ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

APLICACIONES DEL SOFTWARE *MATLAB/SIMULINK* EN MODELADO DE SISTEMAS BÁSICOS DE TELECOMUNICACIONES

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES

JAIRO WLADIMIR CONGO PASTRANA

vlad_di@hotmail.com

DIRECTOR: Wilson Enríquez López

wenriquez@igepn.edu.ec

Quito, Abril, 2015

DECLARACIÓN

Yo, Jairo Wladimir Congo Pastrana, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____

Jairo Wladimir Congo Pastrana

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Jairo Wladimir Congo Pastrana, bajo mi supervisión.

_____

Ing. Wilson Enríquez

DIRECTOR

ÍNDICE DE CONTENIDO

CAPITULO I 2
Antecedentes y Justificación. 2
1.1 Antecedentes. 2
1.1.2. Justificación del Proyecto de Titulación. 3
1.1.2.1. Justificación Técnica 3
1.1.2.2. Justificación Metodológica. 3
1.1.3. Objetivos. 4
1.1.3.1. Objetivo General. 4
1.1.3.2. Objetivos Específicos. 4
Capitulo II 5
Marco Teórico 5
Introducción. 5
2.1. Estructura de un sistema básico de Telecomunicaciones 6
2.1.1. Fuente de Información 6
2.1.2. Transductor 7
2.1.3. Transmisor 7
2.1.4. Canal 7
2.1.5. Receptor 8
2.1.6. Ancho de banda y potencia de la señal 11
2.2. Modulación 13
2.2.1. ¿Por qué se modula una señal? 13
2.2.2. Modulación de amplitud (AM) 13
2.2.3 Modulación de Frecuencia (FM). 15
2.2.4 Modulación de Fase (PM) 16
2.2.5 Modulación de Pulsos codificados (PCM) 17
2.2.6 Modulación en Cuadratura (QAM) 18
2.2.7. Modos de Transmisión 19
2.2.7.1 Simplex 20

2.2.7.2. Half Duplex o Semi Duplex 20
2.2.7.3. Full Duplex 21
2.3. Demodulación 21
2.4. Problemas en la recepción de señales 22
Capitulo III 23
3.1. Software de simulación *Matlab/Simulink* 23
3.1.1. Origen 24
3.1.2. Descripción del Software *Matlab* 24
3.1.3. Instalación del Software *Matlab* versión estudiantil 25
3.1.4. Entorno de trabajo *Matlab* 28
3.1.4.1. Escritorio de *Matlab* (*Matlab* Desktop) 29
3.1.4.2. Ventana de comandos (Command Window) 30
3.1.4.3. Ventana Histórica de comandos (Command History) 31
3.1.4.4. Espacio de trabajo (Workspace) 31
3.1.4.5. Plataforma de Lanzamiento (Launch Pad) 32
3.1.4.6. Directorio Actual (Current Directory) 32
3.1.4.7. La ventana de ayuda (Help) 32
3.1.4.8. El editor de ficheros y depurador de errores (Editor&Debugger) 33
3.1.5. Ventana de comandos (Command Window) 33
3.1.6. Funciones matemáticas elementales del software *Matlab* 34
3.1.7. *Matlab* Help (Ayuda) 35
3.1.8. Tipos de datos 36
3.1.9. Programación en *Matlab* 38
3.1.9.1. Los operadores relacionales 40
3.1.9.2. Operadores Aritméticos 41
3.1.9.3. Operadores Lógicos 41
3.1.9.4. Sentencias condicionales 41
3.1.9.5. Bucles 44
3.1.9.6. Líneas de comentarios 45
3.1.10. Ficheros *.m 45
3.1.11. Ficheros de comandos (Scripts) 47
3.1.12. Librerías 47

3.1.13. Ayuda (Help) para funciones de usuario 49

3.1.14. Ayuda (Help) de directorios 49

3.1.15. Gráficos bidimensionales 49

3.1.16. Funciones Trigonométricas elementales 50

Capitulo IV 70

4. Modelado y simulación en *Matlab/Simulink* aplicado a la asignatura de Telecomunicaciones I 70

4.1. Simulación # 1: Generación de señales típicas en Telecomunicaciones usando *Matlab/Simulink* 71

4.2. Simulación #2: Serie y Transformada Rápida de Fourier. 79

4.3. Simulación # 3: Modulación y Demodulación de Amplitud (AM) usando *Simulink* 88

4.3.1. Modulación (AM) 88

4.3.2. Demodulación AM 93

4.4. Simulación # 4: Modulación y Demodulación de Frecuencia (FM) usando *Simulink* 96

4.5. Simulación # 5: Transmisión de señales en sistemas básicos de telecomunicaciones. 102

CAPITULO V 110

CONCLUSIONES 110

RECOMENDACIONES 110

BIBLIOGRAFIA 112

REFERENCIAS WEB 112

ANEXOS 115

LISTA DE FIGURAS

Figura 2. 1: Diagrama en bloques de un sistema básico de telecomunicaciones 6
Figura 2. 2 Respuesta en frecuencia Filtro Pasa bajos 9
Figura 2. 3 Respuesta en frecuencia Filtro Pasa Altos 9
Figura 2. 4 Respuesta en frecuencia Filtro Pasa Banda 10
Figura 2. 5 Respuesta en frecuencia Filtro Elimina Banda 10
Figura 2. 6 Diseño de un filtro y su respuesta de frecuencia 11
Figura 2. 7 Representación del espectro en frecuencias de señales Periódicas y no Periódicas 12
Figura 2. 8 Diagrama en bloques modulador AM básico (AM Comercial) 14
Figura 2. 9: Gráfico Modulación de amplitud (AM) 14
Figura 2. 10 Diagrama en bloques de un modulador FM 15
Figura 2. 11: Gráfico Modulación de frecuencia (FM) 15
Figura 2. 12: Gráfico Modulación de fase (PM) 16
Figura 2. 13 Diagrama en bloques de un modulador PM 16
Figura 2. 14 Diagrama en bloques de un modulador PCM 17
Figura 2. 15: Gráfico Modulación de pulsos codificados (PCM) 17
Figura 2. 16: Gráfico Modulación digital en cuadratura (QAM) 18
Figura 2. 17: Gráfico de las posibles constelaciones de la modulación digital en cuadratura (**16-QAM**) 19
Figura 2. 18 Diagrama en bloques del modulador QAM 19
Figura 2. 19: Gráfico del Modelo de Tx SIMPLEX 20
Figura 2. 20: Gráfico del Modelo de Tx HALF DUPLEX O SEMIDUPLEX 20
Figura 2. 21: Gráfico del Modelo de Tx FULL DUPLEX 21
Figura 2. 22 Ejemplo de demodulación y traslación de espectros 22
Figura 3. 1: Ejecución del instalador de ***Matlab*** 25
Figura 3. 2: Aceptación de términos y condiciones de la licencia de Mathworks 26
Figura 3. 3: Colocando información acerca de nuestra cuenta de Matworks 26
Figura 3. 4: Especificando la ruta de instalación de ***Matlab*** 27
Figura 3. 5: Insertando el código de actvacion de nuestro software 27
Figura 3. 6: Seleccionando el tipo de instalación de ***MATLAB*** 28
Figura 3. 7: Instalando ***MATLAB*** 28
Figura 3. 8: Opciones del menú “Layout” 30
Figura 3. 9: ***Matlab*** Desktop (configuración por defecto) 30
Figura 3. 10: Command History acceso al registro de sentencias usando doble clic 31
Figura 3. 11 Plataforma de lanzamiento Launch Pad 32
Figura 3. 12: Ayuda en el Desktop ***Matlab*** 33
Figura 3. 13: Opciones del Help **Matlab** 36

Figura 3. 14: Diagrama de flujo de la estructura “if-end” 42
Figura 3.15: Diagrama de flujo de la estructura “if-else-end” 42
Figura 3.16: Diagrama de flujo de la estructura “if-else-if-else-end” 43
Figura 3. 17: Estructura Switch-case 43
Figura 3. 18: Estructura for-end 44
Figura 3. 19: Estructura while-end 45
Figura 3. 20: Comando *“title”* 51
Figura 3. 21: Comandos “XLABEL-YLABEL” 52
Figura 3. 22: Comando “Grid” 52
Figura 3. 23: Superposición de gráficas 53
Figura 3. 24: Especificadores de línea en la función plot 54
Figura 3. 25: Marcadores de línea en la función *plot* 56
Figura 3. 26: Ajuste de los límites de la ventana de gráficos 56
Figura 3. 27: Gráfico función Stem 57
Figura 3. 28: Librería de bloques de función **Simulink** 58
Figura 3. 29: Creando un nuevo modelo **Simulink** 59
Figura 3. 30: Work-Space **Simulink** 59
Figura 3. 31: Selección de una librería 60
Figura 3. 32: Conectando bloques en el Work-space **Simulink** 61
Figura 3. 33: Bloque Suma 62
Figura 3. 34: Parámetros del bloque Suma 63
Figura 3. 35: Bloque producto 63
Figura 3. 36: Bloque Integrator 63
Figura 3. 37: Bloque Transport Delay 64
Figura 3. 38: Bloque Fcn 64
Figura 3. 39: Bloque **Matlab** Function 64
Figura 3. 40: Bloque Transport Delay 65
Figura 3. 41: Bloque Scope 65
Figura 3. 42: Bloque sine wave 66
Figura 3. 43: Ventana de parámetros del bloque Analog Filter Design 68
Figura 4. 1: Señal continua……..……...72
Figura 4. 2: Señal seno 72
Figura 4. 3: Señal coseno 73
Figura 4. 4: Señal tren de pulsos 73
Figura 4. 5: Señal diente de sierra 73
Figura 4. 6: Señal aleatoria..... 74
Figura 4. 7: Señal continua..... 76
Figura 4. 8: Señal seno 76
Figura 4. 9: Señal tren de pulsos 77
Figura 4. 10: Diente de sierra 77
Figura 4. 11: Señal aleatoria..... 77
Figura 4. 12: Generación de la función dada 81
Figura 4. 13: Resultado de ejecución en el *Command Window* ***Matlab*** 82

Figura 4. 14: Gráfica de los coeficientes de Fourier 82
Figura 4. 15: Gráfica impulso unitario 85
Figura 4. 16: Gráfica impulso unitario con etiquetas manipuladas 86
Figura 4. 17: Resultado obtenido en el workspace 86
Figura 4. 18: Gráfica de la solución encontrada 87
Figura 4. 19: Modulador AM Doble banda lateral con portadora suprimida 90
Figura 4. 20: Señal modulante 90
Figura 4. 21: Señal portadora 90
Figura 4. 22: Señal modulada 91
Figura 4. 23: Espectro señal modulante 91
Figura 4. 24: Espectro señal portadora 91
Figura 4. 25: Espectro señal modulada 91
Figura 4. 26: Espectro señal modulada 92
Figura 4. 27: Modulador/Demodulador AM (DBLPS) 94
Figura 4. 28: Espectro señal modulante 94
Figura 4. 29: Detector de envolvente 94
Figura 4. 30: Salida del filtro pasabajo 95
Figura 4. 31: Modulador FM indirecto 97
Figura 4. 32: Señal modulante 98
Figura 4. 33: Señal modulada en FM 98
Figura 4. 34: Modulador/Demodulador FM indirecto 100
Figura 4. 35: Discriminador de frecuencia 100
Figura 4. 36: Detector de envolvente 100
Figura 4. 37: Filtro Pasabajos 100
Figura 4. 38: Transmisión de una señal y modulación en BPSK 104
Figura 4. 39: Señal moduladora 104
Figura 4. 40: Señal muestreada 104
Figura 4. 41: Señal cuantificada 105
Figura 4. 42: Señal codificada 105
Figura 4. 43: Constelación de bits en BPSK 105
Figura 4. 44: Recepción de una señal y demodulación en BPSK o 2QAM 107
Figura 4. 45: Señal Recibida 108
Figura 4. 46: Señal Decodificada 108
Figura 4. 47: Señal Recuperada 108

LISTA DE TABLAS

Tabla 3. 1: Operadores relacionales válidos en ***Matlab*** 40
Tabla 3. 2: Operadores Aritméticos..... 41
Tabla 3. 3: Operadores Lógicos..... 41
Tabla 3. 4: Especificadores de línea en la función plot..... 54
Tabla 3. 5: Marcadores de línea en la función plot 55
Tabla 3. 6: Configuraciónes posibles entre el tipo y diseño de un filtro en ***Matlab/Simulink*** 69

RESUMEN

El proyecto de titulación presentado a continuación, se enfoca en proponer una metodología de prácticas del laboratorio de Telecomunicaciones I, materia impartida dentro de la carrera de Tecnología en Electrónica y Telecomunicaciones de la Escuela de Formación de Tecnólogos. Esta sugerencia hace énfasis en el empleo del Software ***Matlab/Simulink*** y servirá como introducción a la visualización de características y procesos a través de los cuales, las señales de información son sometidas previo a ser transmitidas o recibidas.

Con el fin de presentar una alternativa actualizada al sistema de prácticas de laboratorio, se ha dividido este proyecto de titulación en 5 capítulos. El primer capítulo describe con especial enfoque, los antecedentes, justificación y objetivos de promover ***Matlab/Simulink*** como software para el laboratorio de Telecomunicaciones I. El segundo capítulo contiene los aspectos conceptuales necesarios para interpretar los sistemas básicos de telecomunicaciones. El tercer capítulo, contiene los aspectos teóricos del Software sugerido para emplearse en laboratorio, mostrando sus características, líneas de ayuda y la descripción de la interfaz gráfica para el usuario. El cuarto capítulo, contiene un formato de prácticas que podrían usarse para el laboratorio de la materia mencionada. El quinto capítulo contiene las conclusiones y recomendaciones que han surgido conforme se ha ido desarrollando el proyecto.

Finalmente se presentan tres Anexos, el primero; contiene ejemplos del uso de funciones de ***Matlab***, mientras que el segundo contiene la configuración de los bloques de ***Simulink*** que se utilizaron en las simulaciones del cuarto capítulo y el tercer Anexo contiene las respuestas del laboratorio propuesto en el cuarto capítulo.

CAPITULO I

Antecedentes y Justificación.

1.1 Antecedentes.

Matlab es un lenguaje de programación de alto nivel, que fue creado para brindar soluciones computacionales técnicas y trabaja basándose en matrices, de allí su nombre que proviene del acrónimo **MAT**rix **LAB**oratory. En un principio ***Matlab*** solo era utilizado por personas con conocimientos en FORTRAN y C pero luego de popularizarse se vio que este software ofrecía facilidades en su sintaxis para todo aquel que tenga conocimiento básico en algún lenguaje de programación.

Por su facilidad ***Matlab*** se ha convertido en el software de programación preferido para adentrar a personas en el mundo de la programación, es por eso que en institutos y universidades ***Matlab*** se ha impuesto sobre otros lenguajes como soporte para cursos básicos y avanzados en los que busca formar profesionales de ciencias informáticas, matemáticas o de ingeniería.

Su amplia funcionalidad permite resolver problemas relacionados con el procesamiento de señales, además del diseño de aplicaciones de control para lo que cuenta con un grupo de librerías especiales denominadas "***Toolbox***". Un ***Toolbox*** es un grupo de instrucciones orientadas a ciertos tipos de cálculos es decir; cada ***Toolbox*** agrupa comandos que usamos para el desarrollo de una solución a un problema específico. Dependiendo la jerarquía del problema que debamos resolver usaremos mayor número de ***Toolbox*** en el código fuente de un programa. Debido a la gran extensión de aplicaciones y soluciones que este software presenta, es imposible detallar cada una de dichas aplicaciones dentro de este proyecto de titulación, por tanto enfocaremos este proyecto, a la aplicación de ***Matlab/Simulink*** en el modelado de partes de sistemas básicos de telecomunicaciones.

El interesado en ampliar su perspectiva acerca de este software puede consultar la línea de aspectos avanzados, accediendo a la **Ayuda** que el software proporciona.

El área de trabajo del software ***Matlab*** ha visto bastantes mejoras desde la versión 6.0, ya que, se logra visualizar desde dicha versión un entorno

más gráfico e intuitivo similar a algunas aplicaciones que corren dentro del sistema operativo Windows.

Elogiando las características y potencia de ***Matlab*** como soporte en aplicaciones técnicas y sabiendo que pone a disposición el uso de herramientas para dar solución a los problemas tratados en tecnología, se da por entendido que este software será una herramienta de gran ayuda porque facilitará impartir a los alumnos las teorías de los sistemas básicos de telecomunicaciones, enfocando de manera práctica los temas dictados dentro de la materia Telecomunicaciones I.

1.1.2. Justificación del Proyecto de Titulación.

En la actualidad la tendencia en la enseñanza dentro de las universidades está orientada a metodologías aplicativas con el objetivo de transmitir a los estudiantes la teoría de manera práctica. Las metodologías aplicativas nos llevan a mejorar continuamente, además de que en los estudiantes genera un interés en la investigación que deriva en el desarrollo de soluciones a problemas. Con esto los ciertos alumnos pueden hallar soporte en aplicaciones de software que simulen dispositivos disponibles solamente en laboratorios, promoviendo a ***Matlab/Simulink*** como software de simulación se logrará una mejor comprensión del funcionamiento de los diversos sistemas básicos presentes en telecomunicaciones, con el fin de alcanzar la excelencia académica, apegados al uso de herramientas actualizadas que se pueden incluir dentro del programa de estudios relacionados con el laboratorio de la materia mencionada.

1.1.2.1. Justificación Técnica

Los campos de investigación de científicos e ingenieros se valen de las computadoras para simular el comportamiento de sistemas y resolver los distintos problemas, desde generar una función sencilla hasta la resolución de un sistema de ecuaciones. El entorno de computación técnica ***Matlab/Simulink*** constituye una buena opción para transmitir la teoría de telecomunicaciones a los estudiantes de tecnologías e incluso ingenierías, ya que por su entorno de trabajo fácil de aprender y usar permite la solución de problemas técnicos a través de simulaciones usando inclusive la notación matemática habitual con la que los estudiantes están familiarizados.

1.1.2.2. Justificación Metodológica.

La metodología de este proyecto de titulación se apoyará en métodos deductivos, puesto que esta metodología permite la integración de la parte teórica de la materia con la parte aplicativa, lo que incentivará un juicio

técnico de las múltiples aplicaciones de los sistemas básicos de telecomunicaciones.
Este proyecto abrirá nuevas puertas a múltiples aplicaciones que encontramos dentro de la teoría de telecomunicaciones, los cuales pueden asimilarse fácilmente simulando partes de sistemas reales y obteniendo detalladamente los resultados observables.

1.1.3. Objetivos.

1.1.3.1. Objetivo General.

Presentar una alternativa que permita manipular, diseñar e interpretar libremente los componentes de un sistema básico de telecomunicaciones empleando las herramientas que ***Matlab/Simulink*** contiene para ayuda de laboratorios de telecomunicaciones.

1.1.3.2. Objetivos Específicos.

- Analizar los tipos de señales y funciones que el software ***Matlab/Simulink*** otorga para sistemas de Telecomunicaciones.
- Proponer un manual de prácticas usando el software ***Matlab/Simulink*** para reproducir de manera práctica la teoría revisada en materia de Telecomunicaciones I.

CAPITULO II

Marco Teórico

Introducción.

Un sistema básico de telecomunicaciones es diseñado para transferir información por medio de ondas electromagnéticas, desde un punto en el espacio denominado “fuente de información” hasta otro punto denominado “destino de la información”, con el mínimo de pérdidas posibles. Queda en evidencia que la reproducción perfecta de un mensaje en el otro extremo de la comunicación no es posible, pero desde un punto de vista práctico es suficiente con una aproximación que dependerá de la aplicación para la que fue diseñado el sistema. Por ejemplo, en una conversación telefónica la aproximación requerida será menos preocupante que en aplicaciones de radiodifusión o televisión. En todo caso, la traslación de información siempre experimentará degradación a grandes distancias.

El objetivo de los sistemas de telecomunicaciones es comunicar a las personas entre dos extremos o inclusive hacer una comunicación entre máquinas. Se necesita la presencia de componentes electrónicos que ayuden a codificar los mensajes de forma que la información se represente en niveles de voltaje y puedan ser interpretados por los equipos terminales que simplemente son diseñados para hacer una prolongación de nuestros sentidos, por ejemplo el teléfono celular hace posible la comunicación de dos personas moviéndose en cualquier lugar del espacio como si estuviesen frente a frente. Los métodos que usan los sistemas básicos de telecomunicaciones para el traslado de información se dividen en dos:

1. Análogo.- Es un sistema en el que la energía electromagnética se transmite y recibe como una señal variando en el tiempo continuamente como lo es una onda sinusoidal.
2. Digital.- Es un sistema en el que la energía electromagnética se envía y recibe en niveles de voltaje o corriente discretos tales como 1L= (+5v) o 0L= (0v).

En la fuente que emite el mensaje, la información puede ser analógica “la voz humana”, o puede ser discreta “código morse”, sin embargo para que la información sea propagada de emisor a receptor, necesita ser transformada

a energía electromagnética para hacer un uso adecuado de un canal de comunicación guiado “cables” o no guiado “aire”.[1]

2.1. Estructura de un sistema básico de Telecomunicaciones

Los sistemas básicos de telecomunicaciones se dividen en 7 puntos:[2]

- Fuente de información
- Transductor de entrada
- Transmisor
- Canal
- Receptor
- Transductor de salida
- Destino

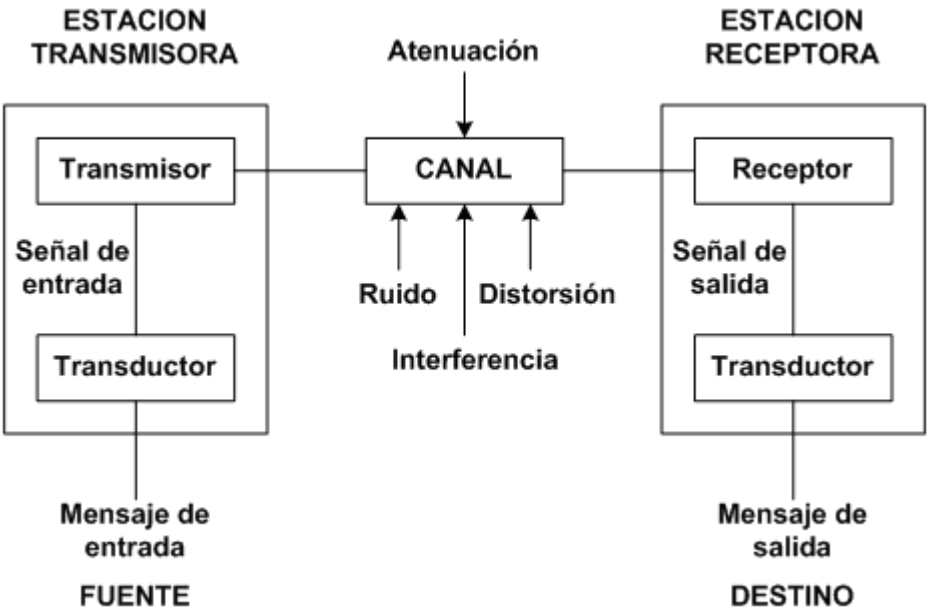


Figura 2. 1: Diagrama en bloques de un sistema básico de telecomunicaciones

2.1.1. Fuente de Información

Se puede entender como una fuente de información, a todo elemento capaz de generar una secuencia de mensajes con suficiente inteligibilidad. Existen varios tipos de fuentes de información como lo son las máquinas o incluso las personas. Cada persona o máquina desarrollará un algoritmo que genera una secuencia de mensajes de acuerdo a los protocolos de comunicación que conozcan. La secuencia de mensajes generada, está dirigida a otros elementos que puedan manejar y entender sus mismos protocolos de

[1] José E. Briceño M. *Principios de las comunicaciones Tercera Edición (2005) Pags. 261-263*

[2] http://www.taringa.net/posts/ciencia-educacion/14052069/Introduccion-a-las-telecomunicaciones.html

comunicación. Pasando este concepto a un entorno práctico, se ha encontrado la manera de relacionar la secuencia de mensajes que emite una fuente de información a una secuencia de impulsos eléctricos, cada impulso tendrá una magnitud diferente relacionando a la información que represente. La secuencia de mensajes emitidos por la fuente de información puede tener característica análogas variables en el tiempo, o características digitales. Sin importar las características y protocolos que la fuente de información use para emitir una secuencia de mensajes, el objetivo siempre será tratar de obtener en el receptor una réplica casi exacta del mensaje original.[1]

2.1.2. Transductor

Se encarga de asociar una secuencia de mensajes a una magnitud eléctrica, esto se debe a que los mensajes que producen las fuentes de información no contienen naturaleza eléctrica, para transformar el mensaje a una "señal eléctrica" se usan transductores. Usando transductores, los mensajes son convertidos en pulsos eléctricos variables en el tiempo como lo son: "corriente y voltaje". En términos generales un sistema básico de telecomunicaciones usa estos dos parámetros para traducir el mensaje original y prepararlo para establecer una comunicación entre emisor y receptor. Habiendo mencionado el transductor básicamente como un traductor, debe saberse que el sistema básico de telecomunicaciones consta de dos de ellos, el primero se lo conoce como "Transductor de entrada o codificador ", el cual es situado en el extremo del transmisor; al segundo se lo conoce como "Transductor de salida o decodificador" situado en el extremo receptor del sistema de comunicación.[1]

2.1.3. Transmisor

En un sistema básico de telecomunicaciones por lo general, el transductor de entrada se encuentra acoplado directamente al medio de transmisión, podemos tomar como ejemplo un teléfono de marcado fijo, por otro lado, ya que el objetivo es comunicar dos extremos distantes lo que significa llevar la señal hasta el receptor se requiere utilizar los conocidos medios de transmisión, cada uno de estos poseen características especiales las cuales generan una respuesta específica de acuerdo a la señal que los atraviese. El transmisor se encarga precisamente de producir una señal acorde a las características del medio de transmisión, modificando algún parámetro de la misma *(proceso conocido como modulación)*, de manera que se adecue la señal de manera que sea compatible con el medio de transmisión y esta se pueda propagar sin pérdidas hacia su destino.[1]

2.1.4. Canal

El término **"Canal"** en telecomunicaciones hace referencia al medio de transmisión por el cual se propaga la información hacia el destino, en el cual

la información viaja en forma de pulsos eléctricos o como ondas electromagnéticas.

El canal de comunicación proporciona las características eléctricas y mecánicas para enlazar al transmisor y al receptor en un entorno de intercambio de información. El traslado de información entre ellos dependerá del modo de transmisión que se esté usando, con lo que se puede dividir a los canales de telecomunicaciones en dos grupos: los fundamentados en *propagación guiada* (canal telefónico, cables coaxiales y fibras ópticas) y los que se basan en la *propagación libre* (canales de transmisión inalámbrica, canales de radio móvil y canales satelitales).[1]

2.1.5. Receptor

El receptor es quien trabaja en conjunto con el transmisor encargándose de restaurar la señal que fue enviada desde el lado transmisor para obtener una réplica casi exacta de la señal de origen, una vez recuperada la señal, este la envía hacia el transductor de salida con la finalidad de devolverla a su forma original y enviarla hacia el destino. En el receptor es fundamental el empleo de filtros para recuperar la señal, los filtros se aplican como selectores de frecuencia esto ayuda a discriminar (filtrar) las frecuencias de interés para tener una correcta respuesta del sistema.[1]

2.1.5.1 Filtros

En ocasiones la señal de información se encuentra mezclada con otro tipo de señales las cuales harían imposible obtener en el receptor la señal enviada, para combatir este percance se emplea el uso de filtros, los cuales cumplen la función de separar la señal de interés de las señales invasoras. Existen varios tipos de filtros cuyas características son empleadas para limitar el paso de cierta banda de frecuencia, además la respuesta del filtro frente a las frecuencias que lo atraviesan están ligadas al diseño elegido para el mismo.

