Escuela Politécnica Superior de Linares

Grado en Ingeniería de Tecnologías de Telecomunicación

UNIVERSIDAD DE JAÉN

Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

APLICACIÓN DIDÁCTICA PARA EL ANÁLISIS DE MODULACIONES DIGITALES EN UN SISTEMA DE COMUNICACIÓN DIGITAL

Alumno: **Juan Antonio Encinas Díaz**

Tutor: D. Francisco Jesús Cañadas Quesada

Depto.: Ingeniería de Telecomunicación

Septiembre, 2022

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En este Trabajo de Fin de Grado se propone la creación de una herramienta didáctica que permita facilitar el estudio y comprensión de los distintos tipos de modulaciones digitales estudiadas en la asignatura de Teoría de la comunicación, la cual se imparte en la Escuela Politécnica Superior de Linares, perteneciente a la universidad de Jaén (UJA).

La herramienta creada se compone de las siguientes modulaciones digitales estudiadas en dicha asignatura:

- PSK.
- QAM.
- FSK.
- MSK.
- DBPSK.

La creación de esta herramienta se ha desarrollado mediante la creación de un Guide usando el programa Matlab, que ha permitido crear un entorno visual en donde se han creado varios algoritmos que permiten modular y demodular tanto imágenes como secuencias de bits en la modulaciones anteriormente citadas, mostrando, comparando y arrojando diferentes resultados que proporcionan estas como, por ejemplo las representaciones de las distinta señales moduladas enviadas y recibidas, diagramas de constelación, espectros en frecuencia, grafica de BER y resultados de señales demoduladas.

GLOSARIO DE ACRÓNIMOS

AM	*Amplitude Modulation*	(modulación en amplitud)
AMPS	*Advanced Mobile Phone System*	(sistema telefónico móvil avanzado)
BER	*Bit Error Rate*	(ratio de bits erróneos)
CDMA	*Code Division Multiple Access*	(código de división de acceso múltiple)
CDPD	*Cellular Digital Packet Data*	(datos de paquetes digitales celulares)
DECT	*Digital Enhanced Cordless Telecommunications*	(telecomunicaciones inalámbricas digitales mejoradas)
DPSK	*Differential Phase Shift Keying*	(manipulación por desplazamiento de fase diferencial)
DVB	*Digital Video Broadcasting*	(retransmisión de vídeo digital)
DVB-S	*Digital Video Broadcasting Satellital*	(retransmisión de vídeo digital satélite)
DVB-T	*Digital Video Broadcasting Terrestrial*	(retransmisión de vídeo digital terrestre)
DVC-C	*Digital Video Broadcasting Cable*	(retransmisión de vídeo digital cable)
FM	*Frequency Modulation*	(modulación en frecuencia)
FSK	*Frequency Shift Keying*	(modulación por desplazamiento de frecuencia)
GSM	*Global System For Mobile Communications*	(sistema global para las comunicaciones móviles)
MMDS	*Mutichannel Multipoint Distribution Service*	(servicio multicanal de distribución multipunto)
MSK	*Minimum-Shift Keying*	(modulación por desplazamiento mínimo)
NRZ	*Non-Return-To-Zero*	(no retorno a cero)
PSK	*Phase-Shift Keying*	(modulación por desplazamiento de fase)
QAM	*Quadrature Amplitude Modulation*	(modulación de amplitud en cuadratura)
RAM	*Random Access Memory*	(memoria de acceso aleatorio)
RZ	*Return-To-Zero*	(retorno a cero)
WLAN	*Wireless Local Area Network*	(red inalámbrica local)

Capítulo 1.

1.INTRODUCCIÓN.

Las telecomunicaciones surgieron con el descubrimiento de la electricidad y de las ondas electromagnéticas lo que permitió una nueva forma de transmisión de información a través de señales electromagnéticas, lo que supuso una revolución. Es en este punto donde nace el término "Telecomunicación", este término fue acuñado en el año 1904 por el ingeniero francés Édouard Estaunié. Este término se compone del nexo– de la palabra latina *communicare*, que significa compartir, con el prefijo griego *tele-*, que significa distancia. El nuevo termino se creó para denominar la transmisión de información a distancia mediante el uso de la electricidad. [7]

Una señal es un flujo de información proveniente de una fuente, la cual puede tener una naturaleza diversa: mecánica, óptica, magnética, eléctrica, acústica… Se tratan de señales eléctricas modeladas por medio de funciones matemáticas como por ejemplo una sinusoide. Surgen nuevas disciplinas de procesado de señales, que se trata de una disciplina que estudia y desarrolla técnicas de tratamiento de señales como por ejemplo amplificación, filtrado, clasificación… para mejorar la calidad y optimización de las comunicaciones.

Las primeras telecomunicaciones empleaban señales con valores continuos, es decir, con números o valores infinitos, como por ejemplo la captación de un micrófono o la temperatura de un termómetro, es aquí es donde surgen los primeros sistemas analógicos. Un sistema analógico es cualquier sistema cuyas señales se representan con valores continuos, es decir, que admite números o valores infinitos.

Posteriormente surge el primer sistema digital con la invención de la FM digital de espectro eficiente por Dekker y de Jager en el año 1978 [8]. La invención de esto supuso una auténtica revolución y tras este se propusieron muchos métodos de modulación digital en un período muy corto como pueden ser por ejemplo la modulación MSK o QPSK entre otras.

Los sistemas digitales se caracterizan en que solo admite señales con valores discretos, es decir, que solo admite señales con un conjunto limitado de números. Esto se logra gracias a la frecuencia de muestreo que es el número de muestras por unidad de tiempo que se toman de una señal continua para producir una señal discreta, durante el proceso necesario para convertirla de analógica en digital.

Los sistemas digitales, como por ejemplo el ordenador, usan la lógica de dos estados representados por dos niveles de tensión eléctrica, uno alto, H y otro bajo, L (de High y Low, respectivamente, en inglés). Por abstracción, dichos estados se sustituyen por ceros y unos, lo que facilita la aplicación de la lógica y la aritmética binaria. Si el nivel alto se representa por 1 y el bajo por 0, se habla de lógica positiva y en caso contrario de lógica negativa.

Las principales ventajas que se introdujeron con los sistemas digitales frente a los sistemas analógicos son: mayor tolerancia al ruido, menor degradación de la señal, mayor precisión, almacenamiento de la información menos costoso… pero también tiene desventajas como, por ejemplo: perdida de información durante el muestreo, sistemas más caros y complejos y también se requiere tiempo durante los procesos de conversión.

1.1. Motivación

De la necesidad de procesado de señales surge la motivación para la realización de este Trabajo de Fin de Grado, el cual se centra especialmente en el la modulación se señales digitales las cuales son un núcleo de los sistemas de telecomunicaciones actuales ya que se encuentran en muchos ámbitos como, por ejemplo: móviles, GPS, Televisión.

La modulación digital es muy importante para estos sistemas ya que posibilita importantes mejoras como, por ejemplo:

- Mejor aprovechamiento del canal de comunicación, lo que posibilita transmitir más información de forma simultánea gracias a la codificación y al multiplexado.
- Mejor aprovechamiento del espectro electromagnético, ya que permite multiplexación por frecuencias.
- Añadir robustez y mejorar la resistencia contra posibles ruidos e interferencias del medio.
- A determinadas frecuencias mejora la eficiencia en la trasmisión en función del medio en el que se emplee.
- En caso de transmisiones inalámbricas, permite disminuir las dimensiones de las antenas de transmisión ya que el tamaño de estas depende de la longitud de onda, por eso a mayores frecuencias menor tamaño de antenas.

A lo largo de mis estudios en el Grado de Ingeniería en Telecomunicaciones he desarrollado un interés particular sobre esta temática, observando, de manera guiada por mi tutor, la posibilidad de desarrollar una aplicación didáctica que permita a futuros estudiantes comprender de manera más eficiente parte de las modulaciones digitales existentes.

1.2. Organización del TFG

Capítulo 1. Introducción explicando historia de las telecomunicaciones y señales, surgimiento del mundo analógico y digital, motivaciones para el desarrollo de este TFG.

Capítulo 2. Objetivos a realizar en este TFG

Capítulo 3. Explicación teórica y estructura de los sistemas de telecomunicación, explicación y parámetros de las señales temporales y señales en frecuencia, breve introducción del concepto modulación y ruido e interferencias.

Capítulo 4. Desarrollo teórico de los diferentes tipos de modulación digitales, explicación del concepto coherencia, antecedentes y aplicaciones de los diferentes tipos de modulaciones digitales.

Capítulo 5. Elección del programa Matlab para el desarrollo de este TFG, Implementación de las modulaciones expuestas en el capítulo 4 en Matlab, problemas que han surgido durante el desarrollo de este TFG.

Capítulo 6. Resultado y observaciones de las simulaciones ejecutadas en Matlab con el código implementado en el capítulo 5.

Capítulo 7. Conclusiones observadas de las simulaciones realizadas en el capítulo 6.

Capítulo 8. Exposición de posibles desarrollos futuros para este TFG y posibles ampliaciones.

Capítulo 9. Código de Matlab implementado para la realización de este TFG y contiene manual de usuario de los diferentes menús implementados utilizando la función guide de Matlab.

Capítulo 2.

2. OBJETIVOS

El objetivo general de este Trabajo Fin de Grado es la creación de una herramienta visual y auditiva para facilitar al alumnado la compresión de las diferentes técnicas de modulación digital como M-PSK, M-QAM, M-FSK, MSK y DBPSK.

El presente Trabajo Fin de Grado se ha desarrollado en distintas etapas. Estas pueden sintetizarse en los siguientes objetivos específicos, de los cuales se destacan:

- Recopilación y estudio de material bibliográfico relacionado con modulaciones digitales.
- Implementación y análisis del modelo matemático en el dominio temporal.
- Implementación y análisis del modelo matemático en el dominio de la frecuencia.
- Implementación y análisis de las constelaciones digitales transmitidas y recibidas.
- Validación de las implementaciones en el dominio temporal y espectral.
- Evaluación de las diferentes modulaciones digitales en diferentes escenarios de comunicación.
- Desarrollo e implementación de una interfaz didáctica enfocada al uso por el alumnado de Grado en Ingeniería de Tecnologías de Telecomunicación y Grado en Ingeniería Telemática, con las funcionalidades adecuadas, que le permita analizar el comportamiento de la modulación en tiempo, frecuencia y constelación variando los parámetros más importantes que caracterizan a dicha modulación.
- Creación una aplicación a través de Matlab en la que, a partir de las modulaciones anteriores, se pueda analizar cada señal modulada asociada a una señal de usuario, en el dominio temporal y espectral mediante diversos parámetros del sistema (régimen binario, frecuencia de portadora, ...), probabilidad de error y análisis de las constelaciones transmitidas y recibidas simulando diferentes escenarios de comunicación. La aplicación permitirá realizar una comparativa entre diferentes modulaciones atendiendo a conceptos como ancho de banda, probabilidad de error de bit.
- Redacción de la memoria acorde a la normativa de la Escuela Politécnica Superior de Linares.

Capítulo 3.

3. FUNDAMENTOS DE LOS SISTEMAS DE TELECOMUNICACIÓN

3.1. Estructura de un sistema de comunicación

La estructura de un sistema de comunicación se compone de varios elementos [7]:

- Fuente de información: produce un mensaje o secuencia de mensajes para ser comunicados al receptor.

- La información o mensaje: Es la información que tratamos de transmitir, debe llegar de manera fidedigna y eficaz. Existe dos tipos:
    - Analógica: soportan la información sobre una señal que puede modelarse como una función continua en el tiempo y amplitud.
    - Digital: generan señales digitales (discretas en tiempo y amplitud). Las señales analógicas pueden transformarse en digitales mediante muestreo. Cuantificación y codificación de muestras.

- El transmisor de información: Es aquel que contiene una la información y es el encargado de transformar y realizar la codificación de esta. También se encarga de su adaptación a la naturaleza del canal para que pueda ser enviada con la mayor calidad posible. Algunas funciones del transmisor son:
    - Amplificación de potencia.
    - Filtrado.
    - Adaptación de la señal al medio (modulación).
    - Protección frente a ruido e interferencias.
    - Compartición del medio de transmisión (multiplexación).
    - Simplificación del proceso de transmisión.

- Medio de transmisión: Es el elemento a través del cual se envía la información del transmisor al receptor y está compuesto por el medio físico que existe entre los dos intervinientes, como por ejemplo cableado, aire o agua. El medio tiene obstáculos que impiden o merman la comunicación tales como el ruido y las interferencias. Los medios de transmisión pueden clasificarse como:
    - Guiados (fibra óptica, cableado).
    - No guiados (radio, microondas).

- El receptor: Es el elemento al que está destinada la información, este realiza el proceso inverso al transmisor para obtener la información original. Es decir, es quien capta la información.

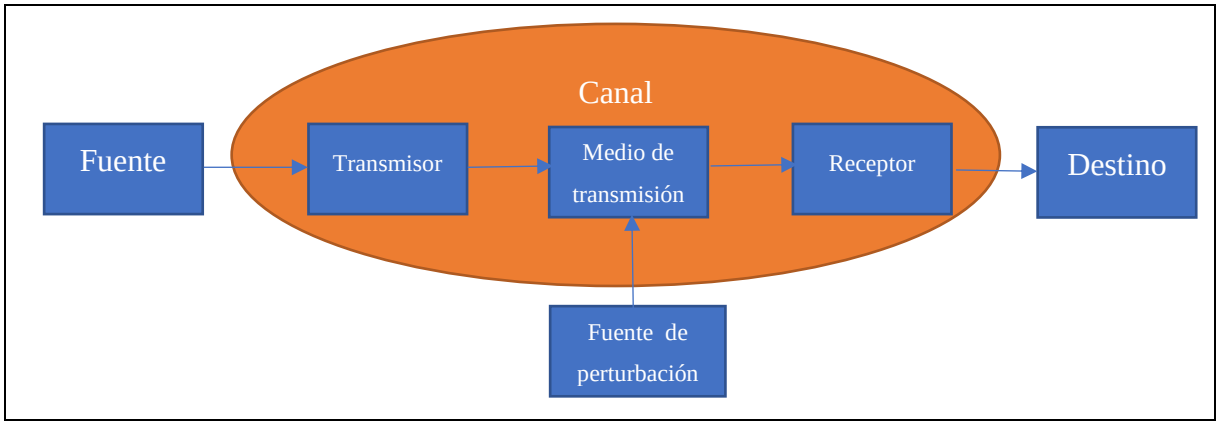


Figura 3.1 Estructura básica de un sistema de comunicación

3.2. Definición de señal temporal

Las señales temporales son aquellas generadas por algún fenómeno electromagnético a lo largo del tiempo, se tratan de una variación de una magnitud física en el tiempo. Por ejemplo: señales de audio, señales de vídeo o ruido ambiental, Es decir, están definidos en todos los instantes o durante un determinado periodo de tiempo.

Una señal periódica se trata de una señal temporal en la cual se encuentra un patrón de repetición después de un determinado tiempo, es decir, que después de un tiempo vuelven a repetirse los valores anteriores una y otra vez, a estos intervalos se les denomina periodo.
La frecuencia, cantidad de periodos por segundo y se mide en Hercios (Hz). Por lo que se implica que la frecuencia es el inverso del periodo. Un ejemplo de señal periódica:

$$x(t) = A * \sin(2\pi f_0 t + \varphi) \qquad (1)$$

3.2.1. Parámetros de una señal temporal.

Los parámetros que componen una señal temporal son:

- Amplitud: es el valor máximo que esta puede llegar a alcanzar y su magnitud suele medirse en voltios (V) o decibelios (dB).

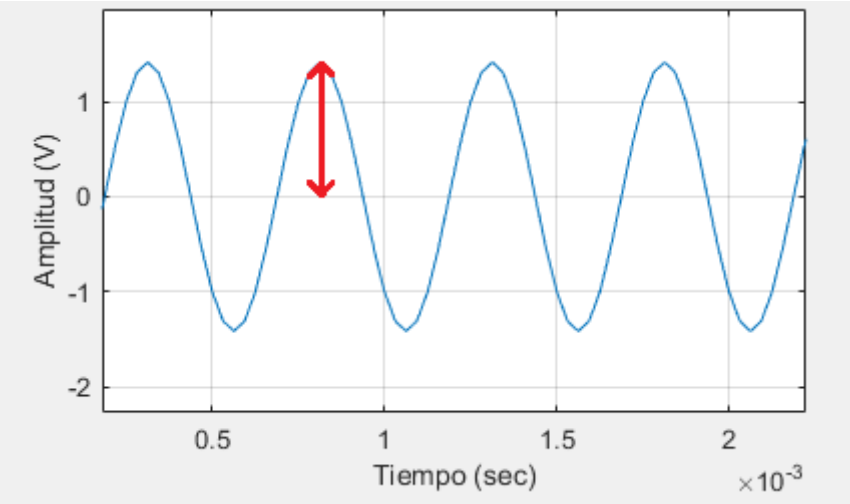


Figura 3.2.1.1 Ejemplo de amplitud de señal periódica.

Se puede observar en el ejemplo que la amplitud es de 1.3v (voltios).

- Periodo: el tiempo en el que se vuelve a repetir la forma de onda a partir de un punto de referencia de la señal. Se calcula mediante la inversa de la frecuencia.

  Periodo (seg) = $\frac{1}{f_0}$.

- Frecuencia: número de periodos por segundo de la señal y se mide en Hz

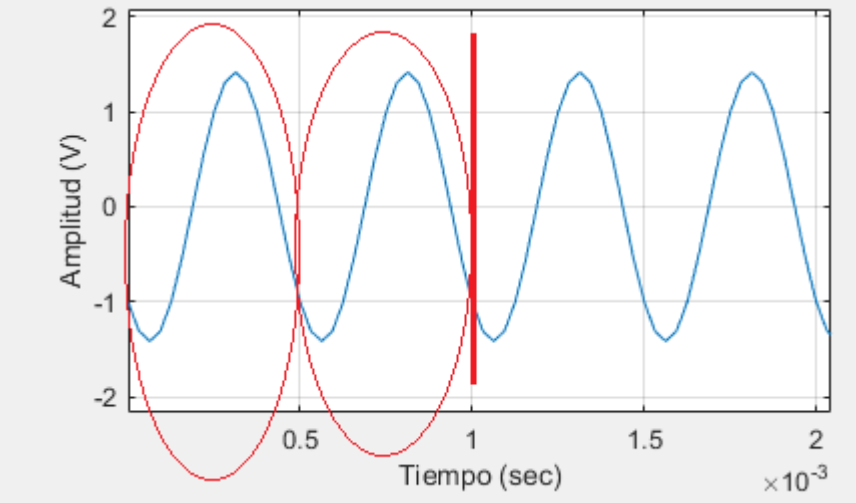


Figura 3.2.1.3 Ejemplo de frecuencia de señal periodica

Se puede observar en el ejemplo que tenemos una frecuencia de 2 ciclos por segundo.

- Fase: se refiere a su desplazamiento hacia la derecha o la izquierda con respecto a una referencia. Se mide en grados (°) o radianes (rad). Sería en valor φ de la función del ejemplo.

Ejemplo de señal periódica:

$$x(t) = A * \sin(2\pi f_0 t + \varphi)$$

$$A = 5\,V, \quad f_0 = 3\,Hz, \quad \varphi = \pi\,(rad)$$

$$x(t) = 5 * \sin(2\pi 3t + \pi)$$

Observamos que la Amplitud son 5v

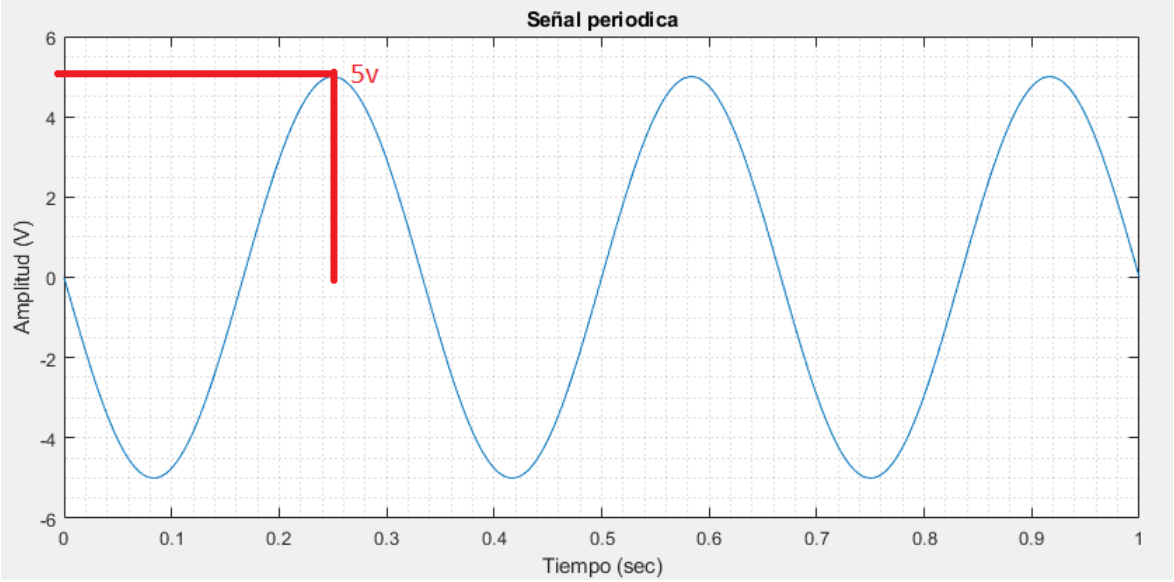


Figura 3.2.1.6 Ejemplo de amplitud de la señal periódica.

Podemos observar 3 ciclos por segundo. 3HZ

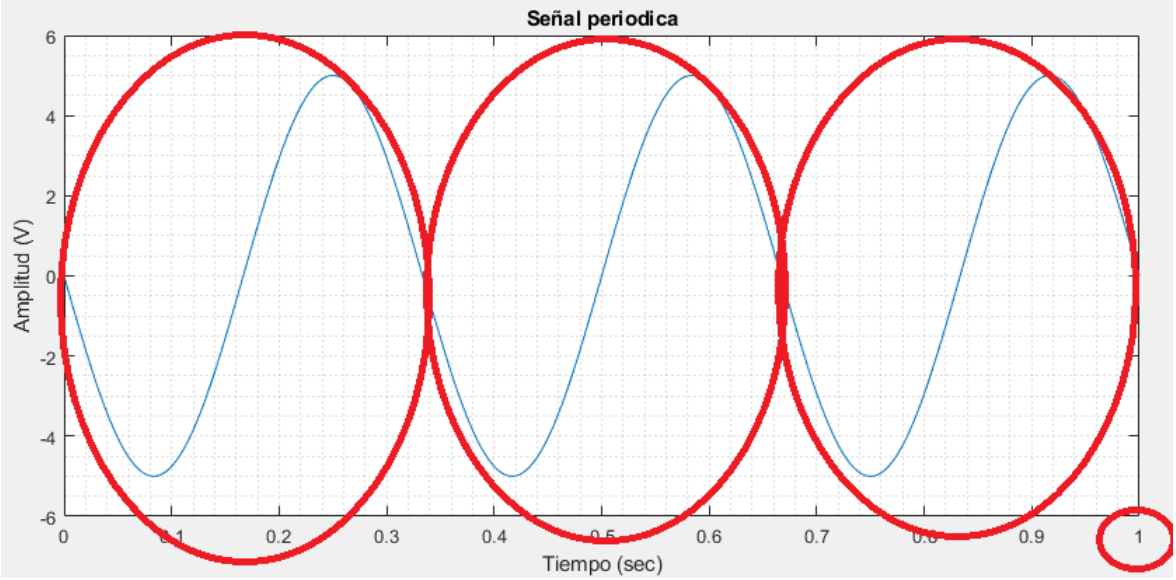


Figura 3.2.1.7 Ejemplo de frecuencia de la señal periódica.

Podemos observar el periodo (seg) = $\frac{1}{f_0} = \frac{1}{3} = 0.33\ s$.

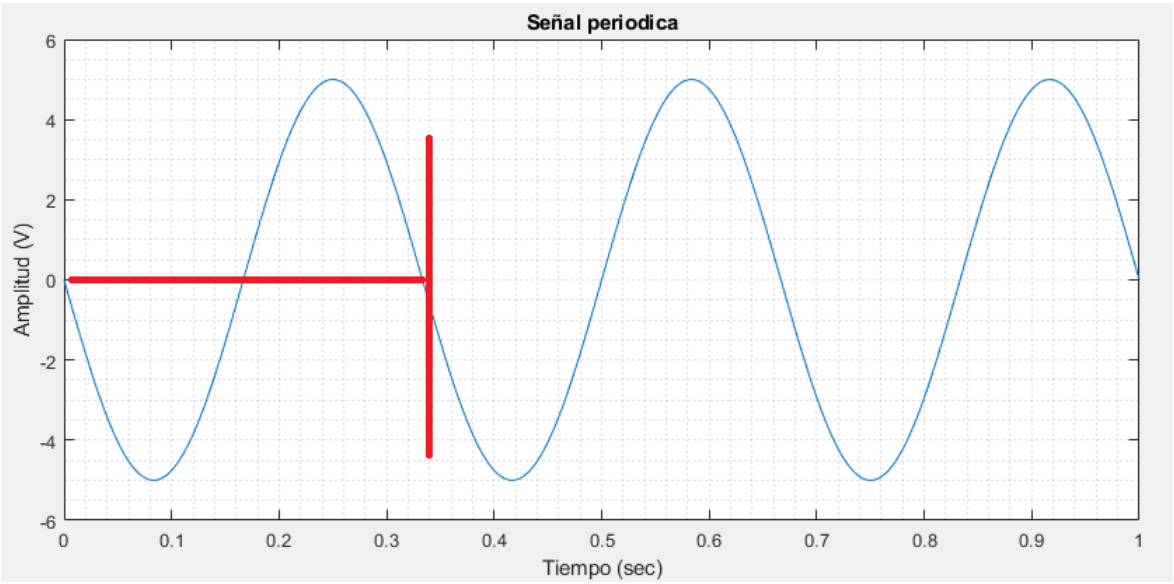


Figura 3.2.1.8 Ejemplo de periodo de la señal periódica.

Podemos el desfase de $\varphi = \pi\ (rad)$ respecto a un seno.

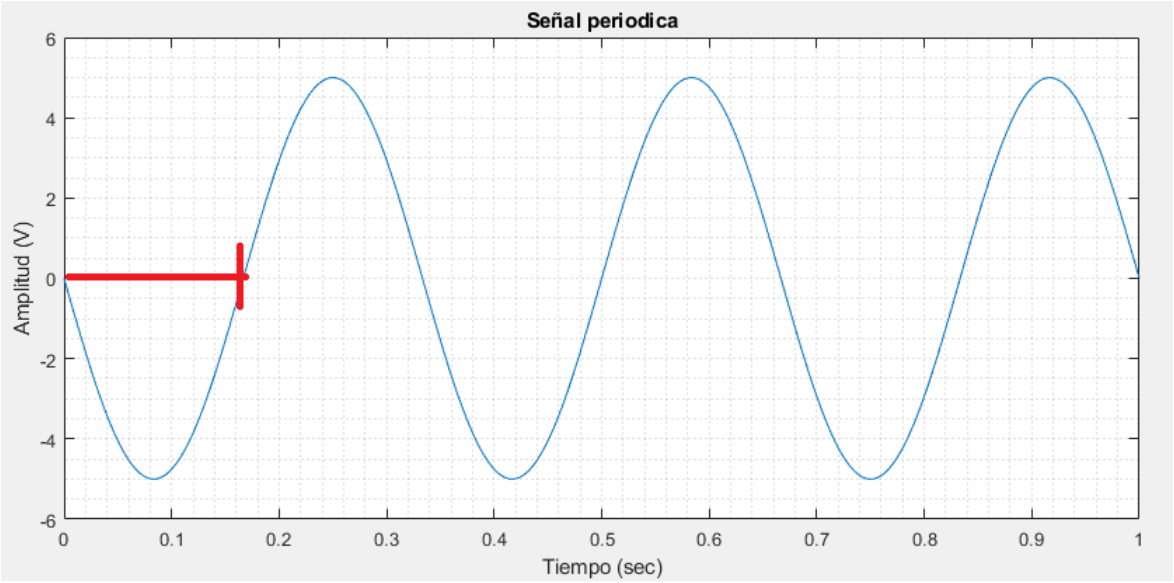


Figura 3.2.1.9 Ejemplo de fase de la señal periódica.

3.2.2. Muestreo

El muestreo se trata de un proceso que se realiza a las señales que son continuas en el tiempo como por ejemplo las señales analógicas, este proceso consiste en la selección de un número determinado de valores discretos los cuales se toman en unos instantes determinados o en un tiempo concreto de la señal continua en el tiempo, convirtiéndola en una señal con valores discretos. Esto se consigue mediante la multiplicación de la señal que se quiere muestrear con un tren de deltas

Tras realizar este proceso la señal obtenida sólo tendrá valores en instantes de tiempo en concreto (valores discretos de la señal). El número de muestras por segundo se conoce como frecuencia de muestreo (f_s) la cual viene condicionada por el ancho de banda de la señal analógica (Teorema de Nyquist).

El Teorema de Muestreo de Nyquist explica la relación entre la velocidad de muestreo y la frecuencia de la señal medida. Afirma que la velocidad de muestreo fs debe ser mayor que el doble del componente de interés de frecuencia más alto en la señal medida. Esta frecuencia por lo general se conoce como la frecuencia Nyquist, f_N.

$$f_s > 2 * f_N$$

La frecuencia de muestreo tiene una incidencia directa sobre la cantidad de recursos necesarios en el almacenaje (espacio en disco) y la transmisión (ancho de banda). El no cumplimiento del Teorema de Nyquist introduce un error llamado solapamiento. El concepto de muestreo temporal se puede entender como una multiplicación de la señal original con un tren de deltas (caso ideal) o un tren de pulsos (caso real).

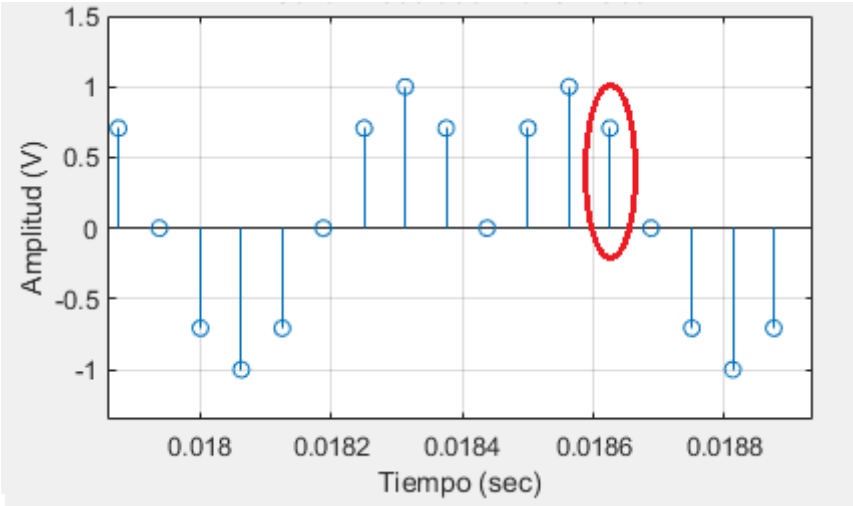


Figura 3.2.2 Ejemplo de muestreo de señal temporal

las señales de naturaleza analógica son imposibles de manipular por los sistemas digitales, por lo tanto, la finalidad del muestreo es transformar señales analógicas en digitales para tal fin.

3.3. Definición de señal en frecuencia

La definición de una señal en frecuencia se trata de un método que permite el análisis de señales respecto a su frecuencia, esto se consigue mediante la Transformada de Fourier.

- Ecuación de síntesis:

$$x(t) = \sum_{n=-\infty}^{\infty} c_n \cdot e^{j \cdot \mathrm{n} \cdot \frac{2\pi}{T_0} \cdot \mathrm{t}} \qquad (2)$$

- Ecuación de análisis:

$$c_n = \frac{1}{T_0} \int_{T_0} x(t) \cdot e^{j \cdot \mathrm{n} \cdot \frac{2\pi}{T_0} \cdot \mathrm{t}} dt\,, \qquad n = 0, \pm 1, \pm 2 \ldots \qquad (3)$$

La transformada de Fourier, denominada por su descubridor Joseph Fourier, es una transformación matemática empleada para transformar señales entre el dominio del tiempo (o espacial) y el dominio de la frecuencia mediante la siguiente ecuación.

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi f t} dt \qquad (4)$$

Mientras que un gráfico en el dominio del tiempo muestra la evolución de una determinada señal a lo largo del tiempo, un gráfico en el dominio de la frecuencia muestra las componentes de dicha señal según sus frecuencias.

