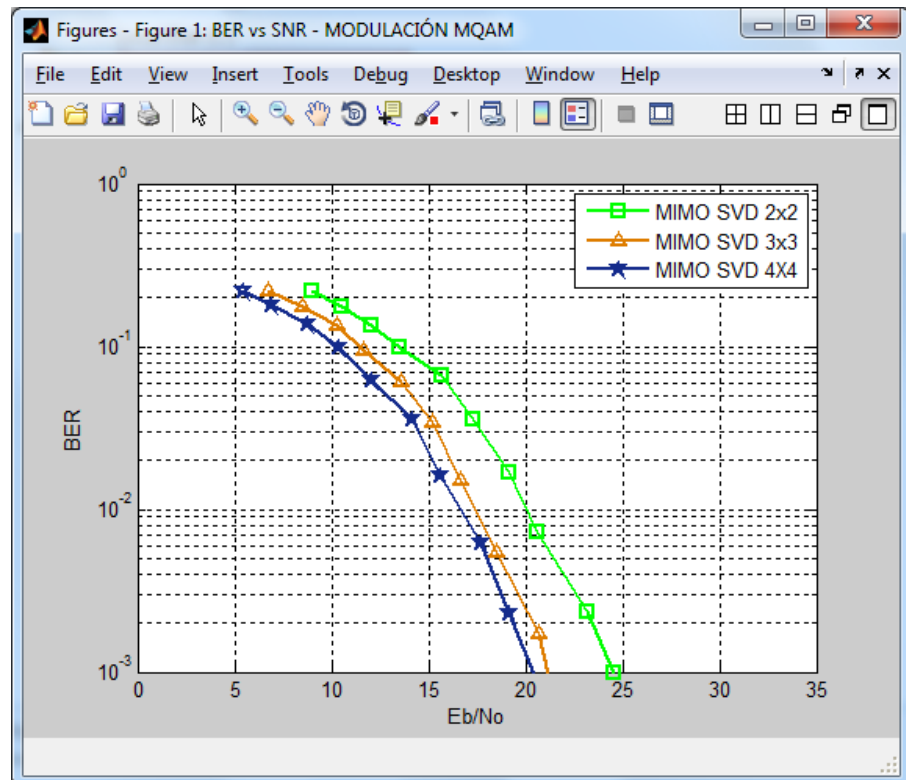


Figura 4.3. Desempeño del BER para diferente combinación de antenas.

La Figura 4.3 muestra que sistema MIMO de 4x4 tiene el mejor desempeño, le sigue el MIMO 3x3 y luego el sistema MIMO 2x2, todos con modulación adaptiva y usando el método de SVD. Esta gráfica nos permite comprobar experimentalmente que existe una mejora significativa en el rendimiento a medida que se aumenta la combinación del número de antenas.

Una diferencia notable en el rendimiento se puede apreciar al incrementar de un sistema MIMO 2x2 a un MIMO 3x3, aproximadamente se presenta una mejora de 4dB para poder alcanzar un mismo valor de BER. Al comparar el MIMO 3x3 con el MIMO 4x4 la mejora de ganancia es modesta, apenas 1 dB de

mejora entre el uno y otro. El estándar 802.11n sugiere que hasta un número máximo de 4 antenas y 4 receptores se use en la práctica, ya que su desempeño es relativamente bueno y un mayor número de combinaciones (ej. 5x5 o 6x6), solo nos daría mínimas mejoras en el rendimiento, lo que no ameritaría debido a su costo.

Como se discutió antes, nuestro sistema MIMO aprovechará el efecto multicamino para mejorar el rendimiento de la transmisión. Un MIMO 2x2 se mantiene constante mientras se aplica un incremento de “taps” al canal para estimular los efectos de la propagación multicamino. Los resultados se muestran a continuación en la Figura 4.4.

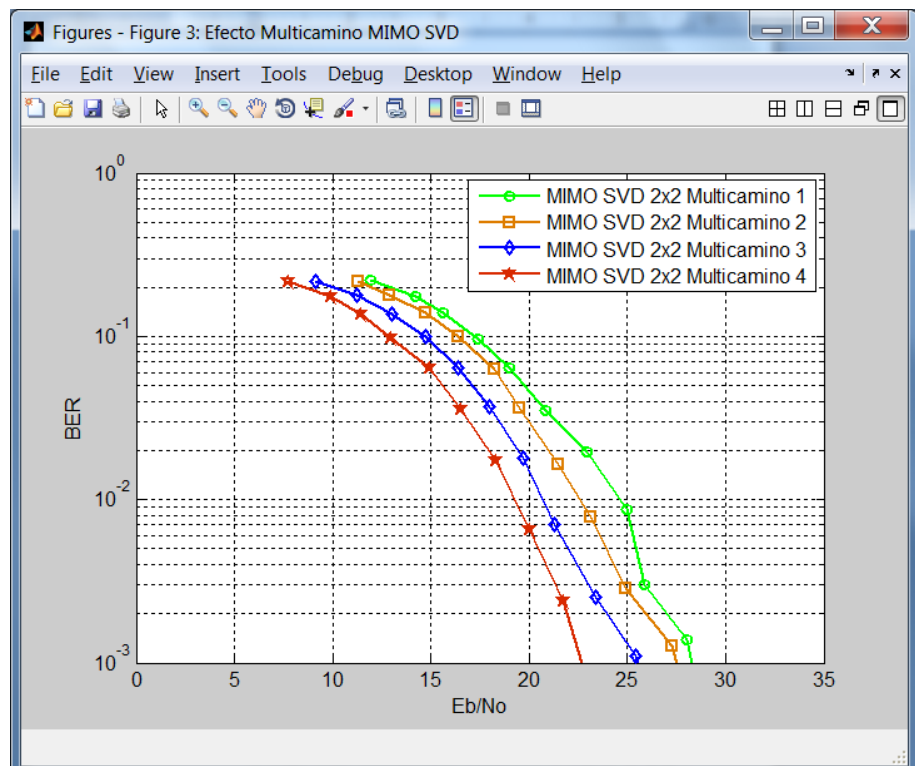


Figura 4.4. Rendimiento del BER para diferentes caminos de propagación.

A continuación procederemos a comparar el método de SVD y modulación adaptiva MQAM, con el método Zero Forcing analizado el trabajo anterior “MODELAMIENTO Y SIMULACION DE SISTEMAS MIMO” [9].

Los dos métodos son desarrollados bajo parámetros similares para que la comparación sea válida, el canal es tipo Rayleigh con NLOS, son sistemas MIMO 2x2 con modulación QAM, solo que en nuestro estudio es una modulación adaptiva mientras que en el anterior era una fija.