2.1.5.2 Tipos de Filtros

El tipo de filtro indica las bandas de frecuencia que serán permitidas en la transición de la señal por el mismo, para esto se tienen las siguientes respuestas:

- **Filtro Pasa bajos.-** Este tipo de filtro atenúa o elimina las frecuencias altas y permite el paso de las frecuencias bajas respecto a una frecuencia fundamental conocida como

frecuencia de corte. La siguiente figura nos muestra la respuesta en frecuencia para este tipo de filtro.[3]

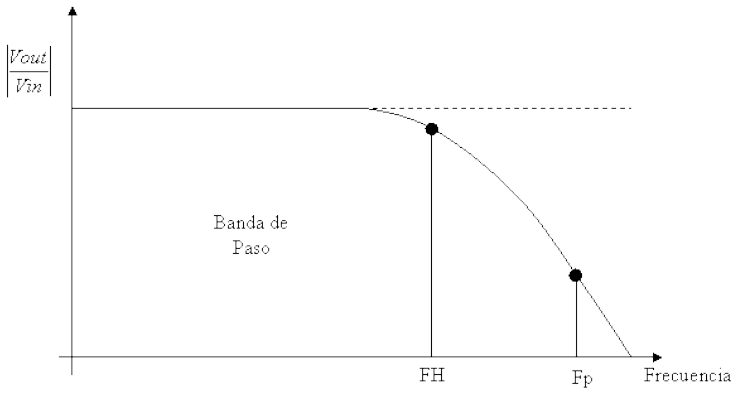


Figura 2. 2 Respuesta en frecuencia Filtro Pasa bajos

- **Filtro Pasa altos.-** Este tipo de filtro, por el contrario del pasa bajos, elimina las frecuencias bajas y permite el paso de las frecuencias altas con respecto a la frecuencia de corte.
  La siguiente figura nos muestra la respuesta en frecuencia para este tipo de filtro.[3]

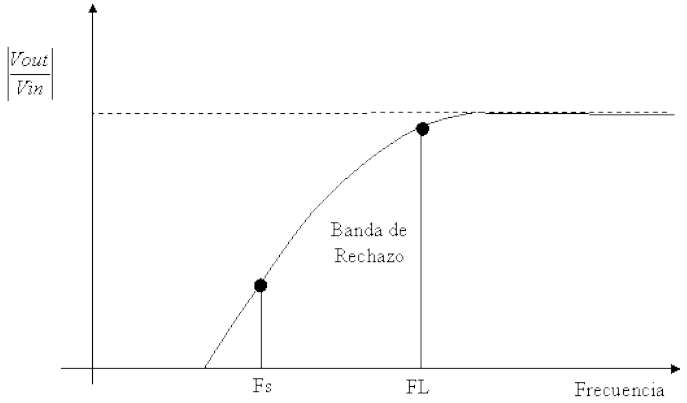


Figura 2. 3 Respuesta en frecuencia Filtro Pasa Altos

- **Filtro Pasa banda.-** Este tipo de filtro utiliza dos frecuencias de corte una superior y una inferior, de manera que atenúa un rango determinado de frecuencias y deja pasar el resto. Atenúa las frecuencias comprendidas antes de la frecuencia de corte inferior y después de la frecuencia de corte superior. La siguiente figura nos muestra la respuesta en frecuencia para este tipo de filtro. [3]

[3] https://lc.fie.umich.mx/~jfelix/InstruII/PB/PB.htm

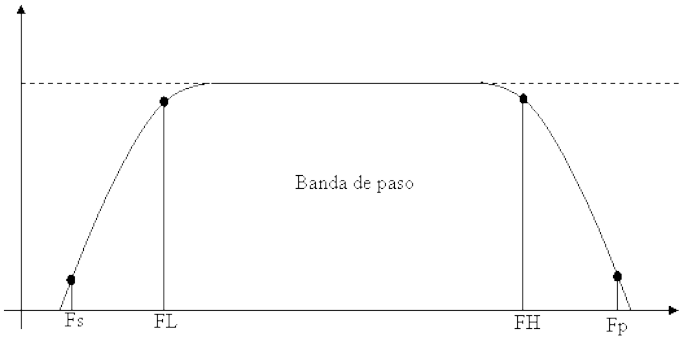


Figura 2. 4 Respuesta en frecuencia Filtro Pasa Banda

- **Filtro Elimina banda.**- Este tipo de filtro utiliza dos frecuencias de corte, una superior y una inferior, de manera que impide el paso de las frecuencias comprendidas entre las frecuencia de corte superior e inferior. La siguiente figura nos muestra la respuesta en frecuencia para este tipo de filtro. [3]

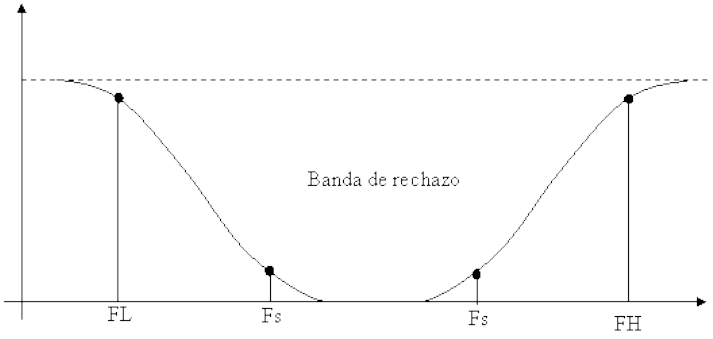


Figura 2. 5 Respuesta en frecuencia Filtro Elimina Banda

2.1.5.3 Diseño de filtro

El diseño de filtro se basa en el tipo de respuesta característica que estos tienen frente a las frecuencias que los atraviesan, en los que encontramos los siguientes:

- **Filtro de butterworth.-** este tipo de diseño es creado para producir la respuesta más plana posible hasta la frecuencia de corte, este filtro genera pequeñas ondulaciones en la banda de paso, es decir genera una respuesta casi lineal en la salida hasta llegar a la frecuencia de corte, a partir de ahí se genera una caída aguda atenuando frecuencias superiores a esta.
- **Filtros de Chebyshev.-** este tipo de filtros generan una respuesta con rizado constante en la banda de paso y una caída de frecuencias monótona.
- **Filtros elípticos.-** son un tipo de filtro mucho más eficiente, porque estrechan la zona de transición de frecuencias entre

bandas. Con esto consigue reducir el rizado que se produce en la transición entre bandas.
La siguiente figura ilustra las respuestas en frecuencia que tienen estos diseños de filtro.[4]

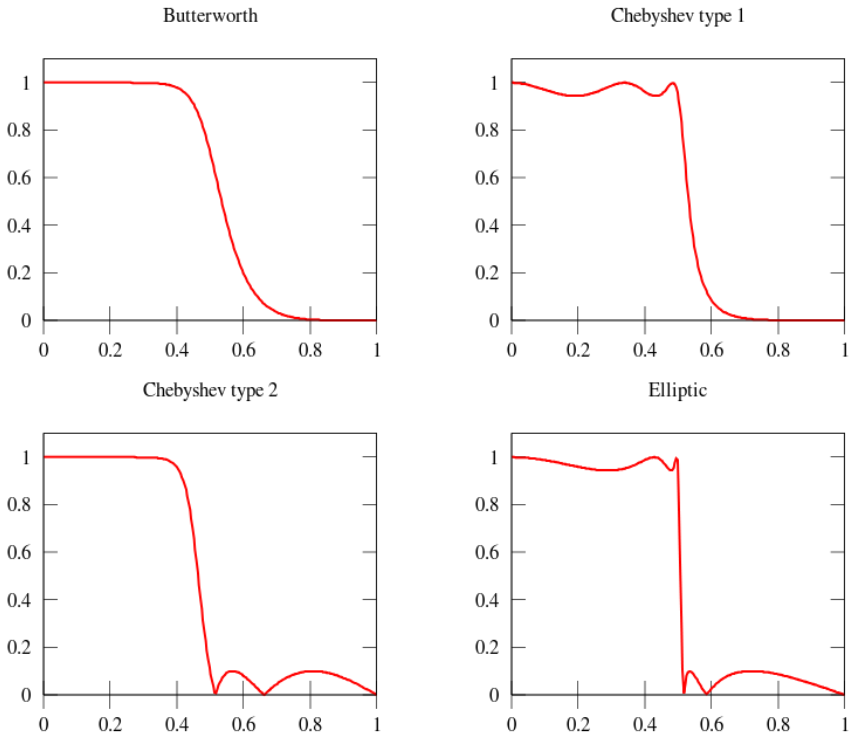


Figura 2. 6 Diseño de un filtro y su respuesta de frecuencia

2.1.6. Ancho de banda y potencia de la señal

Las aplicaciones diseñadas dentro de un sistema básico de telecomunicaciones deben tomar en cuenta dos parámetros fundamentales como lo son el Ancho de Banda y la Potencia de emisión de la señal. Ya que estos factores determinan el presupuesto de un enlace, se debe tratar de optimizar estos recursos sin afectar el rendimiento esperado del sistema. La consigna de un sistema básico de telecomunicaciones es transmitir mayor información en el menor tiempo posible y con el mínimo empleo de potencia.

- El ancho de banda representa la porción de frecuencias permitidas dentro de un sistema o aplicación de un sistema de telecomunicaciones, para lo cual debe ser lo suficientemente grande (ancho) para dejar pasar las frecuencias que contengan la información de interés.
- Potencia de la señal representa el parámetro voltaje o corriente que una fuente entrega para alimentar una carga, en muchas aplicaciones

[4] http://es.wikipedia.org/wiki/Filtro_lineal#/media/File:Electronic_linear_filters.svg

la potencia dependerá de la distancia que exista para enlazar al transmisor y al receptor, puesto que a más distancia se necesitará más potencia para alimentar la carga.[1]

Como se ha visto hasta aquí, es necesario conocer la respuesta en frecuencia de las señales presentes en los sistemas básicos de telecomunicaciones. Generalmente las formas de onda que se aprecian en las señales, son de característica de tiempo continuo, lo que quiere decir que se puede apreciar una forma invariante a lo largo del eje temporal. Un sistema invariante en el tiempo es aquel cuyas características y comportamiento permanecen fijos en el tiempo, por ejemplo: La salida que muestra la pantalla de un osciloscopio al ser atravesada por una señal sinusoidal.
Para conocer la respuesta en frecuencia de cualquier tipo de señal sea esta de característica periódica o no, se necesita obtener su representación fuera del dominio temporal. Esto se consigue mediante la teoría introducida por Joseph Fourier, la cual permite obtener la representación en frecuencia para señales periódicas mediante un algoritmo conocido como ***Serie de Fourier***. Mientras que para señales no periódicas el algoritmo que se introduce es conocido como ***Transformada de Fourier***.
La representación espectral de una señal, indica cómo se distribuye la energía de la misma en las diferentes componentes de frecuencia.
Para una señal periódica el espectro de frecuencias es discreto y la energía se distribuye en frecuencias múltiples de una frecuencia llamada fundamental directamente relacionada con el periodo de la señal.
Por otro lado, la representación espectral de una señal no periódica es una señal de frecuencia continua.[5]

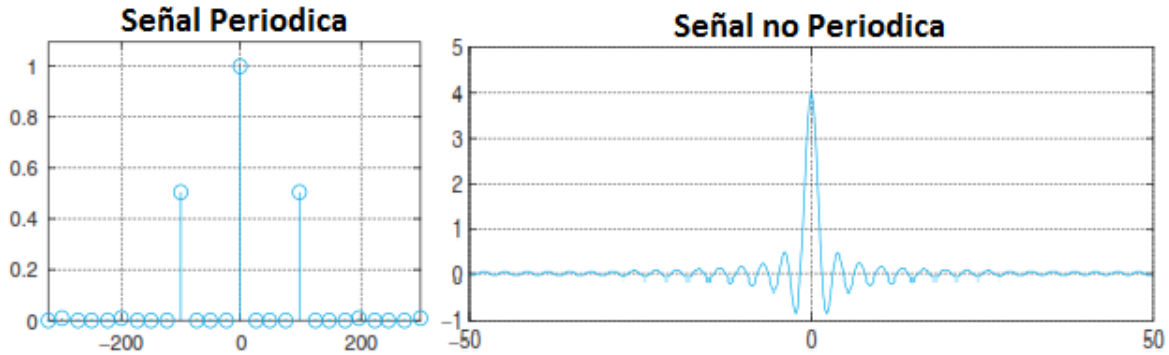


Figura 2. 7 Representación del espectro en frecuencias de señales Periódicas y no Periódicas

[5] Luis F. Chaparro. *Signals and Systems using **Matlab** Primera edición (2011) Pag 237*

2.2. Modulación

Modulación es el conjunto de técnicas desarrolladas para transportar señales de información a través de un canal de transmisión, el concepto de la modulación; consiste en hacer una traslación de frecuencias generalmente bajas a frecuencias superiores aunque no necesariamente. Técnicamente la modulación es cambiar algún parámetro (*Amplitud, Frecuencia, Fase)* de una señal llamada portadora (*Generalmente sinusoidal)*, de manera proporcional a la señal de información o señal modulante.[6]

2.2.1. ¿Por qué se modula una señal?

Es necesario modular las señales por diferentes razones:[7]

1) Si todos los usuarios transmiten a una misma frecuencia la señal original o moduladora, no será posible que en el receptor se reconozca la información contenida en dicha señal, debido a que el receptor captará todas las señales sintonizadas en su misma frecuencia lo que genera interferencia entre las señales transmitidas por diferentes usuarios.

2) A altas frecuencias se mejora la eficiencia en la transmisión y se logra proteger la señal inteligible de la presencia de ruido, de acuerdo al medio de transmisión que se emplee.

3) Se optimiza el espectro electromagnético, ya que permite la multiplicación por frecuencias aprovechando así el canal de comunicación enviando más información por el mismo canal.

4) En caso de transmisión inalámbrica, logramos reducir el tamaño de las antenas a medidas más razonables.

2.2.2. Modulación de amplitud (AM)

Este proceso de modulación es utilizado para variar las propiedades de una señal portadora de frecuencia relativamente alta, debe variar su amplitud de manera continua en el tiempo de acuerdo a la amplitud de la señal modulante o de información, por lo que a esta técnica se la incluye dentro de las llamadas “técnicas de modulación de onda continua”. En AM la amplitud se imprime sobre la portadora en forma de cambios de amplitud.[8]

---

[6] Wayne Tomasi. *Sistemas de Comunicaciones Electrónicas Segunda edición (2006) Pg. 102*

[7] http://www.eveliux.com/mx/Modulacion.html

[8] http://www.monografias.com/trabajos52/modulacion-angular-y-am/modulacion-angular-y-am2.shtml

La modulación de amplitud se caracteriza por ser una forma de modulación relativamente barata esto explica su baja calidad de transmisión y es utilizada en la radiodifusión de señales de audio y video entre otros.

Se obtiene un modulador AM a partir de elementos de características no lineales, introduciendo en el modulador dos señales de entrada de información (una señal portadora de amplitud constante y de frecuencia sencilla) y (una señal que contenga la información).[9]

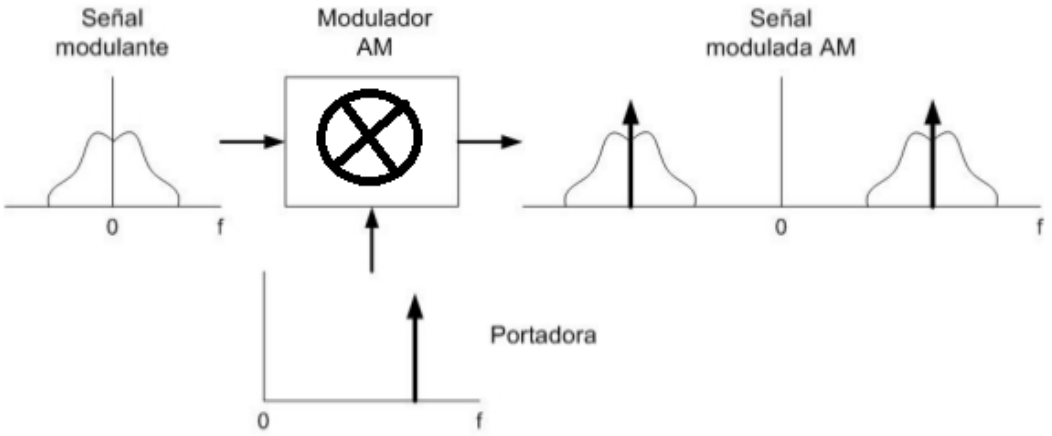


Figura 2. 8 Diagrama en bloques modulador AM básico (AM Comercial)

La información actuará sobre la portadora y nos devolverá una forma de onda de frecuencia simple o compleja compuesta de muchas frecuencias que fueron originadas a través de una o más fuentes. Debido a que la información actúa sobre la portadora, se la llama señal modulante y a la resultante se la llama señal modulada.

Lo mencionado se aprecia en la siguiente imagen:[10]

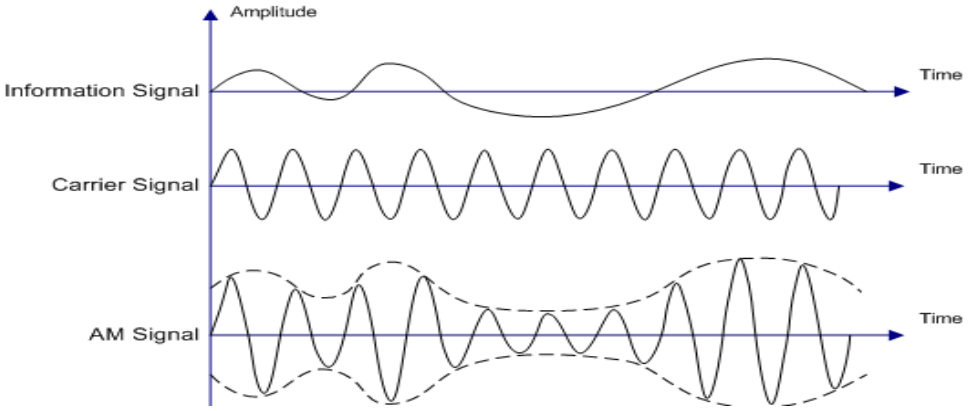


Figura 2. 9: Gráfico Modulación de amplitud (AM)

[9] http://www.elpatio.eu/index.html/Modulos/EST/Modulo-est/UD4-STR/Modulacion.html
[10] http://upload.wikimedia.org/wikipedia/commons/8/8d/Illustration_of_Amplitude_Modulation.png

2.2.3 Modulación de Frecuencia (FM).

En esta forma de modulación la frecuencia será la propiedad que se va a modificar dentro de la señal portadora para transportar la información. La modulación de frecuencia es un tipo de modulación angular. La modulación angular resulta cuando el ángulo de una onda sinusoidal varía con respecto al tiempo, manteniendo la amplitud de la portadora intacta en todo instante, por eso se la denomina "modulación de envolvente constante". La modulación en frecuencia "FM" es usada comúnmente en aplicaciones de radiodifusión que requieren gran fidelidad por ejemplo el sonido de la televisión analógica también es difundido por medio de FM.
De manera sencilla, se puede resumir que para obtener un modulador FM, se debe integrar una señal moduladora y luego esta debe ser usada para modular en fase una señal portadora de frecuencia sencilla. Como resultado, tenemos una señal en FM.[11]

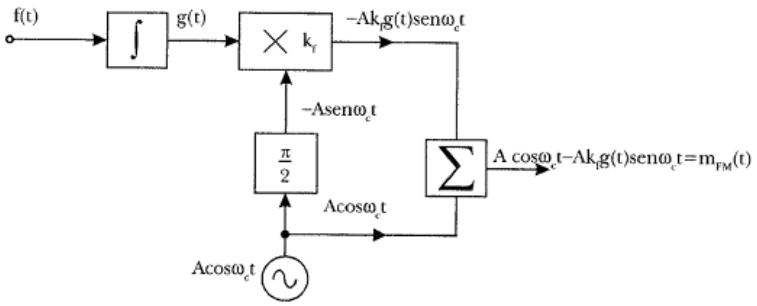


Figura 2. 10 Diagrama en bloques de un modulador FM

La modulación de frecuencia (FM), se consigue entonces: variando la frecuencia de la portadora de amplitud constante, directamente proporcional a la amplitud de la señal modulante. La siguiente imagen ilustra lo antes mencionado:[12]

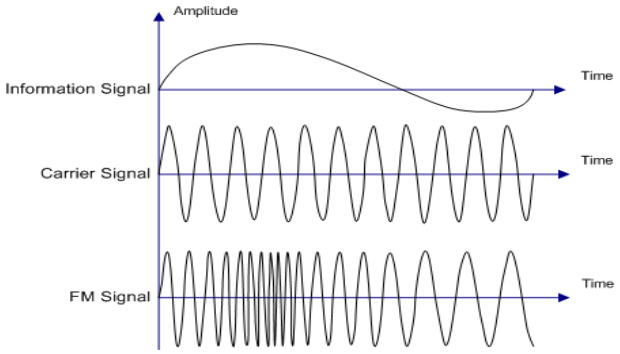


Figura 2. 11: Gráfico Modulación de frecuencia (FM)

[11] http://www.ie.itcr.ac.cr/acotoc/Maestria_en_Computacion/Sistemas%20de%20Comunicacion%20I/Material%20util/3. ModulacionFM.pdf
[12] http://upload.wikimedia.org/wikipedia/commons/d/d4/Illustration_of_Frequency_Modulation.png

2.2.4 Modulación de Fase (PM)

Este también es un caso de modulación angular al igual que la (FM). En este caso la propiedad de la señal portadora que se va a modificar es la fase. Esto es, que se hará variar el ángulo de la señal portadora de manera proporcional a la amplitud de la señal modulante. La modulación de fase (PM) presenta ciertas ventajas frente a la (FM), sin embargo no es muy utilizada principalmente porque el proceso que se requiere para la recepción de la información exige la fabricación de equipos más complejos. En realidad su presentación gráfica en el dominio del tiempo es muy similar a la representación que tiene la señal modulada en (FM), por lo que a simple vista es imposible diferenciar entre uno y otro tipo de modulación.

La figura 2.4 muestra la forma de la modulación de PM.[13]

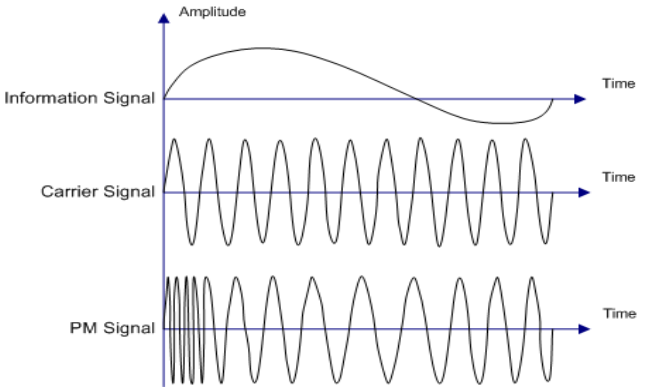


Figura 2. 12: Gráfico Modulación de fase (PM)

Gracias a la relación existente entre FM y PM, se puede resumir que para obtener una señal modulada en fase. Se debe modular la señal de información con una señal portadora de frecuencia sencilla desfasada y el resultado sumarlo con la misma señal portadora en fase. La relación indica que se puede usar el mismo circuito usado para obtener una señal en FM, pero sin integrar la señal moduladora.[14]

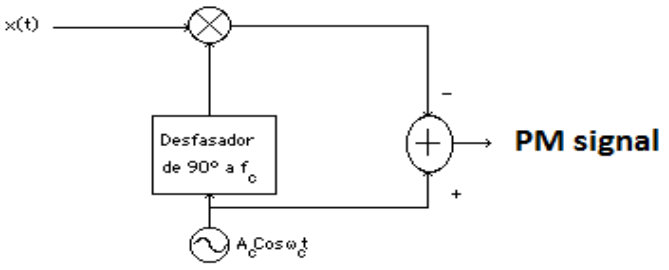


Figura 2. 13 Diagrama en bloques de un modulador PM

[13] http://upload.wikimedia.org/wikipedia/commons/d/d4/Illustration_of_Frequency_Modulation.png
[14] http://prof.usb.ve/tperez/docencia/2422/Capi/cap3/cap33/cap33.htm

2.2.5 Modulación de Pulsos codificados (PCM)

Este tipo de modulación fue diseñado con el propósito de cambiar una señal analógica a una digital, puesto a que la tendencia de las transmisiones de hoy en día necesita justamente hacer esa transición. El proceso en términos simples implica muestrear la señal analógica cada Ts segundos, donde Ts es el intervalo de muestreo o periodo. [15]

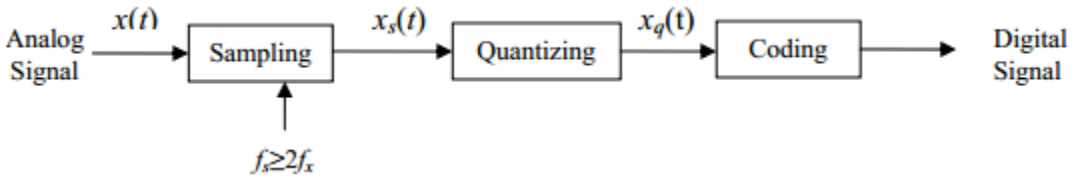


Figura 2. 14 Diagrama en bloques de un modulador PCM

Debido a que el muestreo nos trae una gran cantidad de puntos entre el valor máximo y el mínimo de la señal analógica muestreada y estas suponen valores no enteros infinitos entre dos límites, se deben asumir entonces valores instantáneos aproximando la muestra a los valores cuantificados con una cantidad determinada de bits por cada muestra, seguido a esto se cambia cada muestra a una palabra de longitud fija de un número de *n* bits. El número binario varía de acuerdo a la amplitud de la señal analógica. Siendo esta la modulación de pulsos más utilizada de todas.