Ejemplo de señal en el dominio del tiempo y su transformada de Fourier (señal en frecuencia).

$$x1(t) = \sin(2\pi 1t) \qquad x2(t) = 2 * \sin(2\pi 3t)$$

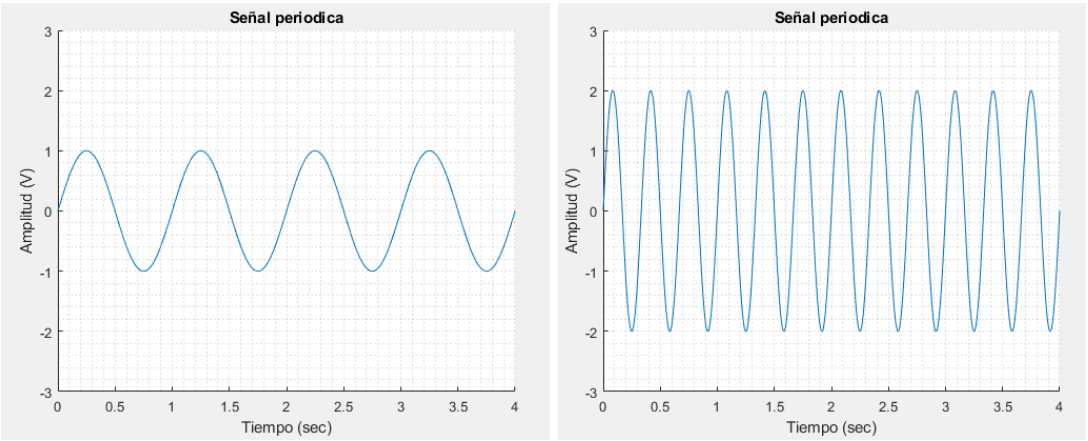


Figura 3.3.1 Señales x_1(t) y x_2(t) en el dominio del tiempo.

$$x(t) = x1(t) + x2(t)$$

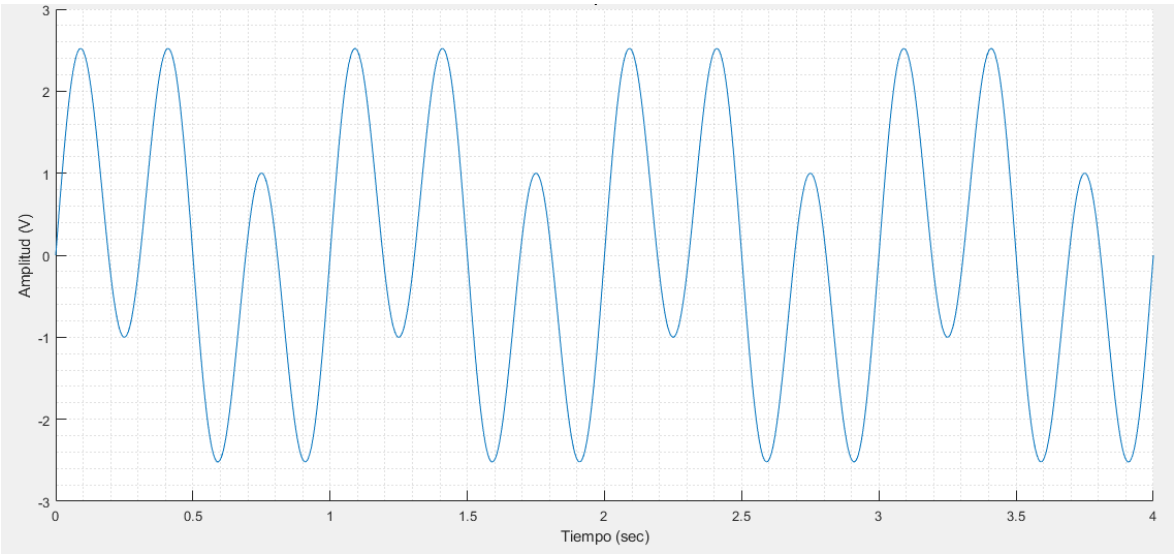


Figura 3.3.2 Señal en el dominio del tiempo $x_{(t)}$

Aplicamos ahora la transformada de Fourier para obtener la señal anterior en el dominio de la frecuencia mediante:

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt \qquad (5)$$

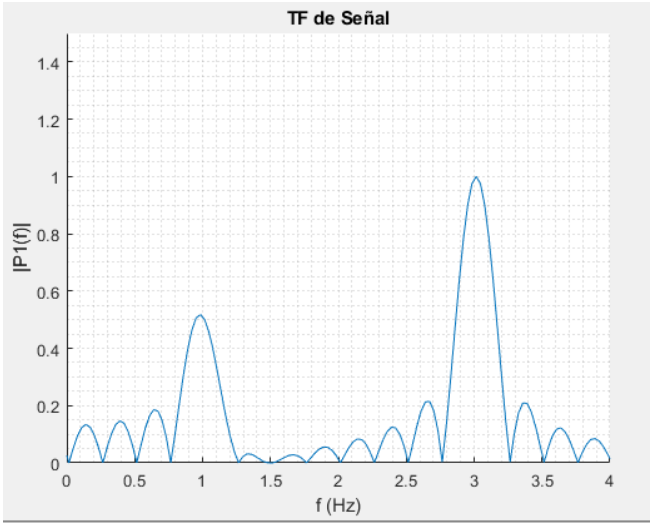


Figura 3.3.3 Señal en el dominio de la frecuencia $X(f)$

Podemos observar ambas señales en frecuencia, una de ellas con f=1hz y con menos amplitud y la otra con f=3 Hz con el doble de amplitud que la anterior.

3.3.1. Parámetros

- Frecuencia: Se basa en la descomposición de una señal en términos de un conjunto de funciones base (sinusoides de diferente frecuencia).

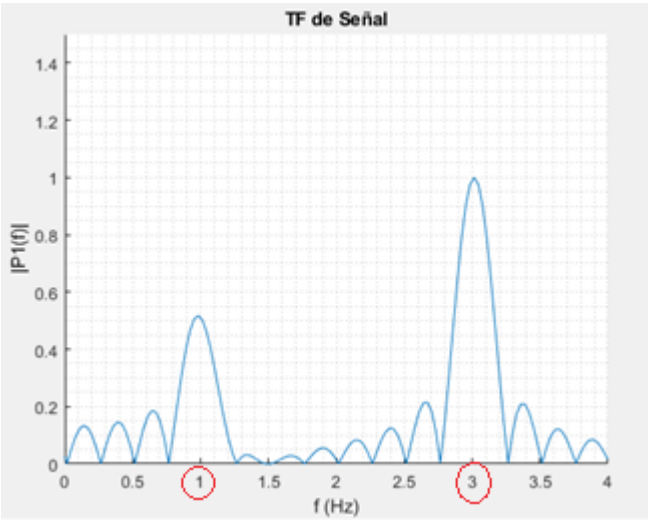


Figura 3.3.1.1 Parámetro frecuencia en $x_{(f)}$

Podemos observar que en las frecuencias 1hz y 3hz tienen mayor amplitud que el resto debido a las dos sinusoides sumadas anteriormente.

- Módulo: El módulo o amplitud de la respuesta en frecuencia (o respuesta en amplitud) representa la ganancia del sistema a cada pulsación ω o componente espectral.

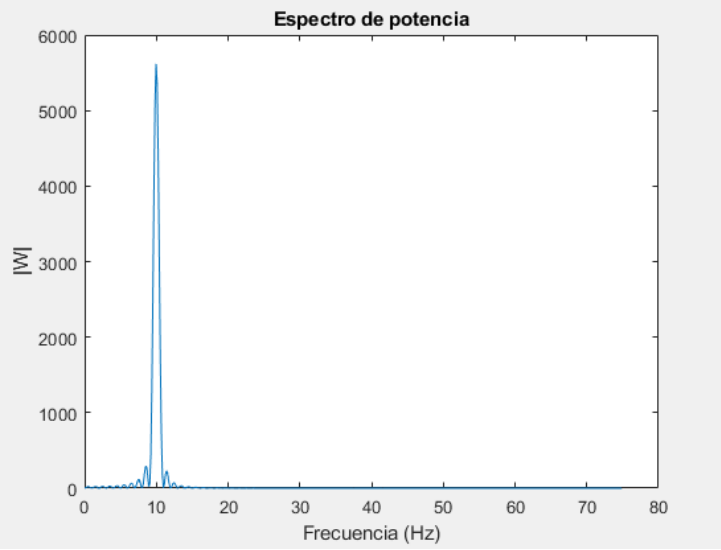


Figura 3.3.1.2 espectro de amplitud.

- Fase: El espectro de fase consiste en la representación en diagrama de fase de todos los fasores que componen la función $f(t)$.

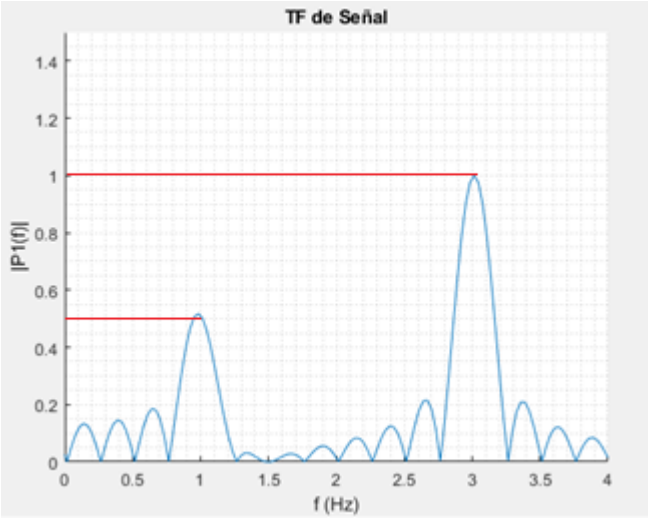


Figura 3.3.1.3 Espectro de fase.

3.3.2. Espectro de potencia

El espectro de potencia describe la distribución de potencia en los componentes de frecuencia que componen una determinada señal. Según el análisis de Fourier, cualquier señal física se puede descomponer en varias frecuencias discretas o en un espectro de frecuencias en un rango continuo. El promedio estadístico de una señal (incluido el ruido), analizado en términos de su contenido de frecuencia, se denomina espectro.

$$S_{xx}(f) = |X(f)|^2 \qquad (6)$$

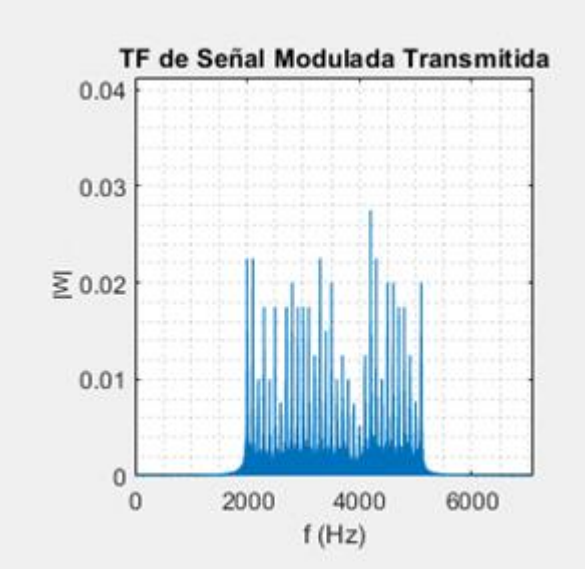


Figura 3.3.2.1 Ejemplo de espectro de potencia

Cuando la energía de la señal se concentra alrededor de un intervalo de tiempo finito, especialmente si su energía total es finita, se puede calcular la densidad espectral de potencia. La densidad espectral de potencia se refiere entonces a la distribución de energía espectral que se encontraría por unidad de tiempo, ya que la energía total de dicha señal en todo el tiempo generalmente sería infinita. La Sumatoria de las potencias con sus componentes espectrales nos indica la potencia media total, como dicta el teorema de Parseval.

3.3.3. Ancho de banda

El ancho de banda se trata de la diferencia entre las distintas frecuencias que van desde la parte superior e inferior en una banda continua. Esta siempre se mide en Hercios (Hz). El ancho de banda de paso es la diferencia entre las frecuencias de corte superior e inferior de, por ejemplo, un espectro de señal o un canal de comunicación.

Ancho de banda: es el rango de frecuencias que componen una señal temporal. La magnitud se mide en Hertzios (Hz).

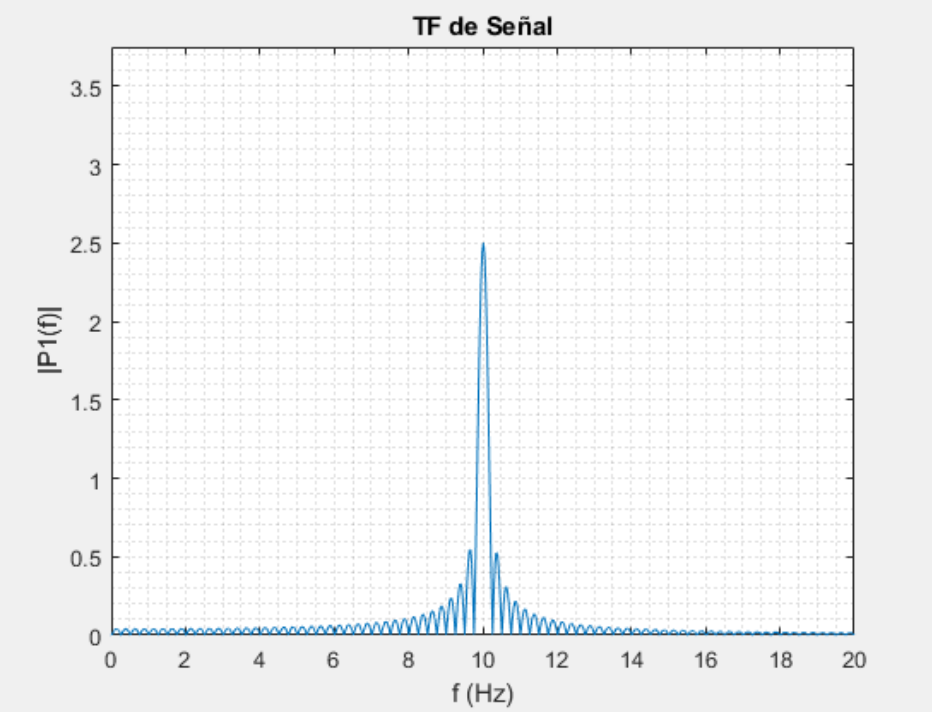


Figura 3.3.3.1 Ancho de banda de tono puro de 10 Hz

Una característica del ancho de banda es que en cualquier banda de un ancho determinado puede contener y transportar la misma cantidad de información, sin distinción de la frecuencia dónde se encuentre esa banda en el espectro. Por ejemplo, en una banda de 2 kHz puede transportar una determinada información o una videollamada ya sea en la banda base (2 kHz) o modulándola dicha videollamada a una frecuencia mucho mayor.

Esto se hace porque trabajando en los anchos de banda más amplios, estos son más fáciles de procesar a frecuencias más altas dado que el ancho de banda fraccional es más pequeño.

Existen diferentes tipos de ancho de banda:

- Ancho de banda absoluto: Este se define como el rango de frecuencias en el que la densidad espectral de potencia es distinta de cero.

- Ancho de banda de nulo a nulo: Este ancho de banda consta de mantener la ocupación espectral entre sus primeros valores nulos en ambas bandas, lo que significa mantener el ancho del lóbulo de espectro principal.

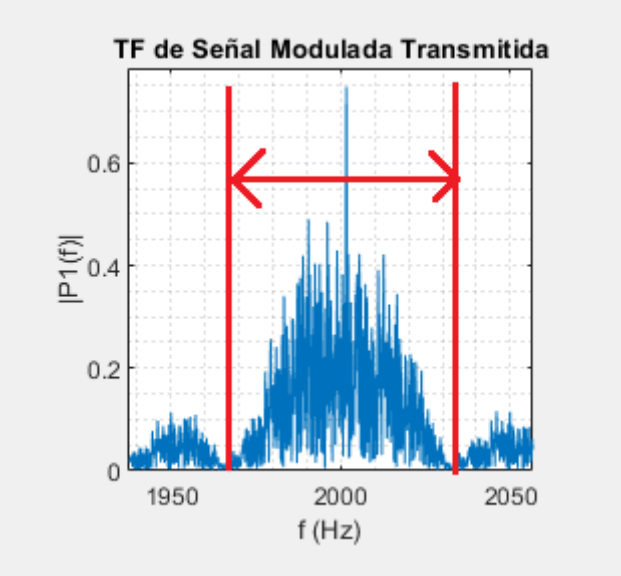


Figura 3.3.3.3 Ejemplo de ancho de banda nulo a nulo

- Ancho de banda de media potencia: Esta se define como el intervalo entre frecuencias en las que la densidad espectral de potencia cae a la mitad de la potencia, o 3dB por debajo del valor pico. Esta también puede definirse como Ancho de banda de 3dB. Esta representa la limitación del menor ancho de banda posible.

- Ancho de banda con base al límite de la densidad espectral de potencia: Este método especifica que el ancho de banda se tome hasta un valor donde la DEP cae por debajo del nivel encontrado para el centro de la banda, sus niveles típicos son de 35 a 50dB.

3.4. Concepto de modulación

La modulación es un proceso cuyo propósito es adherir una determinada información contenida en una señal moduladora en otra señal llamada señal portadora para transmitir dicha información a distancia. Este proceso consiste en modificar una o más características de la señal portadora dando como resultado una tercera señal llamada señal modulada que es similar a la señal portadora, pero contiene la información adherida en ella.

El modulador es un dispositivo o circuito encargado de realizar la modulación, por el contrario, el demodulador es un circuito que realiza la demodulación que es el proceso inverso de la modulación y permite recuperar la señal moduladora que contiene la información en la forma origina de esta.

Los sistemas más recientes usan modulación digital, que añade una señal digital que consta de una secuencia de dígitos binarios (bits) en el portador. En la modulación por modulación por desplazamiento de frecuencia (FSK), utilizada en buses informáticos y telemetría, la señal portadora se desplaza periódicamente entre dos frecuencias que representan los dos dígitos binarios.

3.4.1. Moduladora, portadora, modulada

Moduladora: es la señal que contiene la información a transmitir. Dicha información puede tratarse de una señal sinusoidal analógica como, por ejemplo, una onda de sonido puro, una señal analógica de vídeo que represente imágenes en movimiento de una cámara, una señal de audio de micrófono o puede tratarse de una señal digital que contenga una secuencia de dígitos binarios o un flujo de bits de un ordenador.

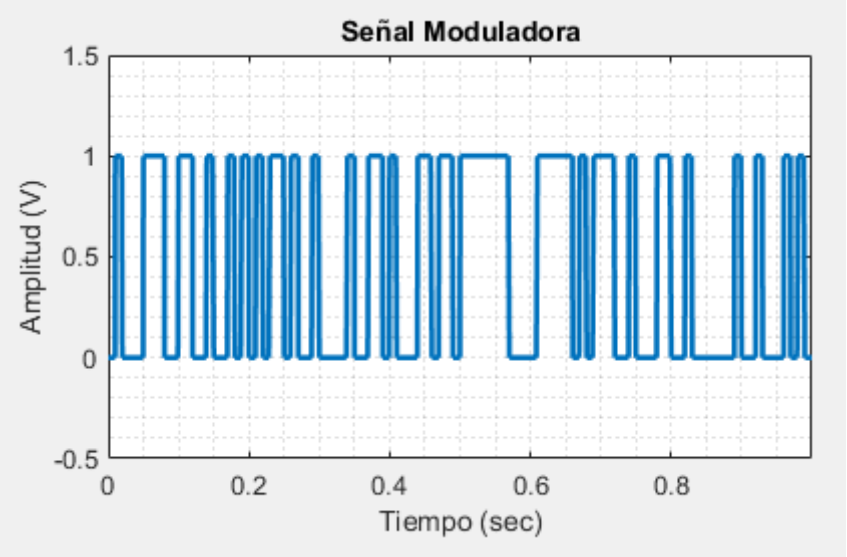


Figura 3.4.1.1 Ejemplo de señal moduladora

Portadora: se trata de una onda periódica, normalmente una señal sinusoidal que se emplea para transportar la información que se encuentra en la moduladora. La portadora tiene una frecuencia más alta que la señal que contiene la información Esta se utiliza para llevar la información a otro punto. Para ello se modifica las características de dicha señal como puede ser fase, amplitud, frecuencia para añadir la información de la moduladora dando como resultado otra señal llamada señal modulada.

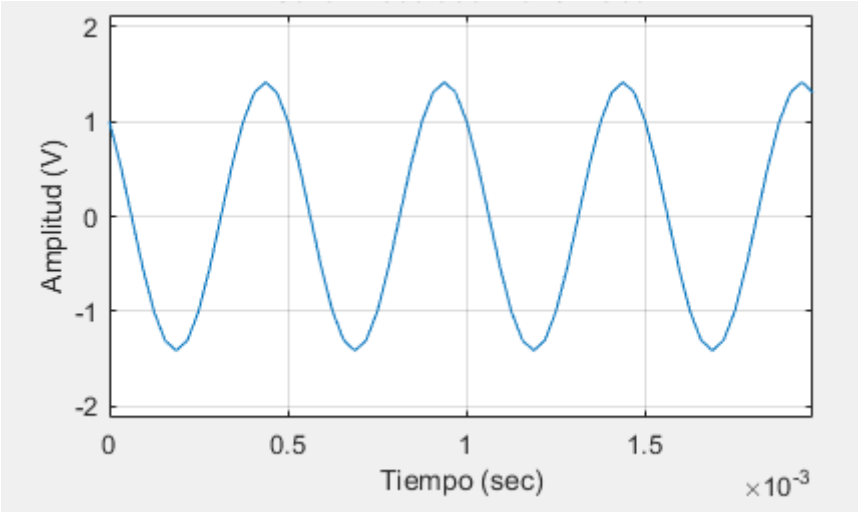


Figura 3.4.1.2 Ejemplo de señal portadora

Modulada: es la señal definitiva que obtenemos de la señal portadora tras añadir la información de la señal moduladora. La señal resultante dependerá del tipo de modulación con la que estemos trabajando. En la comunicación por radio, la señal modulada se transmite a través del espacio como una onda de radio a un receptor de radio.

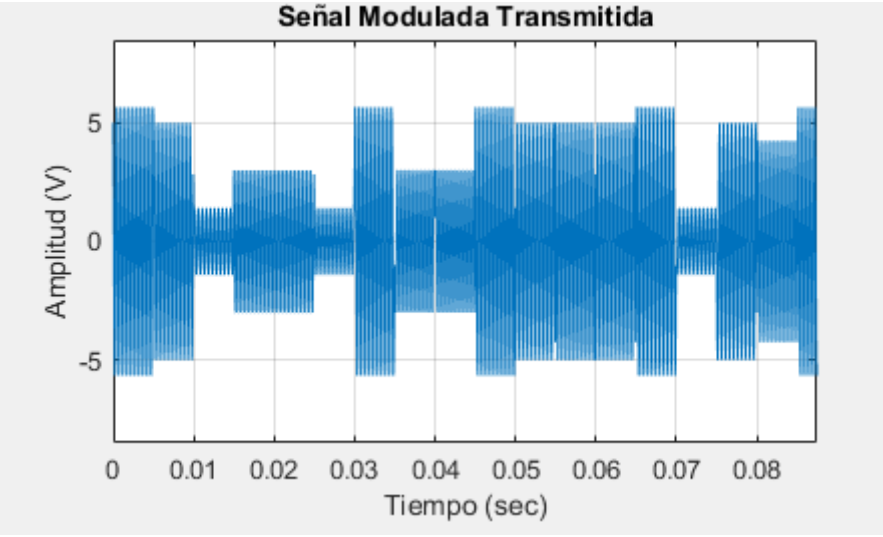


Figura 3.4.1.3 Ejemplo de señal modulada

3.4.2. Ventajas

- Alta capacidad de transmisión de datos: la cantidad de datos transmitidos a través de la modulación digital es mayor que a través de la comunicación analógica.
- Increíble eficiencia de ancho de banda: puede acomodar grandes cantidades de datos dentro de un ancho de banda limitado.
- Señales flexibles: existe la oportunidad de multiplexar varias formas de datos, como información digital, video y voz.
- Menos susceptibilidad a la diafonía, la distorsión de la forma de onda, las no linealidades y el ruido.
- Intensidad de señal mejorada, que evita la intrusión de señal no deseada y los ruidos de comunicación.
- Al aumentar el número de bits por muestra, se puede aumentar el rendimiento de señal a ruido de la modulación digital.
- Es la opción más económica al interactuar con sistemas de conmutación digital.
- Seguridad de datos mejorada: el fácil cifrado y descifrado de señales digitales con alta seguridad lo hacen adecuado para comunicaciones confidenciales y confiables.

3.5. Ruido e interferencias

El ruido es una suma de energía no deseada que produce una perturbación aleatoria involuntaria dentro una señal de información útil causando errores en dicha señal. Este puede ser causado por fuentes naturales existentes en el medio durante el recorrido por un canal de comunicación, o, por fuentes artificiales debido a las interferencias de señales externas no deseadas que se adhieren a la señal útil. También puede producirse por otras fuentes artificiales como componentes electrónicos (amplificadores), al ruido térmico de los resistores que se hallan en el interior de los dispositivos de modulación y demodulación.

Tipos de ruido:

- Crosstalk o diafonía. Acoplamiento no deseado de las señales eléctricas presentes en un medio de transmisión con las de otro próximo.
- Interferencia de canal adyacente. Es una interferencia causada por una radiación electromagnética

Es imposible eliminar totalmente el ruido, ya que los componentes electrónicos no son perfectos. Sin embargo, es posible limitar su valor de manera que la calidad de la comunicación resulte aceptable.

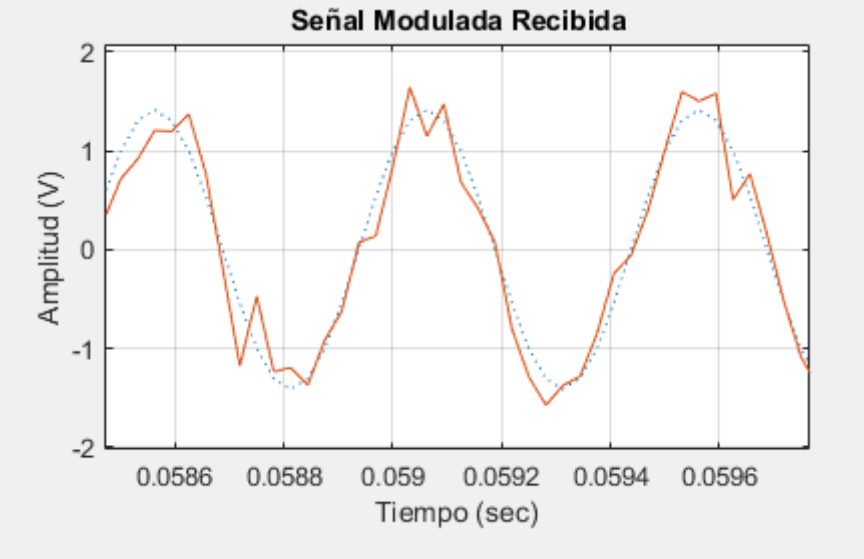


Figura 3.5.1 Ejemplo de señal modulada con ruido

Podemos observar la señal origina en azul punteado y la misma señal pero con ruido en

naranja de una señal modulada.

El nivel de ruido en un sistema de telecomunicaciones generalmente se mide utilizando la relación señal/ruido, que, por lo regular, se maneja en decibelios (dB). Como potencia de la señal se adopta normalmente la potencia de un tono de pruebas que se inyecta en el canal.

La potencia del ruido suele medirse a la entrada del receptor, cuando por él no se emite dicho tono. Cuando se transmiten señales digitales por un canal, el efecto del ruido se pone de manifiesto en el número de errores que comete el receptor. Se deduce inmediatamente que dicho número es tanto mayor cuanto más grande sea la probabilidad de error.
La probabilidad de error depende del valor de la relación señal/ruido. Cuanto mayor sea esta relación, más destaca la señal sobre el ruido y, por tanto, menor es la probabilidad de error. Cuando el ruido se añade una señal con distorsión, la probabilidad de error crece rápidamente.

La distorsión que produce el ruido en una determinada comunicación depende de su potencia, de su distribución espectral respecto al ancho de banda de la señal, y de la propia naturaleza de información que transporta. El ruido afecta de diferente manera a la información que transportan las señales detectadas que a la codificada mediante señales digitales. Esta es la causa por la que se ha establecido una tipificación básica de los canales: los canales analógicos no son buenos (con amplificación) y los canales digitales (con regeneración).

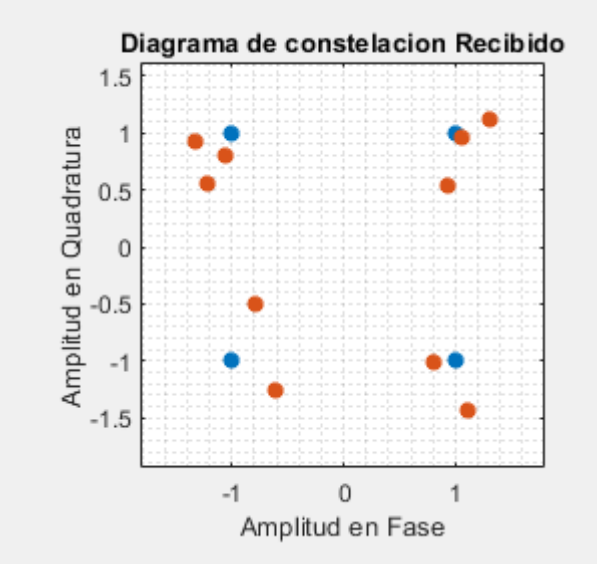


Figura 3.5.2 Ejemplo de diagrama de constelacción con ruido

Capítulo 4.

4. FUNDAMENTOS DE LAS MODULACIONES DIGITALES.

4.1. Clasificación de los diferentes tipos de modulaciones digitales

Los métodos de modulación digital más frecuentes se basan en la manipulación de las siguientes características:

- Modulación por desplazamiento de fase (PSK): se utiliza un número finito de fases.
- modulación por desplazamiento de frecuencia (FSK): se utiliza un número finito de frecuencias.
- modulación por desplazamiento de amplitud) (ASK): se utiliza un número finito de amplitudes.
- modulación de amplitud en cuadratura): (QAM)se utiliza un número finito de al menos dos fases y al menos dos amplitudes.

En QAM Consiste en modular por desplazamiento en amplitud (ASK) de forma independiente dos señales portadoras que tienen la misma frecuencia pero que están desfasadas entre sí 90º.Puede verse como un sistema de dos canales, cada canal usando ASK. La señal resultante es equivalente a una combinación de PSK y ASK.

En todos los métodos anteriores, a cada una de estas fases, frecuencias o amplitudes se le asigna un patrón único de bits binarios. Normalmente, cada fase, frecuencia o amplitud codifica un número igual de bits. Este número de bits comprende el símbolo que está representado por la fase, frecuencia o amplitud particular.

Si el alfabeto consta de M=2^n símbolos alternativos, cada símbolo representa un mensaje que consta de N bits. Si la tasa de símbolo (también conocida como tasa de baudios) es f_s. símbolos/segundo, la tasa de datos es N Fs. bit/segundo.

Por ejemplo, con un alfabeto que consta de 16 símbolos alternativos, cada símbolo representa 4 bits.

En el caso de PSK, ASK o QAM, donde la frecuencia portadora de la señal modulada es constante, el alfabeto de modulación a menudo se representa convenientemente en un diagrama de constelación, que muestra la amplitud de la señal en fase en el eje x y la señal en cuadratura en el eje y, para cada símbolo.

4.1.1. Concepto de coherencia

La coherencia una característica se encuentra en algunos demoduladores la cual les permite obtener la información de fase y frecuencia que se emplearon por parte del modulador previamente a la transmisión lo que hace que exista una sincronización entre ambos, esto añade cierta complejidad a este, pero lo hace más optimo y disminuye la tasa de error al decodificar la información original.

Un modulador coherente obtiene la fase y frecuencia mediante la multiplicación de modulada recibida con otra señal portadora generada localmente en este y mediante un proceso posterior de filtrado se llega a obtener la señal original portadora de la señal modulada recibida.

Gracias a este método se ha conseguido que la fase y frecuencias en transmisor y receptor sean idénticas ya que si son distintas esto genera una gran atenuación que puede causar que se pierda el mensaje.

La detección coherente se caracteriza por la necesidad de tener un sistema de recuperación de portadora, lo que lo hace un sistema más complejo que los sistemas no coherentes, pero ofrecen una mejor tasa de error de bits.

4.2. Antecedentes y aplicaciones de las modulaciones digitales

Las modulaciones digitales surgieron con una evolución de las anteriores modulaciones analógicas existentes como la modulación FM, PM y la AM ya que los desarrolladores de sistemas de comunicaciones se enfrentan a estas restricciones:

- ancho de banda disponible
- potencia admisible
- nivel de ruido inherente del sistema

El espectro de RF debe ser compartido, pero cada día hay más usuarios de ese espectro a medida que aumenta la demanda de servicios de comunicaciones. Los esquemas de modulación digital tienen mayor capacidad para transmitir grandes cantidades de información que los esquemas de modulación analógica.

En los sistemas de modulación digital podemos encontrar gran cantidad de ventajas respecto a la modulación analógica como:

- Alta capacidad de transmisión de datos: la cantidad de datos transmitidos a través de la modulación digital es mayor que a través de la comunicación analógica.
- Increíble eficiencia de ancho de banda: puede acomodar grandes cantidades de datos dentro de un ancho de banda limitado.
- Señales flexibles: existe la oportunidad de multiplexar varias formas de datos, como información digital, video y voz.
- Menos susceptibilidad a la diafonía, la distorsión de la forma de onda, las no linealidades y el ruido.
- Intensidad de señal mejorada, que evita la intrusión de señal no deseada y los ruidos de comunicación.
- Al aumentar el número de bits por muestra, se puede aumentar el rendimiento de señal a ruido de la modulación digital.
- Es la opción más económica al interactuar con sistemas de conmutación digital.
- Seguridad de datos mejorada: el fácil cifrado y descifrado de señales digitales con alta seguridad lo hacen adecuado para comunicaciones confidenciales y confiables.

Las aplicaciones de estos son diversas y dependen del tipo de modulación que se emplee suelen tener distintos fines, a continuación, se muestran los más comunes:

BPSK	Telemetría del espacio profundo, módems de cable
QPSK	Telefonía móvil CDMA (acceso múltiple por división de código), bucle local inalámbrico, Iridium (un sistema satelital de voz/datos) y DVB-S (difusión de video digital - satélite).
8PSK	satélites, aeronaves, telemetría para monitorear sistemas de video de banda ancha
16 QAM	Radio digital microondas, módems, DVB-C, DVB-T
32 QAM	Microondas terrestre, DVB-T
64 QAM	DVB-C, módems, decodificadores de banda ancha, MMDS
256 QAM	Módems, DVB-C, Video digital
FSK	DECT, RAM datos móviles, AMPS, CT2, ERMES, telefonía terrestre
MSK	GSM, CDPD

Tabla 4.2 Aplicaciones más comunes de las distintas modulaciones. [4]

4.3. M-PSK

La modulación por desplazamiento de fase o PSK (Phase-Shift Keying) es un proceso de modulación angular digital también conocido como modulación por cambio de fase. Esta consiste en hacer variar la fase de la portadora entre un número determinado de valores discretos. Keying significa que el proceso de modulación consiste en cambiar parámetros de señal periódicos, en este caso, una fase. Los datos se representan mediante cambios discretos en la fase de la onda portadora. El número de cambios puede ser aleatorio y suele ser igual a la potencia de 2 (utilizado para diferenciar entre los tipos de modulación PSK, incluidos BPSK y su variante DBPSK, y QPSK y su variante DQPSK).

4.3.1. Definición temporal

$$s_i(t) = Acos(2\pi f_c t + \theta_i), \quad 0 \leq t \leq T, \quad i = 1, 2, \ldots, M \tag{7}$$

$$\theta_i = \frac{(2i - 1)\pi}{M} \tag{8}$$

La frecuencia portadora se elige como un múltiplo entero de la tasa de símbolo, por lo tanto en cualquier intervalo de símbolo, la fase inicial de la señal es también una de las fases M. Normalmente M se elige como una potencia de 2 (es decir, M = 2n, n = log2M). Por lo tanto, el binario flujo de datos se divide en n-secuencias. Cada uno de ellos está representado por un símbolo con una determinada fase inicial.