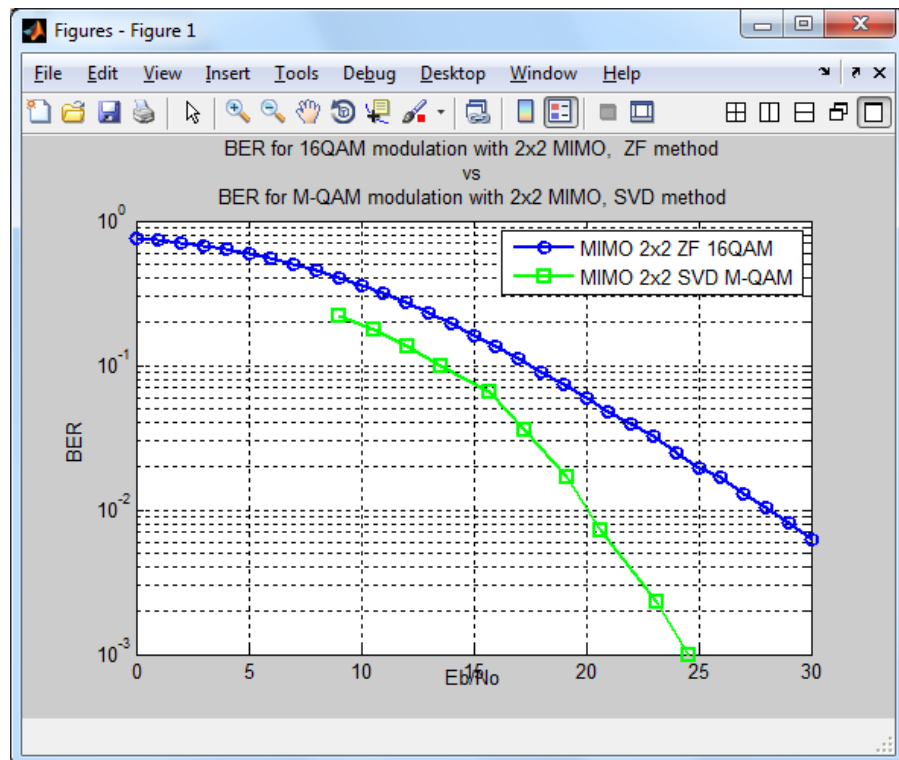


Figura 4.5. Rendimiento del BER, entre ZF y SVD.

La Figura 4.5 muestra una mejora notable en nuestro método, entre 5 – 10db, para un mismo valor de BER. Se comparó con el ZF y su modulación 16QAM, que es una modulación compleja y

requiere de mayor potencia al igual que nuestro algoritmo que posee modulación MQAM adaptiva, sin embargo este último lo supera. Lo que permite aumentar nuestra confianza aún más en nuestro estudio con SVD.

Para concluir este estudio graficamos la capacidad de nuestro sistema de acuerdo a la expresión de Shannon analizada en la sección 3.4.

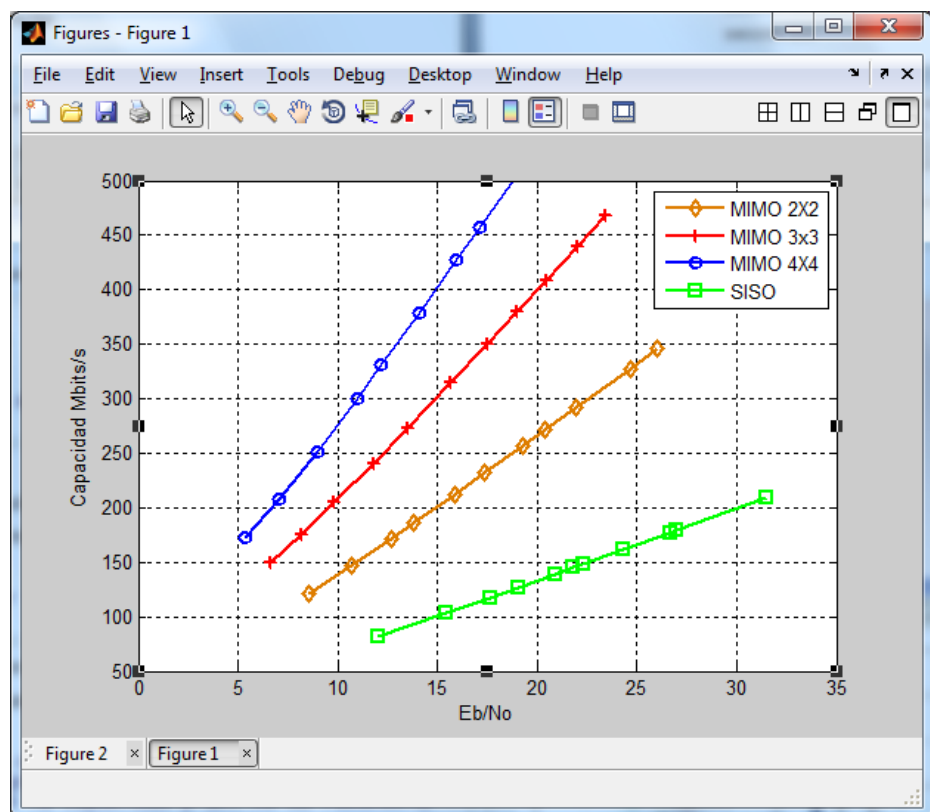


Figura 4.6. Capacidad de nuestro sistema MIMO con distintas combinaciones de antenas.

La Figura 4.6 permite apreciar el rendimiento máximo alcanzable en nuestro sistema (capacidad del canal), este se ve afectado

por el ancho de banda, por el SNR y el número de pares de antenas. Se usa el ancho de banda de 20Mhz que es el valor convencional como indica el estándar 802.11n. La velocidad de transmisión aumentan conforme mejora el SNR y alcanzan valores bastante elevados como se puede apreciar en el sistema MIMO de 4x4 (aprox. 500Mbps), con menor capacidad se presenta el MIMO 3x3 y le sigue el MIMO 2x2 que llega aproximadamente hasta 300Mbps. Con los métodos propuestos a lo largo de este estudio hemos logrado optimizar el desempeño de MIMO y se comprueba que esta tecnología está muy por encima del estándar práctico para redes inalámbricas (54Mbps).

CONCLUSIONES

Mediante la simulación en la plataforma de software Matlab, se ha podido analizar las soluciones propuestas en la sección 3 para la optimización de nuestro sistema MIMO y se pudo llegar a las siguientes conclusiones:

1. Al realizar un estudio analítico de la solución para luego proceder con las simulaciones respectivas se pudo comprobar que el método de Singular Value Decomposition acompañado de las bondades de OFDM, la modulación adaptiva y la propagación multitrayectoria aportaron significativamente para maximizar el rendimiento de los sistemas inalámbricos MIMO logrando velocidades elevadas de transmisión (hasta 500Mbps) y disminuyendo considerablemente el BER, inclusive superando métodos como el Zero Forcing.
2. En la primera simulación analizamos un sistema SISO con un MIMO y la diferencia en el rendimiento fue notable, 4 dB aproximadamente de mejora en el MIMO, confirmando que los múltiples pares de antena mejoran considerablemente el sistema.
3. Al comparar el sistema MIMO bajo las mismas condiciones pero con diferentes combinaciones de antena se puede afirmar que a medida que se incrementan dichos pares de antena, el BER mejora, por lo tanto su relación es directamente proporcional, pero no es necesario incrementar a partir de un 4x4 ya que su costo/beneficio no lo amerita.
4. También se analizaron los efectos multicaminos para ver gráficamente como afectan al rendimiento de nuestro sistema. En la Figura 4.3 comprobamos que MIMO se beneficia de la multitrayectoria ya que disminuye el BER a medida que incrementan dichos efectos.