La figura 2.5 ilustra lo mencionado. [16]

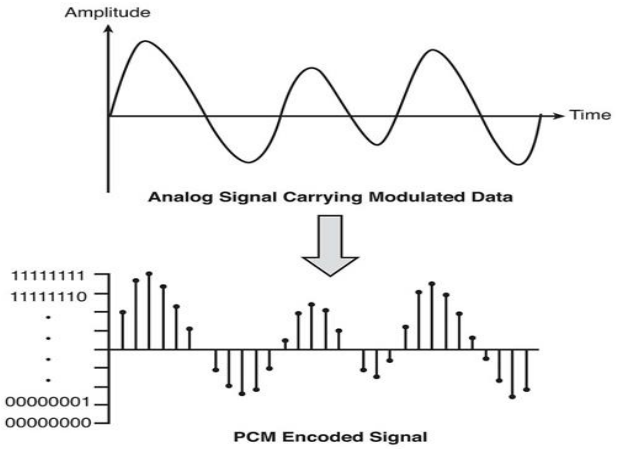


Figura 2. 15: Gráfico Modulación de pulsos codificados (PCM)

[15] http://elementsofpcm.blogspot.com/

[16] http://www.electronicshub.org/wp-content/uploads/2013/10/Pulse-Code-Modulation.jpg

2.2.6 Modulación en Cuadratura (QAM)

La modulación QAM es una forma de modulación digital, la cual tiene como principio enviar dos señales de distintas fuentes de información por un mismo canal de comunicación de manera simultánea. Esta modulación utiliza una señal portadora para que transporte la información tanto en su fase como su amplitud. Esto se consigue modulando una misma portadora, desfasada 90º entre uno y otro mensaje. Esto supone la formación de dos canales ortogonales en el mismo ancho de banda, mejorando la eficiencia del ancho de banda conseguido con esta modulación.[17]

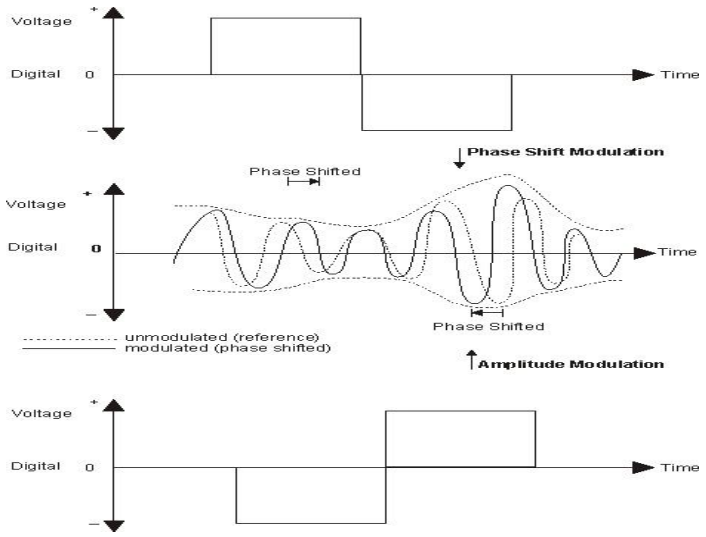


Figura 2. 16: Gráfico Modulación digital en cuadratura (QAM)

Esta técnica es popular frente a otras técnicas de modulación digital, especialmente por sus aplicaciones y bajo costo de transmisión, además que brinda mejores prestaciones en un ancho de banda similar con presencia de ruido blanco. Las ondas que utiliza generalmente son señales sinusoidales en la cual una hace de onda portadora y la otra de señal de datos. El conjunto de símbolos formando por el modulador QAM es un conjunto de palabras de J bits que luego pasan a un mapeo de estas palabras. El mapeo se encarga de seleccionar un símbolo de entre los $M = 2^J$ posibles símbolos. Los símbolos a transmitir son números complejos ubicados sobre un espacio bidimensional, símbolos que se pueden representar en el plano complejo, formando la constelación de la modulación, la figura 2.17 muestra el conjunto de constelaciones posibles para un esquema de modulación **16-QAM**.[18]

Esto significaría que se estaría utilizando palabras de 4 bits formando así una constelación $M = 2^4$. Resultado que nos daría el conjunto de símbolos posibles.

[17] http://www.catvdictionary.com/catv_dictionary_QAM_definition.html

[18] http://en.wikipedia.org/wiki/Quadrature_amplitude_modulation#mediaviewer/File:16QAM_Gray_Coded.svg

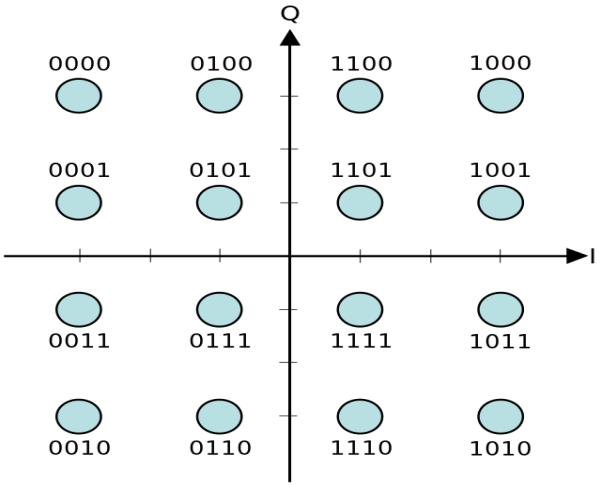


Figura 2. 17: Gráfico de las posibles constelaciones de la modulación digital en cuadratura (**16-QAM**)

Se obtiene un modulador QAM, al separar una palabra de bits codificados en dos grupos de bits iguales, los cuales serán transmitidos por canales distintos, "canal derecho" y "canal izquierdo". Estos bits a su vez serán modulados por una misma señal portadora, pero que tendrá un desfase de 90° entre sí para cada conjunto de bits. Como paso final se sumarán y en la salida tendremos como resultado una señal en modulada en cuadratura.[19]

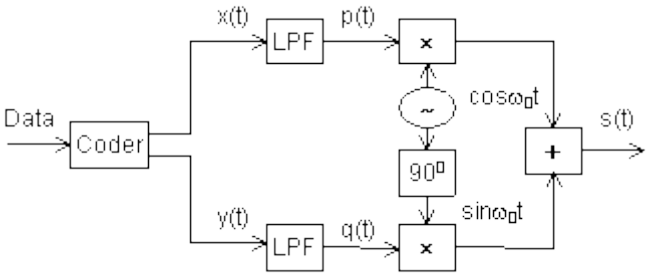


Figura 2. 18 Diagrama en bloques del modulador QAM

2.2.7. Modos de Transmisión

- Los sistemas básicos de telecomunicaciones se pueden diseñar con la intención de manejar la transmisión en una sola dirección, en ambas direcciones pero sólo uno a la vez, o ambas direcciones al mismo tiempo. A esto se le conoce como modos de transmisión que se lo define a continuación:[20]

[19] http://www.dsp.pub.ro/leonardo/ipa/Chapter2/Level1/SubChapter2.10/Subchapter2_10.htm
[20] José E. Briceño M. *Transmisión de Datos Pg. 155*

2.2.7.1 Simplex

La transmisión se realiza en una sola dirección, como una calle de una sola dirección. Los sistemas simplex o sistemas de un sentido, se usan en canales dedicados a sólo recibir o sólo transmitir mensajes. El mensaje puede ubicarse en un transmisor o un receptor pero, no en ambos. Los ejemplos de transmisión simplex son aplicaciones de radiodifusión como la radio comercial o la televisión; la estación de radio siempre transmite y el usuario siempre recibe. El modo simplex puede usar todo el ancho de banda disponible en el canal para enviar datos en una dirección.
En la figura 2.19 se puede observar un ejemplo de la transmisión simplex.[20]

Figura 2. 19: Gráfico del Modelo de Tx SIMPLEX

2.2.7.2. Half Dúplex o Semi Dúplex

La transmisión se realiza en ambas direcciones pero no de manera simultánea, cuando un dispositivo está enviando el otro solo puede recibir y viceversa, podríamos hacer la analogía de este modo de transmisión al compararlo con una calle de un solo carril y tráfico en dos direcciones. Estos sistemas son llamados también: sistemas con alternativa de los sentidos, cualquier sentido, o cambio y fuera. El mensaje puede ubicarse en un transmisor o en un receptor, pero no en los dos al mismo tiempo. Por ejemplo los sistemas de radio de doble sentido que utilizan los botones Push-to-talk (PTT) “oprima para hablar”, para operar sus transmisores, como los radios de banda civil y de banda policiaca o sistemas de comunicación de taxistas. En este modo de transmisión todo el ancho de banda del canal es usado por la estación que se encuentre transmitiendo.
En la figura 2.20 se puede observar un ejemplo de la transmisión Half Dúplex.20

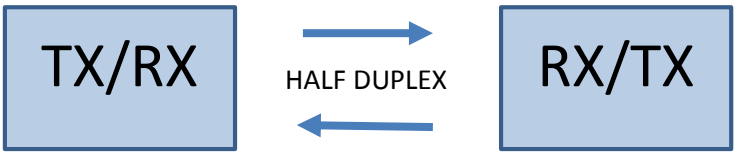


Figura 2. 20: Gráfico del Modelo de Tx HALF DUPLEX O SEMIDUPLEX

2.2.7.3. Full Dúplex

La transmisión se realiza de manera simultánea en ambas direcciones, llamadas también líneas simultáneas de doble sentido, dúplex o de ambos sentidos.
El mensaje puede ubicarse en transmisor y receptor simultáneamente; sin embargo, la comunicación debe ser orientada a un mismo canal; es decir que la estación a la que se está transmitiendo también debe ser la estación de la cual está recibiendo. Por ejemplo un sistema telefónico estándar. En este modo de transmisión el ancho de banda del canal está dividido entre ambas estaciones
En la figura 2.21 se puede observar un ejemplo de la transmisión Full Duplex.[20]

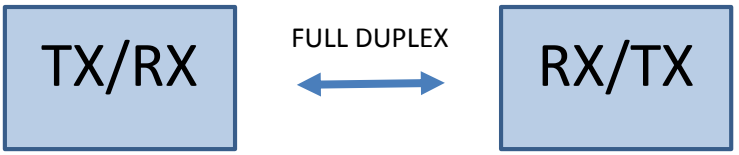


Figura 2. 21: Gráfico del Modelo de Tx FULL DUPLEX

2.3. Demodulación

Es el proceso de revertir los cambios hechos a la portadora analógica, con el fin de recuperar la señal de información originada en la fuente. La demodulación se realiza en el receptor, en un circuito llamado demodulador o detector como también se le conoce.

El detector o demodulador debe ser capaz de recibir y amplificar una señal. Un detector también debe tener la capacidad de limitar las bandas del espectro total de la señal modulada a una banda específica de frecuencias.

La modulación consiste básicamente en hacer una traslación de espectros o lo que es lo mismo, hacer convolucionar los espectros tanto de la señal modulante como de la señal moduladora, se puede obtener un demodulador haciendo convolucionar nuevamente estos espectros con el espectro de la señal portadora y aplicar un filtro pasa bajos centrado en la frecuencia de la señal moduladora. Es necesario centrar el filtro a la frecuencia de la moduladora ya que justamente el espectro de esta señal fue trasladado a la frecuencia de la señal portadora. Con esto se eliminarán las frecuencias más altas, el ejemplo más claro sobre esta teoría se obtiene de la modulación Am de portadora suprimida. La figura 2.22 ilustra esta teoría.[21]

[21] http://slideplayer.es/slide/25430/

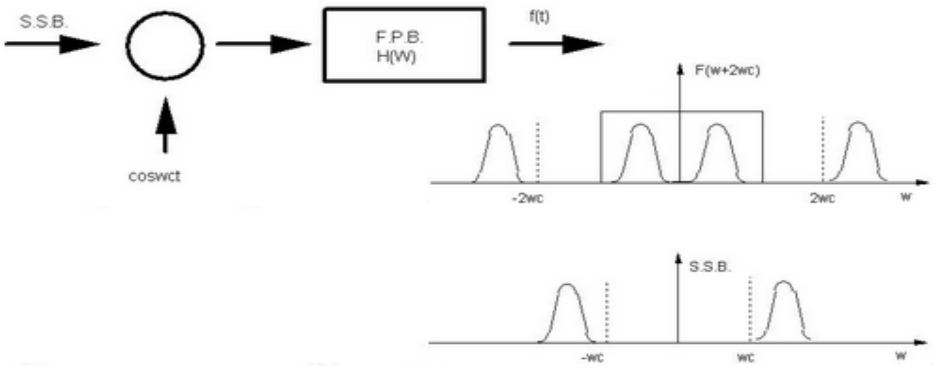


Figura 2. 22 Ejemplo de demodulación y traslación de espectros

Los receptores que se utilizan para las señales de modulación angular son muy similares a los que se usan para la recepción de AM de portadora suprimida, excepto por el método utilizado para extraer la información de audio.

Los métodos usados en la recepción de señales son dos: el primero es conocido como ***Detección Coherente*** y el otro es conocido como ***Detector de Envolvente***.

En la detección coherente lo que se hace para recuperar la señal es multiplicar la señal modulada por un oscilador local a frecuencia de la portadora y pasar el resultado de esta convolución usando un filtro pasa bajos, centrado a la frecuencia de la señal de información.

En la detección de envolvente, aplicamos simplemente un filtro pasa bajos situado a frecuencia de la portadora y obtenemos la señal enviada.[22]

2.4. Problemas en la recepción de señales

Una vez que la señal llega al receptor se presenta con cierta clase de inconvenientes, los cuales merecen ser profundizados en totalidad, pero no son el objeto de este trabajo. Se puede resumir que debido a que la señal se traslada gran distancia para lograr enlazar dos puntos, disminuyen en ella importantes características, como por ejemplo la amplitud, lo cual impide que sea una réplica exacta de la señal original enviada. Esto sucede cada que atraviesa un medio de transmisión sea guiado o no guiado. A este hecho se lo conoce como atenuación y todos los receptores son diseñados para tolerar un rango mínimo de amplitudes con niveles de potencia o voltaje aceptables, con el objetivo de diferenciarlas del ruido. [23]

[22] Wayne Tomasi. *Sistemas de Comunicaciones Electrónicas Segunda edición (2006) Pg. 272-280*
[23] José E. Briceño M. *Principios de las comunicaciones Tercera Edición (2005) Pag 263*

CAPITULO III

3.1. Software de simulación *Matlab/Simulink*

Matlab fue creado por el matemático Cleve Moler dentro de la empresa "MathWorks" líder en desarrollo de software para cálculo técnico. En 1984 se lanza su primera versión, surgiendo con el pensamiento de usar paquetes de subrutinas escritas en Fortran para cursos de álgebra lineal y análisis numérico, sin necesidad de escribir programas en dicho lenguaje. El lenguaje de programación ***Matlab*** fue creado en 1970 para proporcionar un sencillo acceso al software de matrices LINPACK (Linear system package) y EISPACK (Eigen system package) sin tener que usar Fortran.

Desde el año 2004 se logró apreciar que ***Matlab*** ya era usado por más de un millón de personas involucradas en los ámbitos académicos y empresariales.

Matlab es una entre varias sofisticadas herramientas computacionales disponibles en el comercio para la resolución de problemas matemáticos, tales como Maple, Mathematica y MathCad, sin embargo ***Matlab*** fue pensado para trabajar con matrices y proporcionar un entorno sencillo con distintas prestaciones para los usuarios en panoramas como el análisis numérico, cálculo matricial, programación, procesamiento de señales y gráficos.

El nombre mismo de ***Matlab*** es una abreviatura de Matrix Laboratory *"Laboratorio Matricial"*. En un nivel fundamental se puede pensar y promover a este tipo de programas como sofisticadas calculadoras con base en una computadora. Se puede usar ***Matlab*** hasta para las más simples de las operaciones matemáticas, pero el enfoque general del software es remplazar la programación tradicional. Esto significa que los usuarios no deberán aprender previo a ***Matlab*** lenguajes de alto nivel como C o Fortran, ya que ***Matlab*** en estos días es considerado una herramienta estándar en la mayoría de universidades e industrias alrededor del mundo, tal es así que el momento de su publicación se lanzaron 1400 libros ***Matlab*** escritos en 28 idiomas.[24]

[24] http://ordenador.wingwit.com/Programacion/computer-programming-languages/87495.html#.VMOrtEeG9ps

3.1.1. Origen

Cleve Moler era un profesor de matemáticas y ciencias de computación en la Universidad de Nuevo México de Estados Unidos. Cuando desarrolló la primera versión del software ***Matlab***, Moler buscaba que sus estudiantes tengan acceso a los paquetes de software de matices Linpack y Eispack sin tener que usar el lenguaje de programación FORTRAN ya que este era muy complejo. De acuerdo a un artículo de "Computación Científica Mundial", Moler brindo la solución al problema de complejidad de cálculos matriciales a través del software ***Matlab***. [25]
Con el pasar de los años fue complementado y re implementado en lenguaje C. Actualmente la licencia de ***Matlab*** es propiedad de MathWorks Inc.

3.1.2. Descripción del Software *Matlab*

Matlab es un poderoso lenguaje de programación de alto nivel que incluye conceptos comunes a la mayoría de lenguajes de programación. Puesto que ***Matlab*** se basa en la creación de scripts, la creación de programas es mucho más fácil ya que nos brinda un entorno interactivo para el desarrollo de algoritmos, visualización de datos, análisis de datos y cálculo numérico. Sea cual sea la necesidad que se tenga (un análisis, una estadística, un algoritmo, informes o simulación), podemos ejecutar programas o guiones que contengan comandos ***MATLAB***, observar los resultados, ejecutar otro comando ***MATLAB*** que interactúe con la información almacenada en memoria, observar sus resultados y así sucesivamente. Este entorno interactivo no requiere el proceso de compilación, sin embargo, los errores de sintaxis y lógica en un comando, pueden ser causa de errores de ejecución cuando el entorno ***MATLAB*** ejecuta dicha instrucción.
El lenguaje de programación ***Matlab*** permite manipular vectores o matrices como simples variables. Debido a esto es ideal para cálculos que involucran matrices, ya que los ejecuta sustancialmente más rápido que otros lenguajes de alto nivel.
El software de ***Matlab*** crece y se actualiza continuamente, por lo que de manera regular aparecen nuevas versiones; este proyecto de titulación usa la versión estudiantil 8.3.0.532 de ***Matlab*** referida como R2014a para la generación de gráficas y descripción de algunas de sus funcionalidades.
El software estándar de ***MATLAB*** se agrupa en una serie de herramientas que pueden ser usadas para resolver problemas comunes, sin embargo, ***Matlab*** incorpora también otras librerías específicas llamadas (***Toolboxes***) y una colección de funciones y paquetes de bloques para ***Simulink*****,** mismos que están diseñados para resolver problemas muy puntuales y específicos.
Simulink es una extensión de ***Matlab*** que proporciona un entorno gráfico para simulación de procesos muy puntuales, usando herramientas que se colocan dibujando diagramas de bloques que representarán la función de

[25] http://es.0430.com/us/web114842/

algún sistema y cada uno de estos ejecutarán una determinada función específica. Algunas de estas colecciones de herramientas se pueden usar para adquisición, exploración y análisis de datos, visualización y procesado de imágenes, modelado y simulación, programación y desarrollo de aplicaciones. [26]

3.1.3. Instalación del Software *Matlab* versión estudiantil

Requisitos del sistema para instalar ***Matlab*** en consola Windows.[27]

- Memoria Ram de 1GB o superior
- PC con procesador x86 o x64 Intel® o AMD®
- Windows XP (SP3), Windows Vista (SP2), Windows 7(SP1), Windows 8 o Windows 8.1.

Los pasos a seguir para la instalación del software *Matlab* versión estudiantil son ilustrados mediante las siguientes figuras:[28]

1. Entramos al sitio de Mathworks e introducimos el serial de nuestro paquete para asociarlo a una licencia válida para iniciar la descarga de nuestro software estudiantil, descomprimimos los archivos de la descarga y cargamos la imagen ISO que contenga los instaladores de ***Matlab*** y ejecutamos el archivo ***SETUP.EXE***

Figura 3. 1: Ejecución del instalador de ***Matlab***

[26] Amos Gilat. ***Matlab*** *una introducción con ejemplos prácticos Segunda edición (2005) Pg. 1*

[27] http://www.mathworks.com/support/sysreq/current_release/?refresh=true

[28] https://itservices.usc.edu/matlab/windows/

2. Aceptar los términos de licencia y damos en clic en "**Next"**

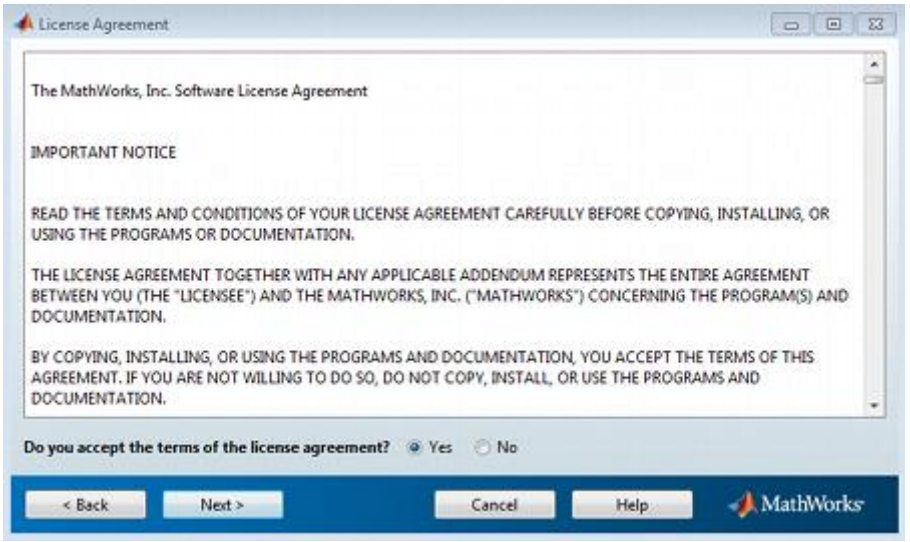


Figura 3. 2: Aceptación de términos y condiciones de la licencia de Mathworks

3. Nos registramos dentro de nuestra cuenta de *Mathworks* con nuestro e-mail y contraseña y damos clic en "**Next"**. Una vez abierto el instalador ***Matlab*** debemos seleccionar la opción "***Log in*** *with a Mathworks account".*

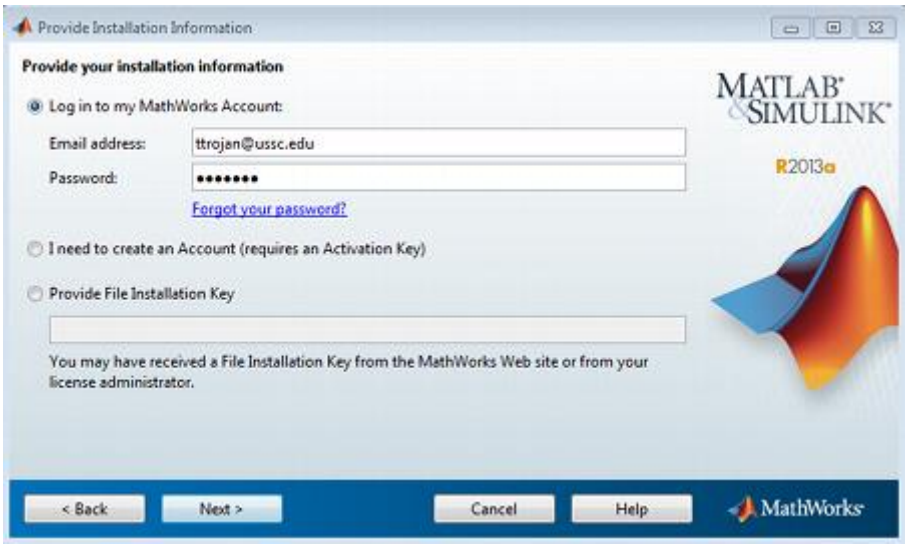


Figura 3. 3: Colocando información acerca de nuestra cuenta de Matworks

4. Elegimos la ruta donde queremos instalar ***Matlab***, normalmente es **c*/* "archivos de programas"** donde se instalan los nuevos paquetes de software y damos clic a "NEXT".

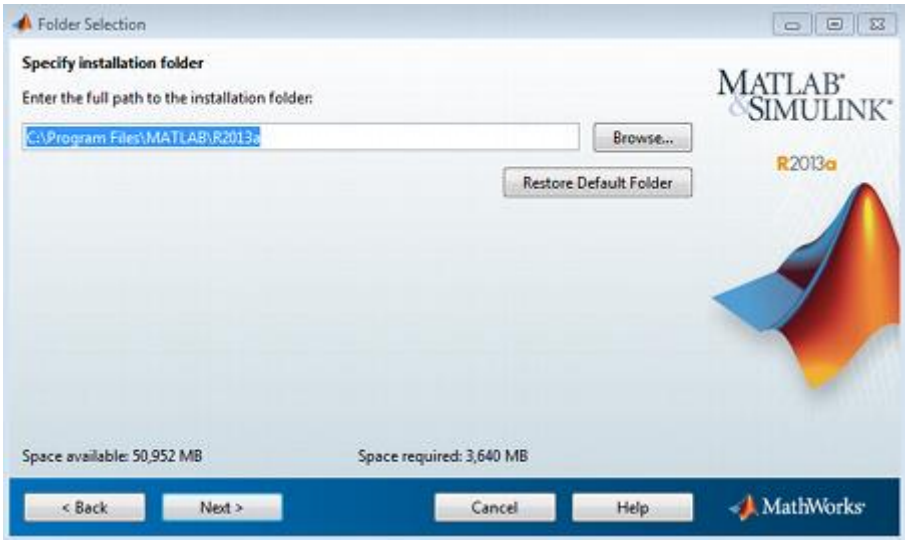


Figura 3. 4: Especificando la ruta de instalación de ***Matlab***

5. Ingresamos nuestro código de activación de licencia, el cual es un grupo de 5 números separados por guiones

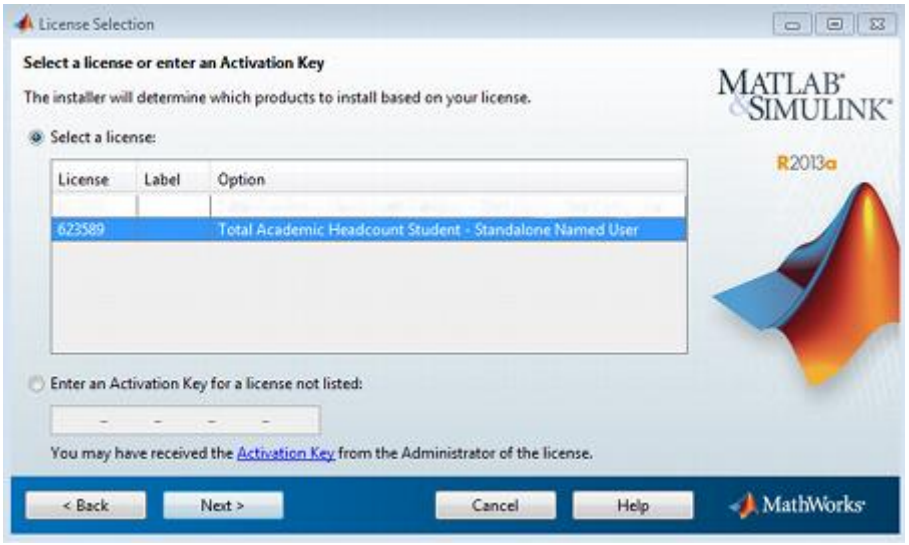


Figura 3. 5: Insertando el código de actvacion de nuestro software

6. Ahora nos mostrará un panel con los componentes que deseamos instalar. Por defecto dejamos seccionada la opción "Typical", la cual nos instala todos los productos que trae el programa, la instalación personalizada nos permite seleccionar que productos instalar y presionamos "Next".

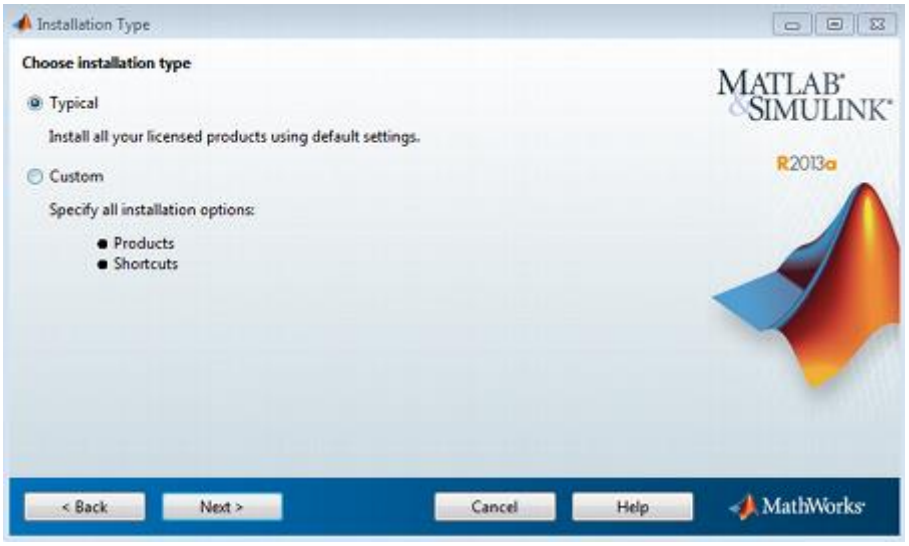


Figura 3. 6: Seleccionando el tipo de instalación de ***MATLAB***

7. Confirmamos la creación del nuevo directorio y nos mostrará la siguiente ventana en donde solo debemos presionar el botón "Install", esperar a que se instale el software ***Matlab*** con todos sus componentes, reiniciar la PC y ***Matlab*** está listo para usarse.