La expresión anterior se puede escribir como:

$$\mathrm{si(t)} = \mathrm{Acos}\ \theta\,\mathrm{i}\cos 2\pi\,\mathrm{fct} - \mathrm{Asin}\ \theta\,\mathrm{i}\sin 2\pi\,\mathrm{fct}$$

$$\mathrm{si(t)} = \mathrm{si1}\,\phi\,1(\mathrm{t}) + \mathrm{si2}\,\phi\,2(\mathrm{t})$$

4.3.2. Espacio de señal (Constelación digital)

Se trata de una representación gráfica de una señal modulada digitalmente que consta de los puntos de constelación utilizados para evaluar la calidad de una señal transmitida.

Además de los datos utilizables, la señal modulada y transmitida también incluye interferencias, ruido y perturbaciones. Para que el demodulador demodule correctamente un bit de señal utilizable, un símbolo debe coincidir con un punto de constelación (bit) correspondiente. De lo contrario, la señal recibida se distorsionará.

$$S_M(t) = \sqrt{E_s} * \cos\left((M-1)\frac{2\pi}{M}\right) * \varphi_1(t) - \sqrt{E_s} * \sin\left((M-1)\frac{2\pi}{M}\right) * \varphi_2(t) \qquad (9)$$

$$\varphi_1(t) = \sqrt{\frac{2}{T}} cos(2\pi f_c t), \qquad 0 \le t \le T \qquad (10)$$

$$\varphi_2(t) = \sqrt{\frac{2}{T}} sin(2\pi f_c t), \qquad 0 \le t \le T \qquad (11)$$

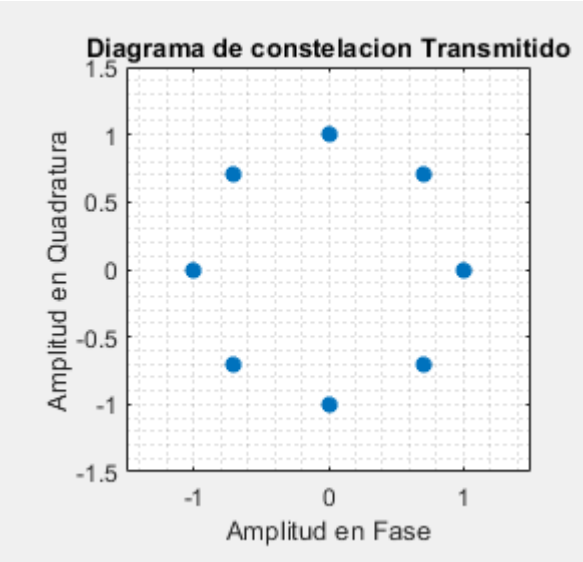


Figura 4.3.2.1 Ejemplo de diagrama de constelacción 8-PSK

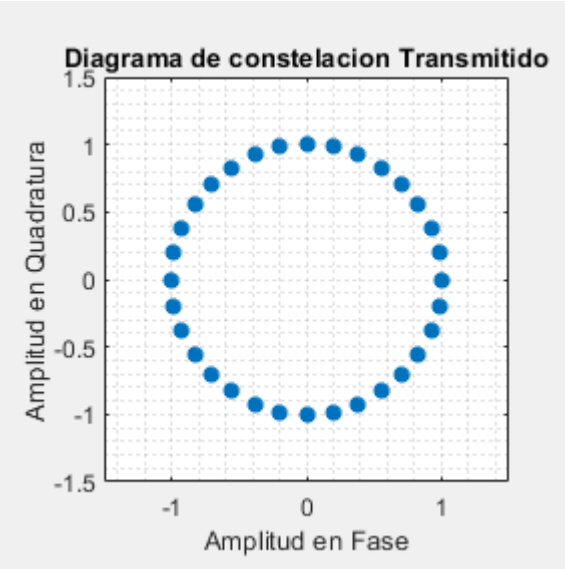


Figura 4.3.2.2 Ejemplo de diagrama de constelacción 32-PSK

4.3.3. Transmisor

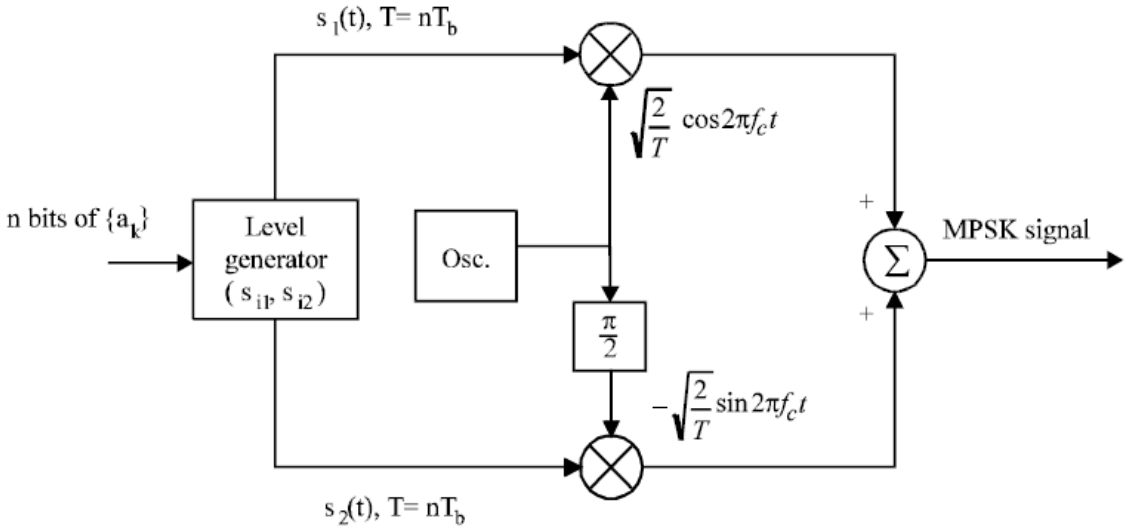


Figura 4.3.3 Diagrama de modulador M-PSK [4]

Q-canales con el signo y el nivel particulares para las coordenadas horizontales y verticales de una señal coordenadas horizontales y verticales, respectivamente. En el caso de QPSK, el generador de nivel es parcialmente sencillo, es simplemente un convertidor serie-paralelo. La tecnología moderna pretende utilizar dispositivos completamente digitales. En este entorno, las señales MPSK se sintetizan digitalmente y se alimentan a un convertidor D/A cuya salida es la señal de fase modulada deseada.

4.3.4. Receptor

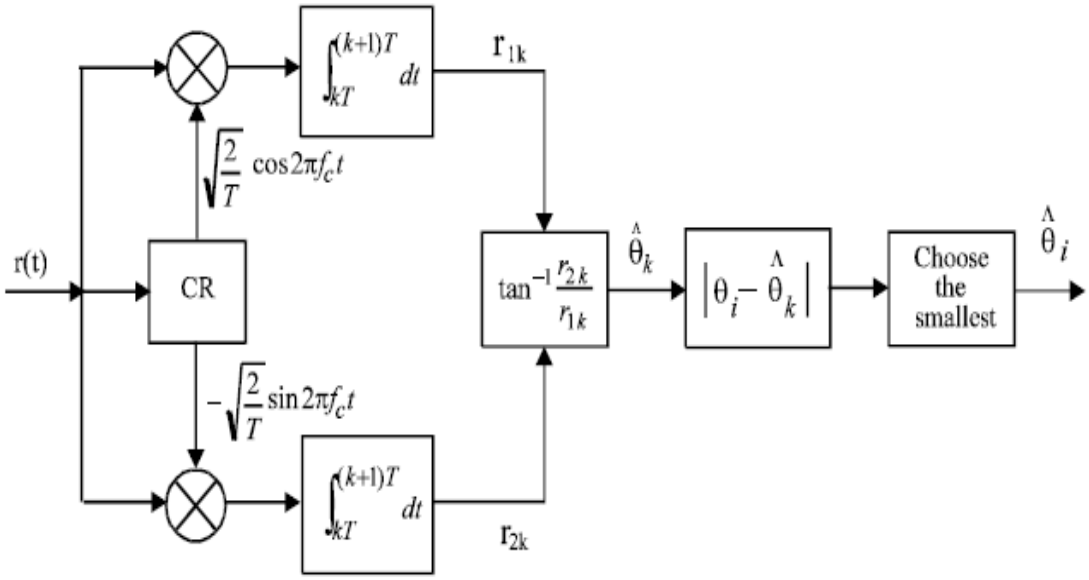


Figura 4.3.4 Diagrama de demodulador M-PSK [4]

La demodulación coherente de MPSK podría implementarse mediante uno de los detectores coherentes para señales M-arias. Dado que el conjunto de señales MPSK tiene sólo dos funciones de base, el receptor más sencillo es el que utiliza dos correladores.

4.3.5. Probabilidad de error

Empleando codificación Gray

$$BER = \frac{P_e}{log_2 M} \qquad (12)$$

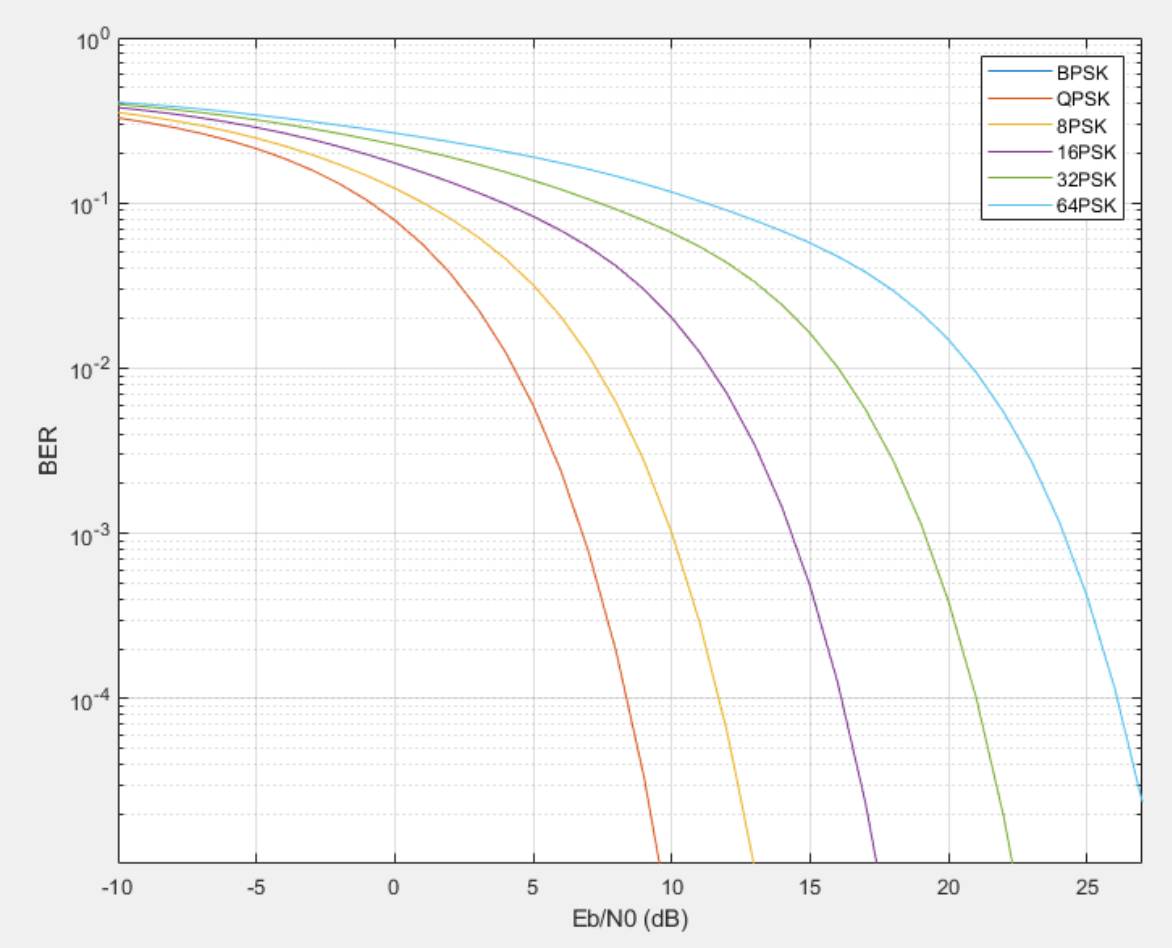


Figura 4.3.4 Curva de BER de M-PSK

4.3.6. Densidad espectral de potencia y ancho de banda

$$B_{n-n} = \frac{2}{T} = \frac{2}{T_b\, log_2(M)} = \frac{2R_b}{log_2(M)} \quad (Hz) \tag{13}$$

$$p_{M-PSK} = \frac{log_2(M)}{2} \quad (bits/s/Hz\,) \tag{14}$$

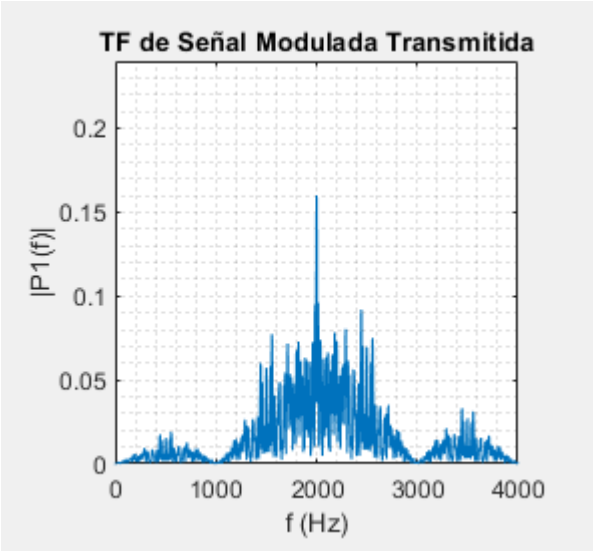


Figura 4.3.6 Densidad espectral de potencia M-PSK

4.4. M-QAM

La modulación de amplitud en cuadratura o QAM (acrónimo de Quadrature Amplitude Modulation, por sus siglas en inglés) es un método que combina dos de las modulaciones digitales más simples: modulación por desplazamiento de amplitud (ASK) y modulación por desplazamiento de fase (PSK). Este modo de codificación implica un cambio simultáneo en la amplitud y la fase de la señal portadora. El número disponible de bits binarios por símbolo es 2n. Si n = 4 (24 = 16), la modulación se denomina 16-QAM. Si n = 6 (26 = 64), la modulación es 64-QAM. Cuanto mayor sea el valor de n, más datos se "empaquetarán" en el canal; aumenta la capacidad de la red y permite velocidades de transmisión de datos más altas.

4.4.1. Definición temporal

$$s_{(t)} = I_{(t)} \cos(2\pi f_o t) + Q_{(t)} \cos(2\pi f_o t - 90^{\circ}) = I_{(t)} \cos(2\pi f_o t) + Q_{(t)} \sin(2\pi f_o t) \quad (15)$$

4.4.2. Espacio de señal (Constelación digital)

En QAM, los puntos de la constelación generalmente se organizan en una cuadrícula con el mismo espaciado vertical y horizontal, aunque son posibles otras configuraciones (por ejemplo, una cuadrícula hexagonal o triangular). En las telecomunicaciones digitales los datos suelen ser binarios, por lo que el número de puntos de la cuadrícula suele ser una potencia de 2 (2, 4, 8...), correspondiente al número de bits por símbolo. Las constelaciones de QAM más simples y más utilizadas consisten en puntos dispuestos en un cuadrado, es decir, 16-QAM, 64-QAM y 256-QAM (potencias pares de dos). Las constelaciones no cuadradas, como Cross-QAM, pueden ofrecer una mayor eficiencia, pero rara vez se usan debido al costo de una mayor complejidad del módem.

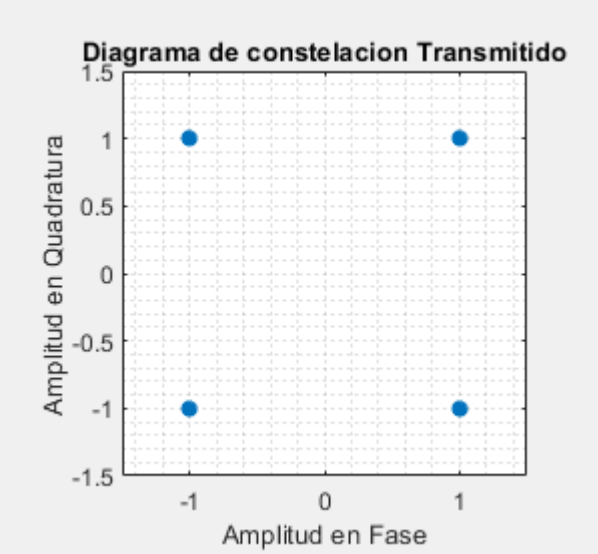


Figura 4.4.2.1 Diagrama de constelacción de 4-QAM

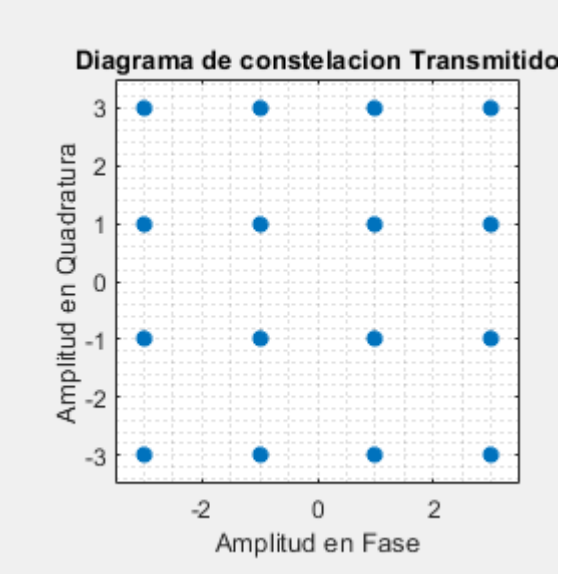


Figura 4.4.2.2 Diagrama de constelacción de 16-QAM

4.4.3. Transmisor

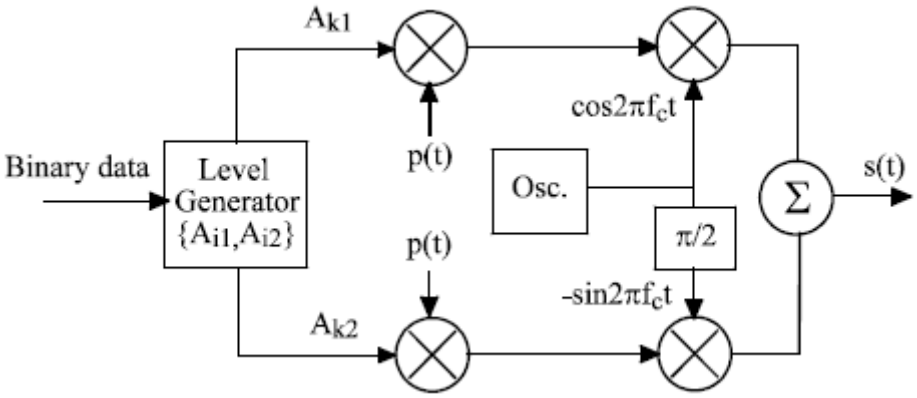


Figura 4.4.3 Diagrama de bloques de modulador M-QAM [4]

La secuencia de bits de datos se divide en n-secuencias de n bits. Hay M = 2^n n-secuencias distintas. Cada n-secuencia de los bits de entrada se utiliza para controlar el generador de nivel. El generador de nivel proporciona a los canales I y Q el signo y el nivel particulares para las coordenadas horizontales y verticales de una señal (A_{k1}, A_{k2}), respectivamente. El mapeo de las n-secuencias a los puntos QAM se normalmente codificados en Gray para minimizar los errores de bit.

4.4.4. Receptor

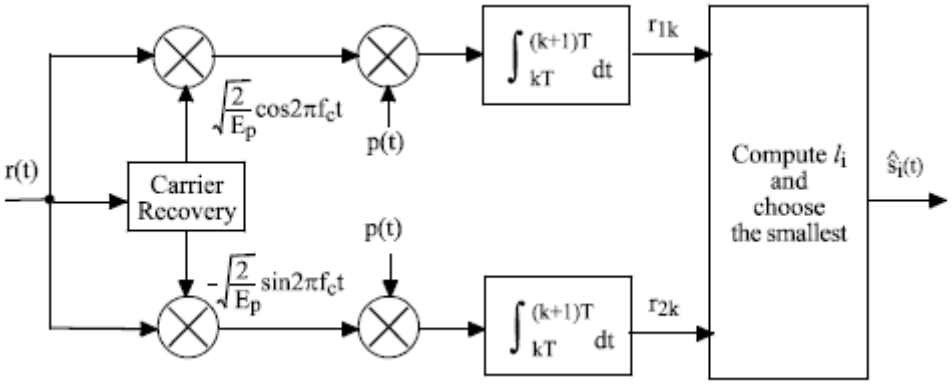


Figura 4.4.4 Diagrama de bloques de demodulador M-QAM [4]

El detector compara las distancias de (r1, r2) a todos los pares de (si1, si2) y elige el más cercano. La figura 4.4.4 es el demodulador basado en la regla de decisión anterior, donde el subíndice k indica el k período de símbolo. Obsérvese que la amplitud de las señales de

referencia puede ser cualquier valor, que es $\sqrt{2/E_p}$ siempre que (s_{i1}, s_{i2}) también se escalen en consecuencia.

4.4.5. Probabilidad de error

La probabilidad de error de bit pb de M-QAM sobre un canal AWGN viene dada por

$$PB = \frac{4}{k}\left(1 - \frac{1}{\sqrt{M}}\right) Q\left(\sqrt{\frac{3k}{M-1} x \frac{E_B}{N_0}}\right) \qquad (16)$$

$$k = Bits\ por\ simbolo$$

$$M = orden\ de\ modulación$$

$$E_B = Epergia\ por\ bit$$

$$N_0 = Potencia\ de\ ruido$$

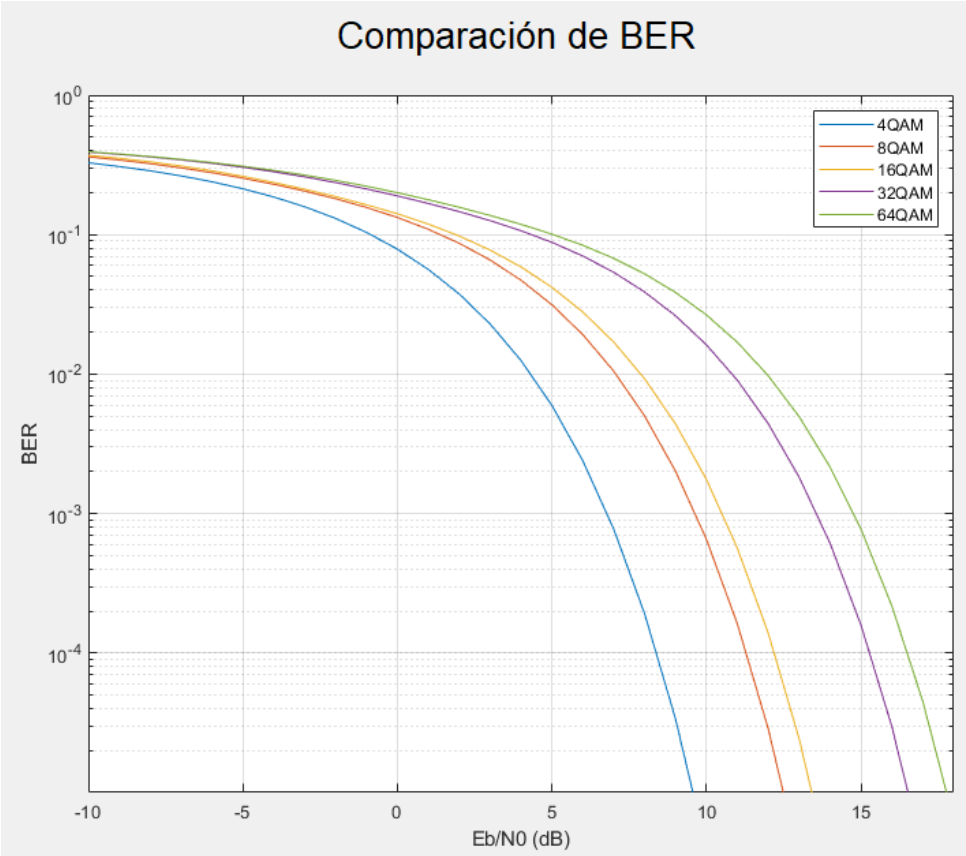


Figura 4.4.5 Curva de BER de M-QAM

4.4.6. Densidad espectral de potencia y ancho de banda

$$B_{n-n} = \frac{2}{T} = \frac{2}{T_b\ log_2(M)} = \frac{2R_b}{log_2(M)}\ \ (Hz) \tag{17}$$

$$p_{M-QAM} = \frac{log_2(M)}{2}\ \ (bits/s/Hz\ ) \tag{18}$$

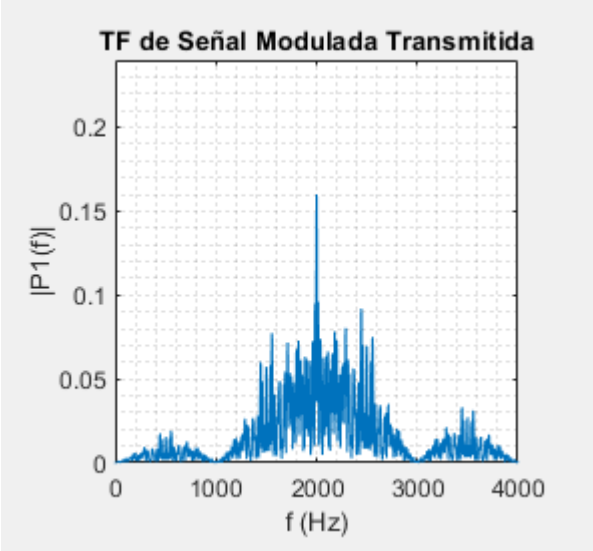


Figura 4.4.6 Densidad espectral de M-QAM con Fc=2000 y Rb=1000

4.5. M-FSK

La modulación por desplazamiento de frecuencia o FSK (Frequency Shift Keying) es una técnica de modulación para la transmisión digital de información utilizando dos o más frecuencias diferentes para cada símbolo. La señal moduladora solo varía entre dos valores de tensión discretos formando un tren de pulsos donde uno representa un "1" o "marca" y el otro representa el "0" o "espacio".

4.5.1. Definición temporal

$$s_{(t)} = A\sin\left(2\pi\left(f + X_{(t)}\Delta f\right)t\right) \qquad (19)$$

4.5.2. Espacio de señal (Constelación digital)

El diagrama de constelación a menudo se considera como un diagrama fasorial y esta analogía funciona la mayor parte del tiempo, pero no funciona para la modulación FSK. Un diagrama fasorial describe un fasor que tiene una frecuencia fija. Si el fasor se modula muy lentamente en fase y/o amplitud, entonces esta aproximación es buena. La modulación FSK no se puede representar en un diagrama fasorial, ya que la información está en la frecuencia en el tictac del reloj y no en la fase y/o amplitud de un fasor.

Figura 4.5.2.Diagrama de constelacción de 8-FSK

4.5.3. Transmisor

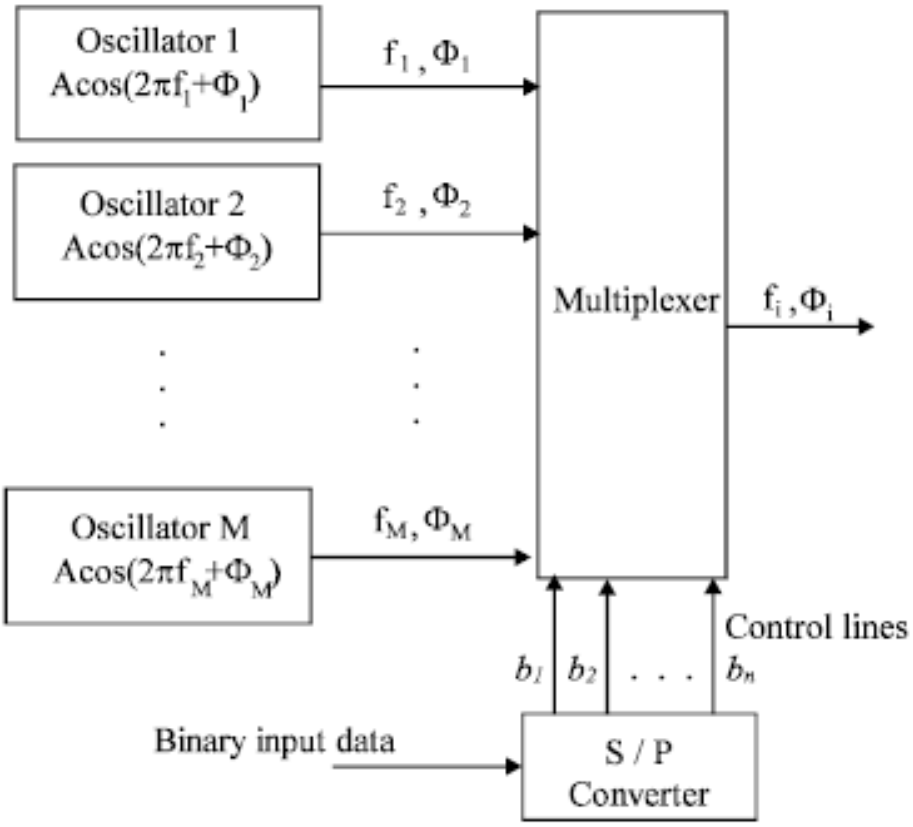


Figura 4.5.3 Diagrama de bloques de modulador M-FSK [4]

la mayor salida de los filtros emparejados. Debemos señalar que los receptores MFSK coherentes sólo requieren que las señales MFSK sean equiprobables y de igual energía, y no requieren que sean ortogonales. Para casos más generales en los que el conjunto de señales M-arias no es equiprobable y/o no es igual en energía.

4.5.4. Receptor

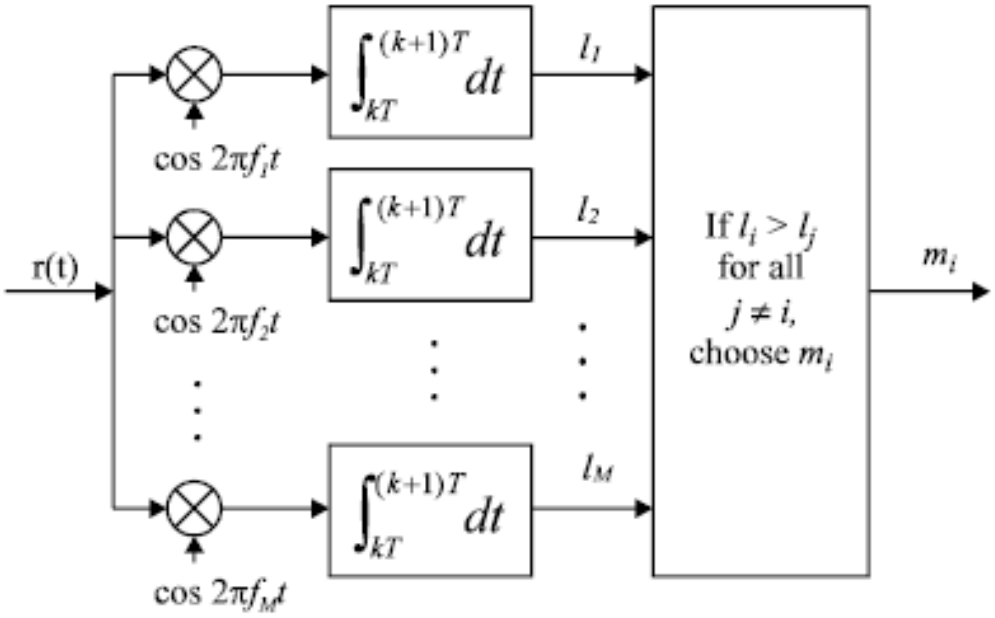


Figura 4.5.4 Diagrama de bloques de demodulador M-FSK [4]

4.5.5. Probabilidad de error

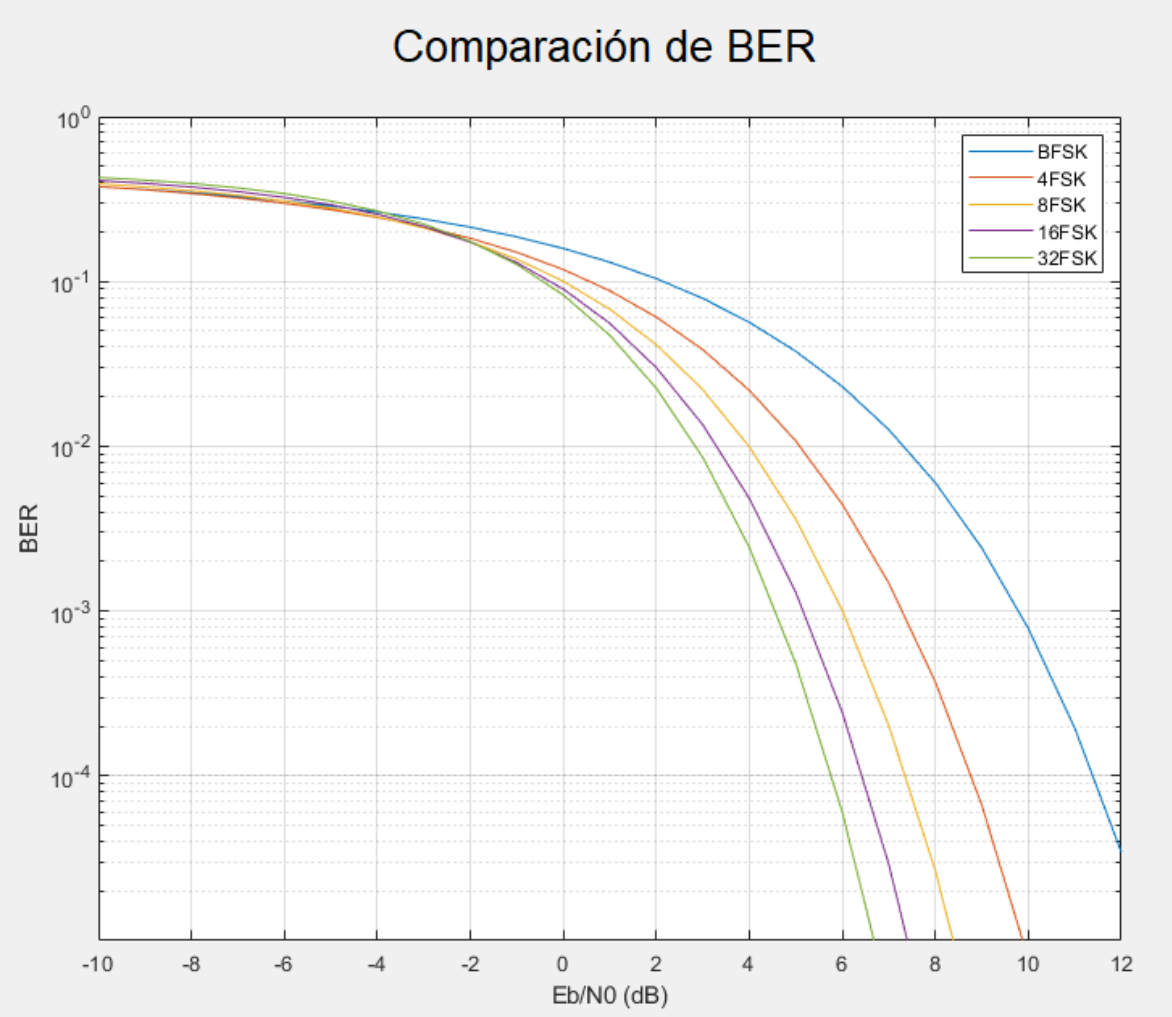


Figura 4.5.5 Curva de BER de M-FSK

4.5.5. Densidad espectral de potencia y ancho de banda

La eficiencia espectral con mínima separación en frecuencia $\Delta f = 1/(2T)$

$$B = \frac{M}{2T} = \frac{M}{T_b\ 2 * log_2(M)} = \frac{M * R_b}{2 * log_2(M)}\ \ (Hz) \qquad (20)$$

$$p = \frac{R_b}{B}\ = 2\frac{log_2(M)}{M}(bits/s/Hz\ ) \qquad (21)$$

M	2	4	8	16	32	64
p	1	1	0,75	0,5	0,3125	0,1875

Tabla 4.5.5 Relación densisdad espectral

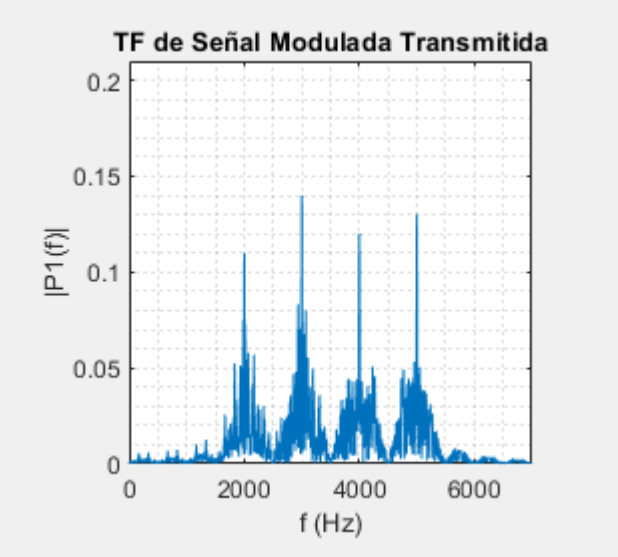


Figura 4.5.5 Curva de BER de 4-FSK

4.6. MSK

La modulación por desplazamiento mínimo, también conocida como MSK (acrónimo inglés de Minimum-shift keying) es un tipo de modulación por desplazamiento de frecuencia de fase continua, inventado en 1956 y patentado en 1961 por los estadounidenses Melvin Doelz y Earl Heald.Al igual que OQPSK, MSK está codificado con bits alternantes entre los componentes de cuadratura, con el componente Q retrasado por la mitad del periodo de símbolo. Sin embargo, en lugar de los pulsos cuadrados que utiliza QPSK, MSK codifica cada bit como una media sinusoide, resultando una señal de envolvente constante, lo que reduce los problemas causados por la distorsión no lineal. Además de ser considerada como relacionada con OQPSK, MSK también puede ser vista como una señal de modulación por desplazamiento de frecuencia de fase continua (CPFSK) con una separación de frecuencias de la mitad de la tasa de bits.