5. Se usó el método Zero Forcing del curso anterior para ilustrar sus ventajas y poder compararlo con nuestro estudio del SVD. Las diferencias entre las curvas de la Figura 4.4 claramente indica que el método de SVD mejora notablemente el rendimiento del sistema entre 5 – 10 dB en comparación. Estos diferentes métodos se encuentran bajo condiciones similares pero el método de ZF posee modulación 16QAM fija, mientras que para el SVD una modulación MQAM fue elegida como el esquema de modulación adaptativa, por lo que su capacidad para asignar los bits de datos y energía de forma dinámica usando la subportadora con mayor ganancia del canal mejora el BER.

6. Por último, se obtuvo la Figura 4.6 y se volvió a comparar el desempeño del sistema con distintos arreglos pero desde otro punto de vista, el de capacidad del canal, y claramente se pudo apreciar que MIMO con SVD presenta excelentes resultados, logrando velocidades de transmisión elevadas (aprox. de 60Mbps – 500Mbps) mejorando el throughput a medida que incrementamos el número de antenas en la configuración, lo que permite optimizar tecnologías como Wi-Fi y WIMAX que en la actualidad pueden llegar a soportar tasas de transferencia de 300Mbps y 128Mbps respectivamente usando un bandwidth de 20Mhz.

RECOMENDACIONES

Las recomendaciones son:

1. Al usar modulación adaptativa y de mayor complejidad como 128QAM o 256QAM se puede mejorar la velocidad de transmisión del sistema.
2. Analizar combinaciones de antenas MIMO de hasta 4x4 ya que un mayor número de combinaciones solo nos dará mínimas mejoras en el rendimiento y esto no es beneficioso en la práctica.
3. Usar SVD siempre para diagonalizar la matriz del canal y eliminar interferencia en el sistema.
4. Un factor importante a considerar en los sistemas MIMO son las reflexiones de la señal, puesto que esto incidirá en su rendimiento, por lo que se considera no tener una clara línea de vista entre las antenas transmisoras y receptoras.
5. Como futuro proyecto se puede recomendar un análisis de sistemas MIMO enfocado a su implementación en tecnologías de telefonía móvil de última generación como es el Long Term Evolution.

ANEXOS

<main_gui.m>

```
function varargout = main_gui(varargin)
% MAIN_GUI M-file for main_gui.fig
%     MAIN_GUI, by itself, creates a new MAIN_GUI or raises the
existing
%     singleton*.
%
%     H = MAIN_GUI returns the handle to a new MAIN_GUI or the
handle to
%     the existing singleton*.
%
%     MAIN_GUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%     function named CALLBACK in MAIN_GUI.M with the given input
arguments.
%
%     MAIN_GUI('Property','Value',...) creates a new MAIN_GUI or
raises the
%     existing singleton*. Starting from the left, property value
pairs are
%     applied to the GUI before main_gui_OpeningFunction gets
called. An
%     unrecognized property name or invalid value makes property
application
%     stop. All inputs are passed to main_gui_OpeningFcn via
varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows
only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help main_gui

% Last Modified by GUIDE v2.5 23-Nov-2010 01:52:05

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',  gui_Singleton, ...
                  'gui_OpeningFcn', @main_gui_OpeningFcn, ...
                  'gui_OutputFcn',  @main_gui_OutputFcn, ...
                  'gui_LayoutFcn',  [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```

else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before main_gui is made visible.
function main_gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to main_gui (see VARARGIN)

% Choose default command line output for main_gui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes main_gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = main_gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Mt = get(handles.txpopup, 'Value');
Mr = get(handles.rxpopup, 'Value');
Adapt = get(handles.modulatepopup, 'Value');
Model = get(handles.modelpopup, 'Value');
Multi = get(handles.multipathpopup, 'Value');
Titer = str2num(get(handles.Tedit, 'String'));
Citer = str2num(get(handles.Cedit, 'String'));
[Errors, TotEbNo]=ofdm_main(Mt,Mr,Adapt,Model,Multi,Titer,Citer);

```

```

% --- Executes on selection change in txpopup.
function txpopup_Callback(hObject, eventdata, handles)
% hObject    handle to txpopup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns txpopup contents
as cell array
%         contents{get(hObject,'Value')} returns selected item from
txpopup

input = get(hObject,'value');
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function txpopup_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txpopup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on
Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in rxpopup.
function rxpopup_Callback(hObject, eventdata, handles)
% hObject    handle to rxpopup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns rxpopup contents
as cell array
%         contents{get(hObject,'Value')} returns selected item from
rxpopup

input = get(hObject,'Value');
guidata(hObject, handles);

```

```
% --- Executes during object creation, after setting all properties.
function rxpopup_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rxpopup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: popupmenu controls usually have a white background on
Windows.
```

```
%     See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on selection change in modulatepopup.
```

```
function modulatepopup_Callback(hObject, eventdata, handles)
% hObject    handle to modulatepopup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: contents = get(hObject,'String') returns modulatepopup
contents as cell array
%     contents{get(hObject,'Value')} returns selected item from
modulatepopup
```

```
input = get(hObject, 'Value');
guidata(hObject, handles);
```

```
% --- Executes during object creation, after setting all properties.
```

```
function modulatepopup_CreateFcn(hObject, eventdata, handles)
% hObject    handle to modulatepopup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: popupmenu controls usually have a white background on
Windows.
```

```
%     See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```

% --- Executes on selection change in modelpopup.
function modelpopup_Callback(hObject, eventdata, handles)
% hObject      handle to modelpopup (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns modelpopup
contents as cell array
%          contents{get(hObject,'Value')} returns selected item from
modelpopup

tempM = get(handles.modelpopup, 'Value');
tempP = get(handles.multipathpopup, 'Value');
if tempM ~= 4
    if tempP ~= 1
        msgbox('Multipath only apply for delay model E');
    end
end

input = get(hObject, 'Value');
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function modelpopup_CreateFcn(hObject, eventdata, handles)
% hObject      handle to modelpopup (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on
Windows.
%          See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in multipathpopup.
function multipathpopup_Callback(hObject, eventdata, handles)
% hObject      handle to multipathpopup (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns multipathpopup
contents as cell array
%          contents{get(hObject,'Value')} returns selected item from
multipathpopup

tempM = get(handles.modelpopup, 'Value');

```

```

tempP = get(handles.multipathpopup, 'Value');

if tempM ~= 4
    if tempP ~= 1
        msgbox('Change this for delay model E only');
    end
end

input = get(hObject, 'Value');
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function multipathpopup_CreateFcn(hObject, eventdata, handles)
% hObject    handle to multipathpopup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on
Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUiControlBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

function Tedit_Callback(hObject, eventdata, handles)
% hObject    handle to Tedit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject, 'String') returns contents of Tedit as text
%         str2double(get(hObject, 'String')) returns contents of Tedit
as a double

% --- Executes during object creation, after setting all properties.
function Tedit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Tedit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.