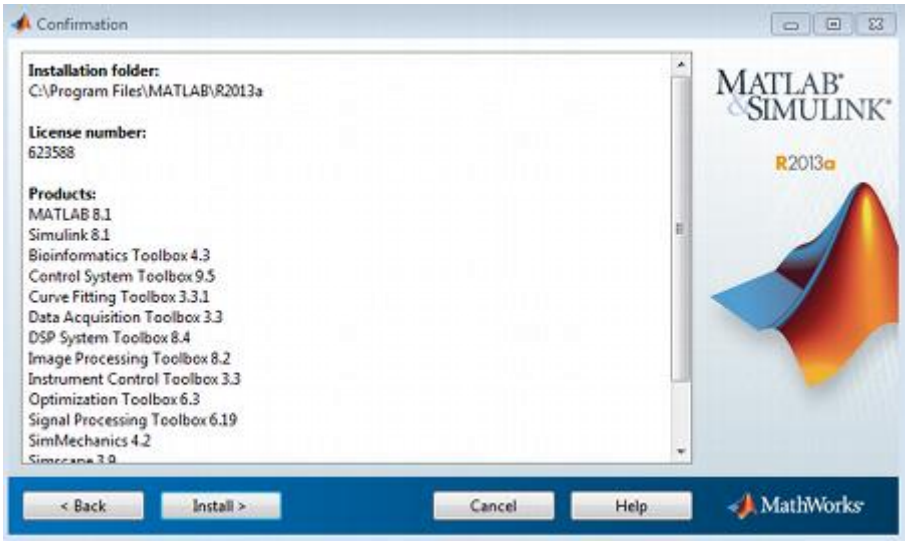


Figura 3. 7: Instalando ***MATLAB***

3.1.4. Entorno de trabajo *Matlab*

A partir de la versión 6.0 de ***Matlab*** se han hecho evidentes muchas mejoras dentro del software, dentro de estas, una de las más destacadas es el entorno de trabajo. Se puede encontrar un entorno mucho más interactivo, similar a las aplicaciones profesionales de **Windows.** Cuando se accede a ***Matlab*** por primera vez necesitamos familiarizarnos con varios componentes de su presentación.

Los componentes más destacados del entorno de trabajo ***Matlab*** son listados a continuación:[29]

1. **El *Escritorio de Matlab*** (***Matlab Desktop***)
   Es la ventana o contenedor de máximo nivel mediante la que se pueden llegar hacia las demás componentes del software.

2. **Las componentes individuales orientadas a tareas concretas**
   Dentro de estas, podemos citar:
   a. La ventana de comandos (***Command Window***)
   b. La ventana histórica de comandos (***Command History***)
   c. El espacio de trabajo (***Workspace***)
   d. La plataforma de lanzamiento (***Launch Pad***)
   e. El directorio actual (***Current Directory***)
   f. La ventana de ayuda (***Help***)
   g. El editor de ficheros y depurador de errores (***Editor&Debugger***)
   h. El editor de vectores y matrices (***Array Editor***)
   i. La ventana que permite estudiar cómo se emplea el tiempo de ejecución (***Profiler***)

Se debe tener en cuenta que desarrollar aplicaciones y programas ***Matlab***, es mucho más sencillo si se está familiarizado con el entorno de trabajo, por lo que a continuación se describen brevemente las componentes listadas anteriormente, de manera que se pueda llegar lo más rápido posible a una alta productividad personal en el uso de ***Matlab*****.**

3.1.4.1. Escritorio de *Matlab* (*Matlab* Desktop)

Se trata de la ventana más general del ***Matlab***, dentro de esta se alojan el resto de componentes antes citados ya sea como sub-ventanas independientes, o como pestañas dentro de las sub-ventanas, sin embargo, gracias a la flexibilidad de ***Matlab***, se nos permite decidir libremente la apariencia de nuestro **"*Escritorio*"**.
En la versión 8.3 de ***Matlab*** la apariencia de nuestro escritorio la podemos editar desde la ventana "**Layout**" la cual se encuentra anclada en la barra de herramientas de ***Matlab***.
Dentro de la pestaña "**Layout**", se nos despliega un menú que permite escoger la apariencia del escritorio que ***Matlab*** trae por defecto, en el grupo marcado como **"*Select Layout*"** se puede probar las distintas apariencias que tendría nuestro "Escritorio". La figura 3.8 ilustra lo descrito anteriormente.[30]

---

[29] Amos Gilat. ***Matlab*** *una introducción con ejemplos prácticos Segunda edición (2005) Pg. 5-6*
[30] Print Screen del programa ***Matlab***

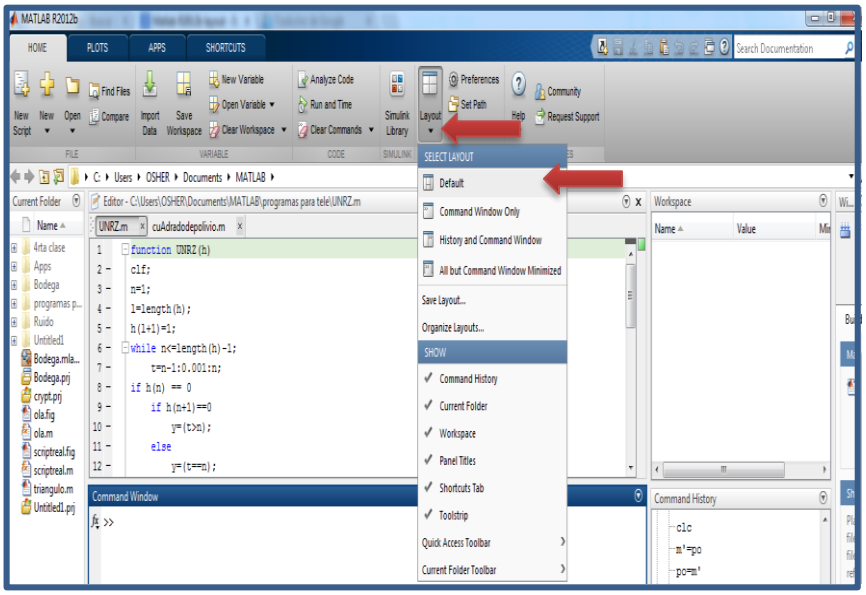


Figura 3. 8: Opciones del menú "Layout"

La figura 3.9 ilustra la vista por defecto.[31]

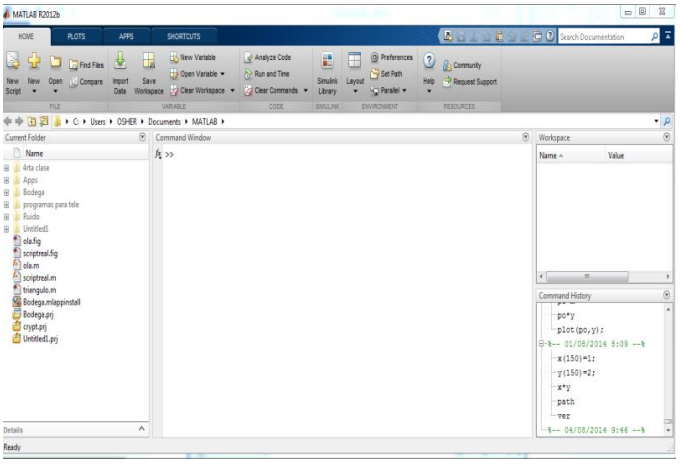


Figura 3. 9: ***Matlab*** Desktop (configuración por defecto)

3.1.4.2. Ventana de comandos (Command Window)

Esta ventana es la que permite la comunicación entre ***Matlab*** y el usuario mediante el ingreso de órdenes (seguidas de "enter"). Los resultados de las órdenes tecleadas se muestran inmediatamente en la misma ventana. Si las órdenes son enviadas desde un programa de ***Matlab*** pre-escrito conocido como **M-file**, la ejecución de estas órdenes también serán mostradas dentro de la ventana mencionada. Esta es la ventana más importante del ***Escritorio Matlab***, tal es así, que es la única que ha estado presente desde versiones pasadas del software. Por esta razón se profundizará más adelante todas las cualidades del ***Command Window***.[29]

[31] Print Screen del programa ***Matlab***

3.1.4.3. Ventana Histórica de comandos (Command History)

Esta ventana es un registro de las órdenes escritas anteriormente en la ***Ventana de Comandos***. A estas sentencias se puede acceder dando *doble clic* en el registro de comandos ejecutados anteriormente, o utilizando los cursores ↑ y ↓ del teclado.
La figura 3.10 ilustra lo descrito anteriormente.[32]

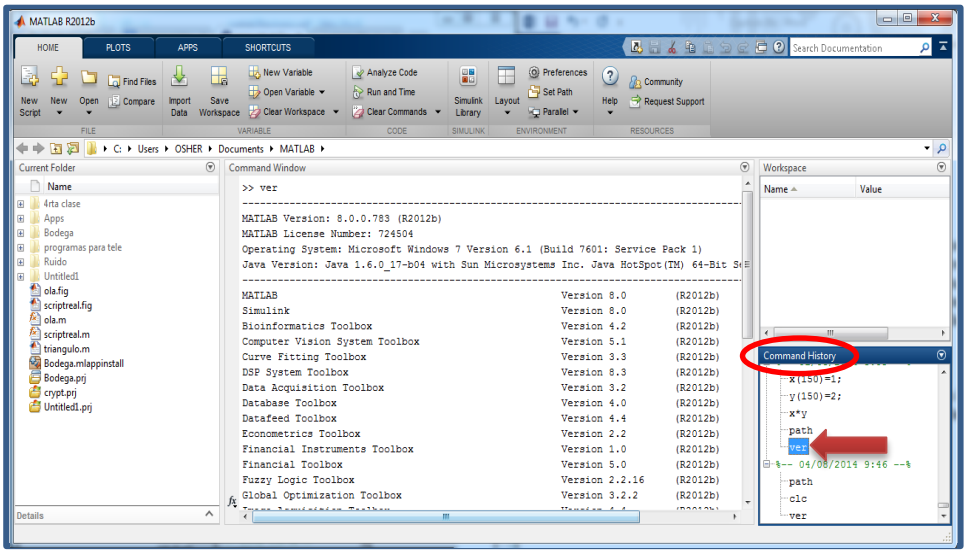


Figura 3. 10: Command History acceso al registro de sentencias usando doble clic

3.1.4.4. Espacio de trabajo (Workspace)

En esta ventana se muestran las variables creadas por el usuario dentro de la sesión ***Matlab*** ya sean estas (vectores, escalares, matrices…).[33]

La información que esta ventana nos proporciona en relación a las variables creadas son: nombre, dimensión, tamaño y tipo de variable. Los comandos dedicados para trabajar con la información existente en El ***Espacio de trabajo*** son los siguientes:

- ***Who:*** Despliega una lista de las variables que se han definido en la sesión ***Matlab*.**
- ***Whos:*** Despliega una lista de las variables definidas en la sesión ***Matlab*** con sus respectivos tamaños en memoria.
- ***Clear "nombre de la variable":*** Borra una variable de nombre específico, creada en la sesión ***Matlab***.
- ***Clear all:*** Borra todas las variables creadas en la sesión ***Matlab*.**

[32] Print Screen del programa ***Matlab***
[33] Moore, Holly. *Matlab para Ingenieros Primera edición (2007Pg. 12-15*

Nota: El complemento del comando Clear debe estar precedido con un espacio.

3.1.4.5. Plataforma de Lanzamiento (Launch Pad)

Este es un recurso muy general que nos facilitaba conocer cuáles son las componentes de ***Matlab*** que tenemos instaladas en nuestro computador y acceder a ellas de manera simplificada. La figura 3.11 ilustra el recurso mencionado.[34]

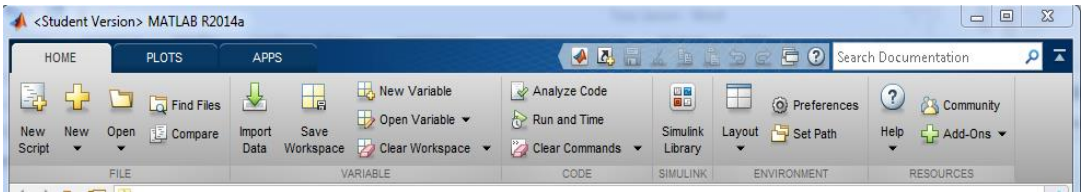


Figura 3. 11 Plataforma de lanzamiento Launch Pad

3.1.4.6. Directorio Actual (Current Directory)

Esta ventana despliega una lista de todos los archivos contenidos en un folder de nuestro ordenador llamado ***Directorio Actual***. Cuando ***Matlab*** ingresa a un registro de archivos o guarda información, usara el ***Directorio Actual*** a menos que se le haya configurado para algo diferente. La dirección del ***Directorio Actual*** varía dependiendo de la versión de software ***Matlab*** que se instaló. Sin embargo, el ***Directorio Actual*** se cita en la parte superior de la ventana principal del ***Matlab Desktop***. El directorio se puede cambiar haciendo uso del fólder de la lista desplegable, similar a las opciones de exploración que corren en la consola de *Windows*.[33]

3.1.4.7. La ventana de ayuda (Help)

Esta ventana es una de las que más destaca a ***Matlab*** como un software interactivo. Contiene toda la información que se desee acerca de ***Matlab***, desde ayudas en la sintaxis de un comando, hasta tutoriales completos posteados en el sitio web de Mathworks (***www.mathworks.com***).
La figura 3.12 muestra la ubicación de la Ayuda, en el Desktop de ***Matlab***.[35]

[34] Print Screen del programa ***Matlab***
[35] Print Screen del programa ***Matlab***

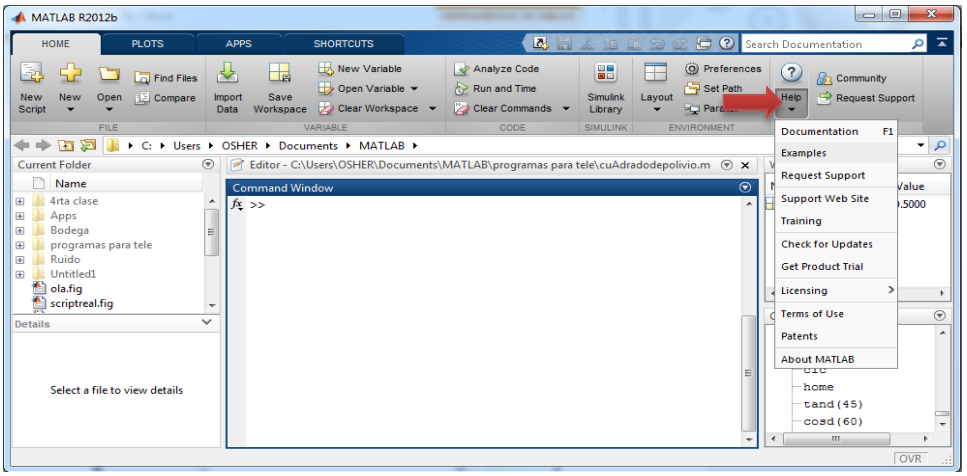


Figura 3. 12: Ayuda en el Desktop ***Matlab***

3.1.4.8. El editor de ficheros y depurador de errores (Editor&Debugger)

En la programación dentro del entorno ***Matlab***, tienen particular importancia los ya mencionados ***M-files***. Estos son ficheros de texto ASCII, con la extensión *.m, que representan un conjunto de comandos. La particularidad de estos es que al teclear su nombre en la línea de comandos y pulsar ***"enter"*** se ejecutan uno detrás de otro todos los comandos contenidos dentro de dicho fichero. Claramente guardar un conjunto de instrucciones y matrices de gran tamaño, permiten ahorrar tiempo en el desarrollo de programas.[33]

3.1.5. Ventana de comandos (Command Window)

Se trata de la ventana más importante del ***Escritorio Matlab***, en esta ventana se ejecutan interactivamente las instrucciones de ***Matlab***, para tener un buen criterio en su uso, se requiere estar familiarizado con las siguientes generalidades:

1. En la ***Ventana de Comandos*** se nos avisa que ***Matlab*** está listo para recibir una instrucción mediante un símbolo característico; ">>" llamado ***prompt*** (aviso). Este símbolo aparece libre después de haberse ejecutado alguna instrucción, o simplemente para marcar que programa está listo para una nueva línea de comandos, los comandos se teclean a continuación del símbolo mencionado.
2. Los comandos de ***Matlab*** se completan automáticamente pulsando la tecla de tabulación, cuando iniciamos a teclear el nombre de un comando y presionamos dicha tecla, ***Matlab*** nos mostrará a acto seguido todos los comandos que comienzan con las letras tecleadas, o simplemente, completará el nombre del comando iniciado.

3. El lado izquierdo del ***prompt*** nos muestra un ícono de función ***"(fx)"*** el cual es una lista desplegable de funciones o comandos contenidos en ***Matlab***, alternativamente también se puede acceder a esta opción usando las teclas **SHIFT+F1**.
4. Los errores de sintaxis en los comandos ejecutados se muestran en la ***Ventana de Comandos*** en color rojo a renglón seguido del ***prompt***. En general para errores de sintaxis, ***Matlab*** nos genera tres avisos: **1)** "*Variable o función no definida"*; **2)** "*Quisiste decir:"*; **3)** "*Entrada de carácter no valido"*.
5. La ejecución de comandos ***Matlab*** se muestra de manera inmediata en la ***Ventana de Comandos*** cuando estos no finalizan en punto y coma.
   Este tipo de respuesta del software ***Matlab*****,** tiende a llenar la pantalla de comandos brevemente, para limpiar la ventana de comandos se ejecuta el comando ***"clc"***. [36]

3.1.6. Funciones matemáticas elementales del software *Matlab*

Matlab contiene en sus librerías varias funciones básicas y trascendentales, algunas de estas funciones las usamos dentro del ámbito tecnológico de los sistemas básicos de telecomunicaciones para representar parte de sus procesos, como por ejemplo el generar una señal o analizar su ancho de banda a partir de las componentes espectrales que contiene la misma.

También hay que considerar que la mayoría de expresiones vistas en ámbitos de Telecomunicaciones son de carácter algebraico, es decir que existe casi una obligación de tratar con expresiones alfanuméricas. Para este propósito ***Matlab*** , tiene una librería llamada *Symbolic Math Toolbox* cuya cualidad, permite trabajar con variables como si fuesen números, tal como se hace en el álgebra tradicional con la que se está familiarizado, con lo cual se puede realizar: integración, diferenciación, simplificación, transformadas y resolución de ecuaciones de manera sencilla.

Las funciones elementales que se listan a continuación, cuando se aplican dentro de una matriz, actúan elemento por elemento como si estos se tratasen de un escalar. Actúan de la misma manera, si se tratan de escalares o vectores.[37]

Sea x un vector de "n" número de elementos:

---

[36] Amos Gilat. ***Matlab*** *una introducción con ejemplos prácticos Segunda edición (2005) Pg. 8-10*

[37] Moore, Holly. *Matlab para Ingenieros Primera edición (2007) Pg. 59-60*

- fft(x) Calcula la transformada de Fourier de una expresión algebraica, para lo cual las variables deben ser declaradas en modo simbólico.
- int(x) Calcula la integral de x, para lo cual x debe ser una variable simbólica
- diff(x) Calcula derivadas parciales y totales de una expresión algebraica para lo cual las variables deben ser declaradas en modo simbólico.
- exp(x) Función exponencial de base *e*
- log(x) Función Logaritmo neperiano
- log10(x) Función Logaritmo decimal
- sqrt(x) Función Raíz cuadrada
- fix(x) Elimina la parte decimal de la real de un número complejo.

3.1.7. *Matlab* Help (Ayuda)

La ***Ayuda*** del software ***Matlab*** es un complemento vital incluso para los programadores experimentados, por esa razón, es fundamental estar familiarizado con él. Cuando se accede al Help por medio del ***Escritorio Matlab***, podremos encontrar una lista de ayudas relacionadas con el programa. Estas ayudas son una compilación documentada en archivos PDF o a su vez, se las puede encontrar posteadas en formato HTML en Internet. Las ayudas que encontramos útiles destacan en los siguientes 3 items:[38]

- ***Documentation (Documentación).-*** Contiene toda la documentación acerca de ***Matlab***, como son sus comandos y ***Toolboxes.*** La información se encuentra respaldada en un conjunto de artículos y ejemplos que muestran soluciones a problemas mediante el uso de comandos del software. Se puede acceder también a esta parte de la ayuda presionando la tecla **F1**.
- ***Examples (Ejemplos)***.- Contiene ejemplos interactivos con video tutoriales acerca del uso de funciones e incluso programas completos realizados en ***Matlab***. Mediante un solo clic, en la ayuda que requiramos. ***Matlab*** nos dirigirá hacia donde se encuentre almacenado el contenido de dicha ayuda, la misma que puede estar localizada en línea o en nuestro mismo computador.
- ***About Matlab (Acerca de Matlab)***.- Despliega información del software ***Matlab*** que tengamos instalado en nuestro computador, incluyendo datos como la licencia activa del producto.

También se puede acceder a la **Ayuda** de ***Matlab*** por medio de una función en la línea de comandos llamada justamente **"*help"***. La forma adecuada de usar esta función es la siguiente:

[38] Moore, Holly. *Matlab para Ingenieros Primera edición (2007) Pg. 57-59*

>> help "comando"

Donde el comando ***Matlab*** del cual se quiere obtener información debe estar precedido por un espacio, una vez completo se presiona "**enter**" y se nos desplegara toda la información acerca del comando tecleado.

La figura 3.13 ilustra el acceso a los ejemplos que la Ayuda de ***Matlab*** proporciona.[39]

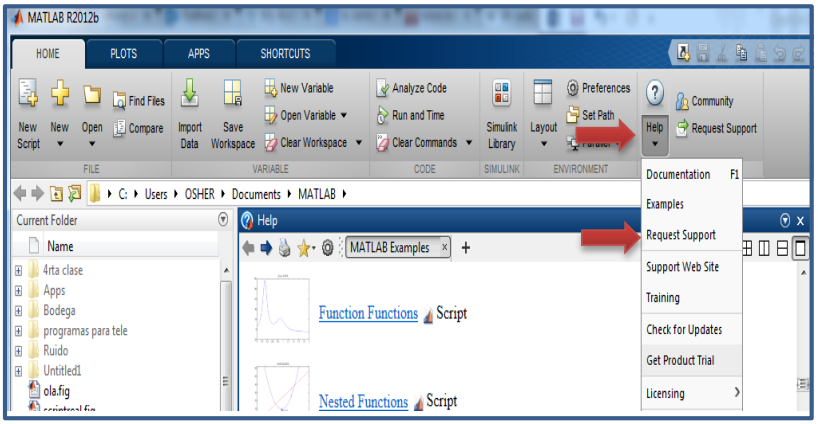


Figura 3. 13: Opciones del Help ***Matlab***

3.1.8. Tipos de datos

El tipo de datos principal para ***Matlab*** es un arreglo o matriz. Dentro de este arreglo el software ***Matlab*** soporta algunos tipos de datos secundarios. En un arreglo todos los datos deben ser del mismo tipo, sin embargo la facilidad que provee ***Matlab*** permite también hacer una combinación de tipos de datos en un mismo arreglo e incluso se puede acceder a comandos que permiten la conversión entre tipos de datos.
A continuación se listan los tipos de datos con los que se puede trabajar en ***Matlab***:[40]

- **Tipos de datos numéricos**
  Números de punto flotante precisión doble:
  Por defecto ***Matlab*** utiliza el tipo de datos de precisión doble, como sugiere el estándar **IEEE** 754, lo que quiere decir que cada variable utiliza en memoria un espacio de 8bytes para cada elemento de su matriz.[41]
  Números de punto flotante de precisión sencilla:

[39] Print Screen del programa ***Matlab***
[40] Moore, Holly. *Matlab para Ingenieros Primera edición (2007.) Pag 342-373*
[41] http://www.mathworks.com/support/sysreq/current_release/?refresh=true

Este tipo de dato viene incluido desde la versión 7 de ***Matlab***. Básicamente ocupan la mitad del espacio en memoria respecto a los datos de precisión doble y por tanto, almacenan solo la mitad de la información.
Para escribir una variable con tipo de dato "*single"* usamos la siguiente sintaxis.

```
>> D=single (5)
```

En caso de que se desee realizar el proceso contrario (*cambiar una variable con tipo de dato sencilla a una variable con tipo de dato doble*), remplazamos la función ***"single"*** por la función ***"double"***.

En realidad, en las aplicaciones que desarrollamos necesitaremos muy poco de estas funciones, puesto a que las computadoras actuales tienen gran espacio de almacenamiento y ejecutarán la mayoría de instrucciones en tiempos muy cortos.

Enteros

Una de las grandes novedades de ***Matlab*** son los muchos tipos de números enteros que contiene. Los mismos que vienen diferenciados por el tamaño en memoria que ocupan. Cuanto mayor sea el espacio de almacenamiento, más grande será el valor de número entero que podremos usar.

Para declarar una variable tipo entero utilizamos la siguiente sintaxis:

```
>>D=int8(5)
```

El comando tecleado nos devolverá una variable entera tipo entero de nombre *"D"* con valor *"5"* y un tamaño de memoria de 8 bits *(1 byte).* Podemos declarar variables tipo entero con valores de; 8, 16,32 y 64 bits *(1, 2,4 y 8 bytes respectivamente)* cambiando solamente el número que acompaña a la función **"*int*"**.

- **Tipos de datos caracter:**
  Matlab puede almacenar datos de tipo carácter, para lo cual los caracteres deben ser colocados dentro de **'apostrofes'** con esto ***Matlab*** los diferenciará de un nombre de variable. De igual manera que con los datos numéricos, cada elemento de la matriz caracter significará un espacio reservado en memoria.
- **Tipos de datos simbólicos:**
  Los datos de tipo simbólico utilizan la caja de herramientas **"*Toolbox*"** de **Matemática simbólica** a través del comando***"syms"***.
  Este tipo de datos se usan en general para hacer cálculos algebraicos.

- **Tipos de datos lógicos:**
  Matlab como otros lenguajes de programación denotan los datos lógicos como unos y ceros para significar verdadero o falso.
  Matlab reconoce perfectamente en este tipo de datos las palabras ***True (****verdadero****) False (****falso****)*** y nos devuelve los números *1* o *0* según corresponda.
- **Tipos de datos complejos:**
  Matlab trabaja sin problemas con números complejos distinguiendo perfectamente la parte real y la parte imaginaria, para lo cual en la entrada de datos podemos usar las letras ***i*** o ***j*** indistintamente para indicar la parte imaginaria. Las funciones para trabajar con variables complejas se muestran a continuación:[42]

  Definiendo una variable compleja por entrada simple.

  *>>D= 2 + 3i o >>D= 2 + 3*i*

  Definiendo una variable compleja mediante el comando *complex* esta opción nos devuelve el mismo resultado del ejemplo anterior.

  >>D=complex(2,3)

  Para separar los componentes real e imaginario de la variable declarado usamos los siguientes comandos:

  >>real(D)
  >>imag(D)

  Para obtener el conjugado de una variable compleja ya declarada usamos la función:

  >>conj(D)

  Para obtener la magnitud absoluta que supone el número complejo usamos la siguiente función:

  >>abs(D)

  Para obtener el valor del ángulo que supone el número complejo usamos la siguiente función:

  >>angle(D)

3.1.9. Programación en *Matlab*

El objetivo de este proyecto de titulación no es profundizar en aspectos de programación, dado que las aplicaciones que planteamos se basan en ingresar cierta fórmula o a su vez componentes que simulen las partes de sistemas básicos de telecomunicaciones y mostrar sus resultados. Sin embargo, mostraremos una especificación leve de los operadores,

[42] Moore, Holly. *Matlab para Ingenieros Primera edición (2007) Pg. 91-95*

sentencias condicionales, bucles y líneas de comentarios, componentes con los que conocedores de cualquier lenguaje de programación deben estar familiarizados, con el fin de mostrar la sintaxis que estos utilizan en ***Matlab*** respecto a otros lenguajes.
También en esta sección se dará una pequeña especificación de requerimientos básicos para iniciarse en ***MATLAB*.** Para empezar a trabajar en el entorno ***Matlab*** se requieren como base los siguientes conocimientos:[43]

1) ¿Cómo definir matrices?