4.6.1. Definición temporal

$$s_{(t)} = \cos\left[2\pi f_c t + b_k(t)\frac{\pi t}{2T} + \varphi_k\right] \qquad (22)$$

4.6.2. Transmisor

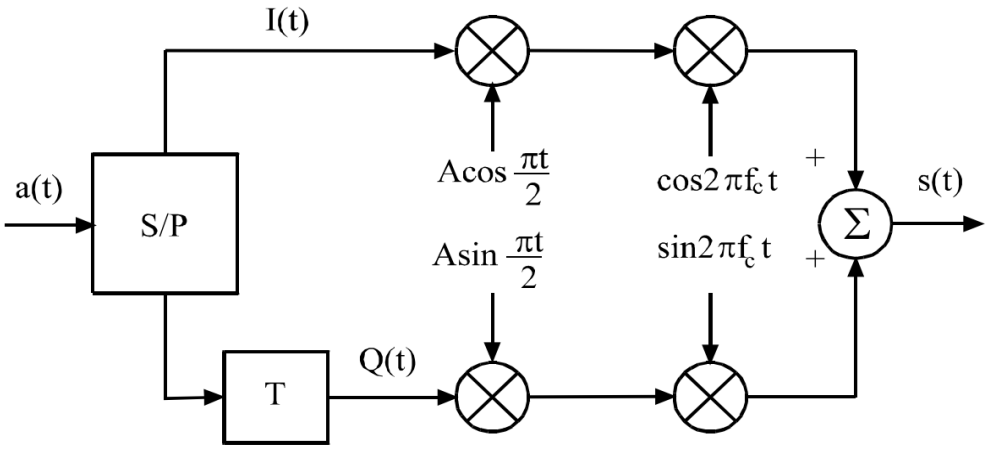


Figura 4.6.2 Diagrama de bloques de modulador MSK [4]

La señal de flujo de datos a(t) es demultiplexada en I(t) y Q(t) por el convertidor serie-paralelo (S/P). La señal de canal en fase I(t) consta de bits pares, y la señal de canal en cuadratura Q(t) consta de bits impares.

4.6.3. Receptor

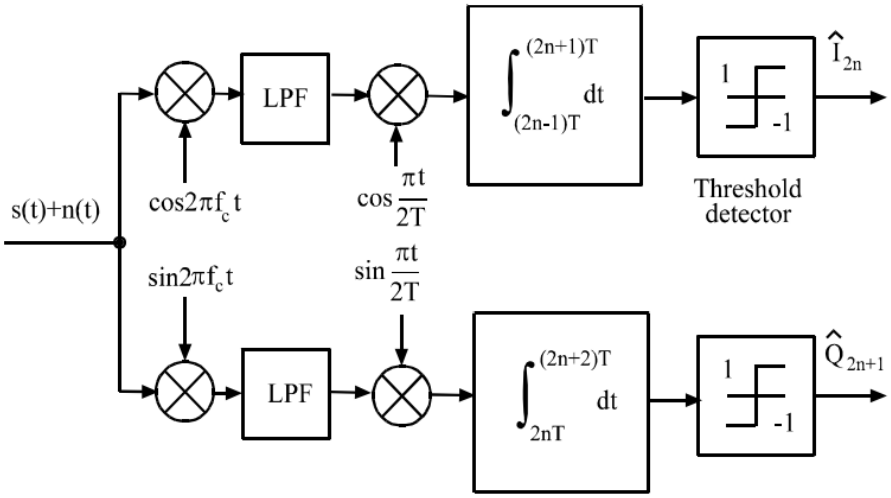


Figura 4.6.3 Diagrama de bloques de demodulador MSK [4]

4.6.4. Probabilidad de error

$$BER = \frac{1}{2}P_e \approx Q\left(\sqrt{\frac{2E_b}{\eta}}\right) = \frac{1}{2} * erfc\left(\sqrt{\frac{E_b}{\eta}}\right) \qquad (23)$$

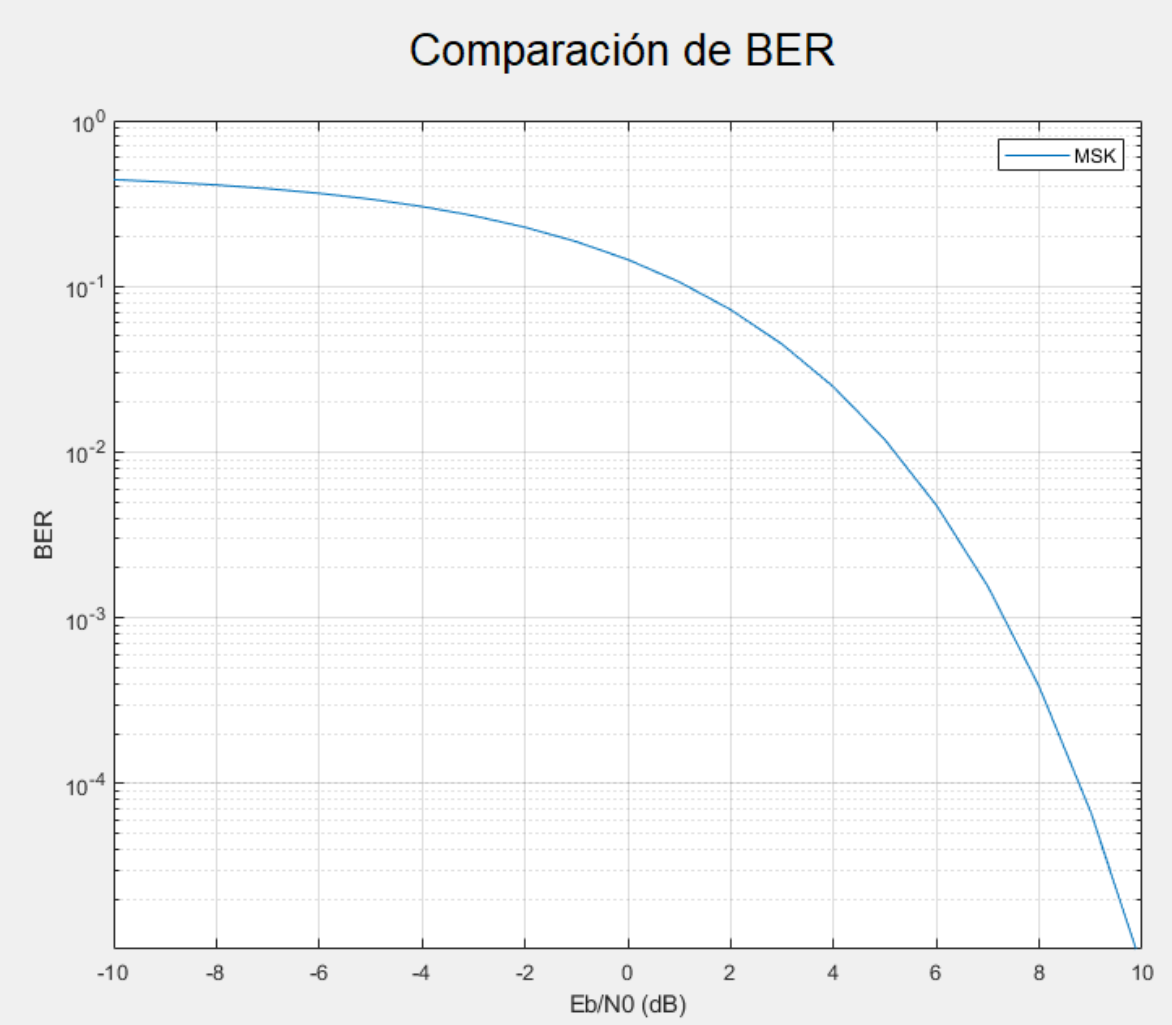


Figura 4.6.4 Curva BER de MSK

4.6.5. Densidad espectral de potencia y ancho de banda

$$B_{n-n\,MSK} = \frac{3}{2}R_b \ (Hz) \qquad (24)$$

$$p_{MSK} = \frac{R_b}{B_{n-n}} = \frac{R_b}{1.5R_b} = \frac{2}{3}(bits/s/Hz) \qquad (25)$$

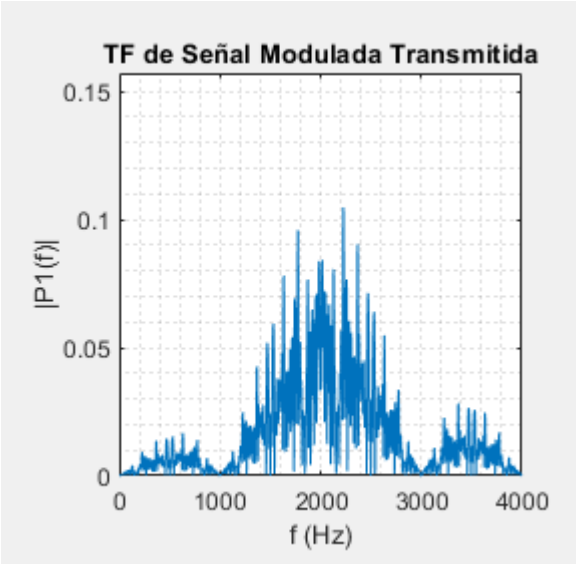


Figura 4.6.5 Represnetación espectro de potencia MSK con Fc=2000 y Rb=1000

4.7. DBPSK

La modulación por desplazamiento diferencial de fase (conocida como DBPSK, por las siglas en inglés de Differential Phase Shift Keying), es una forma de modulación digital, donde la información binaria de la entrada está compuesta en la diferencia entre las fases de dos elementos sucesivos de señalización, y no en la fase absoluta. Se considera una forma no-coherente de BPSK y por ello, en la recepción se evita la necesidad de una señal coherente de referencia para la recuperación de la señal portadora. La implementación del receptor es económica, por lo que es de amplio uso en comunicaciones inalámbricas. En los sistemas DBPSK, el flujo digital de entrada es codificado de forma diferencial y luego es modulado mediante la BPSK binaria.

4.7.1. Definición temporal

$$s_n(t) = \sqrt{\frac{2E_b}{T_b}\,cos(2\pi f_c t + \pi(1-n))}\,, \qquad n = 0,1. \tag{26}$$

4.7.2. Transmisor

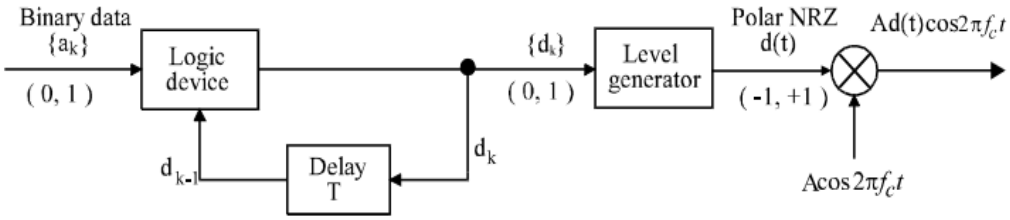


Figura 4.7.2 Diagrama de bloques de modulador DBPSK [4]

El receptor anterior es subóptimo, ya que la señal de referencia es el símbolo anterior, que es ruidoso. La demodulación óptima no coherente, o diferencialmente coherente se presenta ahora la demodulación óptima no coherente o diferencialmente coherente de DEBPSK. Como se ha comentado anteriormente, un bit de mensaje está representado por dos símbolos modulados. Si el bit transmitido es 1, los dos símbolos son iguales. Por lo tanto, podemos definir una señal con una duración de 2T como sigue para representar el 1 binario

4.7.3. Receptor

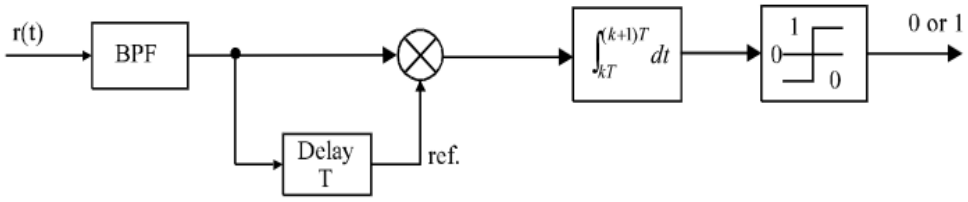


Figura 4.7.3 Diagrama de bloques de demodulador DBPSK [4]

4.7.4. Probabilidad de error

$$P_e = \frac{1}{2} e^{-\left(\frac{E_b}{\eta}\right)} \quad M = 2 \qquad (27)$$

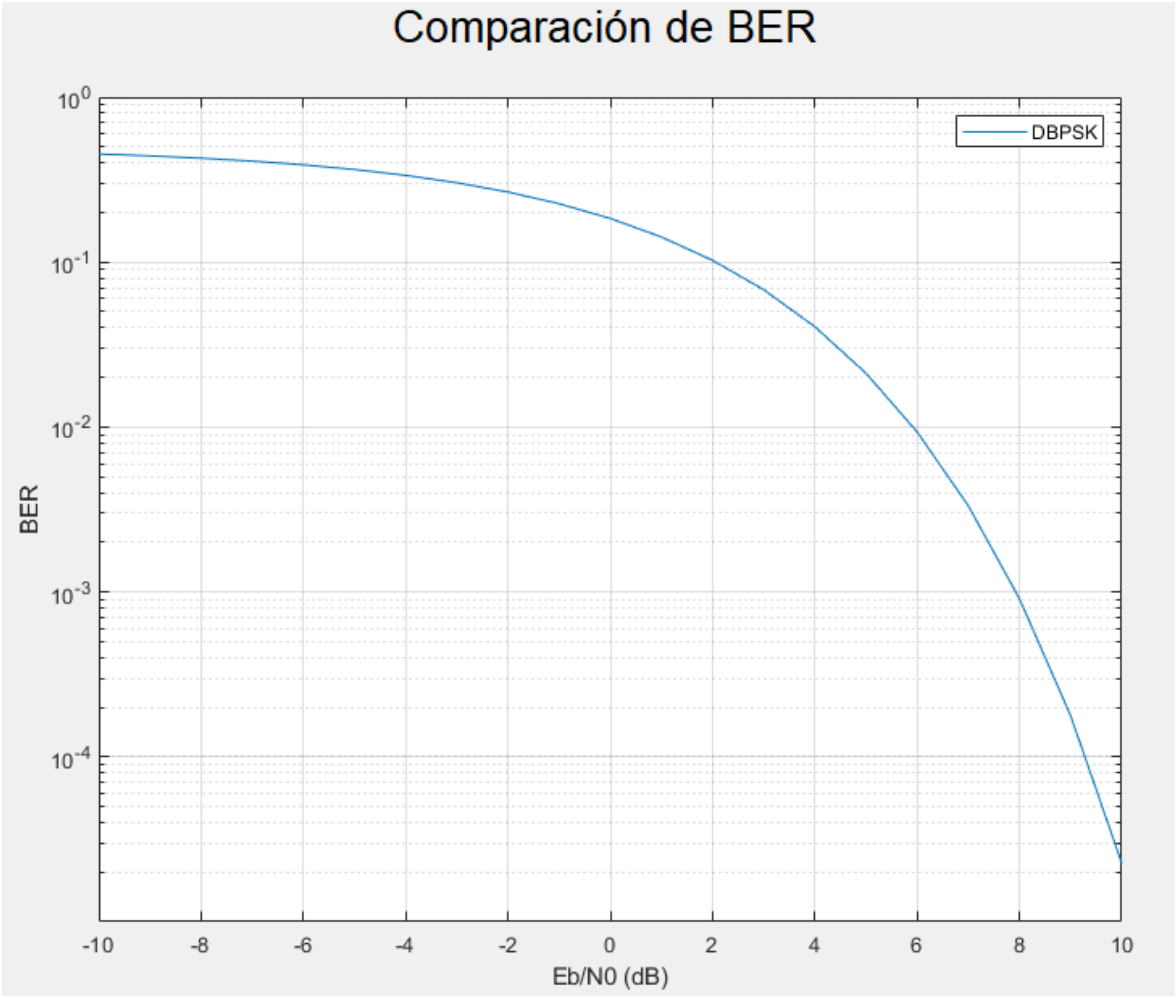


Figura 4.7.4 Curva de BER de DBPSK

4.7.5. Densidad espectral de potencia y ancho de banda

$$B_{n-n} = \frac{2}{T} = \frac{2}{T_b\, log_2(M)} = \frac{2R_b}{log_2(M)} \quad (Hz) \tag{28}$$

$$p_{M-QAM} = \frac{log_2(M)}{2} \quad (bits/s/Hz\,) \tag{29}$$

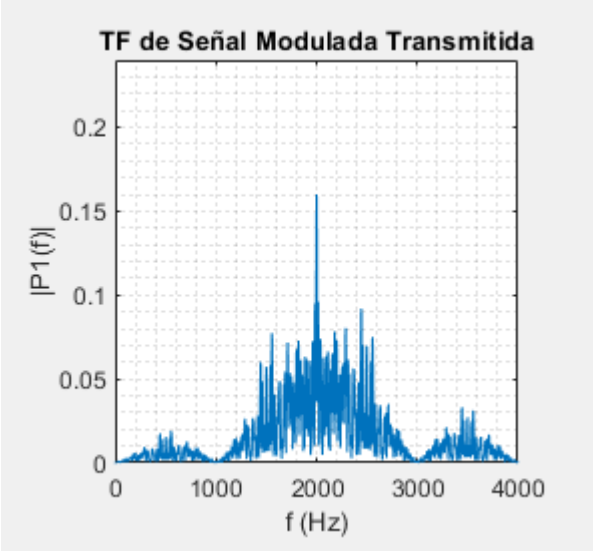


Figura 4.7.5 Densidad espectral del potencia

Capítulo 5.

5. IMPLEMENTACIÓN

5.1. Selección del entorno de programación Matlab

Para el desarrollo de este proyecto se ha decidido emplear el programa Matlab (abreviatura de MATrix LABoratory, laboratorio de matrices) es una herramienta informática y matemática que ofrece un entorno de desarrollo integrado, IDE, (programa formado por un conjunto de herramientas de programación que se dedica a un solo lenguaje de programación o a varios, en este caso, a uno: el lenguaje M).
https://es.mathworks.com/products/matlab-online.html

Espe programa permite la manipulación de matrices, la representación de datos y funciones, el uso de algoritmos, el diseño de interfaces de usuario, GUI, (programa informático que mediante imágenes y elementos gráficos facilita la comunicación con el sistema operativo de la máquina).

Matlab crea archivos para manipulación con extensión .m, archivos para interfaz gráfica del usuario .gui, archivos gráficos .fig, además permite crear sus variables y funciones.
ya que es un lenguaje de programación multiparadigma y un entorno de computación numérica que está adaptado para la trabajar especialmente con matrices empleando un bajo consumo computacional.

En este TFG se ha utilizado concretamente la versión 2020b

Figura 5.1 Icono de Matlab

Sus características más importantes son:

- Lenguaje sencillo, pero a la vez potente y rápido. Además, los ficheros son de texto ocupando poca memoria y no es necesario compilar en una sesión de trabajo estándar.
- Muchas funciones matemáticas y de aplicación como análisis estadístico, optimización o diseño en ingeniería están predefinidas y agrupadas en toolboxes aunque el usuario puede modificarlas e incluso crear las suyas propias como por ejemplo la función fft(), con lo cual es muy útil para las diversas representaciones de señales en el dominio del tiempo, en el dominio de la frecuencia, diagramas de constelación de los distintos moduladores.
- Capacidad de creación de gráficos en 2D y 3D que permiten incorporar efectos y animaciones.
- Desarrollo de aplicaciones mediante el editor de ventanas, menús y GUI lo que permite crear un menú y un entorno gráfico donde podemos seleccionar distintas opciones e interactuar con el propio programa.

5.2. Desarrollo de las modulaciones.

5.2.1 Modulación M-PSK

```
%En primer lugar se obtiene el Numero de Bits de nuestro vector de
entrada, en este caso se llama dataIn
NBits=length(dataIn);



%También se necesita el número de bit que componen un símbolo, este valor
%se obtiene en función del orden de modulación con la siguiente formula
k    = log2(M);



%se calcula los bits extra necesarios para para que no quede ningún bit
%suelto ya que al agruparlos en grupos de k pueden quedar bits solos
BitsExtra=(ceil(NBits/k)*k)-NBits;



%se añade los bits extra calculados al vector dataIn para hacerlo
multiplo de k
dataInExtra = [dataIn,zeros(1,BitsExtra)];



% se crea una matriz en la cual se transforma el vector dataInExtra en
una matriz de grupos de tamaño k
dataInAgru = reshape(dataInExtra,[k ,(NBits+BitsExtra)/k]);



%se transforma los grupos de bits de tamaño k a número decimal
dataInAgru = bin2dec(num2str(dataInAgru.'));
```

En este punto del proyecto se creó una función desde cero para realizar la modulación PSK pero tras avanzar en dicho proyecto se decidió sustituir por la función de Matlab comm.PSKModulator ya que requería un menor conste computacional y hace que el programa se ejecute con mayor rapidez.

```
% se crea un vector para indicar al modulador el tipo de mapa de
% modulación, ya que es necesario para el modulador
MapaMod = [[0:2:M-1] [M-1:-2:0]];



%por ejemplo para M=8 el mapa seria el siguiente
% MapaMod =    0     2     4     6     7     5     3     1
```

```
%se selecciona el tipo de modulador, en este caso con la función
%comm.PSKModulator de matlab
Modulador   = comm.PSKModulator
('ModulationOrder',M,'PhaseOffset',fase,'BitInput',true,
'SymbolMapping','Custom','CustomSymbolMapping',MapaMod);

%se realiza la modulación.
dataMOD = Modulador(dataInExtra');



% se crea el vector tiempo de bits que será lo que dura un bit
tBit=  (0:(k*inc)-1)/Fs;

% se crea una matriz para poner los bits generadores por el modulador en
función de la frecuencia de portadora
% mediante la siguiente formula
A = real(dataMOD)*cos(2*pi*Fc*tBit) + imag(dataMOD)*sin(2*pi*Fc*tBit);

% se transforma la matriz en un vector donde cada bits comience donde
% finaliza el anterior.
dataOUT = reshape(A',[1 ,k * inc * NBitsExtra]);

%se crea un vector de tiempo para la representacion de la señal modulada
t = (0:(NBits+NBitsExtraVector)*inc -1)/Fs;

% señal de entrada añadiendo muestras por bit para representación.
[ModuladoraINC] = muestrasINC(VectorInExtra,inc);



%%%%%%%%%%%%%%%%%%%%% REPRESENTACION Señal
Moduladora%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes2);
plot(t,ModuladoraINC,'linewidth',2);grid minor;

% limites plot
xlim([t(1) t(end)]);
ylim([-0.5, 1.5]);

% etiquetas plot
xlabel('Tiempo (sec)');
ylabel('Amplitud (V)');
title ('Señal Moduladora');
```

5.2.2 Modulación M-QAM

```
%En primer lugar se obtiene el Numero de Bits de nuestro vector de
entrada, en este caso se llama dataIn
NBits=length(dataIn);



%También se necesita el número de bit que componen un símbolo, este valor
%se obtiene en función del orden de modulación con la siguiente formula
k    = log2(M);



%se calcula los bits extra necesarios para para que no quede ningún bit
%suelto ya que al agruparlos en grupos de k pueden quedar bits solos
BitsExtra=(ceil(NBits/k)*k)-NBits;



%se añade los bits extra calculados al vector dataIn para hacerlo
multiplo de k
dataInExtra = [dataIn,zeros(1,BitsExtra)];



% se crea una matriz en la cual se transforma el vector dataInExtra en
una matriz de grupos de tamaño k
dataInAgru = reshape(dataInExtra,[k ,(NBits+BitsExtra)/k]);



%se transforma los grupos de bits de tamaño k a número decimal
dataInAgru = bin2dec(num2str(dataInAgru.'));
```

En este punto del proyecto se creó una función desde cero para realizar la modulación QAM pero tras avanzar en dicho proyecto se decidió sustituir por la función de Matlab qammod() ya que requería un menor conste computacional y hace que el programa se ejecute con mayor rapidez.

```
[dataMOD] = qammod(dataInAgru,M); %Modulo data (data,M, angulo)



% se crea el vector tiempo de bits que será lo que dura un bit
tBit=   (0:(k*inc)-1)/Fs;

% se crea una matriz para poner los bits generadores por el modulador en
función de la frecuencia de portadora
% mediante la siguiente formula
A = real(dataMOD)*cos(2*pi*Fc*tBit) + imag(dataMOD)*sin(2*pi*Fc*tBit);
```

```
% se transforma la matriz en un vector donde cada bits comience donde
% finaliza el anterior.
dataOUT = reshape(A',[1 ,k * inc * NBitsExtra]);

%se crea un vector de tiempo para la representacion de la señal modulada
t = (0:(NBits+NBitsExtraVector)*inc -1)/Fs;

% señal de entrada añadiendo muestras por bit para representación.
[ModuladoraINC] = muestrasINC(VectorInExtra,inc);



%%%%%%%%%%%%%%%%%%%% REPRESENTACION Señal
Moduladora%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes2);
plot(t,ModuladoraINC,'linewidth',2);grid minor;

% limites plot
xlim([t(1) t(end)]);
ylim([-0.5, 1.5]);

% etiquetas plot
xlabel('Tiempo (sec)');
ylabel('Amplitud (V)');
```

5.2.3 Modulación M-FSK

```
%En primer lugar se obtiene el Numero de Bits de nuestro vector de
entrada, en este caso se llama dataIn
NBits=length(dataIn);



%También se necesita el número de bit que componen un símbolo, este valor
%se obtiene en función del orden de modulación con la siguiente formula
k   = log2(M);



%se calcula los bits extra necesarios para para que no quede ningún bit
%suelto ya que al agruparlos en grupos de k pueden quedar bits solos
BitsExtra=(ceil(NBits/k)*k)-NBits;



%se añade los bits extra calculados al vector dataIn para hacerlo
multiplo de k
dataInExtra = [dataIn,zeros(1,BitsExtra)];



% se crea una matriz en la cual se transforma el vector dataInExtra en
una matriz de grupos de tamaño k
dataInAgru = reshape(dataInExtra,[k ,(NBits+BitsExtra)/k]);



%se transforma los grupos de bits de tamaño k a número decimal
dataInAgru = bin2dec(num2str(dataInAgru.'));



%seleccionamos la frecuencia de cada grupo de bits. para esto ponemos las
%secuencias agrupadas en la frecuencia deseada. en este caso Fc y su
%frecuencia de separación.
dataOUT=(dataInAgru.*Fsep)+Fc;



% se crea el vector tiempo de bits que será lo que dura un bit
tBit=  (0:(k*inc)-1)/Fs;
```

En este punto se decidió implementar el modulador por medio de una matriz de las secuencias de bits ya que tenia un menor coste computación que por medio de bucles.

```
%se crea una matriz en la que cada grupo de frecuencias asignadas
%anteriormente se transforma en un seno con dicha frecuencia
dataOUT = sin(2*pi*dataOUT*tBit);
```

```
%se transforma la matriz en un vector donde cada secuencia de bits sigue
a la anterior
dataMOD = reshape(dataOUT',[1 ,k * inc * NBits]);


%se crea un vector de tiempo para la representación de la señal modulada
t = (0:(NBits+NBitsExtraVector)*inc -1)/Fs;

% señal de entrada añadiendo muestras por bit para representación.
[ModuladoraINC] = muestrasINC(VectorInExtra,inc);



%%%%%%%%%%%%%%%%%%%% REPRESENTACION Señal
Moduladora%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes2);
plot(t,ModuladoraINC,'linewidth',2);grid minor;

% limites plot
xlim([t(1) t(end)]);
ylim([-0.5, 1.5]);

% etiquetas plot
xlabel('Tiempo (sec)');
ylabel('Amplitud (V)');
```

5.2.4 Modulación MSK

```
Fmax   = Fc + (RB/4);
F   = Fc - (RB/4);

inc = round(Fs/RB); %muestras por bit. mínimo dos (Teorema de Nyquist)



k=2; %agrupamos de 2 en dos
NBits=length(dataIn); %Numero de Bits



%se calcula los bits extra necesarios para para que no quede ningún bit
%suelto ya que al agruparlos en grupos de k pueden quedar bits solos
BitsExtra=(ceil(NBits/k)*k)-NBits;

%se añade los bits extra calculados al vector dataIn para hacerlo
multiplo de k
VectorInExtra = [dataIn,zeros(1,BitsExtra)];



NBits = length(VectorInExtra); %Numero de Bits


% se crea el vector tiempo de bits que será lo que dura un bit
tBit=  (0:(k*inc)-1)/Fs;

% se agrupan pares e impares en diferentes vectores
bits_pares = dataIn(1:2:NBits)';
bits_impares = dataIn(2:2:NBits)';

% se crean dos vectores para Fmax y Fmin
A=sin(2*pi*Fmax*tBit);
B=sin(2*pi*Fmin*tBit);

%se agrupan la señal con las diferentes combinaciones de grupos de Bits
senal_00 = abs(bits_pares-1).*-(bits_impares-1);
senal_01 = abs(bits_pares-1).*bits_impares;
senal_10 = bits_pares.*-(bits_impares-1);
senal_11 = bits_pares.*bits_impares;

%se suman los vectores
dataOUT = senal_11+senal_00+senal_10+senal_01;

% se transforma la matriz en un vector para representación
ModuladaFc = reshape(dataOUT',[1 , inc * NBits]);



%%%%%%%%%%%%%%%%%%%%% REPRESENTACION Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% VECTOR DE TIEMPO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t = (0:(NBits+NBitsExtraVector)*inc -1)/Fs;
```

```
% señal de entrada añadiendo muestras por bit para representación.
[ModuladoraINC] = muestrasINC(VectorInExtra,inc);


%%%%%%%%%%%%%%%%%%%% REPRESENTACION Señal
Moduladora%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes2);
plot(t,ModuladoraINC,'linewidth',2);grid minor;

% limites plot
xlim([t(1) t(end)]);
ylim([-0.5, 1.5]);

% etiquetas plot
xlabel('Tiempo (sec)');
ylabel('Amplitud (V)');
title ('Señal Moduladora');
```

5.2.5 Modulación DBPSK

```
%En primer lugar se obtiene el Numero de Bits de nuestro vector de
entrada, en este caso se llama dataIn
NBits=length(dataIn);



%También se necesita el número de bit que componen un símbolo, este valor
%se obtiene en función del orden de modulación con la siguiente formula
k    = log2(M);



%se calcula los bits extra necesarios para para que no quede ningún bit
%suelto ya que al agruparlos en grupos de k pueden quedar bits solos
BitsExtra=(ceil(NBits/k)*k)-NBits;



%se añade los bits extra calculados al vector dataIn para hacerlo
multiplo de k
dataInExtra = [dataIn,zeros(1,BitsExtra)];



% se crea una matriz en la cual se transforma el vector dataInExtra en
una matriz de grupos de tamaño k
dataInAgru = reshape(dataInExtra,[k ,(NBits+BitsExtra)/k]);



%se transforma los grupos de bits de tamaño k a número decimal
dataInAgru = bin2dec(num2str(dataInAgru.'));
```

En este punto del proyecto se creó una función desde cero para realizar la modulación DBPSK pero tras avanzar en dicho proyecto se decidió sustituir por la función de Matlab `dpskmod` ya que requería un menor conste computacional y hace que el programa se ejecute con mayor rapidez.

```
[dataMOD] = dpskmod(dataInAgru,2); %Modulo data (data,M)


% se crea el vector tiempo de bits que será lo que dura un bit
tBit=  (0:(k*inc)-1)/Fs;

% se crea una matriz para poner los bits generadores por el modulador en
función de la frecuencia de portadora
% mediante la siguiente formula
A = real(dataMOD)*cos(2*pi*Fc*tBit) + imag(dataMOD)*sin(2*pi*Fc*tBit);

% se transforma la matriz en un vector donde cada bit comience donde
```

```
% finaliza el anterior.
dataOUT = reshape(A',[1 ,k * inc * NBitsExtra]);

%se crea un vector de tiempo para la representacion de la señal modulada
t = (0:(NBits+NBitsExtraVector)*inc -1)/Fs;

% señal de entrada añadiendo muestras por bit para representacion.
[ModuladoraINC] = muestrasINC(VectorInExtra,inc);



%%%%%%%%%%%%%%%%%%%% REPRESENTACION Señal
Moduladora%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes2);
plot(t,ModuladoraINC,'linewidth',2);grid minor;

% limites plot
xlim([t(1) t(end)]);
ylim([-0.5, 1.5]);

% etiquetas plot
xlabel('Tiempo (sec)');
ylabel('Amplitud (V)');
title ('Señal Moduladora');
```

Capítulo 6.