```



```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function Cedit_Callback(hObject, eventdata, handles)
% hObject    handle to Cedit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Cedit as text
%        str2double(get(hObject,'String')) returns contents of Cedit
as a double
```

```
% --- Executes during object creation, after setting all properties.
function Cedit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Cedit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

<ofdm_main.m>

```
function [Errors, TotEbNo] =
ofdm_main(Mt,Mr,Adapt,Model,Multi,Titer,Citer)
% Esta es la función principal que llama a las otras funciones
durante la
% simulación. Toma los datos ingresados en la interfaz gráfica y nos
da las
% gráficas de resultados BER vs. Eb/No y Capacidad vs. Eb/NO

switch Model
    case 1
        A = [1 1/exp(1) 1/exp(2)] % canal 'normal'
    case 2
        A = [1 1/exp(1.25) 1/exp(2.5)] % canal B
    case 3
        A = [1 1/exp(1.4) 1/exp(4.45)] % canal D
    case 4
        A = [1./exp(0.6:0.1:0.9), 1/exp(1.03),
1./exp(1.28:0.3:1.88), ...
1./exp(2.25), 1/exp(2.69), 1./exp(3.2:0.5:5.2)]; % canal E
        switch Multi
            case 1
                A = A(1:7:15)
            case 2
                A = [A(1:5:15),A(15)]
            case 3
                A = [A(1:3:15),A(15)]
            case 4
                A = A(1:2:15)
        end
    end

end

N = 64; % número de símbolos en cada símbolo OFDM
GI = 16; % intervalo guardián
sig2 = 1e-3; % varianza ruido
M = 8; % número máximo de constelaciones de bits
Btot = 128*Mt; % número total de bits por símbolo OFDM
Mgap = 10.^(1:(1.7/10):2.7); % espacio
if Adapt == 2
    Mgap = Mgap(1:7);
end

% # iteraciones de las transmisiones de símbolo p/c instancia de
canal
TransmitIter = Titer

% # iteraciones independientes idénticamente distribuidas p/c
instancia de canal
ChannelIter = Citer

GapIter = length(Mgap);
```

```

%carga las constelaciones MQAM
load ENC2.mat
load ENC4.mat
load ENC16.mat
load ENC64.mat
load ENC256.mat

TotEbNo = [];
Errors = [];
EbNo = [];

for lGap = 1:GapIter
    lGap
    gap = Mgap(lGap);
    totalErrors = 0;
    for lChan = 1:ChannelIter
        % crea el canal
        [H h_f]=create_channel(Mt, Mr, A, N+GI);
        % descompone el canal en el dominio de la frecuencia
        [U S V] = svd_decompose_channel(Mt, Mr, h_f, N);
        % bitloading
        [bits_alloc,energy_alloc] = BitLoad(S,Btot,Mt*N,gap,sig2,M);
        if Adapt == 2
            bits_alloc = 2*ones(size(bits_alloc));
        end

        for lTrans = 1:TransmitIter
            % bits a transmitir, crean los datos aleatoriamente
            x = (randn(1,Btot)>0);
            % modulación
            x_mod = modulate(x,bits_alloc,energy_alloc,
s2,s4,s16,s64,s256);
            % precodificación de la señal modulada
            x_pre = precode(Mt, x_mod, V, N);
            % transformada de Fourier inversa, para cada canal
            ofdm_symbol =[];

            for i=1:Mt
                ofdm_symbol = [ofdm_symbol;
ifft_cp_tx_blk(x_pre(i:Mt:Mt*(N-1)+i),N,GI)];
            end

            ofdm_symbol2 = reshape(ofdm_symbol,Mt*(N+GI),1);
            % canal
            y = transpose(channel(sig2, Mt, Mr, ofdm_symbol2, H,
N+GI));

            % transformada de Fourier
            rec_symbol =[];
            for i=1:Mt
                rec_symbol = [rec_symbol;
fft_cp_rx_blk(y(i:Mt:Mt*(N+GI-1)+i),N,GI)];
            end

```

```

rec_symbol2 = reshape(rec_symbol,1,Mt*N);

% formar señal recibida
shaped_vals = shape(rec_symbol2, Mr, U, N);

% demodular
y_demod = demodulate(shaped_vals, bits_alloc,
energy_alloc, S,s2,s4,s16,s64,s256, c2,c4,c16,c64,c256);

% comparación
totalErrors = totalErrors + sum(xor(y_demod,x));
end
EbNo = [EbNo sum(energy_alloc)/Btot/sig2];
end
%acumulación de los datos para calcular gráficos
Errors = [Errors totalErrors/Btot/ChannelIter/TransmitIter]
TotEbNo = [TotEbNo mean(EbNo)]
EbNo = [];
end

color = ['r';'g';'k';'c'];
notation = ['-O';'d-';'-^';'-s'];

legend_str = [];
legend_str = [ legend_str ;...
{ num2str(Mt), ' x ', num2str(Mr) }]];

%obtención de gráfica BER vs. Eb/No
TotEbNo1 = 10*log10(TotEbNo);
figure(1)
semilogy(TotEbNo1,Errors,notation(Mt,:), 'color',color(Mt,:));
legend(legend_str, 'Location', 'NorthEastOutside')
hold on
grid on
xlabel('Eb/No');
ylabel('BER');
set(gca, 'xlim', [0 35]);

%obtención de gráfica Capacidad vs. Eb/No
Capacity = Mt*log2(1+TotEbNo);
figure(2)
plot(TotEbNo1,Capacity,notation(Mt,:), 'color',color(Mt,:));

legend(legend_str, 'Location', 'NorthEastOutside')
hold on
grid on
xlabel('Eb/No');
ylabel('Capacidad Mbits/s');

hold all

```

<create_channel.m>

```
function [H, H_f]=create_channel(Mt, Mr, A, N);
%
% A - vector que contiene el perfil de retraso de potencia (valor
real)
% Mt - número de antenas Tx
% Mr - número de antenas Rx
% N - número de vectores símbolo a ser enviados en una sola TX de
símbolo OFDM

% Definimos el tamaño del canal, la matriz Rayleigh y linealizamos
% utilizando Fourier.