Definir matrices en ***Matlab*** es conceptualmente sencillo, el software distingue claramente las letras mayúsculas y minúsculas, por lo que al declarar una matriz con nombre "***D***" esta será distinta de una matriz de nombre "***d***".
Una matriz se define al escribir una lista de números encerrados entre corchetes, los números pueden separarse por comas o espacios.
Cuando se define una matriz no hace falta establecer el tamaño de antemano ya que a este lo podemos cambiar posteriormente. ***Matlab*** predice el número de filas o columnas de la matriz, en función del número de elementos que introduzcamos. Al introducir los datos de una matriz, los elementos que se ingresan se colocarán por filas, separando los elementos de cada fila por medio de un **espacio** o una **coma**. Las filas por su parte se las separa usando el signo **punto y coma** (;).
Ejemplo:

>> D=[1 2 3; 4 5 6; 7 8 9]

Al presionar "Enter" se visualizaría:

$$\boldsymbol{D} \gg \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}$$

Esta es una matriz llamada "***D***" de dimensión (3x3).

2) Manipular matrices

Cuando se crea una matriz, esta aparecerá de inmediato en el ***Espacio de Trabajo***, con esto, ***Matlab*** nos indica que ya está lista para utilizarse la matriz creada y podremos hacer cualquier tipo de operación con ella. Por ejemplo podemos usarla para calcular su matriz

[43] Dolores M. Etter. *Solución de problemas de ingeniería con **MATLAB** Segunda edición (1997) Pg. 35-40*

transpuesta, para esto usamos el símbolo de **apostrofe** (‘) de la manera que se muestra:

>>B=D’

Al presionar “Enter” se visualizará:

$$\boldsymbol{B} \gg \begin{vmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{vmatrix}$$

Esto nos guardará la matriz transpuesta de la variable “***D***” en una variable llamada “***B”***.

3) Uso general de las funciones

No hace falta ser un experto en ***Matlab*** para manipular sus funciones o comandos, simplemente debemos asegurarnos de ingresarlos con la sintaxis correcta para obtener la respuesta esperada, para ello podemos acceder a la **Ayuda** del programa.

4) ¿Cómo obtener información de cada función en particular?

Para obtener la información de un comando o función debemos desplegar la **Ayuda** que nos provee el software acerca de la función. El acceso y uso fue descrito en la sección **3.1.7**.

Para relacionar y operar los distintos tipos de datos generados en ***Matlab***, se debe hacer uso de cualquiera de los siguientes operadores mencionados a continuación: *operadores relacionales, operadores aritméticos y operadores lógicos*.

3.1.9.1. Los operadores relacionales[44]

Operador relacional	Significado
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que
==	Igual a
~=	Distinto de

Tabla 3. 1: Operadores relacionales válidos en ***Matlab***

[44] Moore, Holly. *Matlab para Ingenieros Primera edición (2007) Pg. 245*

3.1.9.2. Operadores Aritméticos [45]

Operador aritmético	Significado
+	Suma
-	Resta
*	Multiplicación
.*	Multiplicación elemento por elemento
/	División
./	División elemento por elemento

Tabla 3. 2: Operadores Aritméticos

3.1.9.3. Operadores Lógicos [44]

Operadores lógicos	Significado
&	Y (AND)
|	O (OR)
~	NOT (NO)

Tabla 3. 3: Operadores Lógicos

3.1.9.4. Sentencias condicionales [46]

Una sentencia condicional en ***Matlab*** es un grupo de instrucciones que se ejecutarán siempre y cuando cumplan con cierto requisito, o por el contrario serán omitidas en caso de no cumplir con los mismos. La primera sentencia condicional es la sentencia "*if* "la cual se puede utilizar en estructuras del tipo ***if-end***; ***if-else-end***; ***if-else-if-else-end***.

Las siguientes figuras presentan los diagramas de flujo de todas las estructuras que presenta la sentencia condicional mencionada.

Con los diagramas de flujo mostrados, se logra apreciar como se distribuye el flujo de instrucciones generando condiciones, para controlar la ejecución de las mismas.

[45] Amos Gilat. ***Matlab*** *una introducción con ejemplos prácticos Segunda edición (2005) Pg. 58*

[46] Amos Gilat. ***Matlab*** *una introducción con ejemplos prácticos Segunda edición (2005) Pg. 168-172*

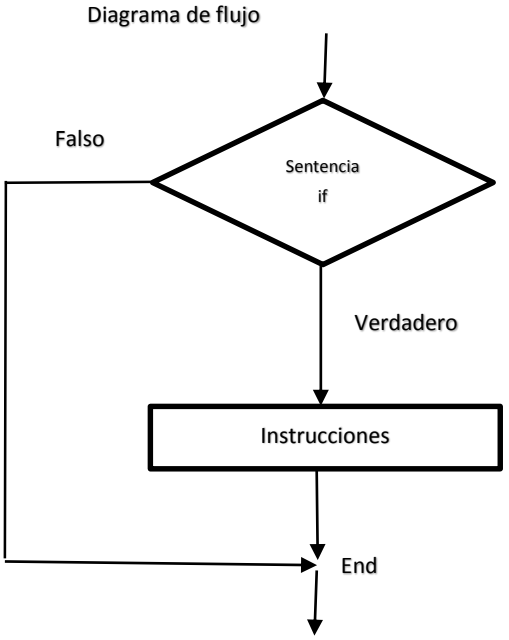


Figura 3. 14: Diagrama de flujo de la estructura “if-end”

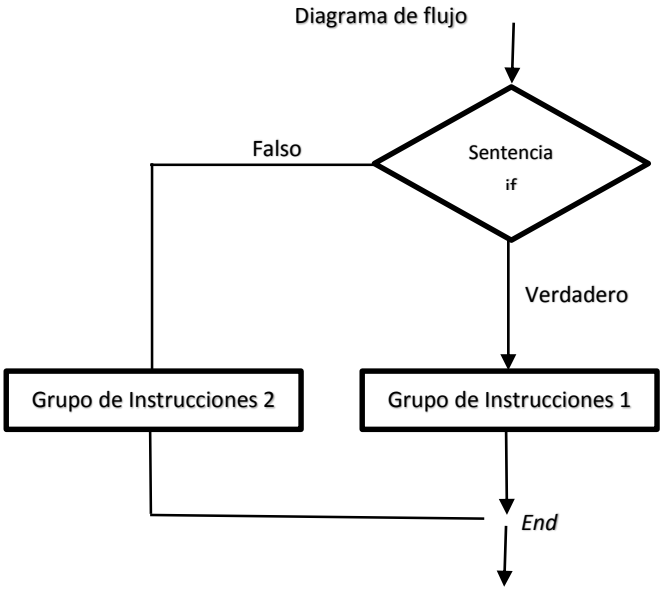


Figura 3.15: Diagrama de flujo de la estructura “if-else-end”

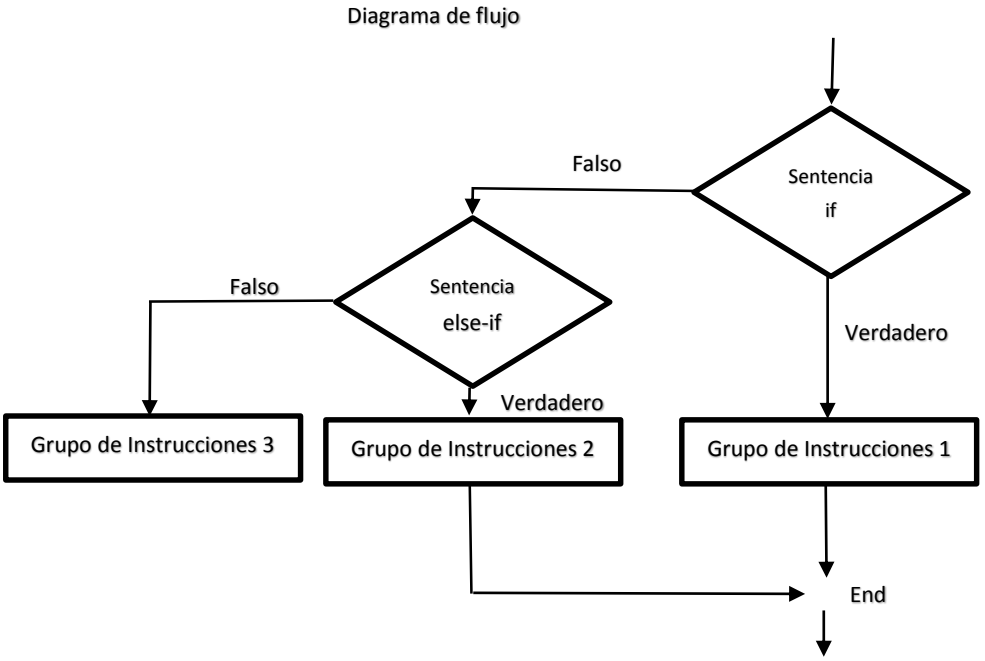


Figura 3.16: Diagrama de flujo de la estructura "if-else-if-else-end"

Otra sentencia de control de flujo conocida es el Switch-Case, la cual permite diseñar una estructura que agrupa múltiples instrucciones repartidas en un grupo de opciones, mismas que serán escogidas de acuerdo a la necesidad planteada para ejecutarse.[47]

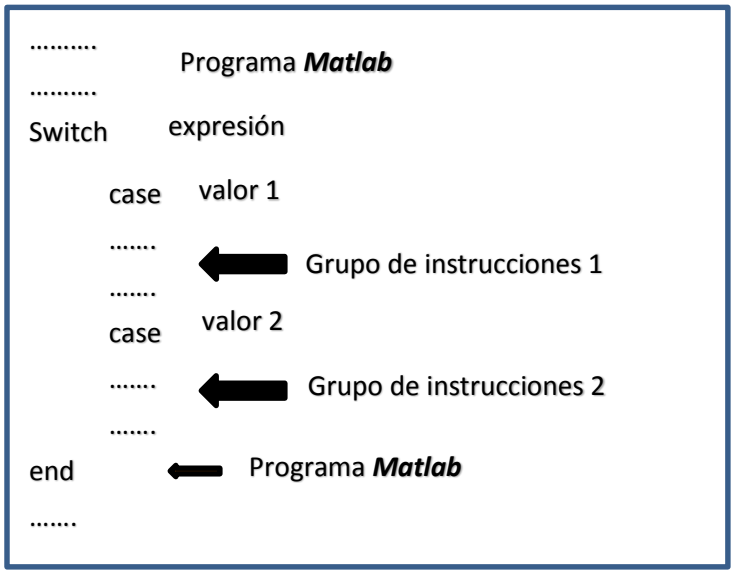


Figura 3. 17: Estructura Switch-case

[47] Amos Gilat. ***Matlab*** *una introducción con ejemplos prácticos Segunda edición (2005)* Pg.173

3.1.9.5. Bucles

Los bucles son también interacciones que controlan el flujo de datos del programa. Estos repiten un grupo de sentencias varias veces de forma consecutiva.
En cada paso se ejecutan grupos de instrucciones que nos servirán para controlar la ejecución de las líneas de programación creadas, de esto se crean 2 grupos de Bucles comunes en casi todos los lenguajes de programación, por lo que son descritos para que se vea la diferencia de la declaración de los mismos dentro de ***Matlab***. Estos bucles son conocidos como ***for-end*** y ***while-end***.[48]

Bucle *for-end*

Este tipo de bucle permite definir desde el inicio, el número de veces que se ejecutarán el grupo de instrucciones.
La estructura de este bucle se muestra en la figura 3.10.

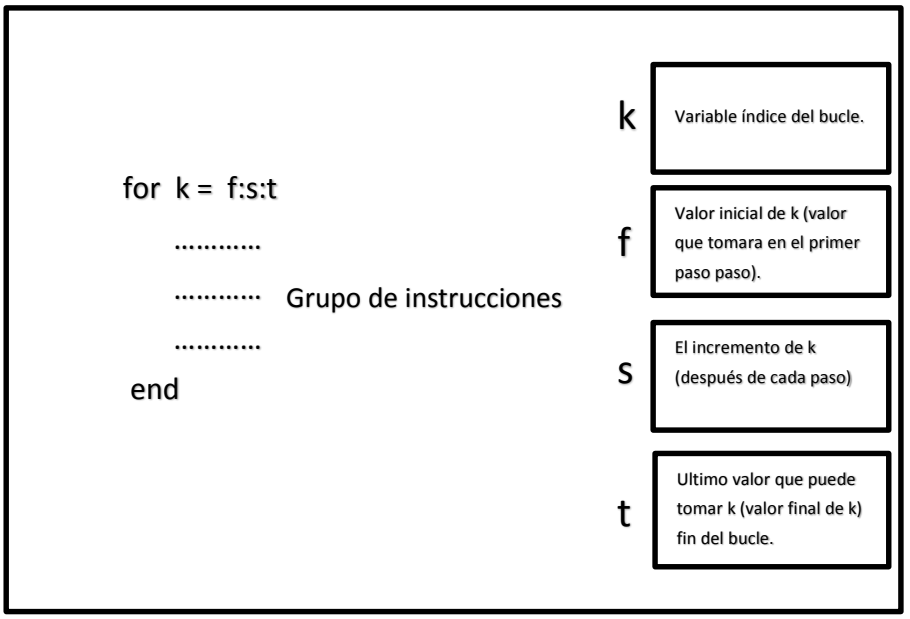


Figura 3. 18: Estructura for-end

Bucle while-end

Este tipo de bucle se utiliza cuando somos desconocedores del número de interacciones que se desean realizar; por lo tanto, las instrucciones se ejecutarán mientras se satisfaga una condición determinada.

[48] Amos Gilat. ***Matlab*** *una introducción con ejemplos prácticos Segunda edición (2005) Pg. 176-180*

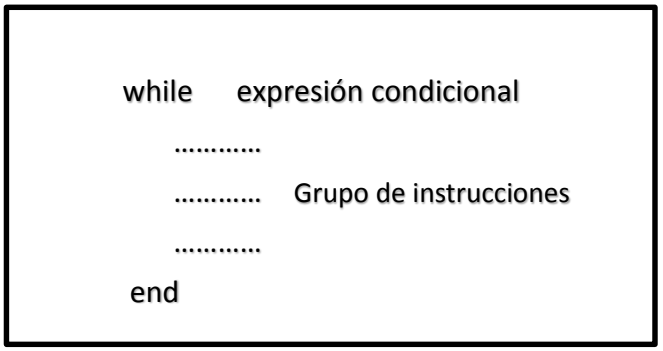


Figura 3. 19: Estructura while-end

3.1.9.6. Líneas de comentarios

Los comentarios en ***Matlab*** se pueden realizar de manera individual o de manera grupal. Los comentarios son útiles para especificar la función de cada línea de comandos o incluso especificar acerca de todo el programa escrito. Este procedimiento en general se usa para ese tipo de programas que requieren seguir una línea de desarrollo más detallada, puesto a que una vez terminados es mucho más difícil saber o recordar que función tiene cada línea de su código.
Las maneras de hacer un comentario en ***Matlab*** es seleccionando las líneas que se desean como comentarios y presionar clic derecho seleccionando después la opción *comment*. La segunda manera de hacer un comentario, es anteponer el símbolo *"%"* previo al texto que especificara la línea de comandos seleccionada, con esto ***Matlab*** reconoce que lo que se hizo fue un comentario y omite estos caracteres en el momento de ejecución de cada línea de comandos.[49]

3.1.10. Ficheros *.m [50]

Estos ficheros en realidad son el centro de programación de ***Matlab***. Se pueden realizar y modificar en un editor de texto cualquiera, después se puede comprobar la validez de su sintaxis mediante el depurador del programa (**Debugger**). Sin embargo, es mucho más razonable usar el propio editor de textos que trae ***Matlab***, mismo que a su vez es **Debugger**.

La extensión *.m es utilizada para crear **ficheros de comandos** denominados "***Scripts"*** en inglés, y también para la creación de **ficheros de función**.

---

[49] Moore, Holly. *Matlab para Ingenieros Primera edición (2007) Pg. 122*
[50] Amos Gilat. ***Matlab*** *una introducción con ejemplos prácticos Segunda edición (2005) Cap. 4 y 6*

1. Los **ficheros de comandos** contienen un conjunto de instrucciones ***Matlab*** que se ejecutan sucesivamente cuando estos son llamados en la línea de comandos del ***Command Window***, tecleando simplemente el nombre con el que fueron creados. Las variables que contengan estos ficheros serán guardadas en el espacio de trabajo de la sesión ***Matlab*** para cuando este haya terminado de ejecutarse.
   La programación en ***Matlab*** se construye alrededor de funciones, estas ayudan a realizar una programación más eficiente evitando escribir el código de programación en cálculos que se realizan de manera frecuente como por ejemplo el cálculo del seno de un ángulo. ***Matlab*** permite que el usuario cree sus propias funciones de manera análoga a las funciones que el software ya trae integrado, es decir; con su nombre, sus argumentos y por supuesto sus valores de retorno.
   Los ficheros *.m que definen funciones ayudan a extender las posibilidades de ***Matlab***, de hecho hay librerías basadas en **ficheros de funciones** creadas por usuarios que se venden o se distribuyen gratuitamente por internet, dentro del mundo de la programación en ***Matlab*** se los conoce como ***(toolkits)***.

2. Los **ficheros de función** en ***Matlab*** se distinguen en una línea, que define que se trata de la creación de una función de usuario, para lo cual se debe usar el siguiente criterio:

   a. Se debe empezar con la palabra ***function***

   b. Seguido de un espacio, se define una variable que será la salida de la función. En caso de utilizar más variables para la salida, estas deben ser colocadas dentro de corchetes ejemplo: "[**variable**1, **variable2**,….**variable***n*]", separadas entre ellas por una coma.

   c. Se le asigna un nombre cualquiera a la función

   d. Se coloca una variable que sirva como argumento de entrada para el usuario, esta variable debe estar dentro de paréntesis y de manera análoga que para las variables de salida, en caso de que se vaya a utilizar más de una variable como argumentos de entrada, estas deben estar separadas entre sí por una coma ejemplo:

   function [y,x,r]=***ejemplo*** [X,Y;R]

Donde hemos declarado una función llamada ***ejemplo*** con variables de entrada “Y,X,R” y obtendremos las salidas en las variables “y,x,r” respectivamente.

Las variables que se definen dentro de los ***ficheros de función*** .*m son conocidas como variables locales, es decir que estas no pueden acceder a las variables creadas dentro del espacio de trabajo ***Matlab*** ni tampoco se puede realizar el proceso en viceversa. Esto evita colisión en el espacio de trabajo es decir, que se puede definir libremente variables con el mismo nombre que contenga el ***fichero de función*** sin problema a no ser, que las variables contenidas en una función hayan sido definidas como variables globales las cuales vendrían a ser las variables de entrada de la función.

3.1.11. Ficheros de comandos (Scripts)

Representan otra forma de ingresar comandos ***Matlab*** usando exactamente la misma nomenclatura que en el espacio interactivo ***Command Window***. Sin embargo, el crear estos ficheros supone una cierta ventaja, sobre todo en programas donde el número de sentencias es demasiado elevado, ya que los comandos que se encuentren guardados dentro de estos pueden ser llamados sucesivamente para ejecutarse, y además, omiten la necesidad de presionar la tecla **Intro** para cada comando. Cuando se ejecutan las instrucciones contenidas en un **fichero de comandos** las variables que se crean se guardan en la sesión ***Matlab*** y se las puede manipular desde el **Command Window** según la necesidad que se tenga.

Es recomendable que las sentencias establecidas dentro de estos ficheros culminen en punto y coma (;), ya que esto evita que ***Matlab*** nos devuelva los resultados de las operaciones escritas en cada línea, de manera que en la salida solamente tendremos el resultado que en realidad nos interesa.[50]

3.1.12. Librerías

Las librerías incluidas en ***Matlab*** reciben el nombre de **“*Toolboxes”***, estas fueron creadas con el fin de resolver problemas muy puntuales y específicos. Existe una gran variedad de librerías útiles para varias ramas que apoyan sus bases en la matemática, sin embargo, para este proyecto enlistaremos solamente algunas de las librerías que pueden encontrarse en ***Matlab***, enfocando de manera específica las librerías útiles para el modelado de las partes de sistemas básicos de telecomunicaciones. Antes de escribir un código ***Matlab*** orientado a resolver cierto problema, es necesario saber si el software que instalamos en nuestro computador contiene el *toolbox* que

puede manejar y aceptar los comandos que resolverán la dificultad planteada.

Para saber cuáles son las librerías con las que contamos se ingresa en el *prompt "*>>*"*; el comando ***"ver"***, con lo que ***Matlab*** nos despliega una lista de las librerías con las que podemos trabajar. Por ejemplo si lo que deseamos es diseñar una aplicación para el procesamiento de imágenes, entonces necesitaremos la librería "*Image Processing Toolbox"*. En caso de ingresar algún comando perteneciente a una librería con la cual no contamos, ***Matlab*** nos devolverá un error, indicando que la función (*comando)* que ingresamos no se encuentra definida.

Las librerías útiles para simular el comportamiento de partes de un sistema de telecomunicaciones son las siguientes:

- ***Simulink*.-** Es una extensión de ***Matlab*** que proporciona un entorno gráfico para crear diagramas en bloques de un sistema y después ajustar los parámetros de cada bloque para proceder a la simulación del mismo.
- **DSP (digital signal processing) system toolbox.-** Esta es una extensa librería que provee herramientas para el diseño, análisis e implementación de filtros, en los que se incluyen los de características pasa banda, pasa altos y pasa bajos. Esta librería proporcionará herramientas de diseño que nos conducen a las respuestas necesarias que un filtro debe tener para generar un algoritmo de recuperación de la señal. Esta librería incluye también la utilidad de visualizar una señal en dominio de la frecuencia mediante los métodos de análisis de Fourier. Además, el **DSP system toolbox** otorga la facilidad de crear formas de onda incluyendo ondas de carácter sinusoidal, rectangular e incluso la generación de señales aleatorias.
- **Communications system toolbox.-** Esta librería provee herramientas específicas para el análisis, diseño e implementación de sistemas de telecomunicaciones. Incluye bloques entre los que destacan: moduladores, demoduladores, canal de comunicación, ruido gaussiano, filtros, entre otros. Esta librería ayuda a la libre implementación de partes de sistemas de telecomunicaciones, con lo que se permite ver su comportamiento y obtener la respuesta de cada una de las partes que conforman estos sistemas. En general las funciones incluidas en esta librería las podemos aplicar desde ***Simulink***.[51]

[51] Ayuda del Software ***Matlab*** sección: Documentación

3.1.13. Ayuda (Help) para funciones de usuario

Cuando se crea una función de usuario es decir un archivo de extensión *.m, ***Matlab*** permite crear un help para esta función de manera análoga al help de funciones que trae el mismo software. Esto se consigue gracias a las líneas que empiezan con el signo "*%*" es decir, las líneas destinadas a ser un comentario, al escribir en el *prompt:*

>> help "Nombre_de_la_funcion_del_usuario"

Matlab nos mostrará el contenido de los comentarios, mismos que nos servirán como ayuda para identificar que trabajo realiza la función. ***Matlab*** reconoce como ayuda solamente las primeras líneas de comentarios, ya sea que estas se encuentren una línea antes de la declaración de la función o una línea después de la declaración de la misma.[38]

3.1.14. Ayuda (Help) de directorios

El help de directorios ***Matlab*** nos permite acceder a una ayuda general de todas las funciones contenidas en un directorio específico, el directorio que usamos por defecto para las funciones que ***Matlab*** trae integrado es el *directorio local*, con lo que para tener una ayuda de las funciones que este directorio contiene hacemos lo siguiente: ingresamos en el *prompt:* >> ***help local***; con lo que ***Matlab*** nos devolverá las funciones contenidas en dicho directorio. Las ayudas de directorios están contenidas en ficheros llamados ***contents.m***.

Para crear nuestro propio directorio de ayuda de funciones, debemos crear un archivo ***contents.m***, los pasos a seguir para lograrlo, los podemos encontrar en la Ventana de ayuda o ***Help*** de ***Matlab*** como: "Create Help Summary Files".[38]

3.1.15. Gráficos bidimensionales

Los gráficos bidimensionales son los de mayor utilidad cuando se trata de análisis de funciones trigonométricas que representan a ciertos tipos de señales presentes en sistemas básicos de telecomunicaciones, las señales se pueden representar en un espacio bidimensional con un conjunto de pares ordenados *x-y*; para identificar los puntos de la gráfica. [52]

Los comandos ***Matlab*** para gráficar vectores, permiten darle una apariencia personalizada al resultado que se mostrará en pantalla, es decir; podemos cambiar el color y grosor de la línea, añadir líneas de referencia y cuadrículas,

[52] Moore, Holly. *Matlab para Ingenieros Primera edición (2007) Cap. 6*

agregar títulos y comentarios. ***Matlab*** también nos permite superponer varios gráficos en una misma ventana.[53]

3.1.16. Funciones Trigonométricas elementales

Dentro del ámbito de las telecomunicaciones varias de las señales presentes son de característica sinusoidal. ***Matlab*** permite trabajar con las funciones trigonométricas elementales, creando vectores de uno o varios valores que representan los puntos sobre los cuales actúa la función trigonométrica, con lo cual, cada valor será tratado como si se fuese de un escalar. La evaluación de cada punto sirve para graficar su valor, o para obtener el resultado en magnitud de cada uno de ellos. Las funciones con las que se puede trabajar son las siguientes: [37]

- sin(x).- Calcula el seno del vector x en radianes
- cos(x).- Calcula el coseno del vector x en radianes
- tan(x).- Calcula la tangente del vector x en radianes
- sind(x).- Calcula el seno del vector x en grados
- cosd(x).- Calcula el coseno del vector x en grados
- tand(x).- Calcula la tangente del vector x en grados
- sec(x).- Calcula la secante del vector x en grados
- csc(x).- Calcula la cosecante del vector x en grados
- atan(x).- Calcula el arcotangente del vector x en grados

3.1.17. Función Plot

La función o comando ***plot*** es justamente la que nos ayuda a crear gráficos bidimensionales, para lo cual, el requisito indispensable es crear vectores unidimensionales y agruparlos en pares para poderlos gráficar. Los vectores con los que se va a trabajar, deben contener el mismo número de elementos en su espacio unidimensional. Una vez que se ejecuta el comando ***plot*** de manera correcta; se abre una nueva ventana con el resultado esperado, a esta ventana se la conoce en el ámbito ***Matlab*** con el nombre de ***"Ventana de Gráficos"***.

La sintaxis para este comando es la que se muestra a continuación:

>> plot (Vector1, Vector2)

El gráfico que nos devuelve corresponde a una curva, en la que el *Vector1* corresponde al eje de abscisas o eje horizontal, mientras que el *Vector2* corresponde a las ordenadas o eje vertical.[52]

[53] Amos Gilat. ***Matlab*** *una introducción con ejemplos prácticos Segunda edición (2005) Cap.5*

3.1.17.1. Función title

La función *title* coloca una cadena de caracteres en el centro de la parte superior de la gráfica cuando utilizamos la siguiente sintaxis:[52]

>> title('Título escogido por el usuario')

El resultado se muestra en la figura 3.20.[54]
Recordar que cuando se trabaja con caracteres, estos deben estar ubicados dentro de apostrofes.