6. SIMULACIONES

Demostraciones sobre funcionamiento de guide:

https://www.youtube.com/watch?v=jvgDgKryQUk

https://www.youtube.com/watch?v=JI29qm-Yniw

Ejemplo de simulación empleando modulador M-PSK

https://www.youtube.com/watch?v=iQrU_TU7unY

Ejemplo de simulación empleando modulador M-QAM

https://www.youtube.com/watch?v=WzHCYLlmYHE

Ejemplo de simulación empleando modulador M-FSK

https://www.youtube.com/watch?v=QsCDebePmuk

Ejemplo de simulación empleando modulador MSK

https://www.youtube.com/watch?v=-OJS6StMcCY

Ejemplo de simulación empleando modulador DBPSK

https://www.youtube.com/watch?v=S5RyE2zMgrM

Ejemplo de simulación de imágenes

https://www.youtube.com/watch?v=U8LT54Loj8A&ab_channel=JuanEnci

Ejemplo de simulación de curva de BER

https://www.youtube.com/watch?v=QU_eaaDm3o0&ab_channel=JuanEnci

A continuación, se muestra un ejemplo de simulación de nuestro guide en M-QAM

Comenzamos con los siguientes valores:

- Fc =2000Hz.
- SNR = 20dB.
- M =4.

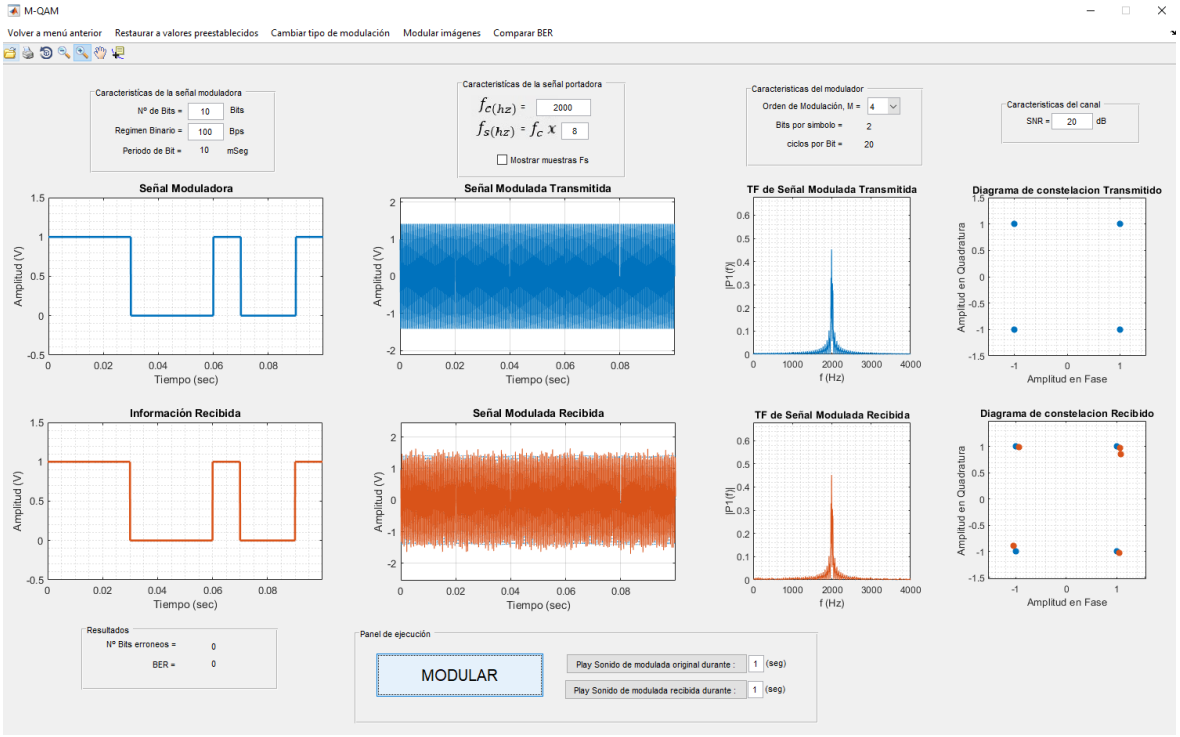
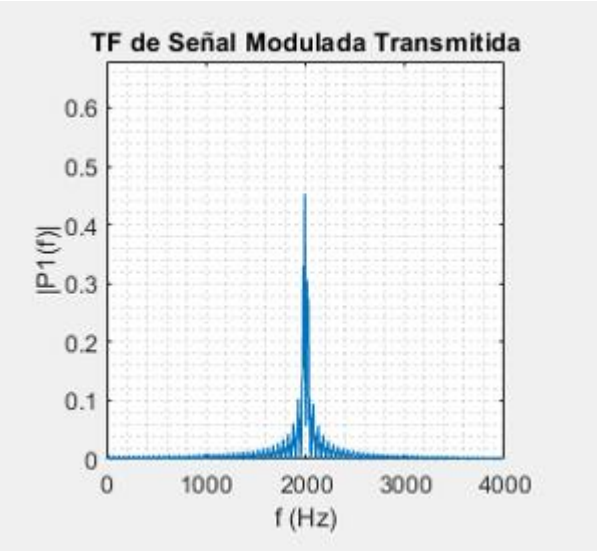


Figura 6.1.1 Simulacion M-QAM

Se aprecia en la Transformada de Fourier que la señal modulada se encuentra en 2000Hz que coincide con la F_c seleccionada, y el ancho de banda con el R_b preestablecido



También se aprecia en el diagrama de constelación el orden de modulación M=4.

Ahora modificamos el valor SNR para añadir ruido a SNR = 0dB.

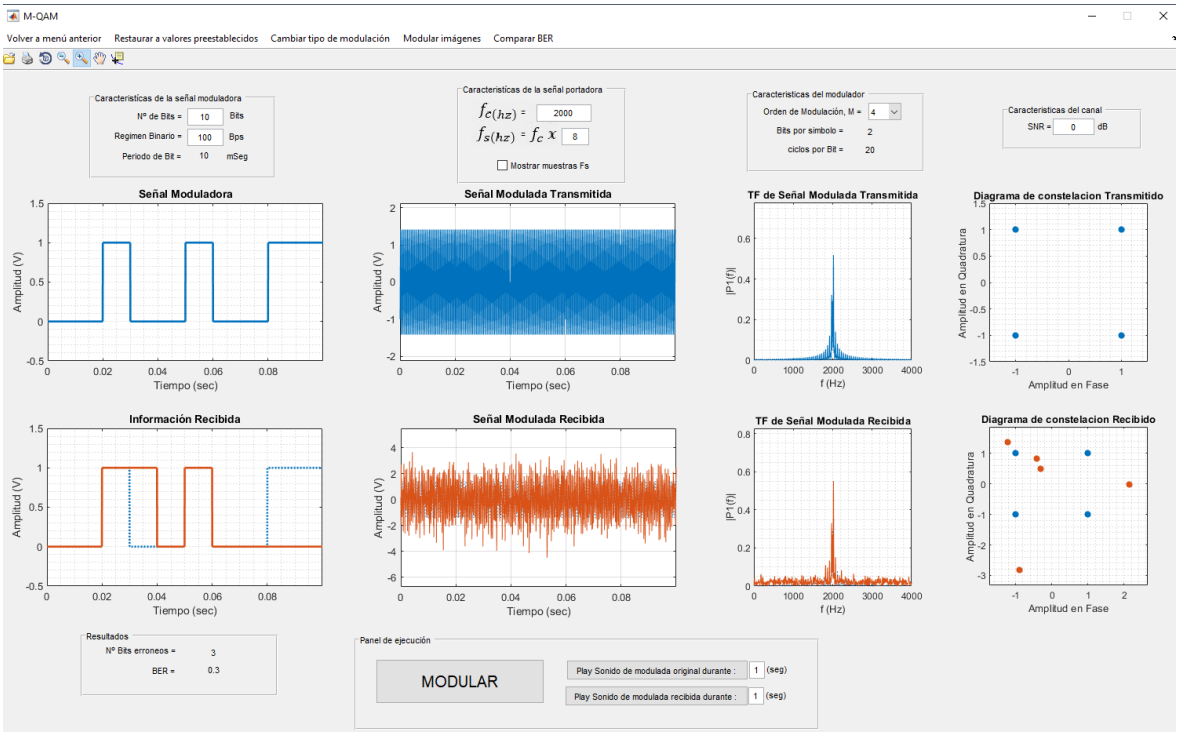


Figura 6.1.2 Simulacion

Observamos en el diagrama de constelación recibido que algunos puntos se salen de los márgenes originales mostrados en azul, lo que hace que la información recibida (la naranja) tenga 3 bits erróneos.

También se observa que la distorsión en la señal modulada recibida ha aumentado considerablemente.

A continuación, se muestra un ejemplo de simulación de nuestro guide A_MENU_2_0.m para modular imágenes.

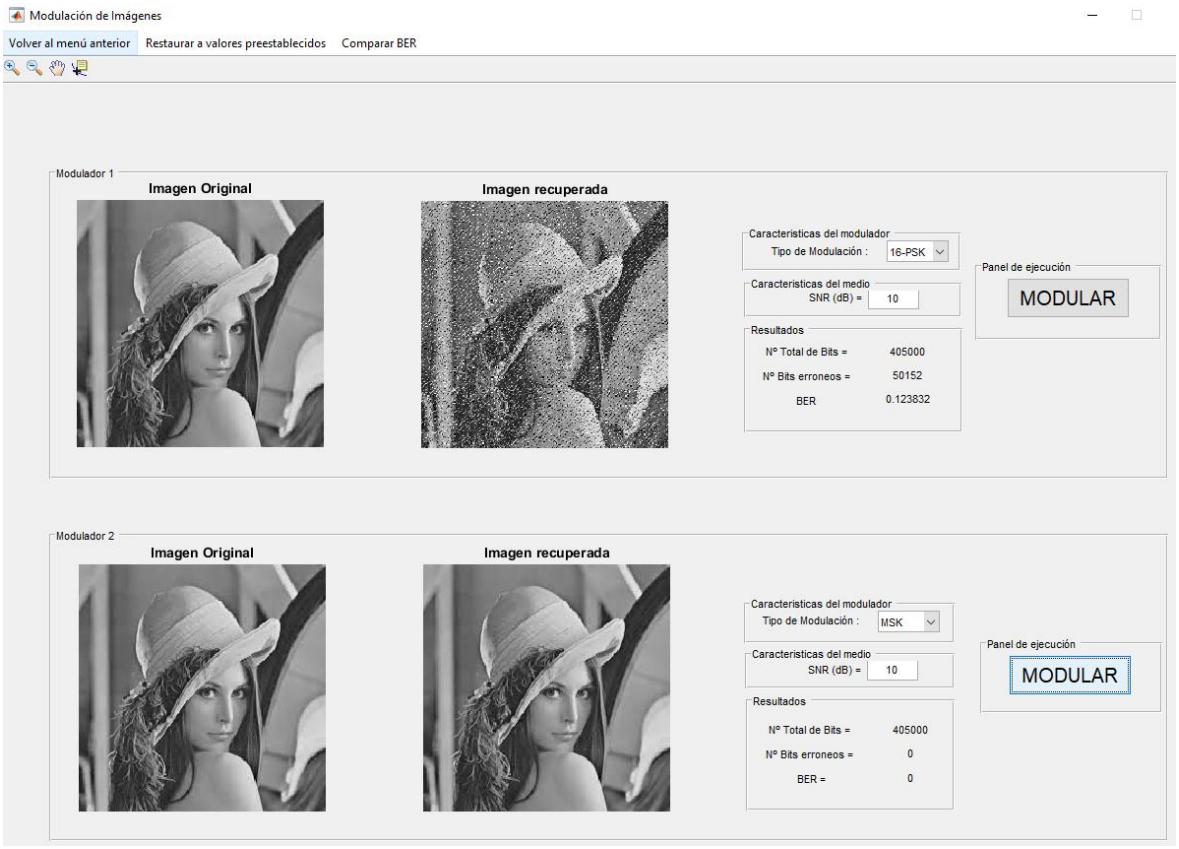


Figura 6.1.3 Simulacion modulación de imágenes

Aquí podemos comparar el resultado de modular una imagen por diferentes tipos de modulación pese.

Observamos que pese a tener el mismo coeficiente SNR la foto superior tiene un gran número de bits erróneos y una BER de 0.123832 debido al comportamiento y las distintas curvas de BER los diferentes modelos utilizados.

Figura 6.1.4 Resultado simulacion modulación de imágenes.

A continuación, se muestra un ejemplo de simulación de nuestro guide A_MENU_3_BER.m para comparar curvas BER.

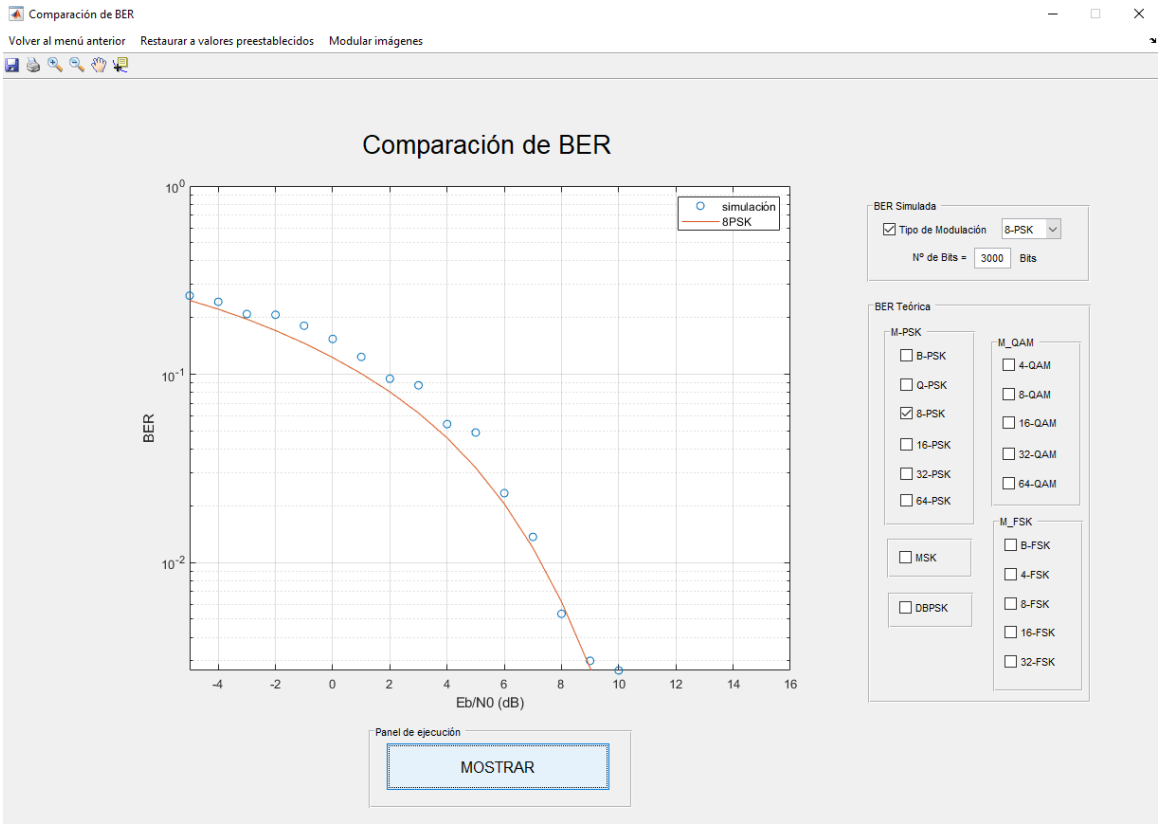


Figura 6.1.3 Simulación cruva BER.

Aquí podemos observar la comparación curva BER teórica con la curva BER simulada de un modulador 8PSK con una señal de 3000 bits de entrada.

A continuación, se muestra un ejemplo de simulación de nuestro guide A_MENU_1_5_DBPSK.m

Comenzamos con Fc = 4.000Hz y también vamos a modificar el valor SNR para añadir ruido, comenzamos modulando con SNR = 12dB.

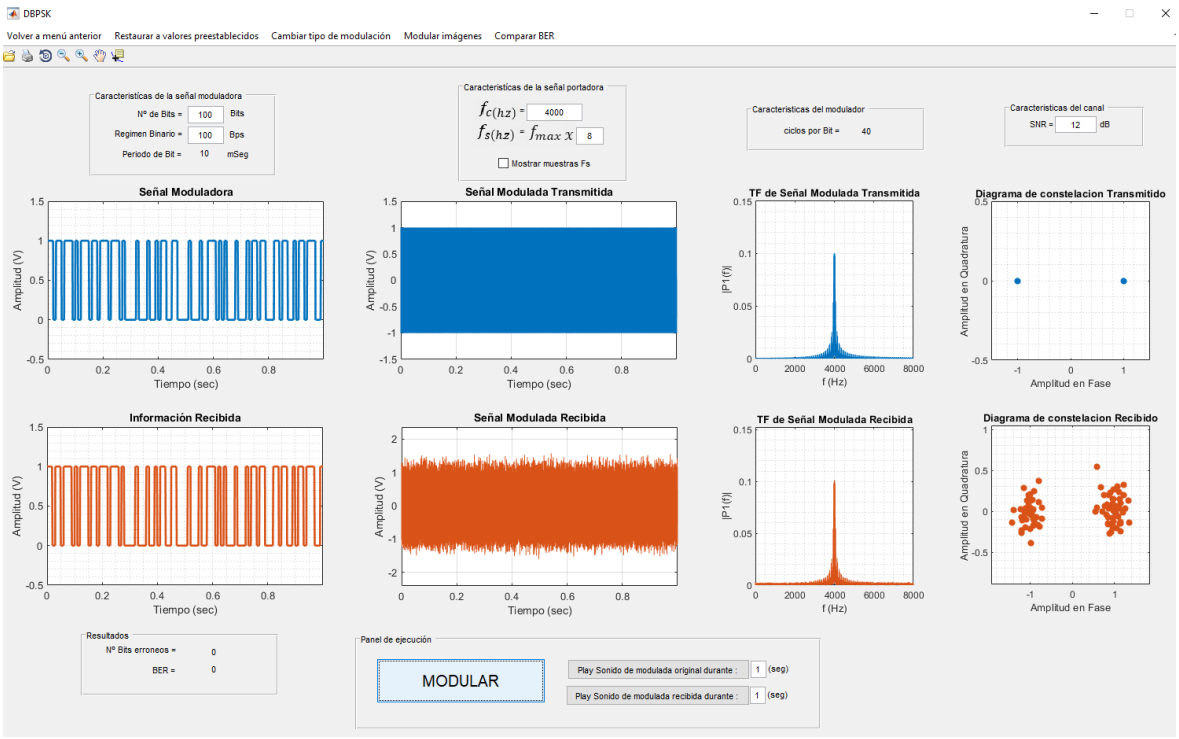


Figura 6.1.4 Simulacion DBPSK

En este tipo de modulación el orden de modulación es fijo M=2, se observa en el diagrama de constelación a la perfección.

También se aprecia Fc la transformada de Fourier.

Capítulo 7.

7. CONCLUSIONES

La principal conclusión llegados a este punto es que se ha logrado alcanzar el principal objetivo de este proyecto final de grado que consistía en la creación de una aplicación en un entorno grafico de Matlab, el cual permita observar las distintas modulaciones digitales fijadas en los objetivos y que esta herramienta tiene un gran valor didáctico que puede ayudar a comprender a fondo la naturaleza de los distintos sistemas digitales de modulación por parte de los estudiantes de la asignatura de Teoría de la Telecomunicación.

Esta aplicación dispone de una gran utilidad ya que permite realizar una comparativa entre diferentes modulaciones digitales atendiendo a conceptos como ancho de banda, probabilidad de error de bit, espectro de frecuencia, espectro de señal temporal, constelaciones y nos enseña algunas de sus ventajas como por ejemplo:

- Permiten mejorar y aumentar las velocidades de transmisión de bits gracias a la multiplexación y codificación.
- Mayor aprovechamiento del canal y del espectro gracias a la adaptación a los distintos medio y canales de transmisión
- Añade robustez y seguridad a la información que se transmite gracias a la codificación de dicha información

También resulta mas esclarecedor para los usuarios gracias a las representaciones y gráficos que permiten hacer zoom sobre los distintos diagramas como por ejemplo en las constelaciones o las representaciones de las señales en el dominio de la frecuencia y dominio del tiempo.

Se han analizado las distintas señales moduladas asociadas a una señal de usuario, en el dominio temporal y espectral mediante diversos parámetros del sistema (régimen binario, frecuencia de portadora, ...), probabilidad de error y análisis de las constelaciones transmitidas y recibidas simulando diferentes escenarios de comunicación. Tanto en señales aleatoria como en el ejemplo de modular la imagen de ejemplo lenna.tif.

También se han obtenido el sonido producido por las diferentes señales moduladas con ruido y sin ruido dependiendo de la frecuencia de portadora que se seleccione, lo que ayuda a comprender los diferentes sonidos producidos por una señal dependiendo de su frecuencia y el rango audible del oído humano.

Finalmente se concluye afirmando que el conocimiento y estudio de las modulaciones tiene una gran importancia a día de hoy ya que el conocimiento de esto permite evitar sus vulnerabilidades y explotar sus ventajas de cara a disminuir sus interferencias y en mi opinión el conocimiento de estas tecnologías y su comprensión ayuda enormemente de cara al desarrollo futuro de posibles nuevas tecnologías como por ejemplo en comunicaciones ópticas o móviles que se puedan llegar a desarrollar en un futuro, las cuales no sería posible de inventar si no se comprenden las tecnologías actuales y las anteriores.

Capítulo 8.

8. LÍNEAS FUTURAS

De cara a futuros estudios relacionados con los moduladores digitales se podrían añadir la posibilidad de introducir nuevas fuentes de información para modularlas aparte de señales binarias e imágenes como por ejemplo pistas de audio u otros elementos.

Profundizar en mejorar la eficiencia de las funciones creadas en ente proyecto para hacerlas más rápidas y eficientes reduciendo el coste computacional de las funciones actuales como por ejemplo la sustitución de algunas de las funciones creadas para los moduladores y demoduladores o las de agrupación de Bits de este proyecto por otras que no contengan bucles en la medida de lo posible.

Añadir nuevos entornos gráficos o nuevos guides con nuevas ilustraciones como puede ser el diagrama de ojo, diagrama de constelación en 3d o nuevos escenarios con otros tipos de ruido ambiental.

Implementación de otros tipos de modulaciones digitales como, por ejemplo:

- ASK
- M-ASK
- M-APK
- GMSK

Implementar o migrar este TFG creado mediante la herramienta Guide a la herramienta App Designer de Matlab, pese a que guide seguirá siendo compatible, esta herramienta será sustituida en futuras versiones de Matlab por la herramienta App Designer que permite nuevas alternativas de diseño como, por ejemplo:

- Gestión de un solo archivo para código e interfaz de usuario
- Aspecto moderno
- Escritorio basado en Toolstrip. Toolstrip organiza la funcionalidad de MATLAB en una serie de pestañas. Las pestañas se dividen en secciones que contienen una serie de controles relacionados. Los controles son botones, menús desplegables y otros elementos de la interfaz de usuario que utiliza para hacer cosas en MATLAB. Por ejemplo, la siguiente imagen muestra la pestaña Inicio con secciones para operaciones en Archivos, Variables, Código, etc. La sección Archivo tiene controles para realizar operaciones relacionadas con archivos, incluida la creación de secuencias de comandos (Nueva secuencia de comandos), la apertura de archivos (Abrir) y la comparación de dos archivos (Comparar).
- Interacciones de Canvas enriquecidas
- Editor de código integrado
- Gestión de metadatos de aplicaciones

Capítulo 9.

9. ANEXOS

Inicialmente para el desarrollo de este proyecto se han creado nueve guides diferentes para el usuario, todos y cada uno de ellos se muestran a continuación en color azul:

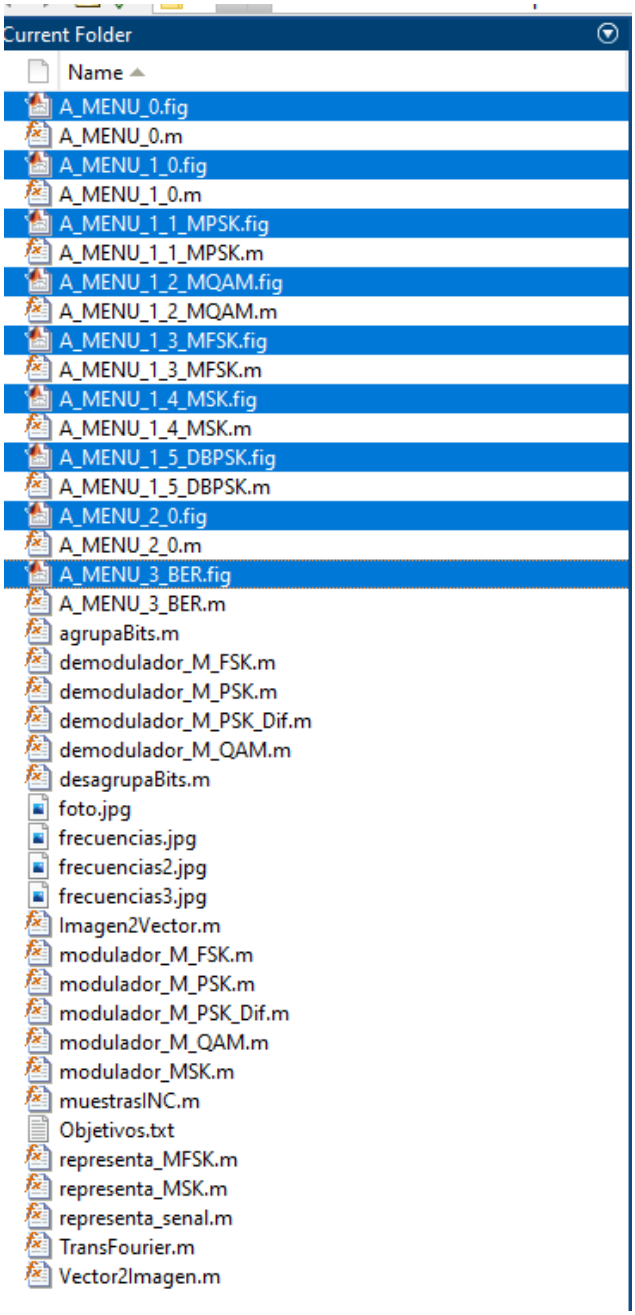


Figura 9.1.1 Lista de guides

- A_MENU_0.fig:

  Este guide es un menú principal donde el usuario elige entre 3 menús más que se muestran a continuación.

- A_MENU_1_0.fig

  Este guide es un menú para modular una señal binaria aleatoria, aquí debemos elegir entre 5 tipos de modulación que son los siguientes guides:

  - A_MENU_1_1_MPSK.fig

    Este guide es un entorno gráfico para modular una señal binaria aleatoria con un modulador M-PSK.
  - A_MENU_1_2_MQAM.fig

    Este guide es un entorno gráfico para modular una señal binaria aleatoria con un modulador M-QAM.

    A_MENU_1_3_MFSK.fig

    Este guide es un entorno gráfico para modular una señal binaria aleatoria con un modulador M-FSK.

    A_MENU_1_4_MSK.fig

    Este guide es un entorno gráfico para modular una señal binaria aleatoria con un modulador MSK.
  - A_MENU_1_5_DBPSK.fig

    Este guide es un entorno gráfico para modular una señal binaria aleatoria con un modulador DBPSK.

- A_MENU_2_0.fig

  Este guide es un entorno gráfico para modular imágenes, en este caso la imagen es Lenna.tif.

- A_MENU_3_BER.fig

  Este guide es un entorno gráfico que muestra la comparación de las curvas BER de los diferentes tipos de moduladores con diferentes ordenes de modulación.

Para el desarrollo del guide A_MENU_1_0.fig donde se modula una señal binaria creada por el usuario se ha necesitado crear varias funciones para todos y cada uno de los moduladores y demoduladores excepto para el demodulador MSK donde se ha empleado la función preexistente de Matlab mskdemod(). Las funciones son las siguientes que aparecen en azul:

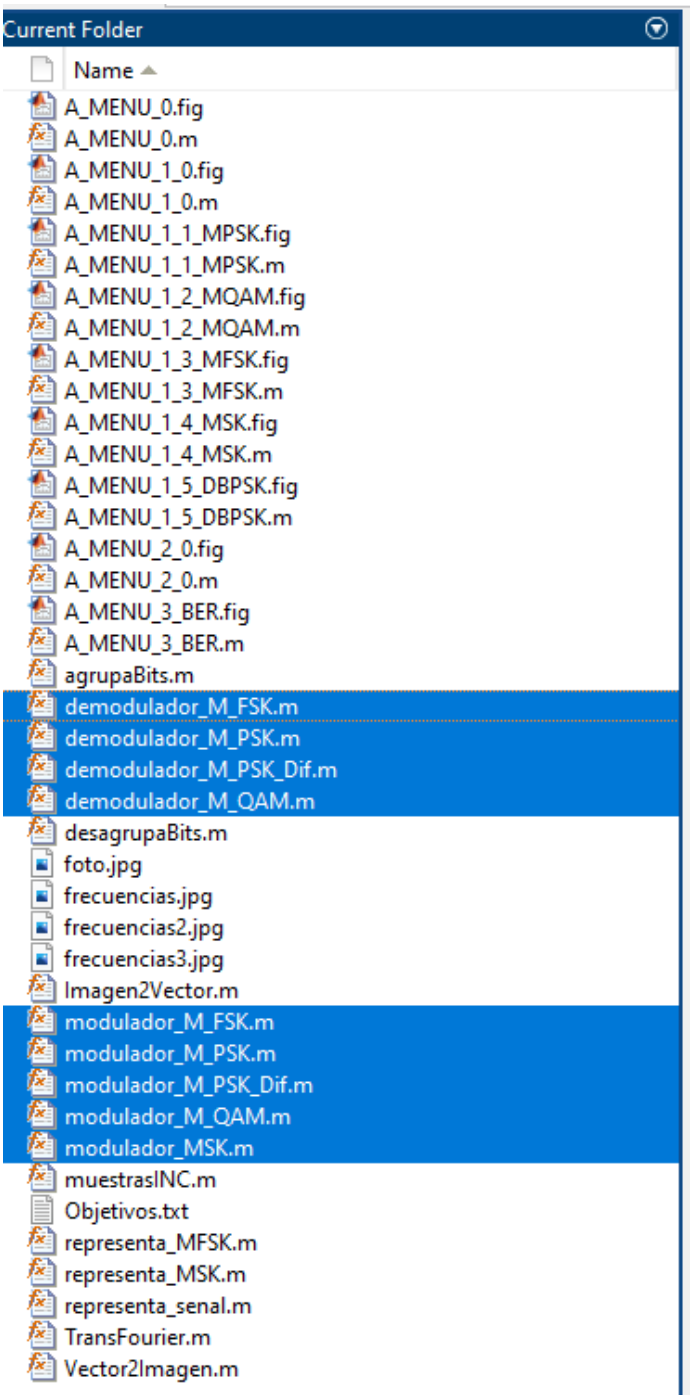


Figura 9.1.2 Lista de algoritmos de moduladores

Y las siguientes funciones para las representaciones gráficas

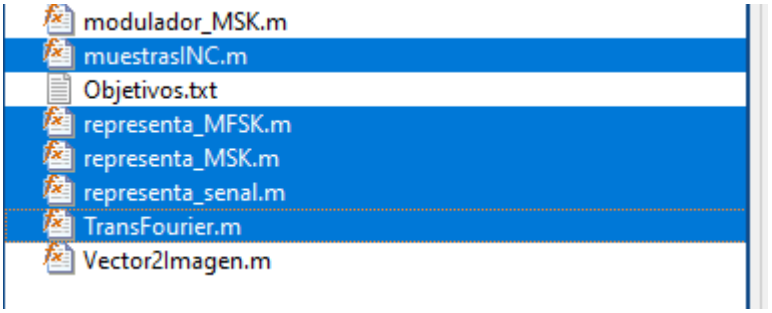


Figura 9.1.2 Lista de funciones para representaciones

- muestrasINC.m
  Esta función añade muestras a la señal binaria en función de fs. para la representación en plot().
- TrasnsFourier.m
  Realiza la transformada de Fourier para la representación.
- Representa_MFSK.m
  Para representar la señal en MFSK.
- Representa_MSK.m
  Para representar la señal en MSK.
- Representa_senal.m
  Para representar la señal de entrada.