%tamaño del canal
H_int = 1/sqrt(2)*(randn(Mr*length(A),Mt) +
j*randn(Mr*length(A),Mt));

% formamos matriz Rayleigh
H_int2=[];
for i = 1:length(A)
    H_int2 = [H_int2;sqrt(A(i))*H_int((i-1)*Mr+1:i*Mr,:)];
end

H_int2 = [H_int2;zeros((N-length(A))*Mr,Mt)];

% linealizamos con la función fft
H_f = zeros(Mr,Mt*(N-16));
for i = 1:Mt
    for jj = 1:Mr
        h_f = fft(H_int2(jj:Mr:(N-16-1)*Mr+jj,i));
        for k = 1:(N-16)
            H_f(jj,i+(k-1)*Mt) = h_f(k);
        end
    end
end

H=[H_int2];

for i = 1:N-1
    H=[H,[zeros(Mr*i,Mt);H_int2(1:(N-i)*Mr,:)]];
end
```

<svd_decompose_channel.m>

```
function [U, S, V] = svd_decompose_channel(Mt, Mr, h_f, N);
%
% Función que descompone el canal en cada subportadora a sus
componentes SVD
%
% Mt - Numero de antenas Tx
% Mr - Numero de antenas Rx
% h_f - Respuesta impulso MIMO - Mr filas, Mt*L columnas, L
canales
% N - Numero de subportadoras

U = [];
S = [];
V = [];

%descomponemos c/canal en subcanales paralelos sin interferencia con
SVD
for i = 1:N
    [Utmp Stmp Vtmp] = svd(h_f(:, (i-1)*Mt+1:i*Mt));
    U=[U Utmp];
    V=[V Vtmp];
    S=[S Stmp];
end

S = sum(S,1);
```

<modulate.m>

```
function y = modulate(x,b,e, s2,s4,s16,s64,s256)
%
% y - salida modulada en forma de un vector fila
% x - vector de bits de entrada para todas las subportadoras
% (vector fila)
% b - asignación de bits de subportadoras (matriz de 64 elementos,
% cada
% uno corresponde al número de bits asignados a la
% subportadora con
% el mismo índice)
% e - asignación de energía de subportadora (matriz de 64
% elementos,
% cada uno corresponde a la energía asignadas a la
% subportadora con
% el mismo índice)
% s_ - el codificador dado un lugar en la constelación
%
% Se realiza la modulación de acuerdo al modelo de modulación que se
% necesite

y=[];

b2 = zeros(1,length(b));
b2(1) = 1;
for i = 1:length(b)
    b2(i+1) = b(i) + b2(i);
end

for i = 1:length(b)
    switch b(i)
    case {1}
        y = [y s2(x(b2(i))+1)*sqrt(e(i))];
    case {2}
        y = [y s4(x(b2(i):(b2(i+1)-1))*[2;1]+1)*sqrt(e(i))];
    case {4}
        y = [y s16(x(b2(i):(b2(i+1)-1))*[8;4;2;1]+1)*sqrt(e(i))];
    case {6}
        y = [y s64(x(b2(i):(b2(i+1)-
1))*[32;16;8;4;2;1]+1)*sqrt(e(i))];
    case {8}
        y = [y s256(x(b2(i):(b2(i+1)-
1))*[128;64;32;16;8;4;2;1]+1)*sqrt(e(i))];
    otherwise
        % cero bits asignados
        y = [y 0];
    end
end
end
```

<demodulate.m>

```
function y = demodulate(x, b, e, h, s2,s4,s16,s64,s256,
c2,c4,c16,c64,c256);
%
% y - salida modulada en forma de un vector fila
% x - vector de símbolos de entrada para todas las subportadoras
% h - valor del canal (frecuencia)
% b - asignación de bits de subportadoras (matriz de 64 elementos,
cada
% uno corresponde al número de bits asignados a la
subportadora con
% el mismo índice)
% e - asignación de energía (matriz de 64 elementos)
% s_ - el índice de complejidad de constelación
% c_ - el código de palabra como un vector de bits para los
índices
%
% Encuentra la distancia mínima estimada de cada señal recibida y
regresa
% el código binario correspondiente. Se usa Zero Forcing por
conveniencia.

y=[];

for i = 1:length(b)
    switch b(i)
        case {1}
            [tmp, index] = min(abs(s2-1/h(i)/sqrt(e(i))*x(i)));
            y = [y c2(index,:)];
        case {2}
            [tmp, index] = min(abs(s4-1/h(i)/sqrt(e(i))*x(i)));
            y = [y c4(index,:)];
        case {4}
            [tmp, index] = min(abs(s16-1/h(i)/sqrt(e(i))*x(i)));
            y = [y c16(index,:)];
        case {6}
            [tmp, index] = min(abs(s64-1/h(i)/sqrt(e(i))*x(i)));
            y = [y c64(index,:)];
        case {8}
            [tmp, index] = min(abs(s256-1/h(i)/sqrt(e(i))*x(i)));
            y = [y c256(index,:)];
        otherwise
            index = 0;
    end
end
end
```


<ComputeSNR.m>

```
function SNR = ComputeSNR(subcar_gains,noise,gap)
% function SNR = ComputSNR(subcar_gains,noise,gap)
%
% esta función calcula el SNR del subcanal

SNR = abs((subcar_gains.^2)./(noise*gap));
```

<chow_algo.m>

```
% Algoritmo de Chow's
% -----
% A Practical Discrete Multitone Transceiver Loading Algorithm
% for Data Transmission over Spectrally Shaped Channels.IEEE Trans
% on Communications. Vol. 43, No 2/3/4, pp. 773-775, Feb/Mar/Apr
% 1995
function [bits_alloc, energy_alloc] = chow_algo(SNR,num_subc,M)
for i = 1:num_subc
% Asumiendo cada canal con flat fading asumimos los bits para cada
subcanal

%calculamos # de bits para el i-ésimo subcanal
tempbits = log2(1 + abs(SNR(i)));
% redondeamos el valor anterior a un entero
roundtempbits = round(tempbits);

% limita que no sobrepase el # máximo de bits permitidos por la
% constelación
if (roundtempbits > M)
    roundtempbits = M;
end

if (mod(roundtempbits,2)== 1 & roundtempbits ~= 1)
    roundtempbits = roundtempbits -1;
end

if roundtempbits > 0 % Calcula la energía requerida
    energy_alloc(i) = (2^roundtempbits-1)/SNR(i) ;
else
    energy_alloc(i) = 0;
end
bits_alloc(i) = roundtempbits; % Actualiza la asignación de bits

end
```

<EnergyTableInit.m>

```
function energytable = EnergyTableInit(SNR,M);
% Entradas:
%     subcar_gains : ganancia subportadora
%           M : tamaño máximo constelación
%           Gap : brecha del sistema
%           Noise : potencia del ruido
% Salidas:
%     energytable : tabla de energía
%
%
% Armamos una tabla para comparar nuestros valores de incremento de
energía
% requerido por cada subportadora para transmitir de 1, 2, 4, 6 u 8
bits,
% para poderlos comparar con las constelaciones MQAM
% Energía = 2^(i-1)/subcar_gains;
% -----
energytable = abs((1./SNR)^(2.^([1:M+1]-1)));