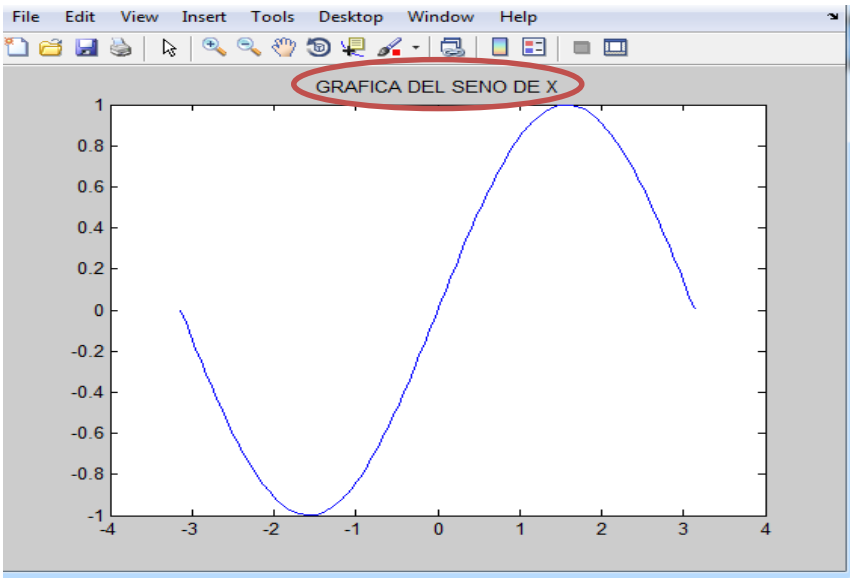


Figura 3. 20: Comando *"title"*

3.1.17.2. Funciones XLABEL-YLABEL

El uso de estos comandos es similar al que tiene la función "*title*", solo que esta vez, el título se colocará en el eje de las *"x"* cuando usemos XLABEL y en el eje de las *"y"* cuando usemos YLABEL.
La sintaxis que manejamos es similar a la del comando *title:*[52]

>> xlabel ('comentario para x')
>> ylabel ('comentario para y')

En la figura 3.21 se ilustra el resultado de la ejecución de los comandos descritos.[55]

[54] Print Screen del programa ***Matlab***
[55] Print Screen del programa ***Matlab***

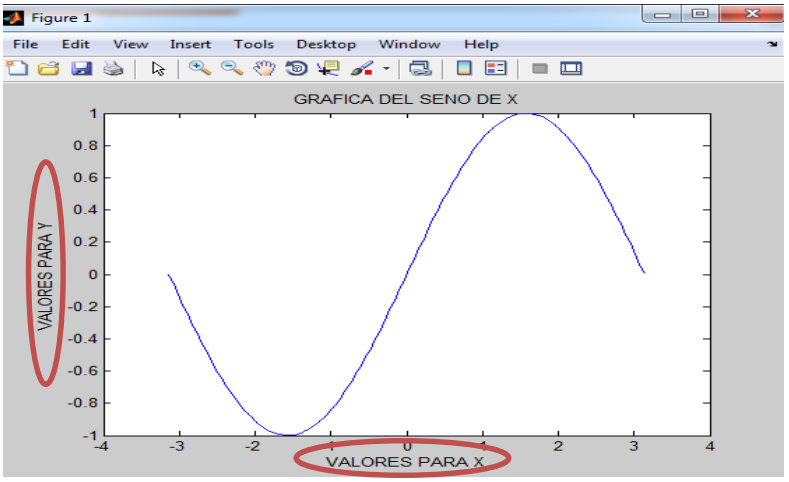


Figura 3. 21: *Comandos* "XLABEL-YLABEL"

3.1.17.3. Función Grid (cuadrícula).

Esta función de ***Matlab***, ayuda a crear una cuadrícula. En la reproducción de nuestra gráfica la sintaxis que usamos es la que sigue:[52]

>> grid on

El resultado que nos devuelve es una gráfica de apariencia más profesional. En la figura 3.22 se ilustra el resultado del comando mostrado.[56]

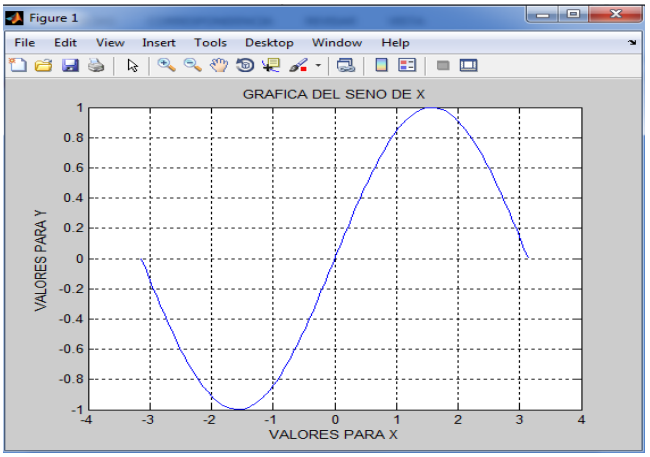


Figura 3. 22: *Comando* "Grid"

3.1.17.4. Superposición de gráficas en una misma ventana

Cuando se generan gráficas suele ser común comparar la influencia que tendría si se alteraran algunos de sus parámetros sean estos: amplitud, fase o frecuencia. De esta manera, se puede observar y entender los cambios que atraviesa la señal en dichos procesos. El programa ***Matlab*** nos permite la

[56] Print Screen del programa ***Matlab***

superposición de gráficas mediante el comando **"hold on"**[52]. El siguiente código ilustra lo descrito anteriormente:[57]

```
>>x= linspace(-pi,pi,100)
>>y1= sin(x), plot(x,y1)
>>hold on
>>y2= cos(x), plot(x,y2)
```

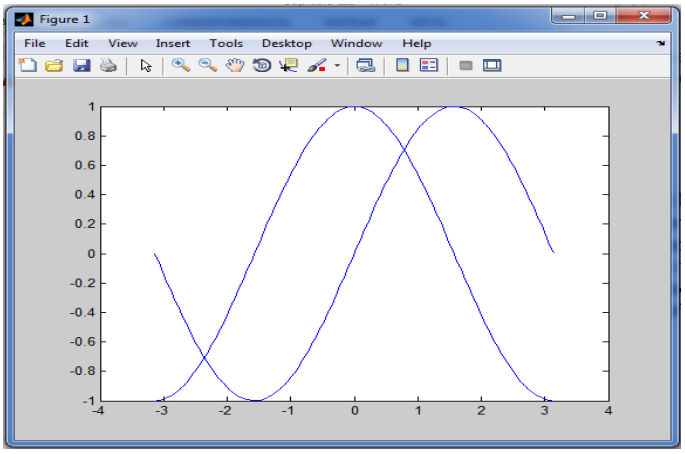


Figura 3. 23: Superposición de gráficas

3.1.18. Estilos de línea y marcadores de la Función Plot

En esta sección nos referimos a la forma en que el software ***Matlab*** nos permite personalizar la apariencia de nuestras gráficas. Cuando se muestra el resultado de una gráfica sin hacer personalizaciones, usamos prácticamente las opciones que ***Matlab*** trae integrado de forma estándar, de manera que la manipulación de la ***"Ventana de gráficos"*** es opcional, por esta razón el resultado esperado entre las operaciones de vectores no es influenciado por nuestras preferencias de personalización. Para modificar los parámetros de impresión de gráficas usamos la siguiente sintaxis en la función *plot:*

>> plot (Vector1,Vector2,'especificador de línea','marcadores','valor de los marcadores')

El especificador de línea puede contener hasta 3 datos dentro de los apostrofes, estos datos determinarán el color y estilo de línea, además del marcador de los puntos de la gráfica.

Las opciones para el **especificador de línea** se muestran a continuación en la siguiente tabla:[58]

[57] Print Screen del programa ***Matlab***

[58] http://www.sc.ehu.es/sbweb/energias-renovables/***MATLAB***/basico/gráficos/gráficos.html

Color	Símbolo	Estilo de línea	Símbolo
Azul (defecto)	b	Sólido(defecto)	-
Verde	g	A puntos	:
Rojo	r	raya-punto	-.
Cian	c	Línea entrecortada	--
Magenta	m		
Amarillo	y		
Negro	k		
Blanco	w		

Tabla 3. 4: Especificadores de línea en la función plot

Un ejemplo de lo descrito se ilustra en la figura 3.24:[59]

>> plot(x,y,'--g')

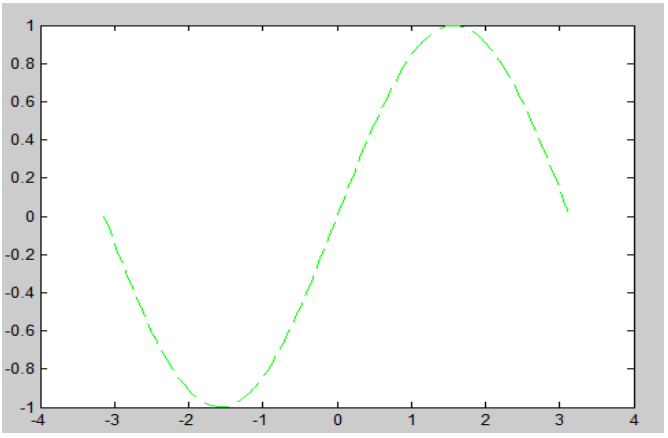


Figura 3. 24: Especificadores de línea en la función plot

En el ejemplo x e y son los vectores, se utiliza línea entrecortada y color verde. Nótese que toda nuestra personalización para el **especificador de línea** se coloca dentro de apostrofes, por otro lado si se quiere utilizar la línea solida (por defecto), no es necesario colocar su símbolo, basta solamente colocar entre los apostrofes el símbolo del color con el que queremos que la línea aparezca.

Las opciones de símbolos para marcar los puntos de gráficas se muestran en la siguiente tabla:[57]

[59] Print Screen del programa ***Matlab***

o	círculo	v	Triángulo (hacia abajo)
.	punto	X	Triángulo (hacia arriba)
x	marca X	<	Triángulo (hacia la izquierda)
+	más	>	Triángulo (hacia la derecha)
*	estrella	p	pentagrama
s	cuadrado	h	hexagrama
d	diamante		

Tabla 3. 5: Marcadores de línea en la función plot

Las especificaciones que podemos usar para los **marcadores** de línea se manejan con las siguientes palabras clave:

- **linewidth:** especifica la anchura de línea, cuyo valor por defecto es: *0.5*
- **markersize:** especifica el tamaño con el que aparecerá el símbolo que marca los puntos
- **markeredgecolor:** especifica el color de borde del símbolo que marca los puntos
- **markerfacecolor:** especifica el color de relleno del símbolo que marca los puntos

Ejemplo:

```
>> plot(x,y,'--g>','linewidth',2,'markersize',8)
```

En el ejemplo x e y son los vectores, se utiliza color verde y línea entrecortada para especificadores de línea y también el símbolo de triangulo hacia la derecha para marcar los puntos en la gráfica, la figura 3.25 muestra la ejecución del comando. [60]

[60] Print Screen del programa ***Matlab***

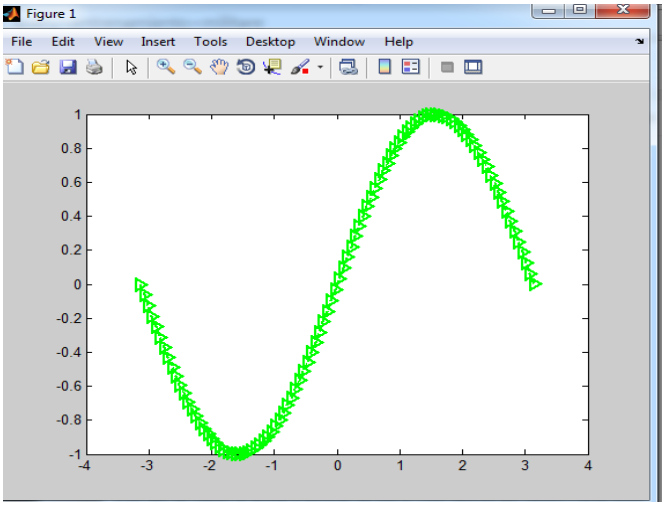


Figura 3. 25: Marcadores de línea en la función *plot*

En la gráfica se logra apreciar un color de línea más fuerte y definido gracias a los marcadores de línea, pese a esto también se ve como la gráfica escapa de los límites de la ventana de gráficos, para especificar y ajustar los límites de la ventana de gráficos existen los siguientes comandos:

'xlim' para el eje *x*
'ylim' para el eje *y*

Para ajustar los valores del eje en este ejemplo se utiliza:

```
>> xlim([-pi pi])
>> ylim([-2 2])
```

El resultado es el que se ilustra en la figura 3.26:[61]

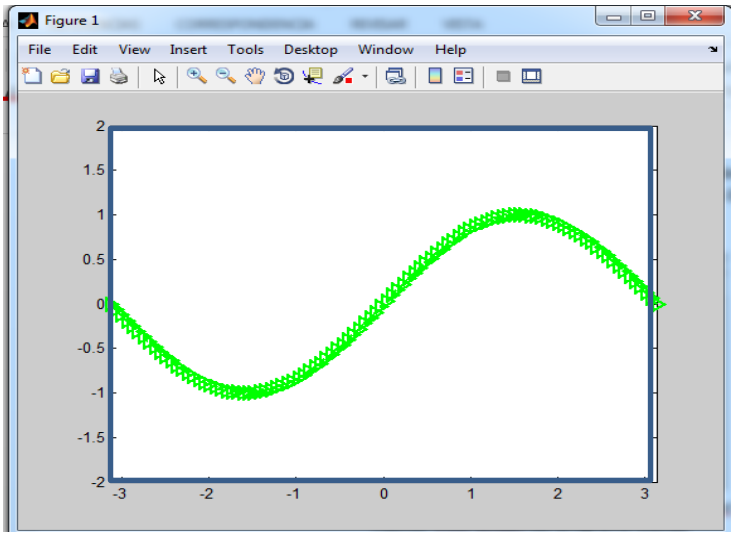


Figura 3. 26: Ajuste de los límites de la ventana de gráficos

[61] Print Screen del programa ***Matlab***

Observar que los límites tanto del eje *x* como del eje *y* cambian con la ejecución de los comandos mostrados se ha conseguido ajustar el resultado de la gráfica a la ventana.

En telecomunicaciones se requiere no solo la gráfica de señales análogas, también necesitamos gráficar señales discretas, ya sea para la representación del espectro de frecuencias de una señal o para otros fines. ***Matlab*** nos proporciona la función *stem* para realizar gráficar discretas. Esta función genera un conjunto de tallos que se unen con el eje de las abscisas su sintaxis es la siguiente:

>>stem (x,y)

Si se utilizan las mismas variables que para el ejemplo anterior el resultado que se devuelve es el que se muestra en la figura 3.19.[62]

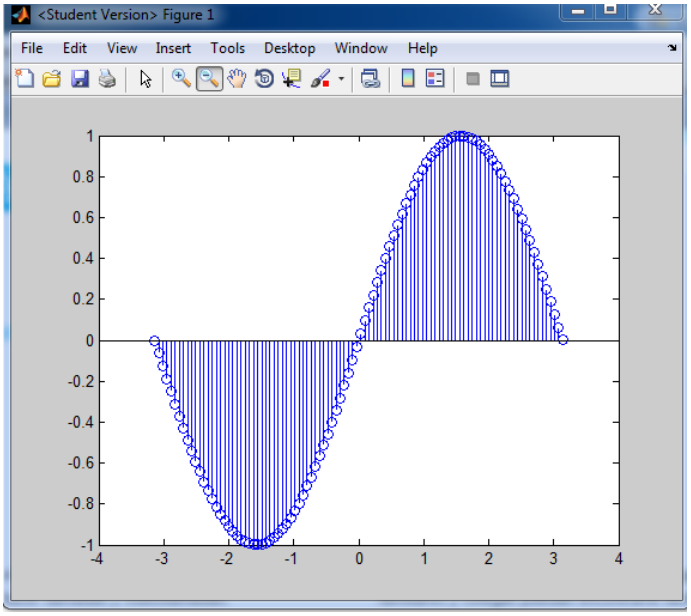


Figura 3. 27: Gráfico función Stem

3.1.19. Introducción a *Simulink*

Simulink es una extensión de ***Matlab*** la cual proporciona un entorno de simulación para sistemas dinámicos lineales y no lineales. Definiendo a la simulación como el tiempo que un computador tarda en resolver un grupo de ecuaciones determinado. Por su dinamismo y velocidad, actualmente esta herramienta se ha convertido en un estándar de simulación en varios ámbitos dentro de las industrias. ***Simulink*** permite representar los sistemas mediante diagramas de bloques, de los cuales, cada uno cumple una función

[62] Print Screen del programa ***Matlab***

específica. Básicamente este programa permite colocar diferentes bloques de funciones dibujando un sistema y observar el resultado que nos devuelve. Tanto ***Matlab*** como ***Simulink*** ayudan a analizar, simular y revisar los procesos que suceden dentro de un sistema. ***Simulink*** trae todos los procesos integrados dentro de bloques, por ejemplo, el algoritmo para generar una señal AM requiere varias líneas de comandos en ***Matlab*** mientras que este mismo algoritmo en ***Simulink***, lo podemos realizar usando unos cuantos bloques.[63]

3.1.20. Descripción y acceso

Para acceder a ***Simulink*** se conocen dos métodos, cualquiera de ellos devuelve el mismo resultado. La primera es teclear en la ventana de comandos de ***Matlab*** la palabra "***Simulink***", mientras que la segunda forma de entrar, es utilizando la pestaña ***"Simulink Library"*** que se encuentra en el escritorio de ***Matlab*** anclado en la barra de herramientas.

Una vez que arranca ***Simulink*** se nos muestra una ventana, cuyo lado izquierdo contiene toda la librería de bloques que nuestra versión de ***Matlab*** soporta, mismos que; tienen mucho que ver con la lista de componentes que hayamos escogido al instalar el software. Para ilustrar lo descrito refiérase a la figura 3.28.[64]

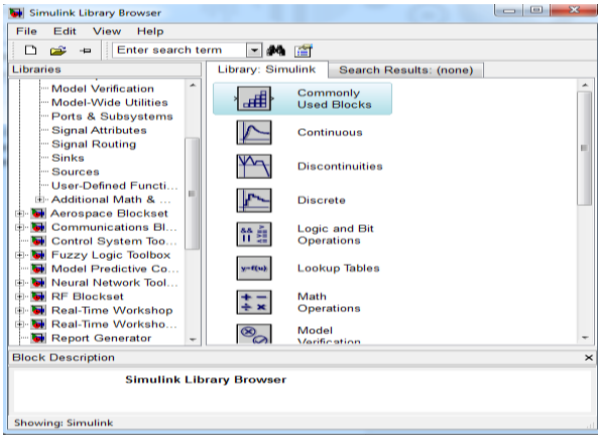


Figura 3. 28: Librería de bloques de función ***Simulink***

Para comenzar a crear un modelo de sistema en ***Simulink*** debemos abrir el área de trabajo del mismo, de manera que podamos dibujar mediante bloques el sistema buscado y observar los resultados, para lograrlo debemos dar ***clic*** en la pestaña **File** opción **New** y a continuación la opción **Model**. Lo mencionado se ilustra según la siguiente figura:[65]

---

[63] Ayuda del software ***Simulink*** sección: Descripción del Producto
[64] Print Screen del programa ***Matlab/Simulink***
[65] Print Screen del programa ***Matlab/Simulink***

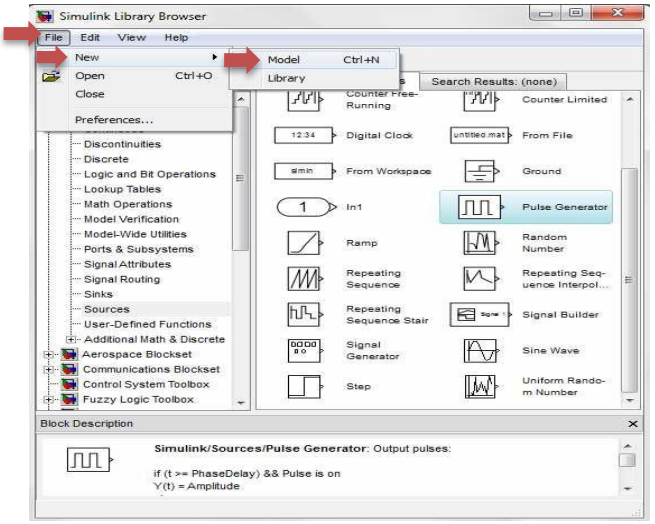


Figura 3. 29: Creando un nuevo modelo ***Simulink***

Inmediatamente se abrirá una nueva ventana conocida como el **Work-space** de ***Simulink***, en esta sección dibujaremos a libertad nuestro modelo de sistema para después observar sus resultados de manera inmediata. Lo descrito lo podemos apreciar en la figura 3.30.[66]

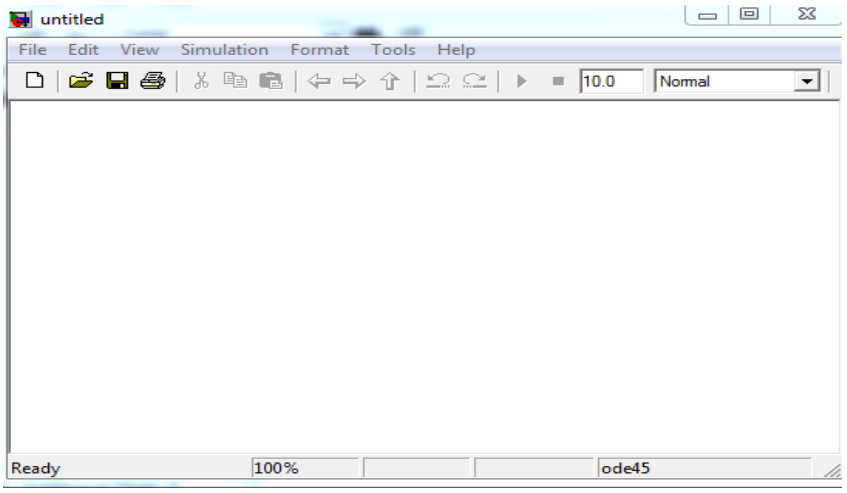


Figura 3. 30: Work-Space ***Simulink***

Para simular nuestro modelo de sistema, es importante tomar el tiempo que tardará el programa ***Simulink*** en resolver las ecuaciones que representan cada uno de los bloques. El tiempo de simulación dependerá de la complejidad y precisión del modelo dibujado, es decir en sistemas más complejos el tiempo de simulación deberá ser más amplio para lograr obtener resultados satisfactorios, por defecto el tiempo de simulación nos viene marcado en segundos **(10.0.)**

[66] Print Screen del programa ***Matlab/Simulink***

Para dar inicio a la simulación una vez terminado el modelo presionamos el icono “run”.[67]

3.1.21. Construcción de modelos en *Simulink*

Para empezar a dibujar en el **Work-space** de ***Simulink***, debemos seleccionar una librería de la ventana principal de ***Simulink***, luego seleccionamos el bloque que queremos extraer de dicha librería e inmediatamente se lo arrastra hacia el **Work-space** del ***Simulink***.

Si se conoce el nombre del bloque, entonces se puede acceder a él de manera inmediata usando el buscador de la ventana de ***Simulink*.** De igual manera que con las funciones de ***Matlab***, los bloques de ***Simulink*** contienen una extensa línea de ayuda a la que podemos ingresar para enterarnos propiamente de la función que realiza, para lo cual presionamos “*clic derecho”* sobre el bloque y se nos desplegará una barra de opciones, de estas seleccionamos la opción “*help”.* Inmediatamente se mostrará el contenido de ayuda para el correcto uso del bloque seleccionado, en la información destaca la descripción y parámetros de configuración del mismo.[66]

La figura 3.31 muestra como seleccionar una librería.[68]

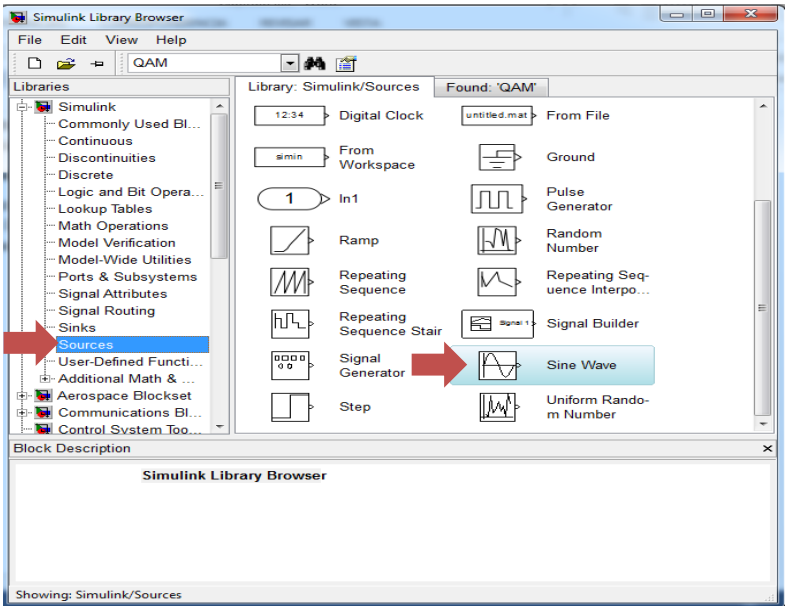


Figura 3. 31: Selección de una librería

[67] Ayuda del software ***Simulink*** sección: Tutoriales
[68] Print Screen del programa ***Matlab/Simulink***

Cuando se ha arrastrado un bloque al área de trabajo a este se lo puede conectar con otros bloques para dibujar nuestro diagrama. Para esto, lo que primero debemos hacer es identificar si es un bloque de entrada, salida o ambas. Una vez identificado que tipo de bloque es el que escogimos, podemos conectarlo con otro haciendo lo siguiente: **1)** posicionar el puntero del mouse sobre el puerto de entrada o salida del bloque, **2)** mantener presionado el botón del ratón, **3)** desplazar el puntero hacia el puerto de entrada del siguiente bloque, **4)** soltar el botón del ratón y tendremos los bloques conectados.

Simulink dibuja una línea para identificar que se han conectado los bloques. La línea de conexión usa una flecha indicando la dirección del flujo de señal, los bloques se pueden desplazar para acomodarlos de acuerdo a la estética que busquemos darle al sistema que estamos dibujando.

También se pueden cambiar los nombres que por defecto muestran los bloques, esto es útil en sistemas en los que se modela estructuras bastante complejas y en las que necesitamos recordar que función realiza cada bloque.[66]

La figura 3.32 muestra el resultado de dos bloques conectados.[69]

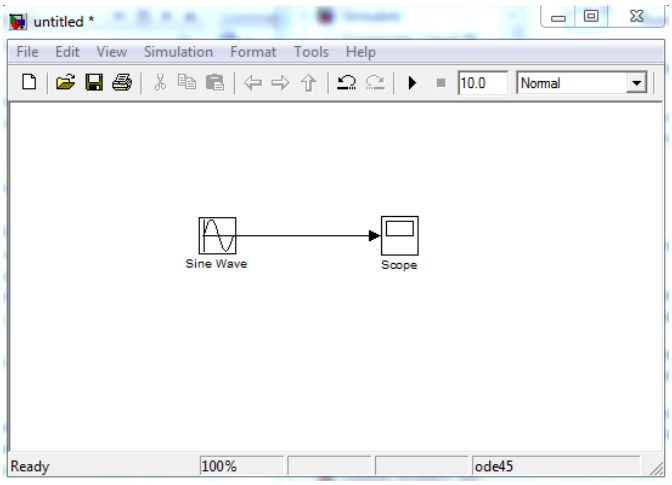


Figura 3. 32: Conectando bloques en el Work-space ***Simulink***

3.1.22. Librerías referenciales de *Simulink*

Simulink nos presenta entre sus librerías un extenso contenido de bloques donde podemos elegir aquellos que nos ayuden a simular partes de un sistema básico de telecomunicaciones. Estas se nombran a continuación:[70]

[69] Print Screen del programa ***Matlab/Simulink***
[70] Ayuda del software ***Simulink*** sección: Bloques

3.1.22.1 Math Operations

Contiene operadores matemáticos que ayudan rápidamente a realizar operaciones entre funciones de matrices, funciones de vectores y funciones complejas. Por ejemplo, en telecomunicaciones es importante conocer acerca de los efectos de la suma y multiplicación entre ellas. Conocer este fenómeno es mucho más sencillo si se puede observar el espectro que otorgaría el resultado de dichas operaciones entre señales; y gracias a esta librería podremos experimentar la interacción de sumar o multiplicar señales de diferentes fuentes.