Para la segunda parte de este proyecto en el guide A_MENU_2_0.fig fueron necesarias funciones para transformar las imágenes en vectores y funciones para agrupar la información en binario. Para esta misión se han creado las siguientes funciones que se muestran en azul:

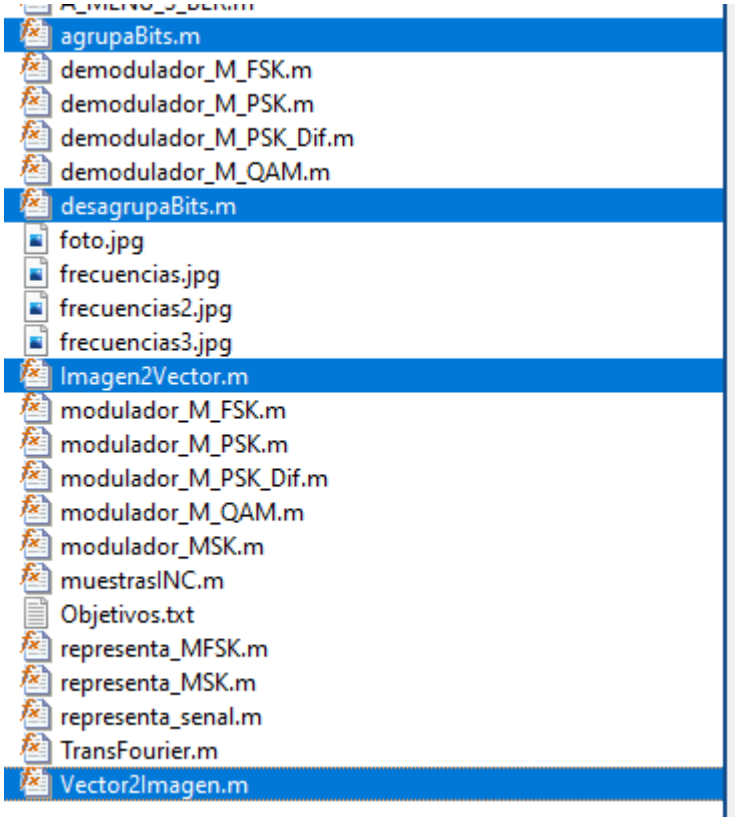


Figura 9.1.4 Lista de algoritmos para modular imagenes

- Imagen2vector.m

  Esta función transforma la matriz de la imagen en un vector
- agrupaBits.m

  Esta función transforma el vector en binario
- desagrupaBits.m (función inversa a agrupaBits.m)
- Vector2imagen.m (función inversa a Imagen2vector.m)

En la tercera parte de este proyecto en el guide A_MENU_3_BER.fig se han empleado las funciones creadas para los menús anteriores para hacer las simulaciones y las siguientes funciones preexistentes de Matlab para los resultados teóricos.

- berawgn() para curva de BER teórica de Matlab.
- semilogy() para la representación en dB.

El código perteneciente a los guides y las funciones creadas para el desarrollo de este proyecto se encuentran a continuación.

9.1. Código MATLAB

Función del Menú Principal Guide

```
function varargout = A_MENU_0(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @A_MENU_0_OpeningFcn, ...
                   'gui_OutputFcn',  @A_MENU_0_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
function A_MENU_0_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = A_MENU_0_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



function boton1_Callback(hObject, eventdata, handles)
close all
A_MENU_1_0

function boton2_Callback(hObject, eventdata, handles)
close all
A_MENU_2_0

function boton4_Callback(hObject, eventdata, handles)
close all
A_MENU_3_BER
```

Función M-PSK

```
function botonEjecutar_Callback(hObject, eventdata, handles)

%reloj del cursor on
set(handles.figure1, 'pointer', 'watch')
```

```
drawnow;


%%%%%%%%%%%%%%%%%%%%%%%%%% RESETS PLOTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
cla(handles.axes9)
cla(handles.axes8)
cla(handles.axes7)
cla(handles.axes6)
cla(handles.axes5)
cla(handles.axes4)
cla(handles.axes3)
cla(handles.axes2)


%%%%%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS de la visualización
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
muestraFs = get(handles.MuestraFs,'Value');  %ver muyestreo

%%%%%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS del Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
NBits = str2double(get(handles.NBits,'String'));  %Numero de Bits

RB =    str2double(get(handles.RB,'String'));  %Regimen Binario (Bits
por segundo) (minimo 10 para poder reproducir sonido)

Tb =  1000/RB;    %duracion de bit

set(handles.TB,'String',Tb);


%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS MODULADOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M =    str2double(get(handles.M,'String'));  %Orden de modulacion
(2,4,8,16,32,64,...)
  switch get(handles.M,'Value')
    case 1; M=2;
    case 2; M=4;
    case 3; M=8;
    case 4; M=16;
    case 5; M=32;
    case 6; M=64;
    otherwise
  end

fase =   pi* str2double(get(handles.fase,'String'));

Fc   = str2double(get(handles.Fc,'String'));  %frecuencia de la
portadora.
Fmax = Fc;
Fs   = Fmax * str2double(get(handles.mul,'String')); %frecuencia de
muestreo.


%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS DEL RUIDO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
SNR = str2double(get(handles.SNR,'String'));        %factor de ruido en
dB



%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS FIJAS MODULADOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
k    = log2(M); %numero de bit que componen un símbolo
set(handles.K,'String',k);

n    = Fc/RB; %ciclos por Bit (fijar como numero entero)
set(handles.CICLOS,'String',n);

inc = round(Fs/RB); %muestras por bit. minimo dos (Teorema de Nyquist)



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[Moduladora] = randi([0 1],1,NBits);              %Vector aleatorio de
entrada de NBits



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MODULADOR
VECTOR%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[Modulada, NBitsExtraVector, VectorInExtra] =
modulador_M_PSK(Moduladora,M,fase);
[ModuladaFc] = representa_senal(Modulada,M,Fc,Fs,inc);



%%%%%%%%%%%%%%%%%%%%%% REPRESENTACION Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% VECTOR DE TIEMPO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t = (0:(NBits+NBitsExtraVector)*inc -1)/Fs;
% señal de entrada añadiendo muestras por bit para representacion.
[ModuladoraINC] = muestrasINC(VectorInExtra,inc);



%%%%%%%%%%%%%%%%%%%%%% REPRESENTACION Señal
Moduladora%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes2);
plot(t,ModuladoraINC,'linewidth',2);grid minor;

% limites plot
xlim([t(1) t(end)]);
ylim([-0.5, 1.5]);

% etiquetas plot
xlabel('Tiempo (sec)');
ylabel('Amplitud (V)');
title ('Señal Moduladora');
```

```
%%%%%%%%%%%%%%%%%%%%% REPRESENTACION señal modulada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes3);

if muestraFs==1
    stem(t,ModuladaFc,'o');
else
    plot(t,ModuladaFc);
end
grid on;

% limites plot
xlim([t(1) t(end)]);
ylim([min(ModuladaFc)*1.5, max(ModuladaFc)*1.5]);

% etiquetas plot
xlabel('Tiempo (sec)');
ylabel('Amplitud (V)');
title ('Señal Modulada Transmitida');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
[Y,f] = TransFourier(ModuladaFc,Fs);

axes(handles.axes4);
plot(f,fftshift(Y)); grid minor;

% etiquetas plot
xlabel('f (Hz)');
ylabel('|P1(f)|');
title ('TF de Señal Modulada Transmitida');

% limites plot
ylim([0 1.5*max(fftshift(Y))]);
xlim([0 Fc+Fmax]);

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

%Diagrama de constelacion
axes(handles.axes5);
plot(real(Modulada),imag(Modulada),'*','linewidth',2); grid minor;

% etiquetas plot
xlabel('Amplitud en Fase')
ylabel('Amplitud en Quadratura')
title ('Diagrama de constelacion Transmitido');

% limites plot
xlim([(min(real(Modulada))-0.5) (max(real(Modulada))+0.5)]);
ylim([(min(imag(Modulada))-0.5) (max(imag(Modulada))+0.5)]);
```

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% AÑADIMOS RUIDO del canal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[ModuladaCanal] = awgn(Modulada,SNR,'measured'); %añadimos ruido blanco
[ModuladaCanalFc] = awgn(ModuladaFc,SNR,'measured'); %añadimos ruido
blanco


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

%Diagrama de constelacion con ruido
axes(handles.axes6);
plot(real(Modulada),imag(Modulada),'*','linewidth',2);
hold on
plot(real(ModuladaCanal),imag(ModuladaCanal),'*','linewidth',2); grid
minor;

% etiquetas plot
xlabel('Amplitud en Fase')
ylabel('Amplitud en Quadratura')
title ('Diagrama de constelacion Recibido');

% limites plot
xlim([(min(real(ModuladaCanal))-0.5) (max(real(ModuladaCanal))+0.5)]);
ylim([(min(imag(ModuladaCanal))-0.5) (max(imag(ModuladaCanal))+0.5)]);
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

[Yr,fr] = TransFourier(ModuladaCanalFc,Fs);

axes(handles.axes7);
plot(f,fftshift(Y),':',fr,fftshift(Yr)); grid minor;

% limites plot
ylim([0 1.5*max(fftshift(Yr))]);
xlim([0 Fc+Fmax]);

% etiquetas plot
xlabel('f (Hz)')
ylabel('|P1(f)|')
title ('TF de Señal Modulada Recibida');



%%%%%%%%%%%%%%%%%%%%% REPRESENTACION señal modulada recibida
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes8);


if muestraFs==1
    plot(t,ModuladaFc,':o');  grid on;
    hold on
    plot(t,ModuladaCanalFc, '-o');
else
```

```
    plot(t,ModuladaFc, ':'); grid on;
    hold on
    plot(t,ModuladaCanalFc);
end
hold off


% limites plot
xlim([t(1) t(end)]);
ylim([min(ModuladaCanalFc)*1.5, max(ModuladaCanalFc)*1.5]);

% etiquetas plot
xlabel('Tiempo (sec)')
ylabel('Amplitud (V)')
title (sprintf('Señal Modulada Recibida',M));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% DEMODULADOR VECTOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[VectorOUT] = demodulador_M_PSK(ModuladaCanal,M,fase);


% señal de salida añadiendo muestras por bit para representacion.
[datadataOUTinc] = muestrasINC(VectorOUT,inc);

axes(handles.axes9);
plot(t,ModuladoraINC,':',t,datadataOUTinc,'linewidth',2);  grid minor;
xlim([t(1) t(end)]);
xlabel('Tiempo (sec)')
ylim([-0.5, 1.5]);
ylabel('Amplitud (V)')
title('Información Recibida');





%%%%%%%%%%%%%%%%%%%%% opcion para ampliar graficas al mismo
tiempo%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Link the axes
linkaxes([handles.axes2,handles.axes3,handles.axes8,handles.axes9],'x')
;
% Link the axes
linkaxes([handles.axes4,handles.axes7],'x');





[err,ber] = biterr(VectorInExtra,VectorOUT);
set(handles.ERR,'String',err);
set(handles.BER,'String',ber);

%reloj del cursor off
set(handles.figure1, 'pointer', 'arrow')
```

Función M-QAM

```
%reloj del cursor on
set(handles.figure1, 'pointer', 'watch')
drawnow;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%% RESETS PLOTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
cla(handles.axes9)
cla(handles.axes8)
cla(handles.axes7)
cla(handles.axes6)
cla(handles.axes5)
cla(handles.axes4)
cla(handles.axes3)
cla(handles.axes2)


%%%%%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS de la visualización
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
muestraFs = get(handles.MuestraFs,'Value');  %ver muyestreo

%%%%%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS del Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
NBits = str2double(get(handles.NBits,'String'));  %Numero de Bits

RB =    str2double(get(handles.RB,'String'));  %Regimen Binario (Bits
por segundo) (minimo 10 para poder reproducir sonido)

Tb =  1000/RB;    %duracion de bit

set(handles.TB,'String',Tb);


%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS MODULADOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M =    str2double(get(handles.M,'String'));  %Orden de modulacion
(2,4,8,16,32,64,...)
  switch get(handles.M,'Value')
    case 1; M=4;
    case 2; M=8;
    case 3; M=16;
    case 4; M=32;
    case 5; M=64;
    otherwise
  end


Fc   = str2double(get(handles.Fc,'String'));  %frecuencia de la
portadora.
Fmax = Fc;
Fs   = Fmax * str2double(get(handles.mul,'String')); %frecuencia de
muestreo.


%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS DEL RUIDO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SNR = str2double(get(handles.SNR,'String'));       %factor de ruido en
dB
```

```
%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS FIJAS MODULADOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
k    = log2(M); %numero de bit que componen un simbolo
set(handles.K,'String',k);

n    = Fc/RB; %ciclos por Bit (fijar como numero entero)
set(handles.CICLOS,'String',n);

inc = round(Fs/RB); %muestras por bit. minimo dos (Teorema de Nyquist)



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[Moduladora] = randi([0 1],1,NBits);              %Vector aleaotrio de
entrada de NBits







%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MODULADOR
VECTOR%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[Modulada, NBitsExtraVector, VectorInExtra] =
modulador_M_QAM(Moduladora,M);
[ModuladaFc] = representa_senal(Modulada,M,Fc,Fs,inc);



%%%%%%%%%%%%%%%%%%%%%% REPRESENTACION Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% VECTOR DE TIEMPO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t = (0:(NBits+NBitsExtraVector)*inc -1)/Fs;
% señal de entrada añadiendo muestras por bit para representacion.
[ModuladoraINC] = muestrasINC(VectorInExtra,inc);



%%%%%%%%%%%%%%%%%%%%%% REPRESENTACION Señal
Moduladora%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes2);
plot(t,ModuladoraINC,'linewidth',2);grid minor;

% limites plot
xlim([t(1) t(end)]);
ylim([-0.5, 1.5]);

% etiquetas plot
xlabel('Tiempo (sec)');
ylabel('Amplitud (V)');
title ('Señal Moduladora');
```

```
%%%%%%%%%%%%%%%%%%%%% REPRESENTACION señal modulada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes3);

if muestraFs==1
    plot(t,ModuladaFc,'-o');
else
    plot(t,ModuladaFc);
end
grid on;

% limites plot
xlim([t(1) t(end)]);
ylim([min(ModuladaFc)*1.5, max(ModuladaFc)*1.5]);

% etiquetas plot
xlabel('Tiempo (sec)');
ylabel('Amplitud (V)');
title ('Señal Modulada Transmitida');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
[Y,f] = TransFourier(ModuladaFc,Fs);

axes(handles.axes4);
plot(f,fftshift(Y)); grid minor;

% etiquetas plot
xlabel('f (Hz)');
ylabel('|P1(f)|');
title ('TF de Señal Modulada Transmitida');

% limites plot
ylim([0 1.5*max(fftshift(Y))]);
xlim([0 Fc+Fmax]);


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

%Diagrama de constelacion
axes(handles.axes5);
plot(real(Modulada),imag(Modulada),'*','linewidth',2); grid minor;

% etiquetas plot
xlabel('Amplitud en Fase')
ylabel('Amplitud en Quadratura')
title ('Diagrama de constelacion Transmitido');

% limites plot
xlim([(min(real(Modulada))-0.5) (max(real(Modulada))+0.5)]);
ylim([(min(imag(Modulada))-0.5) (max(imag(Modulada))+0.5)]);
```

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% AÑADIMOS RUIDO del canal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[ModuladaCanal] = awgn(Modulada,SNR,'measured'); %añadimos ruido blanco
[ModuladaCanalFc] = awgn(ModuladaFc,SNR,'measured'); %añadimos ruido
blanco


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

%Diagrama de constelacion con ruido
axes(handles.axes6);
plot(real(Modulada),imag(Modulada),'*','linewidth',2);
hold on
plot(real(ModuladaCanal),imag(ModuladaCanal),'*','linewidth',2); grid
minor;

% etiquetas plot
xlabel('Amplitud en Fase')
ylabel('Amplitud en Quadratura')
title ('Diagrama de constelacion Recibido');

% limites plot
xlim([(min(real(ModuladaCanal))-0.5) (max(real(ModuladaCanal))+0.5)]);
ylim([(min(imag(ModuladaCanal))-0.5) (max(imag(ModuladaCanal))+0.5)]);
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

[Yr,fr] = TransFourier(ModuladaCanalFc,Fs);

axes(handles.axes7);
plot(f,fftshift(Y),':',fr,fftshift(Yr)); grid minor;

% limites plot
ylim([0 1.5*max(fftshift(Yr))]);
xlim([0 Fc+Fmax]);

% etiquetas plot
xlabel('f (Hz)')
ylabel('|P1(f)|')
title ('TF de Señal Modulada Recibida');




%%%%%%%%%%%%%%%%%%%%% REPRESENTACION señal modulada recibida
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes8);


if muestraFs==1
    plot(t,ModuladaFc,':o');  grid on;
    hold on
```

```
    plot(t,ModuladaCanalFc, '-o');
else
    plot(t,ModuladaFc, ':'); grid on;
    hold on
    plot(t,ModuladaCanalFc);
end
hold off


% limites plot
xlim([t(1) t(end)]);
ylim([min(ModuladaCanalFc)*1.5, max(ModuladaCanalFc)*1.5]);

% etiquetas plot
xlabel('Tiempo (sec)')
ylabel('Amplitud (V)')
title (sprintf('Señal Modulada Recibida',M));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% DEMODULADOR VECTOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[VectorOUT] = demodulador_M_QAM(ModuladaCanal,M);


% señal de salida añadiendo muestras por bit para representacion.
[datadataOUTinc] = muestrasINC(VectorOUT,inc);

axes(handles.axes9);
plot(t,ModuladoraINC,':',t,datadataOUTinc,'linewidth',2);  grid minor;
xlim([t(1) t(end)]);
xlabel('Tiempo (sec)')
ylim([-0.5, 1.5]);
ylabel('Amplitud (V)')
title('Información Recibida');




%%%%%%%%%%%%%%%%%%%%%% opcion para ampliar graficas al mismo
tiempo%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Link the axes
linkaxes([handles.axes2,handles.axes3,handles.axes8,handles.axes9],'x')
;
% Link the axes
linkaxes([handles.axes4,handles.axes7],'x');

[err,ber] = biterr(VectorInExtra,VectorOUT);
set(handles.ERR,'String',err);
set(handles.BER,'String',ber);

%reloj del cursor off
set(handles.figure1, 'pointer', 'arrow')
```

Función M-FSK

```
%reloj del cursor on
set(handles.figure1, 'pointer', 'watch')
drawnow;


%%%%%%%%%%%%%%%%%%%%%%%%%% RESETS PLOTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
cla(handles.axes9)
cla(handles.axes8)
cla(handles.axes7)
cla(handles.axes6)
cla(handles.axes5)
cla(handles.axes4)
cla(handles.axes3)
cla(handles.axes2)


%%%%%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS de la visualización
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
muestraFs = get(handles.MuestraFs,'Value');  %ver muyestreo

%%%%%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS del Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
NBits = str2double(get(handles.NBits,'String'));  %Numero de Bits

RB =    str2double(get(handles.RB,'String'));  %Regimen Binario (Bits
por segundo) (minimo 10 para poder reproducir sonido)

Tb =  1000/RB;    %duracion de bit

set(handles.TB,'String',Tb);


%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS MODULADOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M =    str2double(get(handles.M,'String'));  %Orden de modulacion
(2,4,8,16,32,64,...)
  switch get(handles.M,'Value')
    case 1; M=2;
    case 2; M=4;
    case 3; M=8;
    case 4; M=16;
    case 5; M=32;
    otherwise
  end


Fc   = str2double(get(handles.Fc,'String'));  %frecuencia de la
portadora.
Fsep = RB;
Fmax = Fc + Fsep*(M-1);
Fs   = Fmax * str2double(get(handles.mul,'String')); %frecuencia de
muestreo.
```

```
%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS DEL RUIDO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SNR = str2double(get(handles.SNR,'String'));       %factor de ruido en
dB



%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS FIJAS MODULADOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
k   = log2(M); %numero de bit que componen un simbolo
set(handles.K,'String',k);

n   = Fc/RB; %ciclos por Bit (fijar como numero entero)
set(handles.CICLOS,'String',n);

inc = round(Fs/RB); %muestras por bit. minimo dos (Teorema de Nyquist)



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[Moduladora] = randi([0 1],1,NBits);             %Vector aleaotrio de
entrada de NBits







%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MODULADOR
VECTOR%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[Modulada, NBitsExtraVector, VectorInExtra] =
modulador_M_FSK(Moduladora,M,Fsep,inc,Fs)

[ModuladaFc] = representa_MFSK(Moduladora,M,Fsep,inc,Fs,Fc);




%%%%%%%%%%%%%%%%%%%%% REPRESENTACION Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% VECTOR DE TIEMPO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t = (0:(NBits+NBitsExtraVector)*inc -1)/Fs;
% señal de entrada añadiendo muestras por bit para representacion.
[ModuladoraINC] = muestrasINC(VectorInExtra,inc);


%%%%%%%%%%%%%%%%%%%%% REPRESENTACION Señal
Moduladora%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes2);
plot(t,ModuladoraINC,'linewidth',2);grid minor;

% limites plot
```

```
xlim([t(1) t(end)]);
ylim([-0.5, 1.5]);

% etiquetas plot
xlabel('Tiempo (sec)');
ylabel('Amplitud (V)');
title ('Señal Moduladora');

%%%%%%%%%%%%%%%%%%%%% REPRESENTACION señal modulada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes3);

if muestraFs==1
    plot(t,ModuladaFc,'-o');
else
    plot(t,ModuladaFc);
end
grid on;

% limites plot
xlim([t(1) t(end)]);
ylim([min(ModuladaFc)*1.5, max(ModuladaFc)*1.5]);

% etiquetas plot
xlabel('Tiempo (sec)');
ylabel('Amplitud (V)');
title ('Señal Modulada Transmitida');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
[Y,f] = TransFourier(ModuladaFc,Fs);

axes(handles.axes4);
plot(f,fftshift(Y)); grid minor;

% etiquetas plot
xlabel('f (Hz)');
ylabel('|P1(f)|');
title ('TF de Señal Modulada Transmitida');

% limites plot
ylim([0 1.5*max(fftshift(Y))]);
xlim([0 Fc+Fmax]);


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

%Diagrama de constelacion
axes(handles.axes5);
plot(real(Modulada),imag(Modulada),'*','linewidth',2); grid minor;

% etiquetas plot
xlabel('Amplitud en Fase')
ylabel('Amplitud en Quadratura')
```

```
title ('Diagrama de constelacion Transmitido');

% limites plot
xlim([(min(real(Modulada))-0.5) (max(real(Modulada))+0.5)]);
ylim([(min(imag(Modulada))-0.5) (max(imag(Modulada))+0.5)]);


% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% AÑADIMOS RUIDO del canal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[ModuladaCanal] = awgn(Modulada,SNR,'measured'); %añadimos ruido blanco
[ModuladaCanalFc] = awgn(ModuladaFc,SNR,'measured'); %añadimos ruido
blanco


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

%Diagrama de constelacion con ruido
axes(handles.axes6);
plot(real(Modulada),imag(Modulada),'*','linewidth',2);
hold on
plot(real(ModuladaCanal),imag(ModuladaCanal),'*','linewidth',2); grid
minor;

% etiquetas plot
xlabel('Amplitud en Fase')
ylabel('Amplitud en Quadratura')
title ('Diagrama de constelacion Recibido');

% limites plot
xlim([(min(real(ModuladaCanal))-0.5) (max(real(ModuladaCanal))+0.5)]);
ylim([(min(imag(ModuladaCanal))-0.5) (max(imag(ModuladaCanal))+0.5)]);
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

[Yr,fr] = TransFourier(ModuladaCanalFc,Fs);

axes(handles.axes7);
plot(f,fftshift(Y),':',fr,fftshift(Yr)); grid minor;

% limites plot
ylim([0 1.5*max(fftshift(Yr))]);
xlim([0 Fc+Fmax]);

% etiquetas plot
xlabel('f (Hz)')
ylabel('|P1(f)|')
title ('TF de Señal Modulada Recibida');




%%%%%%%%%%%%%%%%%%%%% REPRESENTACION señal modulada recibida
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
axes(handles.axes8);


if muestraFs==1
    plot(t,ModuladaFc,':o');  grid on;
    hold on
    plot(t,ModuladaCanalFc, '-o');
else
    plot(t,ModuladaFc, ':'); grid on;
    hold on
    plot(t,ModuladaCanalFc);
end
hold off


% limites plot
xlim([t(1) t(end)]);
ylim([min(ModuladaCanalFc)*1.5, max(ModuladaCanalFc)*1.5]);

% etiquetas plot
xlabel('Tiempo (sec)')
ylabel('Amplitud (V)')
title (sprintf('Señal Modulada Recibida',M));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% DEMODULADOR VECTOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[VectorOUT] = demodulador_M_FSK(ModuladaCanal,M,Fsep,inc,Fs)

% señal de salida añadiendo muestras por bit para representacion.
[datadataOUTinc] = muestrasINC(VectorOUT,inc);

axes(handles.axes9);
plot(t,ModuladoraINC,':',t,datadataOUTinc,'linewidth',2);  grid minor;
xlim([t(1) t(end)]);
xlabel('Tiempo (sec)')
ylim([-0.5, 1.5]);
ylabel('Amplitud (V)')
title('Información Recibida');



%%%%%%%%%%%%%%%%%%%% opcion para ampliar graficas al mismo
tiempo%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Link the axes
linkaxes([handles.axes2,handles.axes3,handles.axes8,handles.axes9],'x')
;
% Link the axes
linkaxes([handles.axes4,handles.axes7],'x');


[err,ber] = biterr(VectorInExtra,VectorOUT);
set(handles.ERR,'String',err);
set(handles.BER,'String',ber);

%reloj del cursor off
set(handles.figure1, 'pointer', 'arrow')
```

Función MSK

```
%reloj del cursor on
set(handles.figure1, 'pointer', 'watch')
drawnow;


%%%%%%%%%%%%%%%%%%%%%%%%%% RESETS PLOTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
cla(handles.axes9)
cla(handles.axes8)
cla(handles.axes7)
cla(handles.axes6)
cla(handles.axes5)
cla(handles.axes4)
cla(handles.axes3)
cla(handles.axes2)


%%%%%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS de la visualización
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
muestraFs = get(handles.MuestraFs,'Value');  %ver muyestreo

%%%%%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS del Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
NBits = str2double(get(handles.NBits,'String'));  %Numero de Bits

RB =    str2double(get(handles.RB,'String'));  %Regimen Binario (Bits
por segundo) (minimo 10 para poder reproducir sonido)

Tb =  1000/RB;    %duracion de bit

set(handles.TB,'String',Tb);


%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS MODULADOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Fc = str2double(get(handles.Fc,'String'));  %frecuencia de la
portadora.
F1   = Fc + (RB/4);
F2   = Fc - (RB/4);
Fmax = max(F1,F2);

Fs = Fmax * str2double(get(handles.mul,'String')); %frecuencia de
muestreo.


%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS DEL RUIDO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SNR = str2double(get(handles.SNR,'String'));       %factor de ruido en
dB



%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS FIJAS MODULADOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n   = Fc/RB; %ciclos por Bit (fijar como numero entero)
```

```
set(handles.CICLOS,'String',n);

inc = round(Fs/RB); %muestras por bit. minimo dos (Teorema de Nyquist)



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[Moduladora] = randi([0 1],1,NBits);              %Vector aleaotrio de
entrada de NBits



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MODULADOR
VECTOR%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[Modulada,NBitsExtraVector,VectorInExtra] =
modulador_MSK(Moduladora,inc);
[ModuladaFc] = representa_MSK(Moduladora,inc,F1,F2,Fs);

%%%%%%%%%%%%%%%%%%%%%% REPRESENTACION Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% VECTOR DE TIEMPO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t = (0:(NBits+NBitsExtraVector)*inc -1)/Fs;
% señal de entrada añadiendo muestras por bit para representacion.
[ModuladoraINC] = muestrasINC(VectorInExtra,inc);


%%%%%%%%%%%%%%%%%%%%%% REPRESENTACION Señal
Moduladora%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes2);
plot(t,ModuladoraINC,'linewidth',2);grid minor;

% limites plot
xlim([t(1) t(end)]);
ylim([-0.5, 1.5]);

% etiquetas plot
xlabel('Tiempo (sec)');
ylabel('Amplitud (V)');
title ('Señal Moduladora');

%%%%%%%%%%%%%%%%%%%%%% REPRESENTACION señal modulada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes3);

if muestraFs==1
    plot(t,ModuladaFc,'-o');
else
    plot(t,ModuladaFc);
end
grid on;
```

```
% limites plot
xlim([t(1) t(end)]);
ylim([min(ModuladaFc)*1.5, max(ModuladaFc)*1.5]);

% etiquetas plot
xlabel('Tiempo (sec)');
ylabel('Amplitud (V)');
title ('Señal Modulada Transmitida');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
[Y,f] = TransFourier(ModuladaFc,Fs);

axes(handles.axes4);
plot(f,fftshift(Y)); grid minor;

% etiquetas plot
xlabel('f (Hz)');
ylabel('|P1(f)|');
title ('TF de Señal Modulada Transmitida');

% limites plot
ylim([0 1.5*max(fftshift(Y))]);
xlim([0 Fc+Fmax]);


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

%Diagrama de constelacion
axes(handles.axes5);
plot(real(Modulada),imag(Modulada),'*','linewidth',2); grid minor;

% etiquetas plot
xlabel('Amplitud en Fase')
ylabel('Amplitud en Quadratura')
title ('Diagrama de constelacion Transmitido');

% limites plot
xlim([(min(real(Modulada))-0.5) (max(real(Modulada))+0.5)]);
ylim([(min(imag(Modulada))-0.5) (max(imag(Modulada))+0.5)]);


% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% AÑADIMOS RUIDO del canal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[ModuladaCanal] = awgn(Modulada,SNR,'measured'); %añadimos ruido blanco
[ModuladaCanalFc] = awgn(ModuladaFc,SNR,'measured'); %añadimos ruido
blanco


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

%Diagrama de constelacion con ruido
axes(handles.axes6);
```

```
plot(real(Modulada),imag(Modulada),'*','linewidth',2);
hold on
plot(real(ModuladaCanal),imag(ModuladaCanal),'*','linewidth',2); grid
minor;

% etiquetas plot
xlabel('Amplitud en Fase')
ylabel('Amplitud en Quadratura')
title ('Diagrama de constelacion Recibido');

% limites plot
xlim([(min(real(ModuladaCanal))-0.5)  (max(real(ModuladaCanal))+0.5)]);
ylim([(min(imag(ModuladaCanal))-0.5)  (max(imag(ModuladaCanal))+0.5)]);
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%

[Yr,fr] = TransFourier(ModuladaCanalFc,Fs);

axes(handles.axes7);
plot(f,fftshift(Y),':',fr,fftshift(Yr)); grid minor;

% limites plot
ylim([0 1.5*max(fftshift(Yr))]);
xlim([0 Fc+Fmax]);

% etiquetas plot
xlabel('f (Hz)')
ylabel('|P1(f)|')
title ('TF de Señal Modulada Recibida');




%%%%%%%%%%%%%%%%%%%%%% REPRESENTACION señal modulada recibida
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes8);


if muestraFs==1
    plot(t,ModuladaFc,':o');  grid on;
    hold on
    plot(t,ModuladaCanalFc, '-o');
else
    plot(t,ModuladaFc, ':'); grid on;
    hold on
    plot(t,ModuladaCanalFc);
end
hold off



% limites plot
xlim([t(1) t(end)]);
ylim([min(ModuladaCanalFc)*1.5, max(ModuladaCanalFc)*1.5]);
```

```
% etiquetas plot
xlabel('Tiempo (sec)')
ylabel('Amplitud (V)')
title (sprintf('Señal Modulada Recibida'));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% DEMODULADOR VECTOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[VectorOUT] = mskdemod(ModuladaCanal,inc);

% señal de salida añadiendo muestras por bit para representacion.
[datadataOUTinc] = muestrasINC(VectorOUT,inc);

axes(handles.axes9);


plot(t,ModuladoraINC,':',t,datadataOUTinc,'linewidth',2);  grid minor;
xlim([t(1) t(end)]);
xlabel('Tiempo (sec)')
ylim([-0.5, 1.5]);
ylabel('Amplitud (V)')
title('Información Recibida');





%%%%%%%%%%%%%%%%%%%%% opcion para ampliar graficas al mismo
tiempo%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Link the axes
linkaxes([handles.axes2,handles.axes3,handles.axes8,handles.axes9],'x')
;
% Link the axes
linkaxes([handles.axes4,handles.axes7],'x');





[err,ber] = biterr(VectorInExtra,VectorOUT);
set(handles.ERR,'String',err);
set(handles.BER,'String',ber);

%reloj del cursor off
set(handles.figure1, 'pointer', 'arrow')
```