% Incrementa el valor de energía para una constelación de tamaño
mayor a M
% a un valor mucho más alto que no esté asignado.

energytable(:,M+1) = Inf*ones(size(energytable(:,M+1)));

for i = 3:2:M
    energytable(:,i) = (energytable(:,i) +energytable(:,i+1))/2;
    energytable(:,i+1) = energytable(:,i);
end
```

<campello_algo.m>

```
% campello_algo.m
% -----
% Esta función utiliza el algoritmo de Campello para asignar bits y
% energía
% para cada subcanal de manera optima.
%
% M - número máximo de constelaciones de bits
%

function [bits_alloc, energy_alloc] =
campello_algo(bits_alloc,energy_alloc,energytable,total_bits,num_sub
c,M)

bt = sum(bits_alloc);

if total_bits > M*num_subc %no se puede transmitir mas de M bits
    total_bits = M*num_subc;
end

while (bt ~= total_bits)
    if (bt > total_bits)
        max_val = 0;
        max_ind = ceil(rand(1)*num_subc);

        for i = 1:num_subc
            if bits_alloc(i) ~= 0
                temp = energytable(i,bits_alloc(i)) ;
            else
                temp = 0;
            end

            if (temp > max_val)
                max_val = temp;
                max_ind = i;
            end
        end

        if (bits_alloc(max_ind) > 0)
            bits_alloc(max_ind) = bits_alloc(max_ind) -1;
            energy_alloc(max_ind) = energy_alloc(max_ind) - max_val;
            bt = bt-1;
        end
    else

        min_val = Inf;
        min_ind = ceil(rand(1)*num_subc);
        for i = 1:num_subc
            if bits_alloc(i) ~=0 & bits_alloc(i) < M+1
                temp = energytable(i,bits_alloc(i) + 1);
```

```

else
    temp = Inf;
end

if (temp < min_val)
    min_val = temp;
    min_ind = i;
end
end

if (bits_alloc(min_ind) < M)
    bits_alloc(min_ind) = bits_alloc(min_ind) +1;
    if (min_val==inf)
        min_val = energytable(min_ind,bits_alloc(min_ind));
    end
    energy_alloc(min_ind) = energy_alloc(min_ind) +min_val;
    bt = bt+1;
end
end

end

for i = 1:length(bits_alloc)
    if (mod(bits_alloc(i),2) == 1 & bits_alloc(i) ~=1)
        [bits_alloc,energy_alloc] =
        ResolvetheLastBit(bits_alloc,energy_alloc,i,energytable,num_subc);
    end
end
end

```

<ResolvetheLastBit.m>

```
function [bits_alloc, energy_alloc] =
ResolvetheLastBit(bits_alloc,energy_alloc,index,energytable,num_subc
)
%
% función que resuelve el ultimo bit que se encuentre con la
restricción de
% violación de bit

max_val = 0;

for i = 1:num_subc
    if (i ~= index & bits_alloc(i) == 1)
        if bits_alloc(i) ~= 0
            temp = energytable(i,bits_alloc(i)) ;
            end

            if (temp > max_val)
                max_val = temp;
                max_ind = i;
            end
        end
    end

min_val = Inf;
for i = 1:num_subc
    if (i~= index & bits_alloc(i) == 1)
        if bits_alloc(i) ~=0
            temp = energytable(i,bits_alloc(i) + 1);
            end
            if (temp < min_val)
                min_val = temp;
                min_ind = i;
            end
        end
    end

if (min_val < max_val)
    bits_alloc(min_ind) = bits_alloc(min_ind) + 1;
    bits_alloc(index) = bits_alloc(index) - 1;
    energy_alloc(index) = energy_alloc(index) - min_val;
else
    bits_alloc(max_ind) = bits_alloc(max_ind) - 1;
    bits_alloc(index) = bits_alloc(index) + 1;
    energy_alloc(index) = energy_alloc(index) + max_val;
end
```

<BitLoad.m>

```
function [bits_alloc,energy_alloc] =
BitLoad(subchan_gains,total_bits,num_subc,gap,noise,M)
% Algoritmo Carga de Bits
% -----
%
% Entradas :
%     subchan_gains : ganancias de subportadoras
%     total_bits : número total de bits
%     num_subc : número de subportadoras
%     gap : brecha del sistema
%     noise : potencia ruido
%     M : tamaño máximo de constelación
% Salidas:
%     bits_alloc : asignación de bits para cada subcanal
%     power_alloc : asignación de potencia total
% -----

% calculo de SNR para cada canal
SNR = ComputeSNR(subchan_gains,noise,gap);

% Esta función inicializa el sistema con una asignación de bits y
% asignación de energía por medio del algoritmo de Chow, el cual es
luego
% optimizado por el algoritmo de Campello
[bits_alloc, energy_alloc] = chow_algo(SNR,num_subc,M);

% Forma la tabla de incrementos de energía basados en las ganancias
del
% canal actual para todos los subcanales ara que pueda ser utilizado
por el
% algoritmo de Campello

energytable = EnergyTableInit(SNR,M);

% Algoritmo optimizado
[bits_alloc,energy_alloc] =
campello_algo(bits_alloc,energy_alloc,energytable,total_bits,num_sub
c,M);
```

<precode.m>

```
function [x_pre] = precode(Mt, x_mod, V, N);
% [x_pre] = precode(Mt, x_mod, V, N);
%
% Función que precodifica los símbolos modulados antes de enviarlos
al
% modulo IFFT. Este precodificador multiplica el canal a un tono que
% elimina la matriz unitaria a la derecha del canal descompuesto por
SVD
%
% Mt - número de antenas transmisoras
% x_mod - símbolo modulado para cada tono (los símbolos están
agrupados
% en tonos, ej.: 3 subcanales MIMO por tono, tienen 3
símbolos
% x_mod consecutivos
% V - la matriz unitaria del canal, en orden de fila
% N - número de subcanales

x_pre = [];
for i = 1:N
    x_pre = [x_pre; [V(:, (i-1)*Mt+1:i*Mt)]*transpose(x_mod((i-
1)*Mt+1:i*Mt))];
end

x_pre = transpose(x_pre);
```

<shape.m>

```
function shaped_vals = shape(rec_symbol, Mr, U, N);
%
% Esta función le da forma a los símbolos recibidos antes de su
% demodulación. Esto completa la descomposición del canal en
subcanales
% paralelos
%
% rec_symbol - símbolo recibido después de realizada la IFFT
% Mr - número de antenas receptoras
% U - matriz de matrices unitarias multiplicando a cada subcanal
MIMO
% N - número de subcanales OFDM

shaped_vals = [];
for i = 1:N
    shaped_vals = [shaped_vals; [U(:, (i-
1)*Mr+1:i*Mr)]'*transpose(rec_symbol((i-1)*Mr+1:i*Mr))];
end
shaped_vals = transpose(shaped_vals);
```

<ifft_cp_tx_blk.m>