Suma[71]

Figura 3. 33: Bloque Suma

El bloque suma contiene 2 entradas por defecto lo que permite llevar a cabo la operación de adición o sustracción entre dos señales colocadas en sus entradas. Se puede personalizar las operaciones que realizará el bloque suma desde el parámetro **List of Signs**. Los signos *más* "(+)" *menos* "(-)" y *espacio* "(|)" indican las operaciones que llevarán a cabo las entradas. Estas entradas otorgan las siguientes opciones:[70]

- Si hay dos o más entradas, entonces el número de caracteres "+" y "-" (*más y menos)* debe ser igual a el número de entradas. Es decir; la configuración "+-+" requiere tres entradas y el bloque será configurado para responder de esa manera.
- El carácter de *espacio* (|) crea un espacio entre los puertos del icono suma.

[71] Print Screen del programa ***Matlab/Simulink***

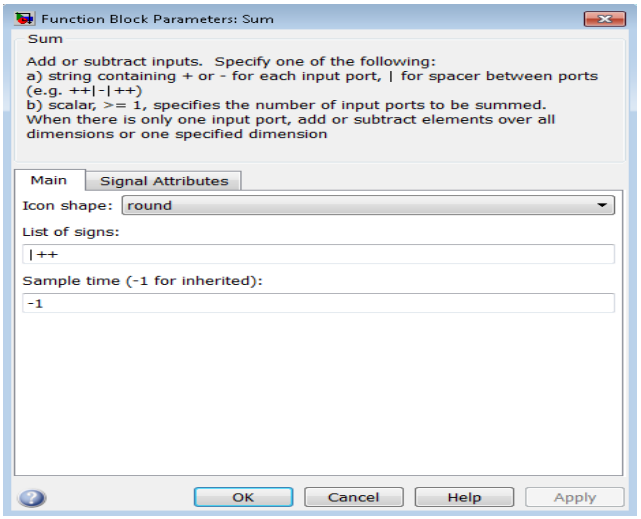


Figura 3. 34: Parámetros del bloque Suma

Producto [72]

Figura 3. 35: Bloque producto

Por default este bloque contiene 2 entradas que multiplican los valores presentes en ellas, de igual manera que el bloque **suma** se puede modificar el número de entradas que se necesite para nuestro diagrama. Por defecto se usan los parámetros que ya vienen establecidos en el bloque ya que lo único que nos interesa es obtener en la salida la multiplicación de sus entradas.[70]

Continuous Library

Esta librería provee de bloques que definen estados para una entrada de datos de carácter continuo, como los son; los integradores, derivadores o funciones de retardo de tiempo o "*Delay".*

Integrator [73]

Figura 3. 36: Bloque Integrator

[72] Print Screen del programa ***Matlab/Simulink***
[73] Print Screen del programa ***Matlab/Simulink***

El bloque “Integrator”, genera a su salida el valor de la integral de la señal de su entrada con respecto al tiempo

Transport Delay [74]

Figura 3. 37: Bloque Transport Delay

El bloque Transport Delay, desfasa la entrada una cierta cantidad en el tiempo. La entrada para este bloque tiene que ser una señal de carácter continuo.

User-Defined Functions Library

Esta es una librería muy particular, y nos permite personalizar el comportamiento de un bloque mediante las funciones que el usuario ingresa. Estas funciones ingresan al bloque ya sea mediante una ecuación especificada por el usuario o mediante un fichero *function* creado en código ***Matlab***.[70]

Fcn [75]

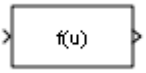


Figura 3. 38: Bloque Fcn

Este bloque le aplica a su entrada una función matemática específica de manera que; los datos de la entrada se regirán bajo las ecuaciones que este bloque contenga, por lo que en su salida nos dará como resultado la resolución de ella.[70]

Matlab **Function** [76]

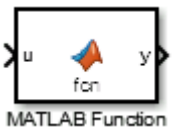


Figura 3. 39: Bloque ***Matlab*** Function

Con este bloque, el usuario puede correr un fichero función creado en ***Matlab*** para usarlo en un modelo de ***Simulink***. Este bloque es una ayuda para los

[74] Print Screen del programa ***Matlab/Simulink***
[75] Print Screen del programa ***Matlab/Simulink***
[76] Print Screen del programa ***Matlab/Simulink***

casos específicos en los que no se dispone de un bloque que ayude a completar el algoritmo que generaría la respuesta esperada del sistema que se quiera simular.[70]

Discontinuities

Esta librería contiene bloques que definen estados discontinuos en una señal, como pueden ser; Saturadores y Cuantizadores.

Transport Delay [77]

Figura 3. 40: Bloque Transport Delay

Este bloque establece los límites del valor máximo y mínimo que tendrá una señal de entrada.[70]

Sinks

Esta librería contiene fuentes de visualización para el usuario con respecto al tiempo de simulación. Esta librería nos provee de una interfaz gráfica para identificar qué operación está cumpliendo cada parte del sistema dibujado, es decir, que al colocarlos a la salida de un proceso, en sus pantallas nos devolverá el resultado de la interacción de los bloques conectados.

Scope [78]

Figura 3. 41: Bloque Scope

Este bloque muestra en su entrada las señales que se generaron durante el tiempo de simulación. En este bloque podemos representar diversas señales a la vez durante el mismo periodo de tiempo. El bloque 'Scope' nos permite ajustar tanto el tiempo como el rango de valores de entrada presentados, además, podemos mover y redefinir el tamaño de la ventana y por último podemos modificar los valores de sus parámetros durante la simulación.

Para visualizar el resultado de la simulación accedemos presionando "doble click" sobre el bloque. La ventana de este bloque contiene varios iconos, pero el más destacado tiene el símbolo de un engranaje y se lo conoce como

[77] Print Screen del programa ***Matlab/Simulink***
[78] Print Screen del programa ***Matlab/Simulink***

'Parameters', en este espacio podemos escoger el número de ejes que se quiere representar mediante la opción 'Number of Axes'. Por otro lado la pestaña '**Data history**' de la ventana del bloque '*Scope*' contiene un parámetro de gran ayuda para la representación de gráficas, parámetro llamado '**Limit data point to last**'. Esta opción nos permite especificar cuantos puntos queremos representar durante la simulación. Por ejemplo, en una simulación de periodo de muestreo reducido, durante la simulación generará un número amplio de puntos, con lo que nos deja claro que esta opción debe tener un número muy elevado para lograr visualizar toda la simulación de forma correcta.

Este bloque dibuja punto a punto la señal producida siempre que la señal sea continua, por otro lado si la señal es discreta el bloque dibujará una señal escalón.[70]

Sources

Esta librería contiene bloques que pueden crear varios tipos de señales mismas que encontramos de manera particular en sistemas básicos de telecomunicaciones.

Sine wave [79]

Figura 3. 42: Bloque sine wave

Este bloque genera la más típica señal presente en los sistemas básicos de telecomunicaciones usando tiempo de simulación como base de tiempo, la cual es una onda sinusoidal. El bloque puede operar en modo de basado en tiempo o basado en muestras.[70]

- **Modo basado en el tiempo**
  La salida del bloque se rige bajo el siguiente comportamiento:

  y= Amplitud * sin (frecuency * time + phase) + bias

  Donde los parámetros mencionados se especifican a continuación:
  - **Amplitud:** Valor pico de nuestra señal.
  - **Frecuency:** Frecuencia en radianes de nuestra señal.
  - **Phase:** Desfasaje de la señal en el eje "*x*" según el ángulo ingresado.

[79] Print Screen del programa ***Matlab/Simulink***

- **Bias:** Valor (DC) agregado a la señal para producir un desfasaje en el eje "*y*".

A su vez el bloque **sine wave** establecido en el modo basado en tiempo tiene dos sub-modos: modo continuo y modo discreto.

El valor del parámetro **Sample time** determina como operara el bloque, ya sea en modo discreto o continuo para lo cual debemos fijarnos lo siguiente:

- Si el valor es 0 (default) deriva en que el bloque opere en modo de tiempo continuo. En este modo de operación debemos ser muy criteriosos, ya que el resultado de la simulación puede ser incorrecto debido a que el bloque pierde precisión en tiempos de simulación muy grandes.
- Si se coloca un valor mayor a 0; el bloque opera en modo de tiempo discreto. En este modo se construye un modelo con una fuente puramente discreta, con esto se evita tener modelos híbridos "continuos/discretos" que demandan mucho más tiempo para simular.

➢ **Modo basado en muestras**

El comportamiento de la salida del bloque viene dado bajo la siguiente influencia:

y= Amplitud * sin (2Π * (k+*o*) / p) + bias

Donde:

- k es un valor integral que se repite en el rango de 0 a p-1
- *o* es el desfasaje de la señal, es decir; su fase
- p es el número de muestras en el tiempo, tomados del periodo de la sinusoide.

Comunications System Toolbox

Esta es una librería, construida en base a varios bloques, que representan los componentes presentes en sistemas de telecomunicaciones. Por tal razón, en ella podemos encontrar bloques que simulan el comportamiento de canales de comunicaciones, generadores de ruido, moduladores, incluso podemos encontrar bloques que simulan las características de los obstáculos que atraviesa una señal de RF.[51]

DSP System Toolbox

Esta es una librería especial, ya que gran parte de ella provee los bloques que simulan componentes presentes en sistemas de telecomunicaciones, de manera que en ella podemos encontrar aquellos bloques que nos ayudan a tener estimaciones de las componentes de frecuencia de las señales. Contiene también bloques que ayudan a idear un algoritmo de resolución para representar canales de comunicaciones, generadores de ruido, moduladores, entre otros.[51]

Diseño de filtro análogo

Los sistemas que se simulan dentro del entorno ***Simulink***, se crean con la intención de que sus características sean lo más posible asemejadas a la realidad, es decir que la respuesta de un filtro creado en este entorno, no tendrá la respuesta de un filtro ideal. El bloque **Diseño de filtro análogo** se encuentra en la librería ***Signal Procesing Blockset*** y exige ciertos requisitos para trabajar con una señal:

1. La señal que ingrese al filtro debe ser una señal continua en el tiempo
2. La señal que ingrese al filtro debe poseer valores reales
3. Los valores de la señal son escalares basados en muestras

La configuración tanto del diseño como de configuración de banda del filtro la escogemos desde "***Design mode***" y **"Filter Type"** desde la ventana principal del bloque. Esta caja de diálogo contiene también los parámetros necesarios para elección del orden del filtro y las frecuencias de borde en la que trabajará el filtro escogido, estos son los parámetros secundarios que deben ser establecidos y probados de acuerdo a las combinaciones posibles entre las pestañas "***Design mode***" y **"Filter Type"**. [70]

La figura 3.43 muestra los parámetros del bloque descrito.[80]

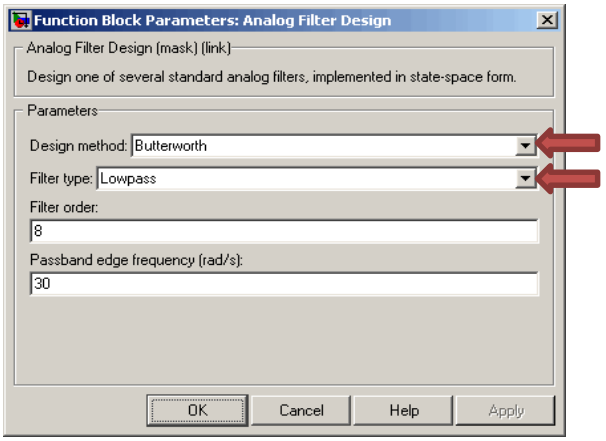


Figura 3. 43: Ventana de parámetros del bloque Analog Filter Design

[80] Print Screen del programa ***Matlab/Simulink***

Existen diversas combinaciones entre el diseño y tipo de filtro las cuales podemos entender con ayuda de la tabla 3.6.[81]

Configuración	Butterworth	Chebyshev I	Chebyshev II	Elíptica
Paso bajo	Ω_p	Ω_p , R_p	Ω_s , R_s	Ω_p , R_p , R_s
Pasa altos	Ω_p	Ω_p , R_p	Ω_s , R_s	Ω_p , R_p , R_s
Pasa banda	Ω_{p1} , Ω_{p2}	Ω_{p1} , Ω_{p2} , R_p	Ω_{s1} , Ω_{s2} , R_s	Ω_{p1} , Ω_{p2} , R_p , R_s
Banda eliminada	Ω_{p1} , Ω_{p2}	Ω_{p1} , Ω_{p2} , R_p	Ω_{s1} , Ω_{s2} , R_s	Ω_{p1} , Ω_{p2} , R_p , R_s

Tabla 3. 6: Configuraciónes posibles entre el tipo y diseño de un filtro en ***Matlab/Simulink***

Los parámetros de la tabla son:

- Ω_p - frecuencia borde de banda de paso
- Ω_{p1} - menor frecuencia de borde de banda de paso
- Ω_{p2} - frecuencia de corte superior
- Ω_s - frecuencia borde de banda rechazo
- Ω_{s1} - menor frecuencia de borde banda de rechazo
- Ω_{s2} - mayor frecuencia borde banda de rechazo
- R_p - ondulación banda de paso en decibelios
- R_s - atenuación de banda de detención en decibelios

3.1.23. Densidad espectral de potencia

El análisis de la densidad espectral de potencia es fundamental cuando se trata de recuperar una señal de información; ya que mediante este proceso sabremos cuales componentes de frecuencia contiene nuestra señal luego de que esta haya pasado por su etapa de filtrado. La medida que se obtiene es la distribución de valores de potencia en función de la frecuencia, de donde la potencia es considerada como el promedio de la señal; que en dominio de la frecuencia esto es el cuadrado de la magnitud de la transformada rápida de Fourier. Para esto ***Simulink*** nos provee de un bloque conocido como: ***Power Spectral Density***, el cual también es un bloque de visualización al igual que el bloque **Scope**. El bloque nos ayudara a tener una estimación usando el algoritmo de la FFT, con lo cual, podemos observar las posibles pérdidas de energía que haya sufrido la señal durante su recuperación.[70]

[81] Ayuda del programa ***Matlab/Simulink***

Capitulo IV

4. Modelado y simulación en *Matlab/Simulink* aplicado a la asignatura de Telecomunicaciones I

Este capítulo contiene una sugerencia de prácticas que ayudarán a complementar la teoría dictada en la materia de Telecomunicaciones I usando el software ***Matlab/Simulink***, para lo cual se utilizan varios de los conceptos abarcados en capítulos anteriores.
Se desarrolla a continuación un pequeño manual que abarca un formato sugerido para el desarrollo total de prácticas mediante simulaciones, en el cual cada simulación contiene:

- Trabajo preparatorio correspondiente
- Desarrollo de la práctica
- Preguntas y conclusiones

La parte correspondiente al trabajo preparatorio se hace necesaria ya que usando este método el estudiante podrá ir con previo conocimiento acerca de la manipulación de una función o bloque relacionado a ***Matlab/Simulink***; mismo que se usará para resolver el problema planteado.

El desarrollo de la práctica o ejecución de la simulación es necesaria para evaluar el conocimiento adquirido en la materia de Telecomunicaciones I.

Después de la ejecución de cada simulación el estudiante responderá a varias preguntas generadas y podrá anotar sus propias conclusiones acerca de los resultados obtenidos.

El capítulo consta de 5 prácticas que se subdividen en dos partes las cuales se listan a continuación:

- Generación de Señales típicas usando ***Matlab/Simulink.***
- Serie y Transformada Rápida de Fourier
- Modulación y Demodulación de Amplitud (AM)
- Modulación y Demodulación de Frecuencia (FM)
- Transmisión y Recepción de señales en sistemas básicos de telecomunicaciones.

4.1. Simulación # 1: Generación de señales típicas en Telecomunicaciones usando *Matlab/Simulink*

4.1.a. Generación de señales típicas en telecomunicaciones usando *Matlab*.

Realizar una simulación en la que se facilite la generación de la señales típicas presentes en sistemas básicos de telecomunicaciones; usando código de ***Matlab*** para la generación y visualización de estas señales.

TRABAJO PREPARATORIO

- Consultar las señales que puede emitir un generador de funciones.
- Investigar como crear un vector usando la función de ***Matlab*** *"linspace"*.
- Investigar en ***Matlab*** como generar un vector de valores aleatorios.
- Consultar las funciones de ***Matlab*** para generar ondas cuadradas, triangulares, trenes de pulsos, señales senos y cosenos.

TRABAJO PRÁCTICO
Sustento Teórico
Conceptos:

Dentro de los sistemas de telecomunicaciones se pueden encontrar diversos tipos de señales entre los cuales conocemos:

- Señal Continua
- Señal Sinusoidal
- Señal Tren de pulsos
- Señal Diente de sierra
- Señal aleatoria

Algunas de las señales, que se presentan dentro del ámbito de las telecomunicaciones, hacen posible predecir cuál será su amplitud en cualquier instante de tiempo siempre que se pueda reconocer su periodo y por tanto su frecuencia. El concepto de periodo es muy importante para los casos de las señales de información.

Se conoce como periodo al número de veces que una señal se repite en un instante de tiempo determinado. El periodo está relacionado de forma inversa a la frecuencia de la señal; siendo esta la magnitud que mide el número de repeticiones por unidad de tiempo de una señal.

OBJETIVOS

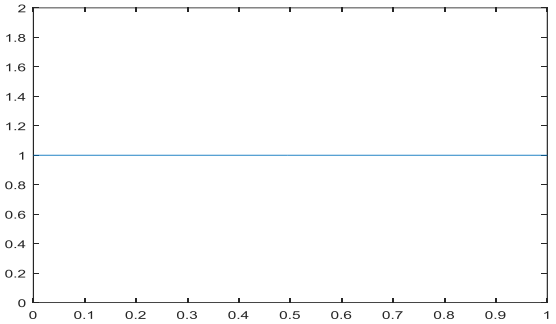
- Conocer cómo acceder y utilizar las funciones básicas del software ***Matlab*** para generar señales.
- Desarrollar un código ***Matlab*** simple que nos ayude a manipular vectores y gráficar sus resultados.
- Representar señales típicas en telecomunicaciones usando ***Matlab*** y su ventana de gráficos.
- Observar y reconocer la relación de la frecuencia y el periodo en las señales.

Librerías utilizadas:

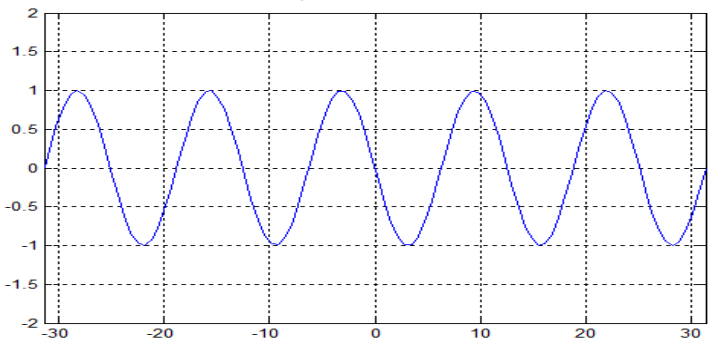
- ***MATLAB***\graph2d
- ***MATLAB***\Elementary math functions.
- ***MATLAB***\DSP

Para obtener las gráficas de las señales típicas presentes en los sistemas de telecomunicaciones, se puede usar el siguiente código:

Señal Continua:

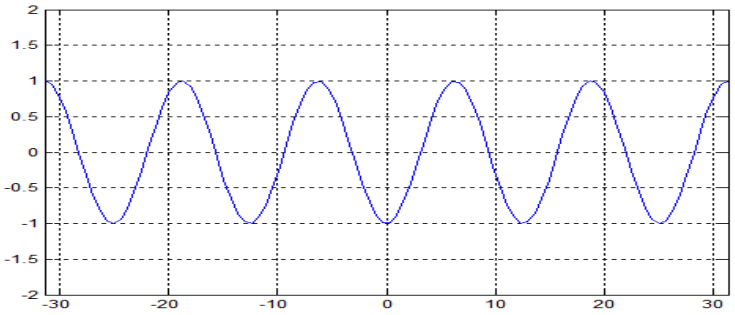
Código:
t=linspace(0,1,100);
y=1+(0*t);
plot(t,y); grid off
 	
Señal a gráficada
 	
$y = 1$
 	
$0 < t < 1$
	Observación/lectura de resultados
	 
	Figura 4. 1: Señal continua
	Fuente: El autor

Señal Seno

Código:
t=linspace(-10*pi,10*pi,100);
w=50;
y=sin (w*t);
plot(t,y); grid on
axis([-10*pi 10*pi -2 2])
Señal a gráficada
 	
$y = \text{seno}(w * t)$
 	
$-10\pi < t < 10\pi$
	Observación/lectura de resultados
	 
	Figura 4. 2: Señal seno
	Fuente: El autor

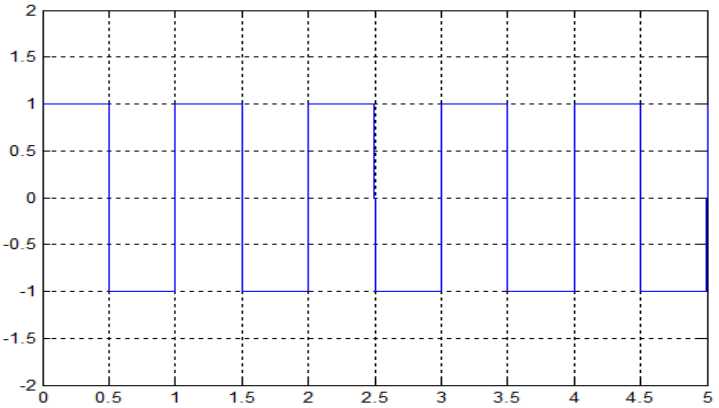
Señal Coseno:

Código	Observación/lectura de resultados

t=linspace(-10*pi,10*pi,100);
w=50;
y=cos(w*t);
plot(t,y); grid on
axis([-10*pi 10*pi -2 2])
Señal a gráficada
$y = \cos(w * t)$
$-10\pi < t < 10\pi$
	
	Figura 4. 3: Señal coseno
	Fuente: El autor

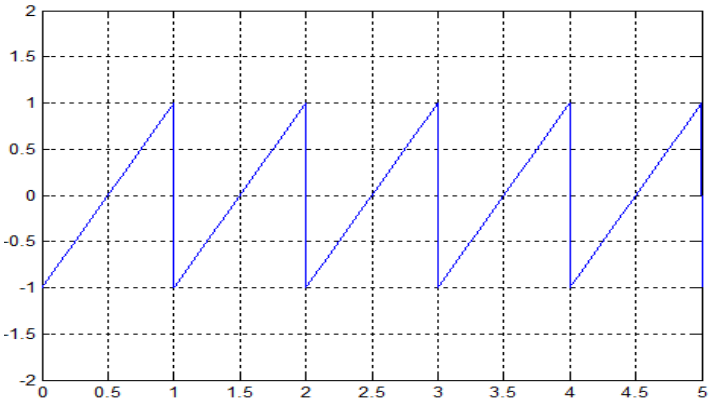
Señal tren de pulsos:

Código	Observación/lectura de resultados

fs=1000;
t=(0:1/fs:5);
y=square(2*pi*t);
plot(t,y); grid off
axis([0 5 -2 2]);
Señal a gráficada
$y = \text{square}(t)$
$0 < t < 1$
	
	Figura 4. 4: Señal tren de pulsos
	Fuente: El autor

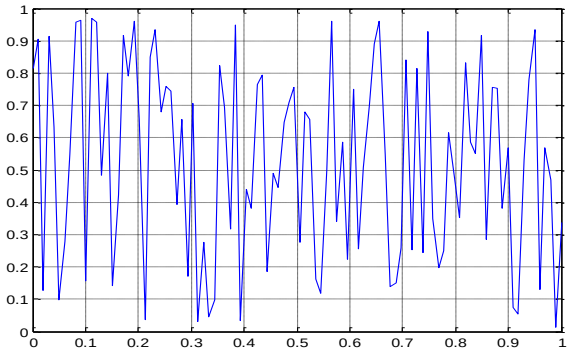
Señal diente de sierra:

Código	Observación/lectura de resultados

fs=1000;
t=(0:1/fs:5);
y=sawtooth(2*pi*t);
plot(t,y); grid on
axis([0 5 -2 2]);
Señal a gráficada
$y = \text{sawtooth}(t)$
$0 < t < 1$
	
	Figura 4. 5: Señal diente de sierra
	Fuente: El autor

Señal aleatoria:

Código	**Observación/lectura de resultados**

x=linspace(0,1,100);
y=rand(1,length(x);
plot(x,y); grid on
 	
Señal a gráficada
 	
$y = \text{rand}(t)$
 	
$0 < x < 1$
	
	Figura 4. 6: Señal aleatoria
	Fuente: El autor

Preguntas
1. ¿Qué representa la variable fs en las señales tren de pulsos y diente de sierra? Explique.
2. Observando en los resultados de las gráficas indique. ¿Cuál es el periodo que tienen las señales representadas?
3. En la señal aleatoria. ¿Es posible calcular el periodo de la señal? Explique.
4. Para las señales no aleatorias, variar su frecuencia y comprobar los resultados para una frecuencia mayor y para una frecuencia menor a la asignada.

CONCLUSIONES

4.1.b. Generación de señales típicas en telecomunicaciones usando *Simulink*.

Realizar una simulación que ayude en la generación de la señales típicas presentes en sistemas básicos de telecomunicaciones; usando los bloques de ***Simulink*** para generar y visualizar los resultados.

TRABAJO PREPARATORIO

- Consultar el bloque ***Signal generator*** de la librería **Sources** de ***Simulink***.
- Consultar como cambiar el aspecto de visualización en el bloque ***Scope***.
- Consultar la relación entre el tiempo de simulación de ***Simulink*** y los parámetros de frecuencia del bloque ***Signal generator***.
- Consultar acerca del bloque ***Constant*** de la librería **Sources** de ***Simulink***.
- Consultar acerca del bloque ***Hit Crossing*** de la librería ***Discontinuities***.

TRABAJO PRÁCTICO
Sustento Teórico
Conceptos:

El bloque ***Signal generator*** es un bloque de la librería “Sources”, que simula a un generador de funciones mediante el cual podemos representar sin mayor dificultad, señales de diversos tipos como son:

- Señal Continua.
- Señal Sinusoidal.
- Señal Tren de pulsos.
- Señal Diente de sierra.
- Señal aleatoria.

Es importante verificar el periodo de la señal generada y la relación que tiene su frecuencia con el tiempo de simulación, ya que con esto se puede obtener la gráfica acorde a la esperada y obtener una visualización óptima.

Se conoce como periodo al número de veces que una señal se repite en un instante de tiempo determinado. El periodo está relacionado de forma inversa a la frecuencia de la señal siendo esta la magnitud que mide el número de repeticiones por unidad de tiempo de una señal.

OBJETIVOS

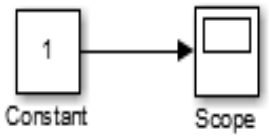
- Generar distintos tipos de señales usando solamente 2 bloques de ***Simulink***.
- Conocer la ventaja de utilizar los bloques de ***Simulink*** frente a la generación de código ***Matlab*** para generar señales.
- Conocer cómo graficar varios periodos de una señal, fijando el tiempo de simulación de ***Simulink***.
- Observar y reconocer la relación de la frecuencia y el periodo en las señales con el tiempo de simulación de ***Simulink***.