Función DBPSK

```
%reloj del cursor on
set(handles.figure1, 'pointer', 'watch')
drawnow;


%%%%%%%%%%%%%%%%%%%%%%%%%% RESETS PLOTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
cla(handles.axes9)
cla(handles.axes8)
cla(handles.axes7)
cla(handles.axes6)
cla(handles.axes5)
cla(handles.axes4)
cla(handles.axes3)
cla(handles.axes2)


%%%%%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS de la visualización
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
muestraFs = get(handles.MuestraFs,'Value');  %ver muyestreo

%%%%%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS del Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
NBits = str2double(get(handles.NBits,'String'));  %Numero de Bits

RB =    str2double(get(handles.RB,'String'));  %Regimen Binario (Bits
por segundo) (minimo 10 para poder reproducir sonido)

Tb =  1000/RB;    %duracion de bit

set(handles.TB,'String',Tb);


%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS MODULADOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS MODULADOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Fc = str2double(get(handles.Fc,'String'));  %frecuencia de la
portadora.
Fmax = Fc;
Fs = Fmax * str2double(get(handles.mul,'String')); %frecuencia de
muestreo.


%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS DEL RUIDO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SNR = str2double(get(handles.SNR,'String'));       %factor de ruido en
dB



%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS FIJAS MODULADOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n    = Fc/RB; %ciclos por Bit (fijar como numero entero)
set(handles.CICLOS,'String',n);
```

```
inc = round(Fs/RB); %muestras por bit. minimo dos (Teorema de Nyquist)



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[Moduladora] = randi([0 1],1,NBits);             %Vector aleaotrio de
entrada de NBits




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MODULADOR
VECTOR%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[Modulada, NBitsExtraVector, VectorInExtra] =
modulador_M_PSK_Dif(Moduladora);
[ModuladaFc] = representa_senal(Modulada,2,Fc,Fs,inc);


%%%%%%%%%%%%%%%%%%%%% REPRESENTACION Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% VECTOR DE TIEMPO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t = (0:(NBits+NBitsExtraVector)*inc -1)/Fs;
% señal de entrada añadiendo muestras por bit para representacion.
[ModuladoraINC] = muestrasINC(VectorInExtra,inc);


%%%%%%%%%%%%%%%%%%%%% REPRESENTACION Señal
Moduladora%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes2);
plot(t,ModuladoraINC,'linewidth',2);grid minor;

% limites plot
xlim([t(1) t(end)]);
ylim([-0.5, 1.5]);

% etiquetas plot
xlabel('Tiempo (sec)');
ylabel('Amplitud (V)');
title ('Señal Moduladora');

%%%%%%%%%%%%%%%%%%%%% REPRESENTACION señal modulada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes3);

if muestraFs==1
    plot(t,ModuladaFc,'-o');
else
    plot(t,ModuladaFc);
end
grid on;

% limites plot
```

```
xlim([t(1) t(end)]);
ylim([min(ModuladaFc)*1.5, max(ModuladaFc)*1.5]);

% etiquetas plot
xlabel('Tiempo (sec)');
ylabel('Amplitud (V)');
title ('Señal Modulada Transmitida');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
[Y,f] = TransFourier(ModuladaFc,Fs);

axes(handles.axes4);
plot(f,fftshift(Y)); grid minor;

% etiquetas plot
xlabel('f (Hz)');
ylabel('|P1(f)|');
title ('TF de Señal Modulada Transmitida');

% limites plot
ylim([0 1.5*max(fftshift(Y))]);
xlim([0 Fc+Fmax]);


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

%Diagrama de constelacion
axes(handles.axes5);
plot(real(Modulada),imag(Modulada),'*','linewidth',2); grid minor;

% etiquetas plot
xlabel('Amplitud en Fase')
ylabel('Amplitud en Quadratura')
title ('Diagrama de constelacion Transmitido');

% limites plot
xlim([(min(real(Modulada))-0.5) (max(real(Modulada))+0.5)]);
ylim([(min(imag(Modulada))-0.5) (max(imag(Modulada))+0.5)]);


% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% AÑADIMOS RUIDO del canal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[ModuladaCanal] = awgn(Modulada,SNR,'measured'); %añadimos ruido blanco
[ModuladaCanalFc] = awgn(ModuladaFc,SNR,'measured'); %añadimos ruido
blanco


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

%Diagrama de constelacion con ruido
axes(handles.axes6);
plot(real(Modulada),imag(Modulada),'*','linewidth',2);
```

```
hold on
plot(real(ModuladaCanal),imag(ModuladaCanal),'*','linewidth',2); grid
minor;

% etiquetas plot
xlabel('Amplitud en Fase')
ylabel('Amplitud en Quadratura')
title ('Diagrama de constelacion Recibido');

% limites plot
xlim([(min(real(ModuladaCanal))-0.5) (max(real(ModuladaCanal))+0.5)]);
ylim([(min(imag(ModuladaCanal))-0.5) (max(imag(ModuladaCanal))+0.5)]);
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

[Yr,fr] = TransFourier(ModuladaCanalFc,Fs);

axes(handles.axes7);
plot(f,fftshift(Y),':',fr,fftshift(Yr)); grid minor;

% limites plot
ylim([0 1.5*max(fftshift(Yr))]);
xlim([0 Fc+Fmax]);

% etiquetas plot
xlabel('f (Hz)')
ylabel('|P1(f)|')
title ('TF de Señal Modulada Recibida');




%%%%%%%%%%%%%%%%%%%%%% REPRESENTACION señal modulada recibida
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes8);


if muestraFs==1
    plot(t,ModuladaFc,':o');  grid on;
    hold on
    plot(t,ModuladaCanalFc, '-o');
else
    plot(t,ModuladaFc, ':'); grid on;
    hold on
    plot(t,ModuladaCanalFc);
end
hold off



% limites plot
xlim([t(1) t(end)]);
ylim([min(ModuladaCanalFc)*1.5, max(ModuladaCanalFc)*1.5]);

% etiquetas plot
```

```
xlabel('Tiempo (sec)')
ylabel('Amplitud (V)')
title (sprintf('Señal Modulada Recibida'));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% DEMODULADOR VECTOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[VectorOUT] = demodulador_M_PSK_Dif(ModuladaCanal);

% señal de salida añadiendo muestras por bit para representacion.
[datadataOUTinc] = muestrasINC(VectorOUT,inc);

axes(handles.axes9);
plot(t,ModuladoraINC,':',t,datadataOUTinc,'linewidth',2);  grid minor;
xlim([t(1) t(end)]);
xlabel('Tiempo (sec)')
ylim([-0.5, 1.5]);
ylabel('Amplitud (V)')
title('Información Recibida');




%%%%%%%%%%%%%%%%%%%% opcion para ampliar graficas al mismo
tiempo%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Link the axes
linkaxes([handles.axes2,handles.axes3,handles.axes8,handles.axes9],'x')
;
% Link the axes
linkaxes([handles.axes4,handles.axes7],'x');




[err,ber] = biterr(VectorInExtra,VectorOUT);
set(handles.ERR,'String',err);
set(handles.BER,'String',ber);

%reloj del cursor off
set(handles.figure1, 'pointer', 'arrow')
```

Función para modular imágenes

```
%reloj del cursor on
set(handles.figure1, 'pointer', 'watch')
drawnow;




%%%%%%%%%%%%%%%%%%%%%%%%%%% RESETS PLOTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc
cla(handles.axes9);


%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS DEL RUIDO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SNR = str2double(get(handles.SNR,'String'));       %factor de ruido en
dB






%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Imagen de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Imagen = imread('foto.jpg');                        %leemos imagen y
obtenemos matriz
Imagen = rgb2gray(Imagen);                          %tranforma la imagen
rgb a estala de grises
[ImagenIn,dimension] = Imagen2Vector(Imagen); % pasamos la imagen a
vector

axes(handles.axes2);
imshow(Imagen,[0 255]);
title('Imagen Original');




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MODULADOR
IMAGEN%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ImagenMod= [1];
  switch get(handles.Mod,'Value')




    case 1
            [ImagenMod] = modulador_M_PSK(ImagenIn,2,0);
    case 2
            [ImagenMod] = modulador_M_PSK(ImagenIn,4,0);
    case 3
            [ImagenMod] = modulador_M_PSK(ImagenIn,8,0);
    case 4
            [ImagenMod] = modulador_M_PSK(ImagenIn,16,0);
    case 5
            [ImagenMod] = modulador_M_PSK(ImagenIn,32,0);
```

```
    case 6
            [ImagenMod] = modulador_M_PSK(ImagenIn,64,0);
    case 7
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 8
            [ImagenMod] = modulador_M_QAM(ImagenIn,4);
    case 9
            [ImagenMod] = modulador_M_QAM(ImagenIn,8);
    case 10
            [ImagenMod] = modulador_M_QAM(ImagenIn,16);
    case 11
            [ImagenMod] = modulador_M_QAM(ImagenIn,32);
    case 12
            [ImagenMod] = modulador_M_QAM(ImagenIn,64);
    case 13
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 14
            [ImagenMod] = modulador_M_FSK(ImagenIn,2,8,8,16);
    case 15
            [ImagenMod] = modulador_M_FSK(ImagenIn,4,8,8,32);
    case 16
            [ImagenMod] = modulador_M_FSK(ImagenIn,8,8,8,64);
    case 17
            [ImagenMod] = modulador_M_FSK(ImagenIn,16,8,8,128);
    case 18
            [ImagenMod] = modulador_M_FSK(ImagenIn,32,8,8,256);
    case 19
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 20
            [ImagenMod] = mskmod(ImagenIn,8);
    case 21
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 22
            [ImagenMod] = modulador_M_PSK_Dif(ImagenIn);

  end




% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%% AÑADIMOS RUIDO EN SEÑALES MODULADAS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[ImagenModR]  = awgn(ImagenMod, SNR,'measured'); %añadimos ruido blanco




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% DEMODULADOR
IMAGEN%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[ImagenDemod] = ImagenIn+1000;

  switch get(handles.Mod,'Value')

    case 1
            [ImagenDemod] = demodulador_M_PSK(ImagenModR,2,0);
```

```
    case 2
            [ImagenDemod] = demodulador_M_PSK(ImagenModR,4,0);
    case 3
            [ImagenDemod] = demodulador_M_PSK(ImagenModR,8,0);
    case 4
            [ImagenDemod] = demodulador_M_PSK(ImagenModR,16,0);
    case 5
            [ImagenDemod] = demodulador_M_PSK(ImagenModR,32,0);
    case 6
            [ImagenDemod] = demodulador_M_PSK(ImagenModR,64,0);
    case 7
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 8
            [ImagenDemod] = demodulador_M_QAM(ImagenModR,4);
    case 9
            [ImagenDemod] = demodulador_M_QAM(ImagenModR,8);
    case 10
            [ImagenDemod] = demodulador_M_QAM(ImagenModR,16);
    case 11
            [ImagenDemod] = demodulador_M_QAM(ImagenModR,32);
    case 12
            [ImagenDemod] = demodulador_M_QAM(ImagenModR,64);
    case 13
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 14
            [ImagenDemod] = demodulador_M_FSK(ImagenModR,2,8,8,16);
    case 15
            [ImagenDemod] = demodulador_M_FSK(ImagenModR,4,8,8,32);
    case 16
            [ImagenDemod] = demodulador_M_FSK(ImagenModR,8,8,8,64);
    case 17
            [ImagenDemod] = demodulador_M_FSK(ImagenModR,16,8,8,128);
    case 18
            [ImagenDemod] = demodulador_M_FSK(ImagenModR,32,8,8,256);
    case 19
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 20
            [ImagenDemod] = mskdemod(ImagenModR,8);
    case 21
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 22
            [ImagenDemod] = demodulador_M_PSK_Dif(ImagenModR);

  end




% recuperar imagen a partir del vector
[ImagenOUT] = Vector2Imagen(ImagenDemod , dimension , size(Imagen));


axes(handles.axes9);
imshow(ImagenOUT,[0 255]);
title(sprintf('Imagen recuperada'));
```

```
%%%%%%%%%%%%%%%%%%%% opcion para ampliar graficas al mismo
tiempo%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Link the axes
linkaxes([handles.axes2,handles.axes9],'xy');


size(ImagenIn)
size(ImagenDemod)



[Bit_err,BER] = biterr(ImagenIn,ImagenDemod);
NBits=length(ImagenIn);


set(handles.NBits,'String',NBits);
set(handles.Bit_err,'String',Bit_err);
set(handles.BER,'String',BER);


%reloj del cursor off
set(handles.figure1, 'pointer', 'arrow')






% --- botonEjecutar 2.
function botonEjecutar2_Callback(hObject, eventdata, handles)


%reloj del cursor on
set(handles.figure1, 'pointer', 'watch')
drawnow;




%%%%%%%%%%%%%%%%%%%%%%%%%% RESETS PLOTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clc
cla(handles.axes13);




%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS DEL RUIDO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
SNR = str2double(get(handles.SNR2,'String'));       %factor de ruido en
dB




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Imagen de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Imagen = imread('foto.jpg');                       %leemos imagen y
obtemenos matriz
Imagen = rgb2gray(Imagen);                         %tranforma la imagen
rgb a estala de grises
[ImagenIn,dimension] = Imagen2Vector(Imagen); % pasamos la imagen a
vector



axes(handles.axes12);
imshow(Imagen,[0 255]);
title('Imagen Original');



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MODULADOR
IMAGEN%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ImagenMod= [1];
  switch get(handles.Mod2,'Value')


    case 1
            [ImagenMod] = modulador_M_PSK(ImagenIn,2,0);
    case 2
            [ImagenMod] = modulador_M_PSK(ImagenIn,4,0);
    case 3
            [ImagenMod] = modulador_M_PSK(ImagenIn,8,0);
    case 4
            [ImagenMod] = modulador_M_PSK(ImagenIn,16,0);
    case 5
            [ImagenMod] = modulador_M_PSK(ImagenIn,32,0);
    case 6
            [ImagenMod] = modulador_M_PSK(ImagenIn,64,0);
    case 7
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 8
            [ImagenMod] = modulador_M_QAM(ImagenIn,4);
    case 9
            [ImagenMod] = modulador_M_QAM(ImagenIn,8);
    case 10
            [ImagenMod] = modulador_M_QAM(ImagenIn,16);
    case 11
            [ImagenMod] = modulador_M_QAM(ImagenIn,32);
    case 12
            [ImagenMod] = modulador_M_QAM(ImagenIn,64);
    case 13
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 14
            [ImagenMod] = modulador_M_FSK(ImagenIn,2,8,8,16);
```

```
    case 15
            [ImagenMod] = modulador_M_FSK(ImagenIn,4,8,8,32);
    case 16
            [ImagenMod] = modulador_M_FSK(ImagenIn,8,8,8,64);
    case 17
            [ImagenMod] = modulador_M_FSK(ImagenIn,16,8,8,128);
    case 18
            [ImagenMod] = modulador_M_FSK(ImagenIn,32,8,8,256);
    case 19
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 20
            [ImagenMod] = mskmod(ImagenIn,8);
    case 21
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 22
            [ImagenMod] = modulador_M_PSK_Dif(ImagenIn);

  end




% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% AÑADIMOS RUIDO EN SEÑALES MODULADAS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[ImagenModR]  = awgn(ImagenMod, SNR,'measured'); %añadimos ruido blanco




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% DEMODULADOR
IMAGEN%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[ImagenDemod] = ImagenIn+1000;

  switch get(handles.Mod2,'Value')

   case 1
            [ImagenDemod] = demodulador_M_PSK(ImagenModR,2,0);
    case 2
            [ImagenDemod] = demodulador_M_PSK(ImagenModR,4,0);
    case 3
            [ImagenDemod] = demodulador_M_PSK(ImagenModR,8,0);
    case 4
            [ImagenDemod] = demodulador_M_PSK(ImagenModR,16,0);
    case 5
            [ImagenDemod] = demodulador_M_PSK(ImagenModR,32,0);
    case 6
            [ImagenDemod] = demodulador_M_PSK(ImagenModR,64,0);
    case 7
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 8
            [ImagenDemod] = demodulador_M_QAM(ImagenModR,4);
    case 9
            [ImagenDemod] = demodulador_M_QAM(ImagenModR,8);
    case 10
            [ImagenDemod] = demodulador_M_QAM(ImagenModR,16);
    case 11
```

```
            [ImagenDemod] = demodulador_M_QAM(ImagenModR,32);
    case 12
            [ImagenDemod] = demodulador_M_QAM(ImagenModR,64);
    case 13
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 14
            [ImagenDemod] = demodulador_M_FSK(ImagenModR,2,8,8,16);
    case 15
            [ImagenDemod] = demodulador_M_FSK(ImagenModR,4,8,8,32);
    case 16
            [ImagenDemod] = demodulador_M_FSK(ImagenModR,8,8,8,64);
    case 17
            [ImagenDemod] = demodulador_M_FSK(ImagenModR,16,8,8,128);
    case 18
            [ImagenDemod] = demodulador_M_FSK(ImagenModR,32,8,8,256);
    case 19
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 20
            [ImagenDemod] = mskdemod(ImagenModR,8);
    case 21
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 22
            [ImagenDemod] = demodulador_M_PSK_Dif(ImagenModR);

  end



% recuperar imagen a partir del vector
[ImagenOUT] = Vector2Imagen(ImagenDemod , dimension , size(Imagen));


axes(handles.axes13);
imshow(ImagenOUT,[0 255]);
title(sprintf('Imagen recuperada'));


%%%%%%%%%%%%%%%%%%%%%% opcion para ampliar graficas al mismo
tiempo%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Link the axes
linkaxes([handles.axes12,handles.axes13],'xy');


[Bit_err2,BER2] = biterr(ImagenIn,ImagenDemod);
NBits2=length(ImagenIn);


set(handles.NBits2,'String',NBits2);
set(handles.Bit_err2,'String',Bit_err2);
set(handles.BER2,'String',BER2);


%reloj del cursor off
set(handles.figure1, 'pointer', 'arrow')
```

Función para comprar BER

```
%reloj del cursor on
set(handles.figure1, 'pointer', 'watch')
drawnow;

clc
clear leyenda
cla(handles.axes1)

SNR = -10:1:35;

leyenda=[];


Simulacion =  get(handles.Simulacion,'Value');

BPSK =  get(handles.BPSK,'Value');
QPSK =  get(handles.QPSK,'Value');
PSK8 =  get(handles.PSK8,'Value');
PSK16 = get(handles.PSK16,'Value');
PSK32 = get(handles.PSK32,'Value');
PSK64 = get(handles.PSK64,'Value');

QAM4  =  get(handles.QAM4,'Value');
QAM8  =  get(handles.QAM8,'Value');
QAM16 =  get(handles.QAM16,'Value');
QAM32 =  get(handles.QAM32,'Value');
QAM64 =  get(handles.QAM64,'Value');

BFSK   =  get(handles.BFSK,'Value');
FSK4   =  get(handles.FSK4,'Value');
FSK8   =  get(handles.FSK8,'Value');
FSK16  =  get(handles.FSK16,'Value');
FSK32  =  get(handles.FSK32,'Value');

MSK  =  get(handles.MSK,'Value');

DBPSK  =  get(handles.DBPSK,'Value');



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

axes(handles.axes1);


% %%%%%%% M_PSK %%%%%%%%%%
%
    if BPSK==1
        BER_BPSK =  berawgn(SNR,'psk', 2,'nondiff');
        semilogy(SNR, BER_BPSK);
        hold on
        leyenda=[leyenda;'BPSK '];
    end
```

```
    if QPSK==1
        BER_QPSK =  berawgn(SNR,'psk', 4,'nondiff');
        semilogy(SNR, BER_QPSK);
        hold on
        leyenda=[leyenda;'QPSK '];
    end

    if PSK8==1
        BER_8PSK =  berawgn(SNR,'psk', 8,'nondiff');
        semilogy(SNR, BER_8PSK);
        hold on
        leyenda=[leyenda;'8PSK '];
    end

     if PSK16==1
        BER_16PSK = berawgn(SNR,'psk',16,'nondiff');
        semilogy(SNR, BER_16PSK);
        hold on
        leyenda=[leyenda;'16PSK'];
     end


     if PSK32==1
        BER_32PSK = berawgn(SNR,'psk',32,'nondiff');
        semilogy(SNR, BER_32PSK);
        hold on
        leyenda=[leyenda;'32PSK'];
     end

    if PSK64==1
        BER_64PSK = berawgn(SNR,'psk',64,'nondiff');
        semilogy(SNR, BER_64PSK);
        hold on
        leyenda=[leyenda;'64PSK'];
    end



% %%%%%%% M_QAM %%%%%%%%%%

    if QAM4==1
        BER_4QAM =  berawgn(SNR,'qam', 4);
        semilogy(SNR, BER_4QAM);
        hold on
        leyenda=[leyenda;'4QAM '];
    end

    if QAM8==1
        BER_8QAM =  berawgn(SNR,'qam', 8);
        semilogy(SNR, BER_8QAM);
        hold on
        leyenda=[leyenda;'8QAM '];
    end

    if QAM16==1
        BER_16QAM =  berawgn(SNR,'qam', 16);
        semilogy(SNR, BER_16QAM);
```

```
        hold on
        leyenda=[leyenda;'16QAM'];
    end

    if QAM32==1
        BER_32QAM =  berawgn(SNR,'qam', 32);
        semilogy(SNR, BER_32QAM);
        hold on
        leyenda=[leyenda;'32QAM'];
    end

    if QAM64==1
        BER_64QAM =  berawgn(SNR,'qam', 64);
        semilogy(SNR, BER_64QAM);
        hold on
        leyenda=[leyenda;'64QAM'];
    end



% %%%%%%% M_FSK %%%%%%%%%%

    if BFSK==1
        BER_BFSK =  berawgn(SNR,'fsk', 2,'coherent');
        semilogy(SNR, BER_BFSK);
        hold on
        leyenda=[leyenda;'BFSK '];
    end

    if FSK4==1
        BER_4FSK =  berawgn(SNR,'fsk', 4,'coherent');
        semilogy(SNR, BER_4FSK);
        hold on
        leyenda=[leyenda;'4FSK '];
    end

    if FSK8==1
        BER_8FSK =  berawgn(SNR,'fsk', 8,'coherent');
        semilogy(SNR, BER_8FSK);
        hold on
        leyenda=[leyenda;'8FSK '];
    end

    if FSK16==1
        BER_16FSK =  berawgn(SNR,'fsk', 16,'coherent');
        semilogy(SNR, BER_16FSK);
        hold on
        leyenda=[leyenda;'16FSK'];
    end

    if FSK32==1
        BER_32FSK =  berawgn(SNR,'fsk', 32,'coherent');
        semilogy(SNR, BER_32FSK);
        hold on
        leyenda=[leyenda;'32FSK'];
    end

% %%%%%%% MSK %%%%%%%%%%
```

```
    if MSK==1
        BER_MSK =  berawgn(SNR,'msk','off');
        semilogy(SNR, BER_MSK);
        hold on
        leyenda=[leyenda;'MSK  '];
    end


%

% ******* DBPSK **********

    if DBPSK==1
        BER_DBPSK =  berawgn(SNR,'dpsk', 2);
        semilogy(SNR, BER_DBPSK);
        hold on
        leyenda=[leyenda;'DBPSK'];
    end






% ******* simulacion **********


if Simulacion==1

%%%%%%%%%%%%%%%%%%%%%%%%%% CARACTERISTICAS del Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
NBits = str2double(get(handles.NBits,'String'));  %Numero de Bits

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Vector de entrada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[Moduladora] = randi([0 1],1,NBits);              %Vector aleaotrio de
entrada de NBits


TipoMod=0;



switch get(handles.Mod,'Value')


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% M-PSK
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 1
      M=2; TipoMod=1;

    case 2
      M=4; TipoMod=1;
    case 3
      M=8; TipoMod=1;
    case 4
```

```
      M=16; TipoMod=1;
    case 5
      M=32; TipoMod=1;
    case 6
      M=64; TipoMod=1;
    case 7
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    case 8
      M=4;   TipoMod=2;
    case 9
      M=8;   TipoMod=2;
    case 10
      M=16;   TipoMod=2;
    case 11
      M=32;   TipoMod=2;
    case 12
      M=64;   TipoMod=2;
    case 13
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 14
      M=2;   freqSep = 20; TipoMod=3;
    case 15
      M=4;   freqSep = 25; TipoMod=3;
    case 16
      M=8;   freqSep = 27; TipoMod=3;
    case 17
      M=16;   freqSep = 27; TipoMod=3;
    case 18
      M=32;   freqSep = 27; TipoMod=3;
    case 19
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 20
               TipoMod=4;
    case 21
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 22
               TipoMod=5;


end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
if TipoMod ==1
    [Modulada, NBitsExtraVector, VectorInExtra] =
modulador_M_PSK(Moduladora,M,0);
Canal = comm.AWGNChannel('BitsPerSymbol',log2(M));
    for i = 1:length(SNR)
       Canal.EbNo = SNR(i);
       Recibida = Canal(Modulada);
       VectorDemod = demodulador_M_PSK(Recibida,M,0);
   %calculo de la BER simulada
    [NBits_erroneos(i),BER_simulada(i)] =
biterr(VectorInExtra,VectorDemod);
%      end
         if BER_simulada(i)==0
            break
        end
```

```
    end
 SNR= SNR(1:length(BER_simulada));
semilogy(SNR, BER_simulada,'o');
leyenda=[leyenda;'simu '];
 end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

if TipoMod==2
    k=log2(M);
    [Modulada,BitsExtra,dataInExtra] = modulador_M_QAM(Moduladora,M);

    for i = 1:length(SNR)
     Recibida = awgn(Modulada,SNR(i)+k,'measured'); %añadimos ruido
blanco
     VectorDemod = demodulador_M_QAM(Recibida,M);

   %calculo de la BER simulada
    [NBits_erroneos(i),BER_simulada(i)] =
biterr(dataInExtra,VectorDemod);
%      end
        if BER_simulada(i)==0
             break
        end

    end
 SNR= SNR(1:length(BER_simulada));
semilogy(SNR, BER_simulada,'o');
leyenda=[leyenda;'simu '];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
if TipoMod==3
    Modulador   = comm.FSKModulator(M,freqSep);
    Demodulador = comm.FSKDemodulator(M,freqSep);

    Modulada = Modulador(Moduladora');
    for i = 1:length(SNR)

        [Recibida] = awgn(Modulada,SNR(i),'measured'); %añadimos ruido
blanco
        VectorDemod = Demodulador(Recibida);

         %calculo de la BER simulada
        [NBits_erroneos(i),BER_simulada(i)] =
biterr(Moduladora',VectorDemod);

         if BER_simulada(i)==0
              break
         end

    end
SNR= SNR(1:length(BER_simulada));
semilogy(SNR, BER_simulada,'o');
leyenda=[leyenda;'simu '];
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
if TipoMod==4
    Modulador = comm.MSKModulator('BitInput', true,
'InitialPhaseOffset', pi/4);
    Demod = comm.MSKDemodulator('BitOutput', true,
'InitialPhaseOffset', pi/4);

    VectorMod = Modulador(Moduladora');
    BER_simulada=0;
    for i = 1:length(SNR)
    canal = comm.AWGNChannel('NoiseMethod',  'Signal to noise ratio
(SNR)','SNR',SNR(i)-9);
    Recibida = step(canal, VectorMod);
    VectorDemod = Demod(Recibida);
    hError = comm.ErrorRate('ReceiveDelay', Demod.TracebackDepth);
    [errorStats] = step(hError, Moduladora', VectorDemod);
    BER_simulada(i)=errorStats(1);
        if BER_simulada(i)==0
            break
        end
SNR= SNR(1:length(BER_simulada));
semilogy(SNR, BER_simulada,'o');
leyenda=[leyenda;'simu '];
    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%


end

if TipoMod==5

[Modulada, NBitsExtraVector, VectorInExtra] =
modulador_M_PSK_Dif(Moduladora)
Canal = comm.AWGNChannel('BitsPerSymbol',log2(2));
    for i = 1:length(SNR)
       Canal.EbNo = SNR(i);
       Recibida = Canal(Modulada);
       VectorDemod = demodulador_M_PSK_Dif(Recibida);
   %calculo de la BER simulada
    [NBits_erroneos(i),BER_simulada(i)] =
biterr(VectorInExtra,VectorDemod);
%     end
         if BER_simulada(i)==0
            break
        end
    end
SNR= SNR(1:length(BER_simulada));
semilogy(SNR, BER_simulada,'o');
leyenda=[leyenda;'simu '];
 end
```

```
 end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


grid on
axis([SNR(1) SNR(length(SNR)) 1e-5 1]);
legend(leyenda);


xlabel('Eb/N0 (dB)');
ylabel('BER');



%reloj del cursor off
set(handles.figure1, 'pointer', 'arrow')
```

9.2. Manual de usuario de GUIDE

9.2.1 Menú principal

Para ejecutar el guide es necesario abrir el menú principal que es el siguiente:

A_MENU_0.m

El guide consta de 3 opciones principales que son las siguientes:

1. Para modular una secuencia aleatoria de bits
2. Para modular imágenes
3. Para ver Graficas de comparación de la BER de los distintos tipos de moduladores

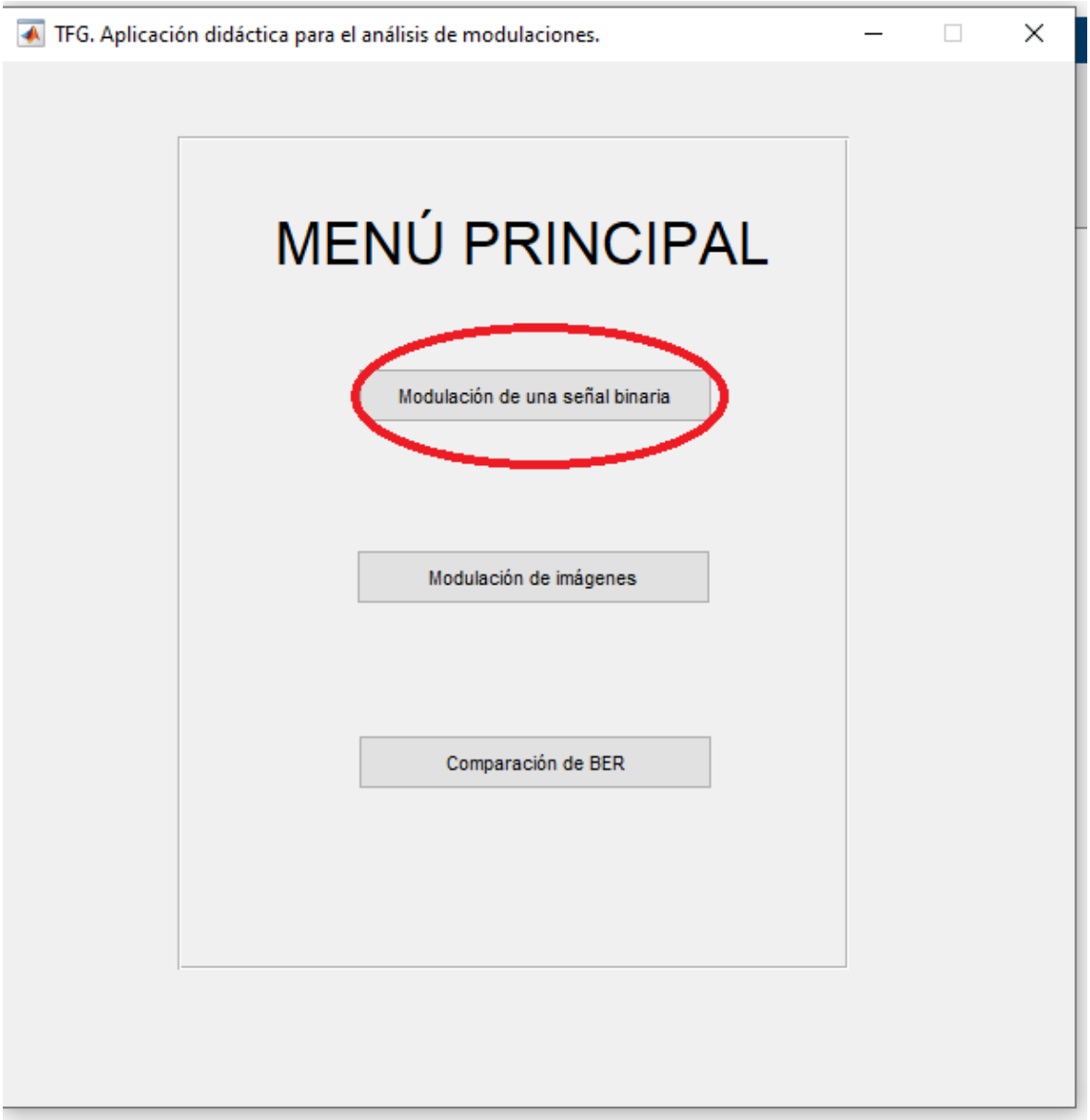


Figura 9.2.1.1 Menu principal de guide

Si pinchamos en la primera opción aparecerán nuevas opciones en las cuales debemos elegir el tipo de modulación con la que vamos a trabajar, explicaremos la primera (M-PSK)

aparece el siguiente menú que es donde se introducen los parámetros.

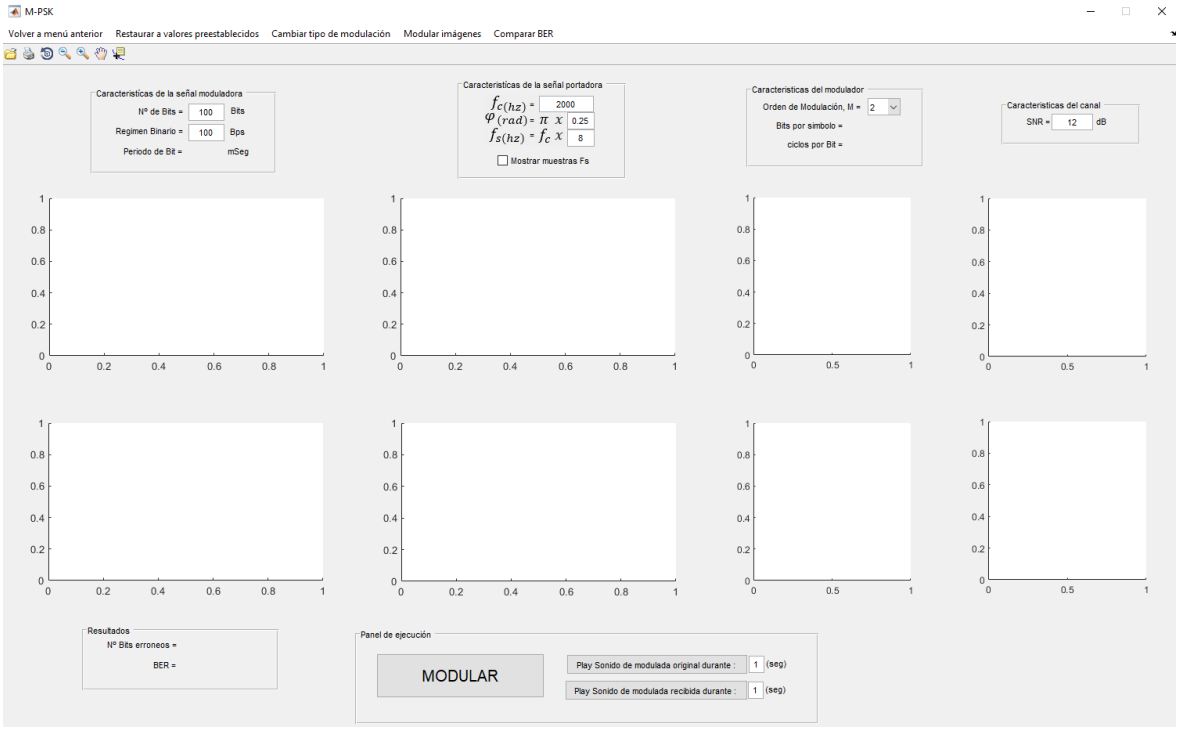


Figura 9.2.2 Modulación M-PSK

En el primer recuadro podemos modificar los parámetros de la señal moduladora

- Como por ejemplo el N.º de bits y también el Régimen Binario

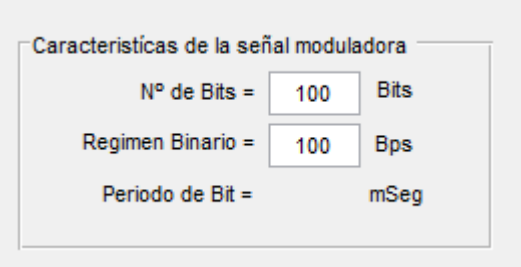


Figura 9.2.3 Selección de caracteristicas de señal moduladora

- En el segundo los parámetros de la portadora como por ejemplo la frecuencia de portadora, fase y frecuencia de muestreo. La casilla de mostrar Fs permite representar la señal en continua o en discreta en el plot generado.