```
function ofdm_symbol =
ifft_cp_tx_blk(inp_symbol,num_subc,guard_interval)
%-----
% IFFT and Circular Prefix Addition Block for Transmitter
% -----
% (EE359 Project: Andrew and Prateek)
% Entradas :
%     inp_symbol : Símbolo de entrada a todas las subportadoras
%     num_subc   : Número de subportadoras
%     guard_interval : Intervalo guardián basando en el símbolo
OFDM
% Salidas :
%     ofdm_symbol : Salida de IFFT e Intervalo guardián
% -----
ofdm_symbol = ifft(inp_symbol,num_subc);
if (guard_interval > length(ofdm_symbol))
    error('The guard interval is greater than the ofdm symbol
duration');
end

% El símbolo guardián es la copia del final del símbolo ofdm al
principio
% del símbolo ofdm.
guard_symbol = ofdm_symbol(end-guard_interval+1:end);

% Añade el prefijo cíclico al símbolo OFDM
ofdm_symbol = [guard_symbol ofdm_symbol];
```

<fft_cp_rx_blk.m>

```
function rec_symbol =
fft_cp_rx_blk(ofdm_symbol,num_subc,guard_interval)
%-----
% IFFT and Circular Prefix Addition Block for Transmitter
% -----
% (EE359 Project: Andrew and Prateek)
% entradas :
%     ofdm_symbol : símbolo ofdm recibido por el canal
%     num_subc   : número de subportadoras
%     guard_interval : intervalo de guardia basado en el símbolo
ofdm
% salidas :
%     rec_symbol : señal recibida en el dominio de la frecuencia.
% -----
if (guard_interval > length(ofdm_symbol))
    error(' The guard interval is greater than the ofdm symbol
duration ');
end
```



```

% El símbolo guardián (prefijo cíclico) es removido del símbolo ofdm
rec_time_symbol = ofdm_symbol(guard_interval+1:end);

% El FFT de la señal en el dominio del tiempo después de remover el
prefijo
% cíclico
rec_symbol = fft(rec_time_symbol,num_subc);

```

<channel.m>

```

function y = channel(sig2, Mt, Mr, x, H, N);
% function y = channel(sig2, Mt, Mr, x, H, N)
%
% Simulador de transmisión de canal
%
% entradas:
%     sig2 - varianza de ruido
%     Mt - número de antenas Tx
%     Mr - número de antenas Rx
%     x - vector de símbolos de entrada complejos (en MIMO es una
matriz
%         en la que cada columna es el valor de salida de antena
en un
%         solo instante de tiempo
%     H - canal de frecuencia selectiva
%     N - número de símbolos transmitidos en un marco OFDM
%
% salidas:
%     y - vector de canales de salida (matriz MIMO, como la matriz
x)
%         crea un vector de secuencia de ruido (cada fila es una
antena
%         diferente, cada columna es un índice diferente de tiempo

noise = sqrt(sig2)*1/sqrt(2)*(randn(Mt*N,1) + j*randn(Mt*N,1));
y = H*x + noise;

```

```

%CODIGO MATLAB DE LA SIMULACIÓN Zero Forcing 16QAM

% Código en Matlab para la modulación 16 QAM con respecto al BER
en un canal
% Rayleigh fading con 2 Tx, 2Rx siendo este un canal MIMO
% y ecualizador Zero Forcing.

clear
N = 10^5; % numero de bits o símbolos

Eb_N0_dB = [0:30]; % multiple Eb/N0 values
nTx = 2; % numero de antenas transmisoras
nRx = 2; ; % numero de antenas receptoras

alpha16qam = [-3 -1 1 3]; % 16-QAM alphabets

ipHat = zeros(1,N);

%En esta primera parte inicializamos las variables que vamos a
utilizar más adelante en la programación,
%los símbolos a transmitir son 10^6 almacenados en la variable N y
las antenas transmisoras y receptoras son 2 respectivamente.

for ii = 1:length(Eb_N0_dB)
    % Transmisor

    ip = randsrc(1,N,alpha16qam) + j*randsrc(1,N,alpha16qam);
    s = (1/sqrt(10))*ip; % normalization of energy to 1

    sMod = kron(s,ones(nRx,1)); %multiplicamos la matriz s por
una matriz (2x1) que contiene unos
    sMod = reshape(sMod, [nRx,nTx,N/nTx]);

    h = 1/sqrt(2)*[randn(nRx,nTx,N/nTx) + j*randn(nRx,nTx,N/nTx)];
% canal Rayleigh

    n = 1/sqrt(2)*[randn(nRx,N/nTx) + j*randn(nRx,N/nTx)]; % ruido
blanco gaussiano
    % Adición del canal y ruido blanco gaussiano
    y = squeeze(sum(h.*sMod,2)) + 10^(-Eb_N0_dB(ii)/20)*n;

    % Receptor
    % Forming the Zero Forcing equalization matrix W =
inv(H^H*H)*H^H

    hCof = zeros(2,2,N/nTx) ;
    hCof(1,1,:) = sum(h(:,2,:).*conj(h(:,2,:)),1); % aquí obtenemos
el termino d

```

```

    hCof(2,2,:) = sum(h(:,1,:).*conj(h(:,1:)),1); % se obtiene el
termino a
    hCof(2,1,:) = -sum(h(:,2,:).*conj(h(:,1:)),1); % el termino c
    hCof(1,2,:) = -sum(h(:,1,:).*conj(h(:,2:)),1); % y por último
se obtiene el termino b

    hDen = ((hCof(1,1,:).*hCof(2,2:)) -
(hCof(1,2,:).*hCof(2,1:))); % aquí multiplicamos y restamos como se
muestra a continuación: ad-bc
    hDen = reshape(kron(reshape(hDen,1,N/nTx),ones(2,2)),2,2,N/nTx);
% formatting for division

    hInv = hCof./hDen; % se obtiene la inversa inv(H^H*H) para el ZF

    % Forming the MMSE equalization matrix W =
inv(H^H*H+sigma^2*I)*H^H
    % H^H*H is of dimension [nTx x nTx]. In this case [2 x 2]
    % Inverse of a [2x2] matrix [a b; c d] = 1/(ad-bc)[d -b;-c a]
    %obtenemos los términos a, b, c y d los que nos ayudaran a
obtener la
    %matriz inversa

    hCofu = zeros(2,2,N/nTx) ;
    hCofu(1,1,:) = sum(h(:,2,:).*conj(h(:,2:)),1) + 10^(-
Eb_N0_dB(ii)/10); % termino d
    hCofu(2,2,:) = sum(h(:,1,:).*conj(h(:,1:)),1) + 10^(-
Eb_N0_dB(ii)/10); % termino a
    hCofu(2,1,:) = -sum(h(:,2,:).*conj(h(:,1:)),1); % termino c
    hCofu(1,2,:) = -sum(h(:,1,:).*conj(h(:,2:)),1); % termino b

    hDenu = ((hCofu(1,1,:).*hCofu(2,2:)) -
(hCofu(1,2,:).*hCofu(2,1:))); % ad-bc term
    hDenu =
reshape(kron(reshape(hDenu,1,N/nTx),ones(2,2)),2,2,N/nTx); %
formatting for division

    hInvu = hCofu./hDenu; % inv(H^H*H) del MMSE

    hMod = reshape(conj(h),nRx,N); %se realiza la siguiente
operación H^H
    yMod = kron(y,ones(1,2)); % formatting the received symbol for
equalization
    yMod = sum(hMod.*yMod,1); % H^H * y
    yMod = kron(reshape(yMod,2,N/nTx),ones(1,2)); % formatting

    yHat = sum(reshape(hInv,2,N).*yMod,1); % se realiza la siguiente
operación: inv(H^H*H)*H^H*y esto es para el ZF

    yHatu = sum(reshape(hInvu,2,N).*yMod,1); % se realiza la
siguiente operación: inv(H^H*H)*H^H*y para el MMSE

```

```

% Para ZF
y_re = real(yHat); % real
y_im = imag(yHat); % imaginaria

%Para MMSE
y_reu = real(yHatu); % real
y_imu = imag(yHatu); % imaginaria

% receiver - se toma la decisi3n del decodificador para el ZF

ipHat_re(find(y_re < -2/sqrt(10))) = -3;
ipHat_re(find(y_re > 2/sqrt(10))) = 3;
ipHat_re(find(y_re > -2/sqrt(10) & y_re <= 0)) = -1;
ipHat_re(find(y_re > 0 & y_re <= 2/sqrt(10))) = 1;

ipHat_im(find(y_im < -2/sqrt(10))) = -3;
ipHat_im(find(y_im > 2/sqrt(10))) = 3;
ipHat_im(find(y_im > -2/sqrt(10) & y_im <= 0)) = -1;
ipHat_im(find(y_im > 0 & y_im <= 2/sqrt(10))) = 1;
ipHat = ipHat_re + j*ipHat_im;

% receiver - se toma la decisi3n del decodificador para el MMSE

ipHat_reu(find(y_reu < -2/sqrt(10))) = -3;
ipHat_reu(find(y_reu > 2/sqrt(10))) = 3;
ipHat_reu(find(y_reu > -2/sqrt(10) & y_reu <= 0)) = -1;
ipHat_reu(find(y_reu > 0 & y_reu <= 2/sqrt(10))) = 1;

ipHat_imu(find(y_imu < -2/sqrt(10))) = -3;
ipHat_imu(find(y_imu > 2/sqrt(10))) = 3;
ipHat_imu(find(y_imu > -2/sqrt(10) & y_imu <= 0)) = -1;
ipHat_imu(find(y_imu > 0 & y_imu <= 2/sqrt(10))) = 1;
ipHatu = ipHat_reu + j*ipHat_imu;

% contando los errores del Zero Forcing
nErr(ii) = size(find([ip- ipHat]),2);

% contando los errores del MMSE
nErru(ii) = size(find([ip- ipHatu]),2);
end

% La variable ipHat_re almacena los valores reales de la matriz
yHat, la cual es una matriz compleja, donde hay valores imaginarios
y reales.

simBer = nErr/N; % se simula el BER del Zero Forcing

simBeru = nErru/N; % se simula el BER para el MMSE

```

```

EbN0Lin = 10.^(Eb_N0_dB/10);

close all

figure
semilogy(Eb_N0_dB,simBer,'mo-','LineWidth',2); % graffito del ZF
hold on

semilogy(Eb_N0_dB,simBeru,'bp-','LineWidth',2);% grafico del MMSE

axis([0 30 10^-3 1])
grid on
legend('sim (nTx=2, nRx=2, ZF)', 'sim (nTx=2, nRx=2, MMSE)');
xlabel('Average Eb/No,dB');
ylabel('Bit Error Rate');
title('BER for 16QAM modulation with 2x2 MIMO, ZF and MMSE
equalizer (Rayleigh channel)');

```

BIBLIOGRAFÍA

- [1] Dr. Edel Garcia, Singular Value Decomposition (SVD) A Fast Track Tutorial, September 11, 2006.
- [2] M. Petrou, P. Bosdogianni, "Image Processing: The Fundamentals", Editorial Wiley Primera Edición, 1999.
- [3] Ersin Sengul, Enis Akay, Ender Ayanoglu, "Diversity Analysis of Single and Multiple Beamforming", University of California, Junio 2006
- [4] Bengt Holter, "On the capacity of the MIMO channel", Norwegian University of Science and Technology, Mayo 2002.
- [5] Hakju Lee, Myeongcheol Shin, Chungyong Lee, "An Eigen-based MIMO Multiuser Scheduler with Partial Feedback Information", IEEE Communications Letters, Vol. 9, No. 4, Abril 2005.
- [6] Oscar Fernández, "Caracterización Experimental y Modelado de Canal MIMO para aplicaciones WLAN", Universidad de Cantabria, Mayo 2007.
- [7] Holger Busche, Alexandre Vanaev, Hermann Rohling, "SVD based MIMO Precoding and Equalization Schemes for Realistic Channel Estimation Procedures", Wireless Personal Communications, Volume 48, Number 3, 347-359, Junio 2008.
- [8] Luna O. Marco, Sánchez G. Jaime, "CDMA Multiportadora en redes inalámbricas de banda ancha para interiores", Instituto Tecnológico de Chihuahua, 2001.

- [9] Viviane Molineros Guevara, "Modelamiento y Simulación de Sistemas MIMO". Escuela Superior Politécnica del Litoral, Marzo 2010.
- [10] Prateek Bansal, Andrew Brzezinski, "Adaptive Loading in MIMO/OFDM Systems", Stanford University, 2001.
- [11] Paul Thomas, Ogunfunmi Tokunbo, "Wireless LAN Comes of Age: Understanding the IEEE 802.11n Amendment," Circuits and System Magazine, 28-54, IEEE, Primer Cuarto 2008.
- [12] W. Yu, J.M. Cioffi, "On Constant Power Water-filling", IEEE International Conference on 2001, Communications, 2001.
- [13] P. Chow, "A Practical Discrete Multitone Transceiver Loading Algorithm for Data Transmission over Spectrally Shaped Channels," IEEE Trans. Comm, Vol. 43, No.2, Paginas 773-775, Febrero 1995.
- [14] J. Campello de Souza, "Discrete Bit Loading for Multicarrier Modulation System," Stanford University, Agosto 1998.
- [15] V.Kuhn, "Wireless Communications over MIMO Channels - Applications to CDMA and Multiple Antenna Systems" ", Editorial Wiley Primera Edición, 2006.
- [16] I. Santamaría, "Máster y Doctorado en Tecnologías de la Información y Comunicaciones en Redes Móviles", Universidad de Cantabria, Octubre 2010.