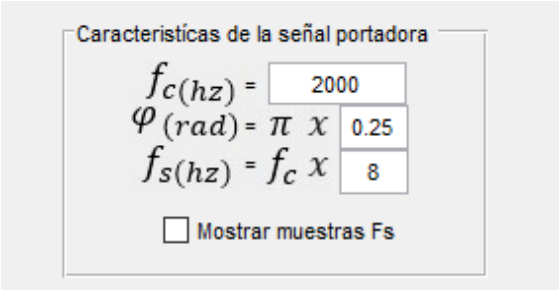


Figura 9.2. 1.4 Selección de caracteristicas de la portadora

- En el tercer apartado seleccionamos el Orden de modulación de nuestro modulador

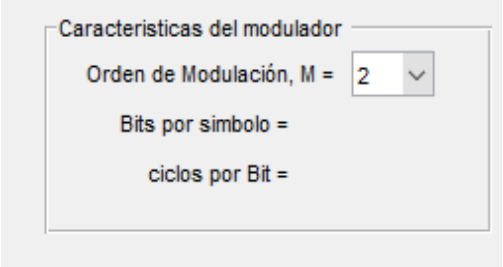


Figura 9.2. 1.5 Selección de caracteristicas de modulador

- Y finalmente en el ultimo apartado seleccionamos la SNR

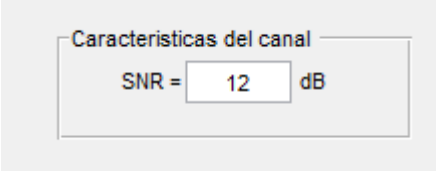


Figura 9.2. 1.6 Selección de SNR

En la parte inferior están las opciones de ejecución, una vez seleccionados todos los parámetros a nuestro gusto tenemos que pinchar en modular y se ejecutará el programa y aparecerá algo tal que así:

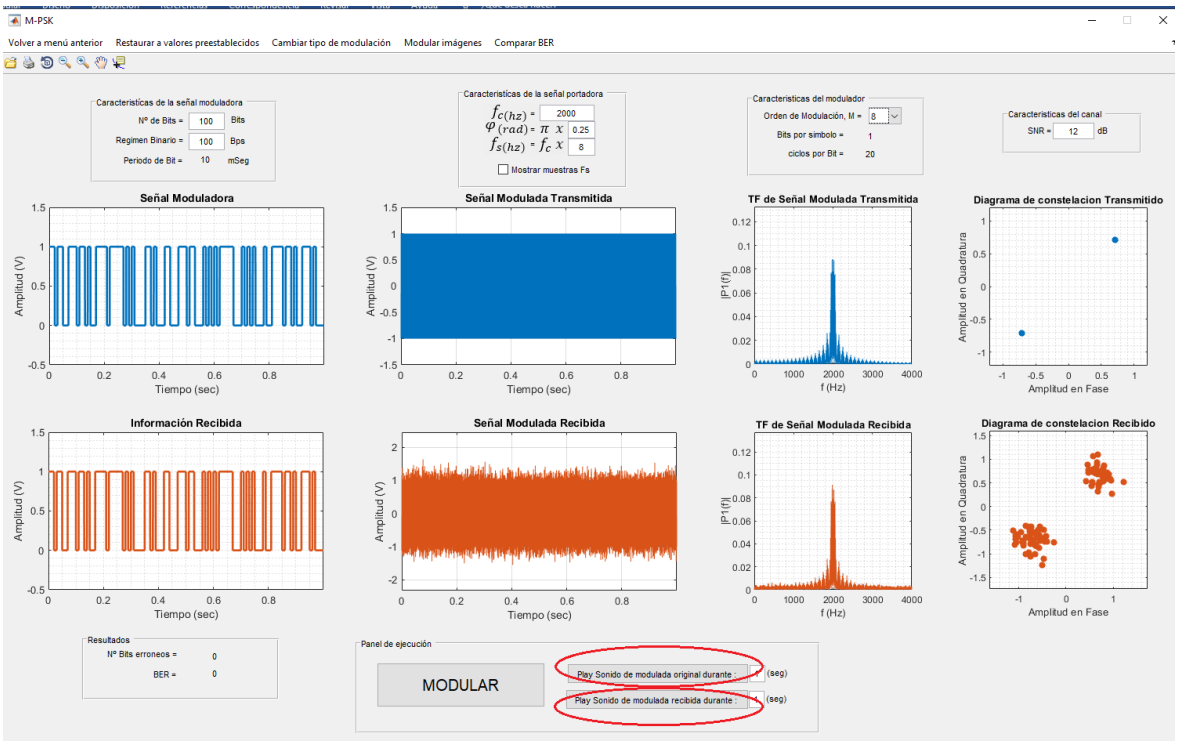


Figura 9.2. 1.7 Ejecución de sonidos de señales moduladoras

Ahora podemos visualizar los resultados en las gráficas y también podemos escuchar el sonido de las señares resultantes pinchando en los botones que están junto al botón ejecutar. El primer botón hace sonar la señal modulada transmitida mientras que el segundo reproduce la señal modulada recibida (que es la misma, pero con el ruido añadido).

- En el primer recuadro aparece la señal moduladora.
- En el segundo recuadro aparece la señal modulada.
- En el tercero la TF
- Y finalmente en el cuarto el diagrama de constelación.

En azul aparecen la señal enviada mientas que en naranja la recibida.

9.2.2 Menú de modulación de imágenes

Ahora volvemos al menú principal y pinchamos en la segunda opción para modular imágenes.

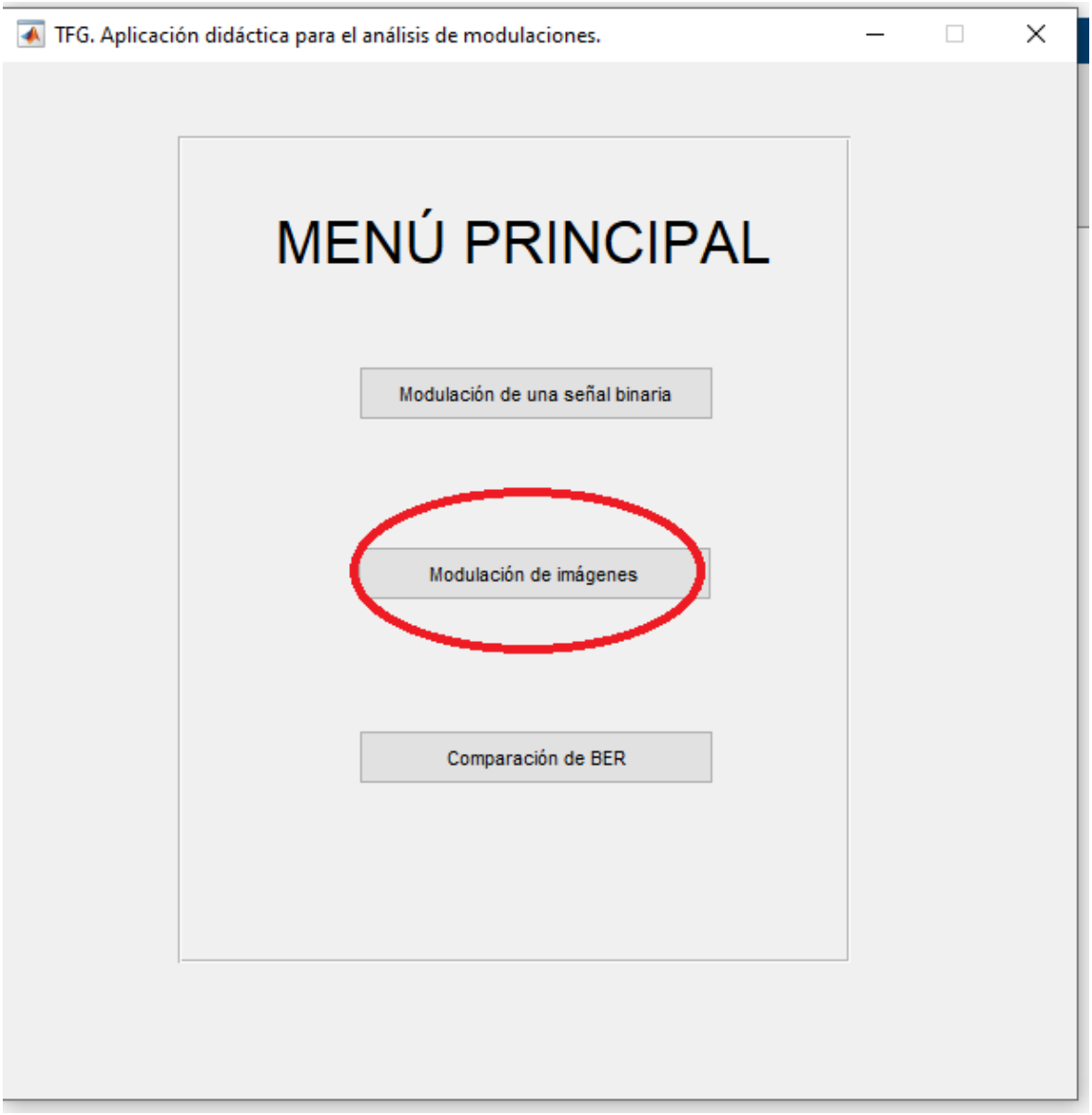


Figura 9.2.2.1 Selección de opcción de modular imágenes en Menu Principal

Aparecerá un segundo menú donde podemos modular imágenes, en este caso se ha trabajado con la imagen incluida en Matlab lenna.tiff

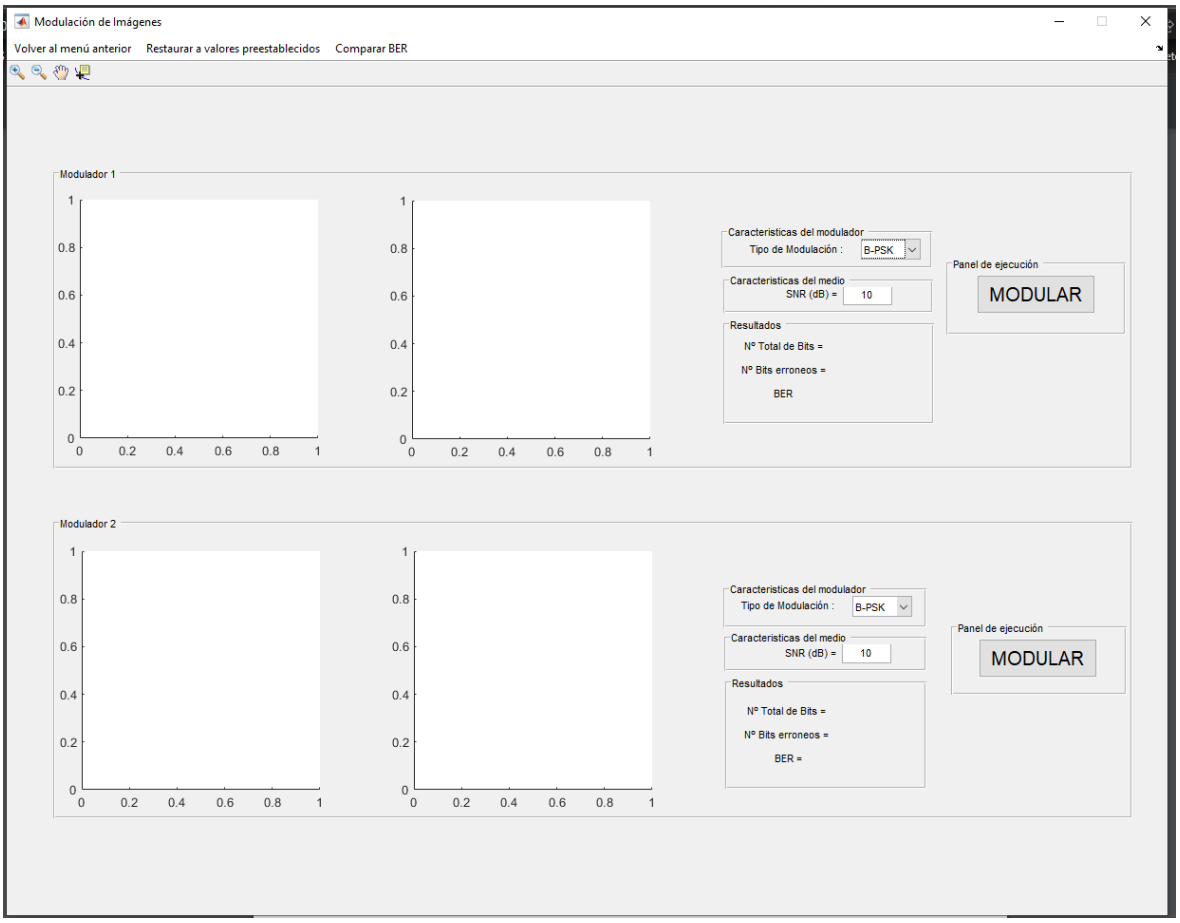


Figura 9.2.2.2 Menu de Modulacción de imágenes

- 1º seleccionamos en tipo de modulación que queremos emplear
- En 2º lugar la calidad señal ruido
- Finalmente pinchamos en modular para visualizar los resultados.

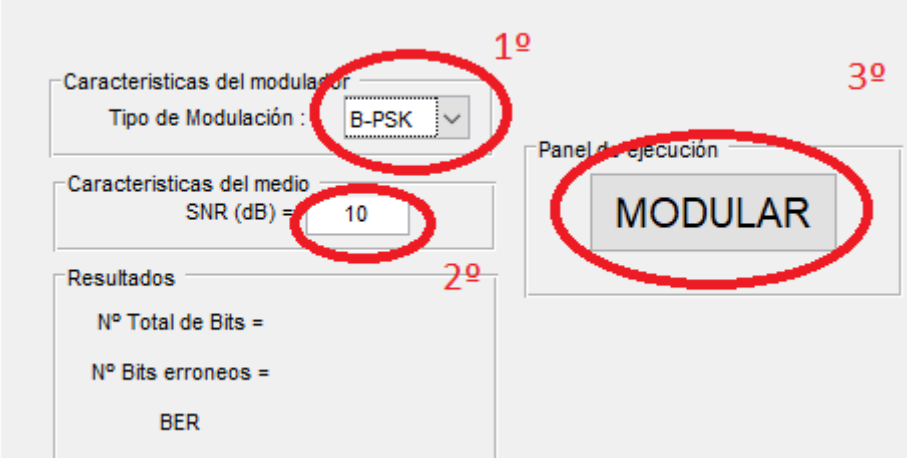


Figura 9.2.2.3 Selección de caracteristicas de señal moduladora

Este es el resultado:

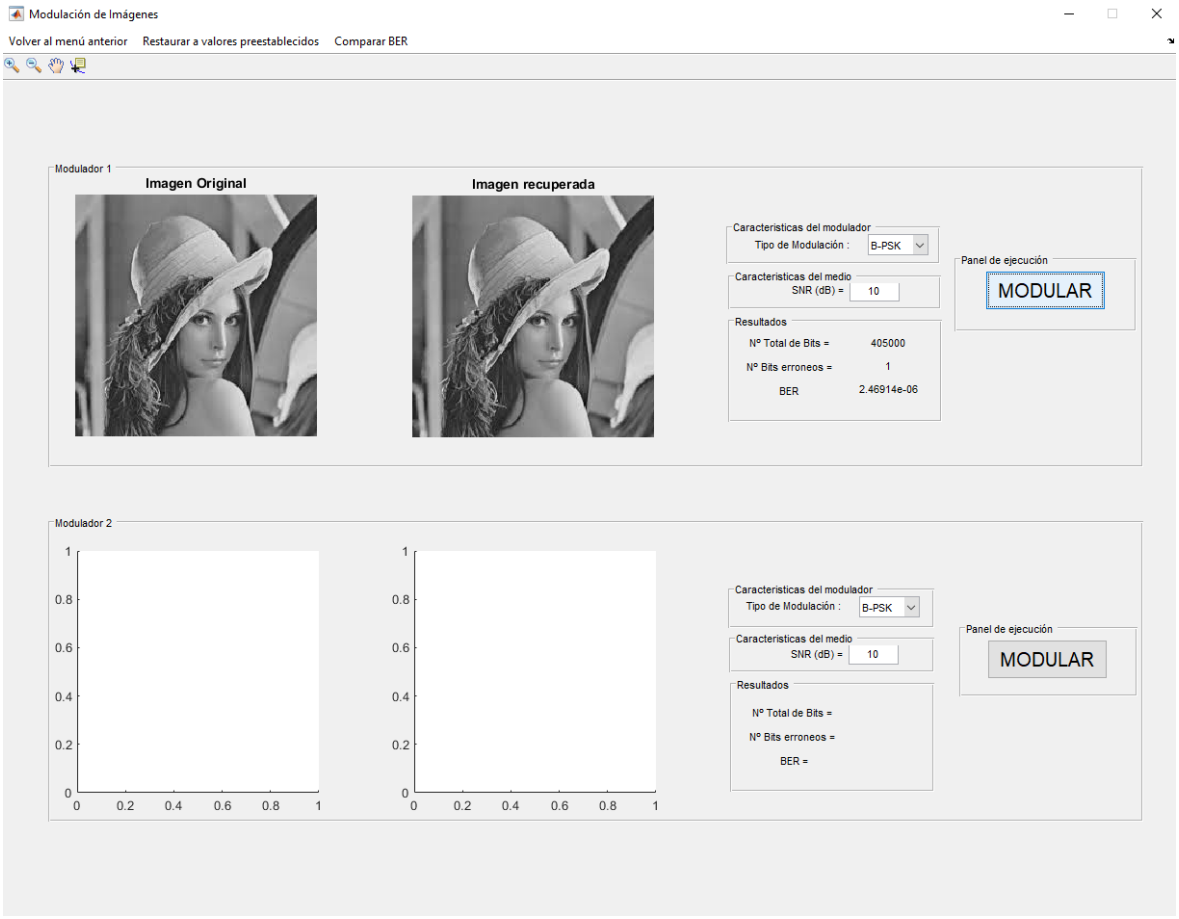


Figura 9.2.2.4 Resultados de imagen modulada

En la ventana de resultados podemos ver que con las características seleccionadas solo ha habido un bit erróneo y la BER.

Figura 9.2.2.5 Resultados de BER de imagen modulada

Ahora hacemos lo mismo en el menú inferior, seleccionando nuevamente un tipo de modulación y SNR para comprar resultados.

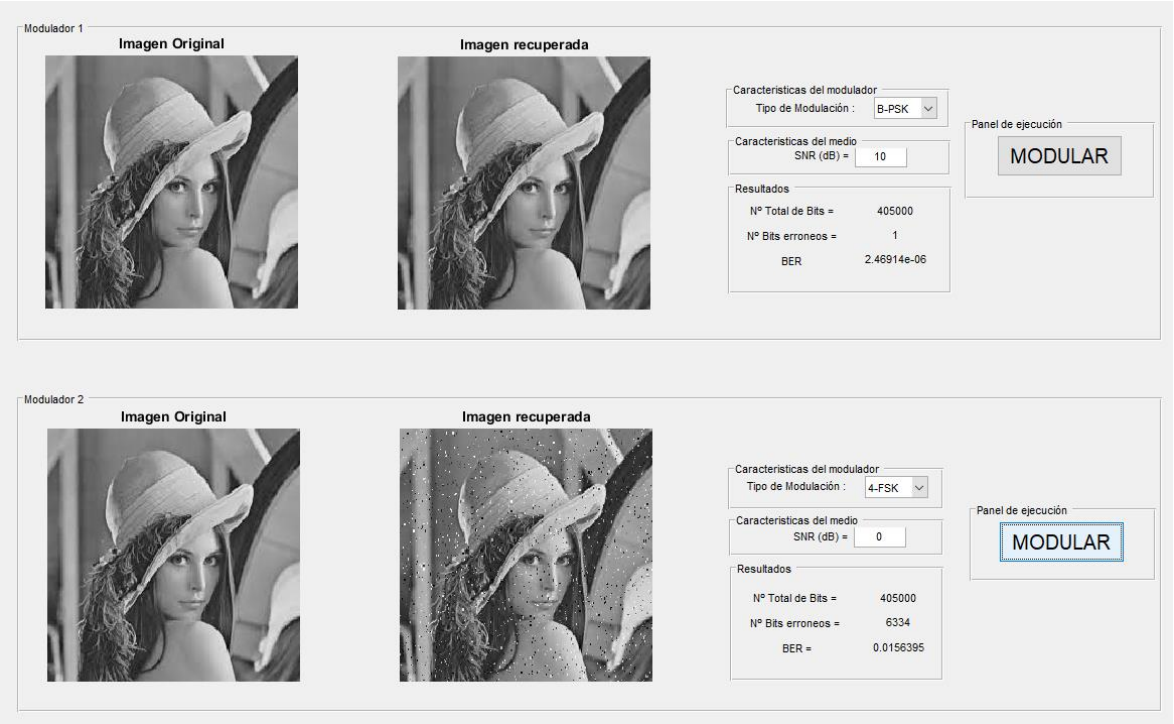


Figura 9.2.2.6 Comparacón entre moduladores de imágenes

9.2.3 Menú de comparación de BER

Para ver Graficas de comparación de la BER volvemos a menú principal y pinchamos la 3º opción.

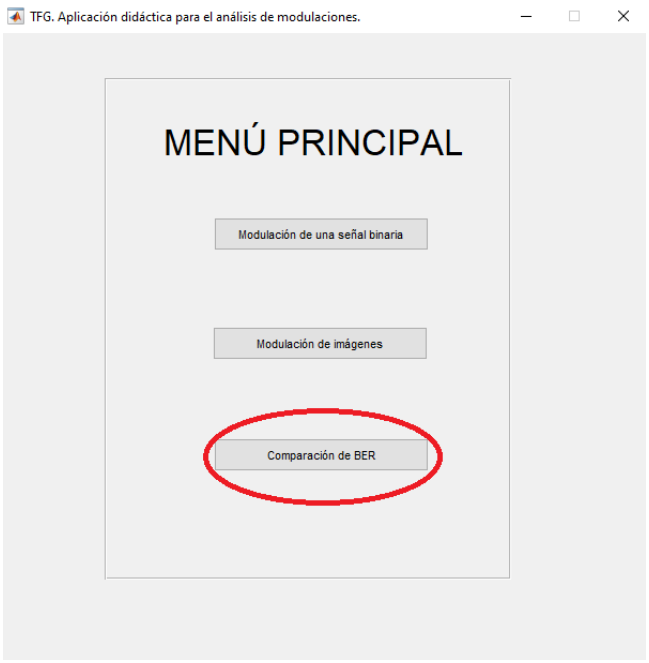


Figura 9.2.3.1 Menú princial, selección de comparación de BER.

Este es el menú para comparar la BER de las distintas modulaciones

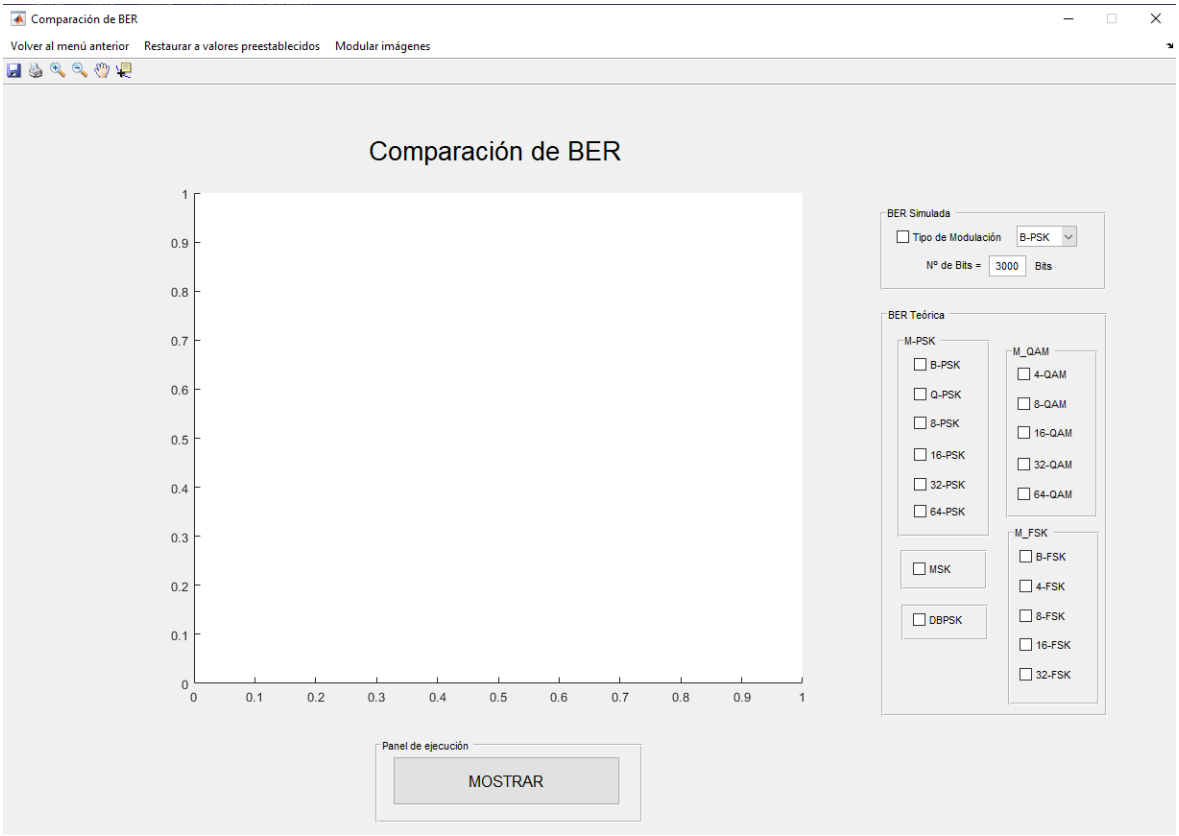


Figura 9.2.3.2 Menú de comparación de BER.

En la opción superior seleccionamos una modulación para realizar una simulación

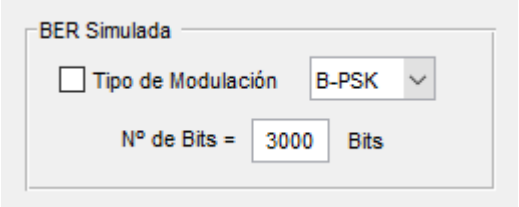


Figura 9.2.3.3 Selección de tipo de modulador para simulación.

Mientras que en la segunda opción seleccionamos una modulación para hace comparación con la BER teórica

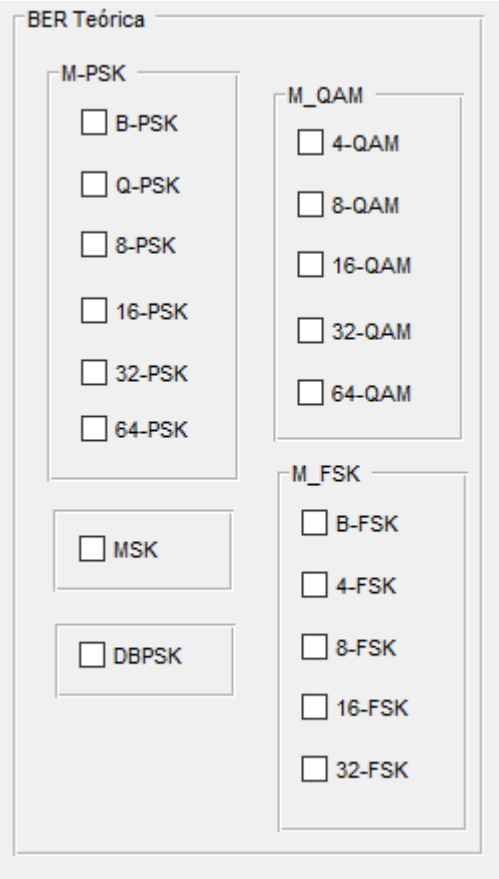


Figura 9.2.3.4 Selección de tipo de modulador para ver resultados teoricos.

Para ejecutar programa es necesario seleccionar las casillas que queremos visualizar y pinchar el botón de MOSTRAR

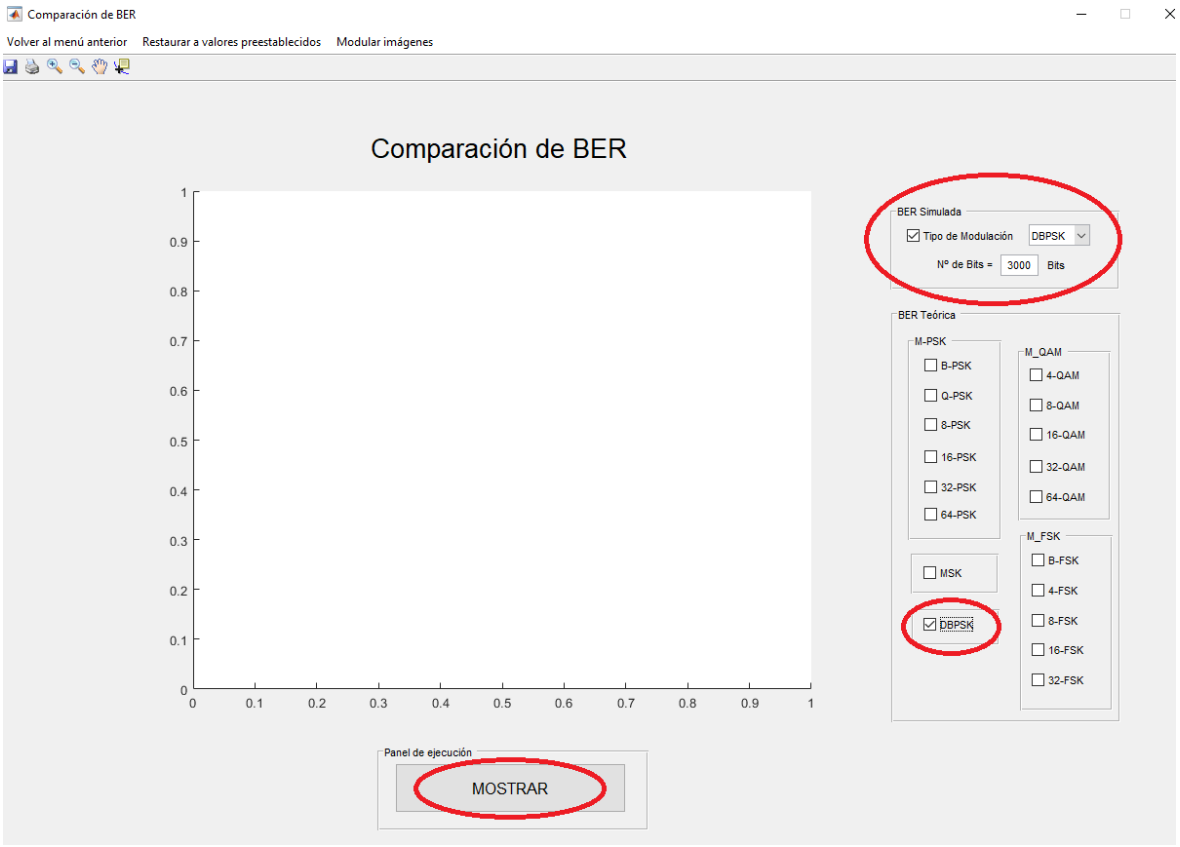


Figura 9.2.3.5 Mostrar resultados teoricos y simulados en gráfico.

Este sería el resultado

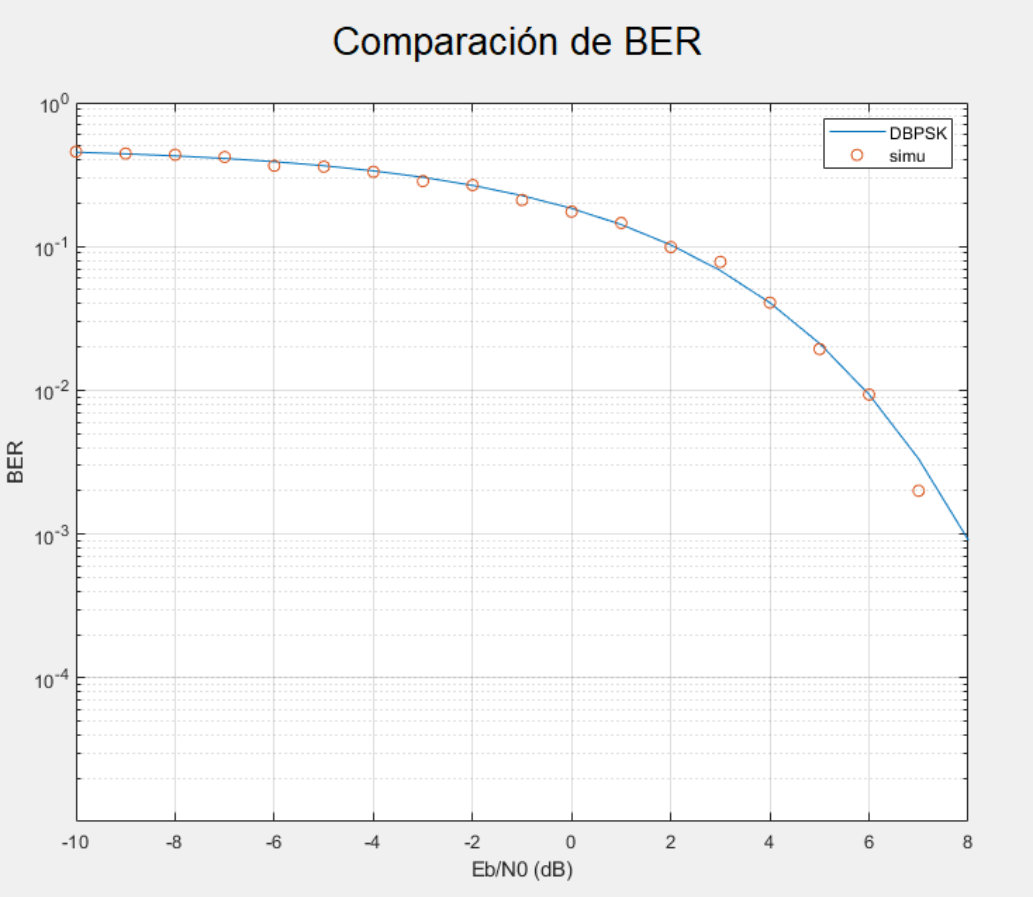


Figura 9.2.3.6 Gráfico con resultados de la BER.

Para salir del programa solo hay que cerrar la ventana.

10. REFERENCIAS BIBLIOGRÁFICAS

[1] Bernard Sklar. Digital communication system performance, 1999.

[2] John B Anderson, Tor Aulin, and Carl-Erik Sundberg. Digital phase modulation. Springer Science & Business Media, 2013.

[3] Manoj Barnela and Dr Suresh Kumar. Digital modulation schemes employed in wireless communication, 2014.

[4] Fuqin Xiong. Digital Modulation Techniques, 2006.

[5] Jianhua Lu. M-PSK and M-QAM BER computation using signal-space concepts, 1999.

[6] Francisco Jesús Cañadas Quesada. Apuntes de la asignatura Teoría de la Comunicación, 2º curso, Grado en Ingeniería de Tecnologías de Telecomunicación y Grado en Ingeniería Telemática, Escuela Politécnica Superior de Linares, Universidad de Jaén, 2022.

[7] Antonio Pérez Yuste. Sobre la etimología de Telecomunicación, 2006.

[8] Yoshihiko Akaiwa. Introduction to digital mobile communication. John Wiley & Sons, 2015.