Librerías utilizadas

- ***Simulink***\Sources.
- ***Simulink***\Sinks.

La distribución de los bloques para generación de señales, es la que sigue:

Señal Continua:



Constant
Constant value: 1
Sample time: inf
Tiempo de simulación: 10s

Observación/lectura de resultados

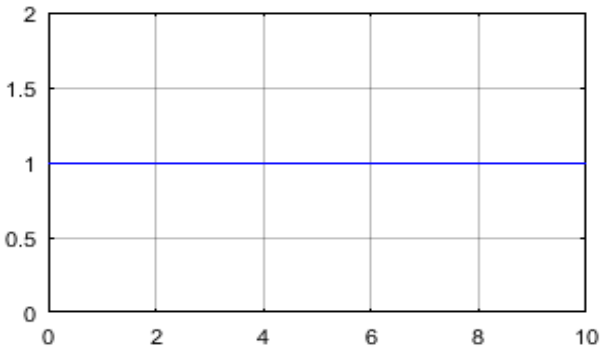
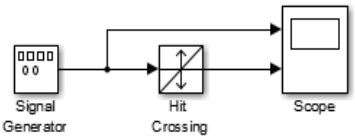


Figura 4. 7: Señal continua
Fuente: El autor

Señal Seno:



Signal Generator
Amplitud: 1
Frecuencia: 50
Unidades: Hertz
Forma de onda: seno
Tiempo de simulación: 0.02s

Observación/lectura de resultados

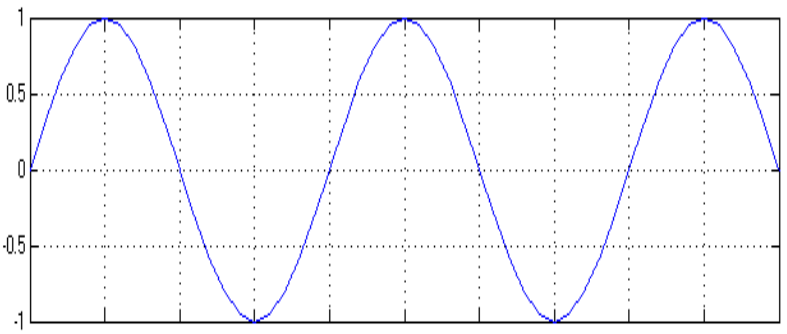
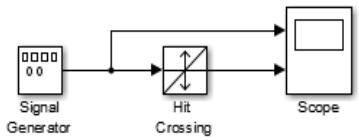


Figura 4. 8: Señal seno
Fuente: El autor

Señal tren de pulsos:



Signal Generator
Amplitud: 5
Frecuencia: 100
Unidades: Hertz
Forma de onda: cuadrada
Tiempo de simulación: 0.05s

Observación/lectura de resultados

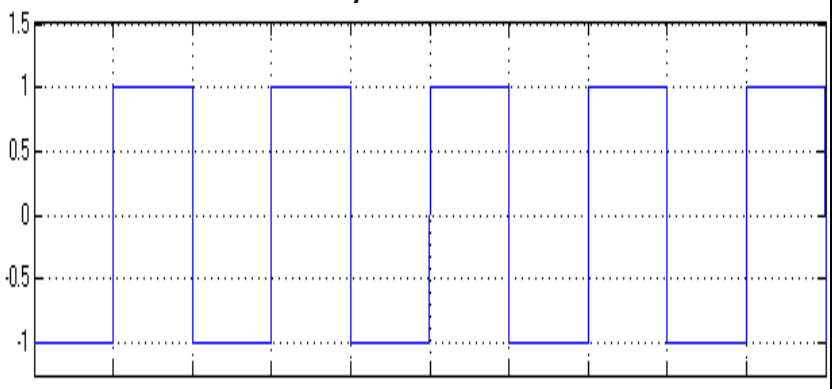
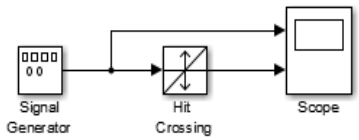


Figura 4. 9: Señal tren de pulsos
Fuente: El autor

Señal tren diente de sierra:



Signal Generator
Amplitud: 1
Frecuencia: 100
Unidades: Hertz
Forma de onda: diente de sierra
Hit Crossing
Hit Crossing offset: 0
Sample time: -1
Tiempo de simulación: 0.05s

Observación/lectura de resultados

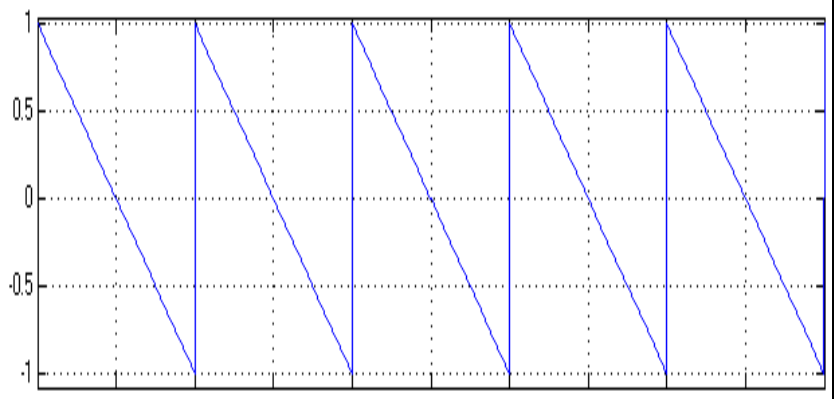
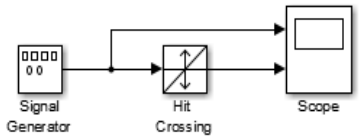


Figura 4. 10: Diente de sierra
Fuente: El autor

Señal aleatoria:



Signal Generator
Amplitud: 1
Frecuencia: 100 Hz
Unidades: rad/seg
Forma de onda: aleatoria
Tiempo de simulación: 0.05s

Observación/lectura de resultados

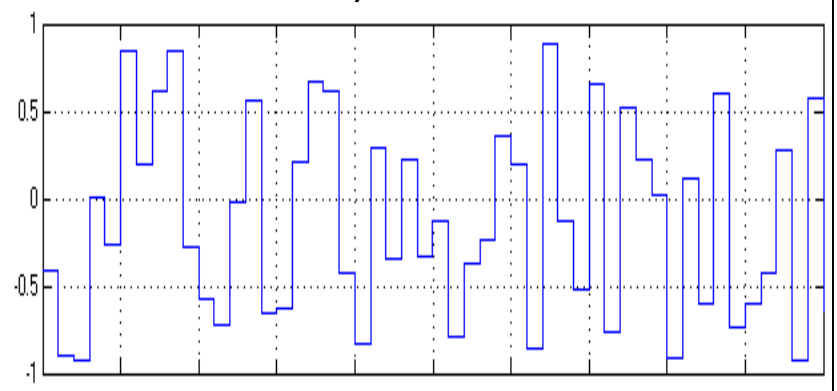


Figura 4. 11: Señal aleatoria
Fuente: El autor

Preguntas
1. Varar el tiempo de simulación a 10s. ¿Qué ocurre con la representación gráfica de la función seno?
2. Repetir el proceso de la pregunta 1 para el resto de tipos de onda. ¿Qué es lo que ocurre?
3. En la señal aleatoria. ¿Por qué no se altera su representación gráfica pese a haber cambiado el tiempo de simulación? Explique.

CONCLUSIONES

4.2. Simulación #2: Serie y Transformada Rápida de Fourier.

Los análisis de las señales presentes en sistemas básicos de telecomunicaciones, utilizan métodos que suponen cierta facilidad para la interpretación de las mismas. Entre estos métodos que ayudan a facilitar su interpretación se encuentran las series y transformadas de Fourier, las cuales nos sirven para pasar una señal del dominio temporal al dominio de la frecuencia.

4.2.1. Serie de Fourier

TRABAJO PREPARATORIO

- Consultar como declarar variables simbólicas.
- Consultar como usar la función de integración numérica *"int"*.
- Consultar la función *"pretty"*
- Consultar cómo cambiar el aspecto de visualización de la ventana de gráficos ***Matlab***, de manera que se pueda manipular y editar las etiquetas de los ejes de coordenadas.
- Consultar cómo graficar señales discretas usando la función *"stem"*
- Consultar las funciones de ***Matlab***, para convertir una variable simbólica en variable double.

TRABAJO PRÁCTICO
Sustento teórico
Conceptos:

La serie de Fourier se aplica en señales periódicas de periodo T. No todas las señales presentes en telecomunicaciones son de característica sinusoidal pero se las puede expresar como una suma de senos y cosenos y obtener una representación espectral de las mismas. Esta representación nos indica como se encuentra distribuida la energía dentro de las diferentes frecuencias que la señal contiene. A esta distribución se la conoce con el nombre de *espectro de la señal*. Para una señal periódica el espectro es discreto, y su energía se concentra en frecuencias múltiplos de una frecuencia llamada *fundamental*, la cual se encuentra directamente relacionada con el periodo de la señal. El teorema de Fourier establece que una función puede escribirse como una serie de términos que incluyen funciones trigonométricas. En general cualquier función periódica puede expresarse como:

$$f(t) = \frac{1}{2}\, a_0 + \, a_1\, cos wt + a_2\, cos 2wt + a_3\, cos 3wt + a_4\, cos 4wt + \cdots$$
$$+ b_1\, sin wt + b_2\, sin 2wt + b_3\, sin 3wt + b_4\, sin 4wt + \cdots$$

De donde los coeficientes de la serie de Fourier se calculan usando las siguientes formulas:

$$a_0 = \frac{1}{T}\int_0^T f(t) * dt$$

$$a_n = \frac{2}{T}\int_0^T f(t) * \cos(nw_0 t) * dt$$

$$b_n = \frac{2}{T}\int_0^T f(t) * \text{sen}(nw_0 t) * dt$$

$$Para\ n = 1,2,3,5, \dots \dots n$$

OBJETIVOS

- Ser capaces de graficar cualquier tipo de función dada utilizando código ***Matlab***.
- Comparar los resultados de los ejercicios desarrollados en clase; con los resultados del cálculo computacional del laboratorio.
- Ser capaces de manipular con facilidad la ventana de gráficos de código ***Matlab*** que permita calcular *"n"* coeficientes de la serie de Fourier de una señal dada.
- Ser capaces de generar código ***Matlab*** que permita dibujar *"n"* armónicos de dicha señal.
- Estar en capacidad de resolver ejercicios de Series de Fourier usando código ***Matlab***.

Librerías utilizadas

- ***MATLAB***\graph2d.
- ***MATLAB***\Elementary math functions.
- ***MATLAB***\Symbolic math toolbox.

Desarrollo

Encontrar los coeficientes de Fourier para la siguiente función:

$$f(x)\begin{cases} A * seno(t) & 0 < t < \pi \\ 0 & \pi < t < 2\pi \end{cases}$$

Para encontrar los coeficientes de Fourier de la función dada, aplicaremos las fórmulas usando código *Matlab*.

Función dada:

$$f(x)\begin{cases} A*seno(t) & 0<t<\pi \\ 0 & \pi<t<2\pi \end{cases}$$

Código que genera la gráfica mostrada:

```
x1=linspace(0,pi,100);
y1=sin(x1);
x2=linspace(pi,2*pi,100);
y2=0*x2;
x3=linspace(2*pi,3*pi,100);
y3=sin(x3);
x=[x1,x2,x3];
y=[y1,y2,y3];
plot(x,y);
```

Observación/lectura de resultados
Forma de onda "Rectificador de media onda"

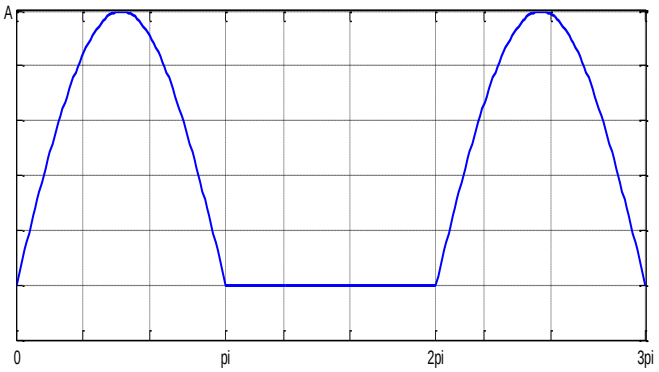


Figura 4. 12: Generación de la función dada
Fuente: El autor

Cálculo de los coeficientes de la Serie de Fourier para la función dada, usando código *Matlab*

```
%CALCULO DE LOS COEFICIENTES DE FOURIER DE UNA SEÑAL PERIODICA
syms t A %Declaramos t y A como variables simbólicas
%Se hará el cálculo para 5 coeficientes de la Serie, ➔ n=1,2,3,4,5
n=1:5; %Creamos un vector "n" de 5 elementos
T=2*pi;%Periodo de la señal (se puede reemplazar)
wo=2*pi/(T);%Calculo de Wo
f1(t)=A*sin(t);%La función "A*sin(t)", depende del ejercicio a resolver
f2(t)=0*t;%La función "0*t", depende del ejercicio a resolver
%Calculamos integrales definidas usando la función "int" de Matlab para
encontrar los términos de la serie
a0=int(f1(t),t,0,T/2)+int(f2(t),t,T/2,T);%Calculamos la integral definida
de a0 respecto a "t" en el intervalo "0 a T/2" para f1(t) y de "T/2 a T"
para f2(t)
a0=(1*a0)/T;% sobrescribimos el valor encontrado de "a0", multiplicando
"a0" por el valor 1/T de la fórmula de "a0" justamente
disp('a0=') %Mostramos en el Command Window de Matlab la palabra a0
pretty(a0)%Se imprime la expresión simbólica "a0"con una mejor
presentación
an=int(f1(t)*cos(n*t*wo),t,0,T/2)+int(f2(t)*cos(n*t*wo),t,T/2,T); %
Calculamos la integral definida de an respecto a "t" en el intervalo "0 a
T/2" para f1(t) y de "T/2 a T" para f2(t)
an=(2*an/T);% sobrescribimos el valor encontrado de "an", multiplicando
"an" por el valor 2/T de la fórmula de "an" justamente
disp('an=') %Mostramos en el Command Window de Matlab la palabra an
pretty(an)% se imprime la expresión simbólica con una mejor presentación
bn=int(f1(t)*sin(n*t*wo),t,0,T/2)+int(f2(t)*sin(n*t*wo),t,T/2,T);
Calculamos la integral definida de bn respecto a "t" en el intervalo "0 a
T/2" para f1(t) y de "T/2 a T" para f2(t)
bn=(2*bn/T);% sobrescribimos el valor encontrado de "bn", multiplicando
"bn" por el valor 2/T de la fórmula de "bn" justamente
disp('bn=') Mostramos en el Command Window de Matlab la palabra bn
pretty(bn)% se imprime la expresión simbólica con una mejor presentación
A0=subs(a0,A,1);%Substituyo A con 1 "es decir A=1" en a0 y lo guardo en A0
A0=double(A0);%Convierto datos simbólicos en doble precisión
An=subs(an,A,1);%Substituyo A con 1 "es decir A=1" en an y lo guardo en An
An=double(An); %Convierto datos simbólicos en doble precisión
Bn=subs(bn,A,1);%Substituyo A con 1 "es decir A=1" en bn y lo guardo en Bn
```

```
Bn=double(Bn);%Convierto datos simbólicos en doble precisión
Coef=An+Bn; %Realizo la operación entre los términos de An y Bn
x=[A0 Coef];%Creo un vector x de 6x1 con los valores de A0 y Coef
ng=0:5;%Creo un vector ng de 6x1
stem(ng,x,'filled');%Dibujo el espectro discreto con los valores
encontrados
xlim([0 6]);, ylim([-0.6 0.6]); grid on %Coloco límites de la gráfica y
activo la cuadrícula
```

Observación/lectura de resultados

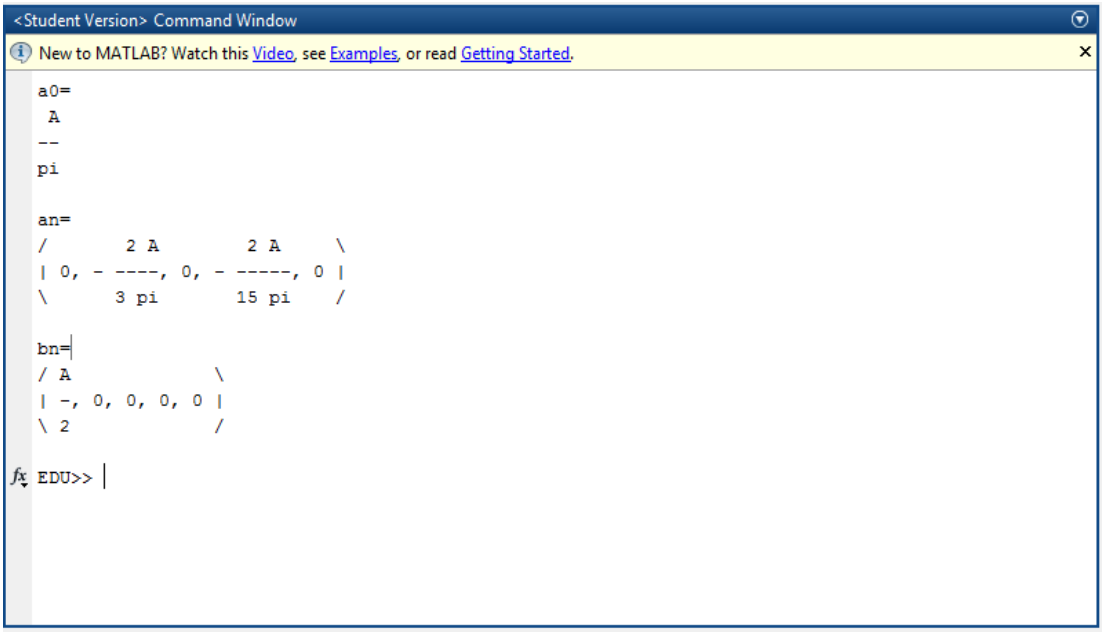


Figura 4. 13: Resultado de ejecución en el *Command Window* ***Matlab***

Fuente: El autor

Expresando en términos matemáticos los valores obtenidos

$$f(t) = \frac{1}{2}\,a_0 + a_1\,coswt + a_2\,cos2wt + a_3\,cos3wt + a_4\,cos4wt + \cdots + b_1\,sinwt + b_2\,sin2wt + b_3\,sin3wt + b_4\,sin4wt + \cdots$$

$$f(t) = \frac{1}{2} * \frac{2A}{pi} + \frac{A}{2}\,coswt - \frac{2A}{3pi}\,cos2wt - 0cos3wt - \frac{2A}{15pi}\,cos4wt - 0cos5wt + \ldots$$

Gráfica del espectro discreto de la señal

$$f(x)\begin{cases} A * seno(x) & 0 < t < \pi \\ 0 & \pi < t < 2\pi \end{cases}$$

Observación/lectura de resultados

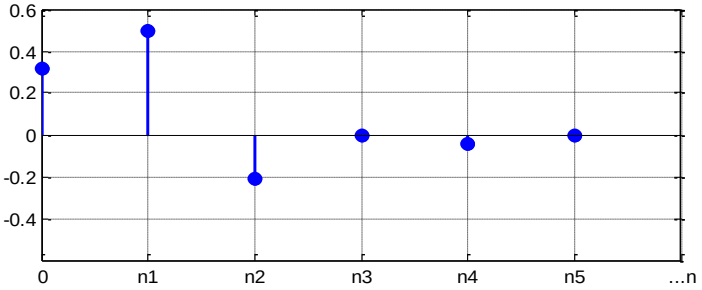


Figura 4. 14: Gráfica de los coeficientes de Fourier

Fuente: El autor

Preguntas
1. ¿Se puede usar la función *stem* para graficar variables simbólicas? Explique.
2. ¿Cuál es el código para encontrar los coeficientes de Fourier para la forma de onda del rectificador de onda completa?
3. Para una función aleatoria. ¿Es posible calcular su serie de Fourier? Explique.

CONCLUSIONES

4.2.2. Transformada Rápida de Fourier

La transformada de Fourier se aplica a señales no periódicas y de igual manera que las series de Fourier, éstas nos ayudan a obtener los parámetros de las componentes de frecuencia que la señal contiene, a diferencia de que el espectro de una señal periódica es discreto y no contiene valores en cero, los espectros que producen las señales no periódicas son continuos.

TRABAJO PREPARATORIO

- Consultar la función "*disp.*".
- Consultar como usar la función para variables simbólicas "*pretty*".
- Consultar la función para gráficos simbólicos *"ezplot"*
- Consultar las condiciones de uso para la función *ffy*.
- Consultar cómo cambiar el aspecto de visualización de la ventana de gráficos ***Matlab***, de manera que se pueda manipular y editar las etiquetas de los ejes de coordenadas.
- Consultar cómo graficar señales discretas usando la función *"stem"*

TRABAJO PRÁCTICO
Sustento teórico
Conceptos:

La transformada de Fourier se aplica en señales no periódicas. En los sistemas de telecomunicaciones, también son señales de interés la función impulso unitario, función rampa unitaria, entre otras, las cuales no tienen definido un periodo de repetición. El espectro que generan estas señales tienen cruces por cero y son de característica continua. Se puede suponer que una señal no periódica proviene de una señal periódica, en donde; el periodo se extiende desde -∞ hasta ∞. Dicho esto; podemos expresar que para una señal que es función del tiempo con periodo -∞ hasta ∞ tenemos la siguiente integral:

$$F(w) = \int_{-\infty}^{\infty} f(t) * e^{-jwt} * dt$$

Para esta condición debemos asumir que la función existe para cada valor de la frecuencia en radianes *"w"*; con esto podemos llamar a esta función como la conocida transformada de Fourier. La transformada de Fourier es en general una función compleja; es decir contiene una parte real y una parte imaginaria.

OBJETIVOS

- Ser capaces de graficar cualquier tipo de función dada; utilizando código ***Matlab***.
- Comparar los resultados de los ejercicios desarrollados en clase; con los resultados del cálculo computacional del laboratorio.
- Estar en capacidad de resolver ejercicios de Transformada de Fourier usando código ***Matlab***.
- Ser capaces de manipular con facilidad la ventana de gráficos de ***Matlab*.**

Librerías utilizadas

- ***MATLAB***\graph2d.
- ***MATLAB***\Elementary math functions.
- ***MATLAB***\Symbolic math toolbox.

Desarrollo

Encontrar la transformada de Fourier para el impulso unitario dado por la siguiente función

$$f(x)\begin{cases} 1 & -T/2 < x < T/2 \\ 0 & x < -\frac{T}{2} \\ 0 & x > \frac{T}{2} \end{cases}$$

Función dada:

$$f(x)\begin{cases} 1 & -T/2 < x < T/2 \\ 0 & x < -\frac{T}{2} \\ 0 & x > \frac{T}{2} \end{cases}$$

Código que genera la gráfica mostrada:

```
%para la gráfica asumimos para T; un valor real cualquiera.
x1=linspace(-3,-1,100);
```

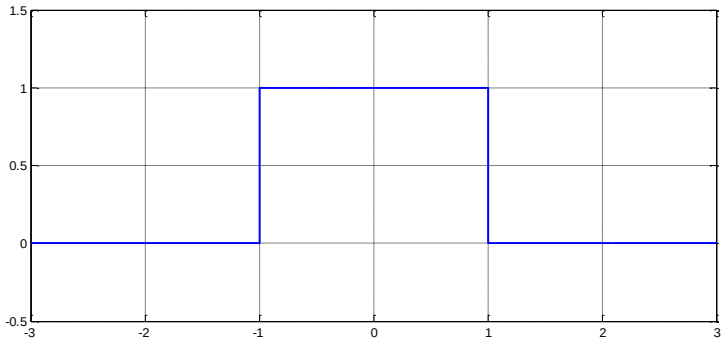


Figura 4. 15: Gráfica impulso unitario

Fuente: El autor

```
y1=0*x1;
x2=linspace(-1,1,100);
y2=1+0*x2;
x3=linspace(1,3,100);
y3=0*x3;
x=[x1,x2,x3];
y=[y1,y2,y3];
plot(x,y,'b','Linewidth',2);
axis([-3 3 -1.5 1.5]);
grid on
```

Ventana de Gráficos Manipulando las etiquetas

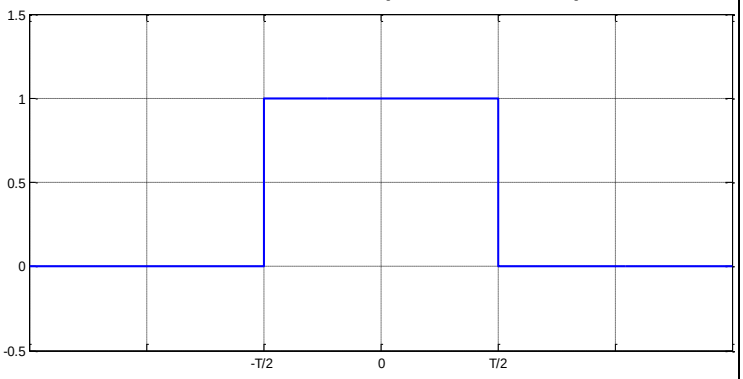


Figura 4. 16: Gráfica impulso unitario con etiquetas manipuladas

Fuente: El autor

Código para el cálculo de la transformada de Fourier de la función impulso cuadrado.

```
syms t T w;%Declaro variables simbólicas
f=1; %La función existe solo en -T/2 a T/2 y es igual a 1
A=int(f*exp(-j*w*t),t,-T/2,T/2);%Usamos la función "int" de Matlab para resolver la integral
disp('F(t)=') %Mostramos en el command window de Matlab la expresión F(t)
pretty(A); %Imprimo con mejor presentación la expresión simbólica "A"
```

Observación/lectura de resultados

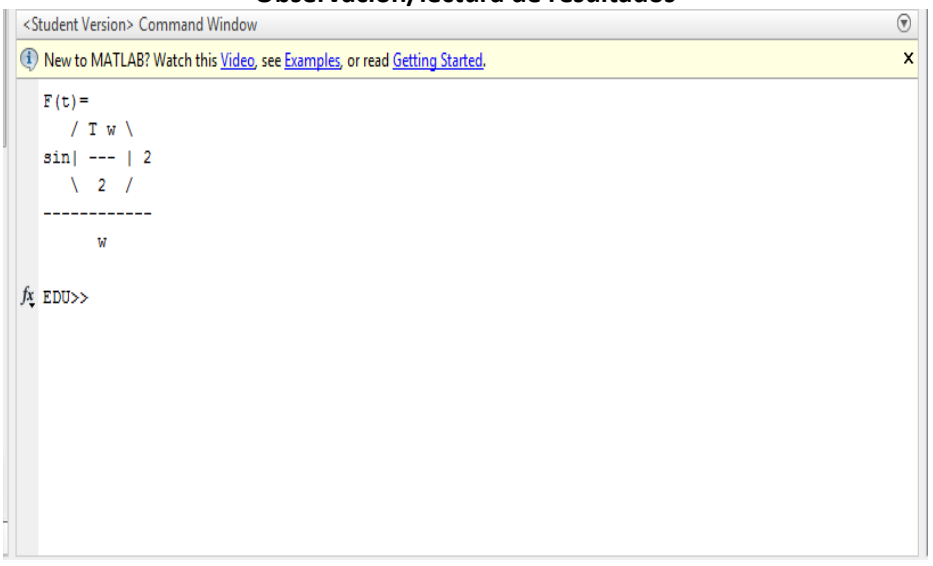


Figura 4. 17: Resultado obtenido en el workspace

Fuente: El autor

Representamos el resultado obtenido en formato matemático

$$F(t) = \frac{sin\left(\frac{wT}{2}\right)*2}{w}$$

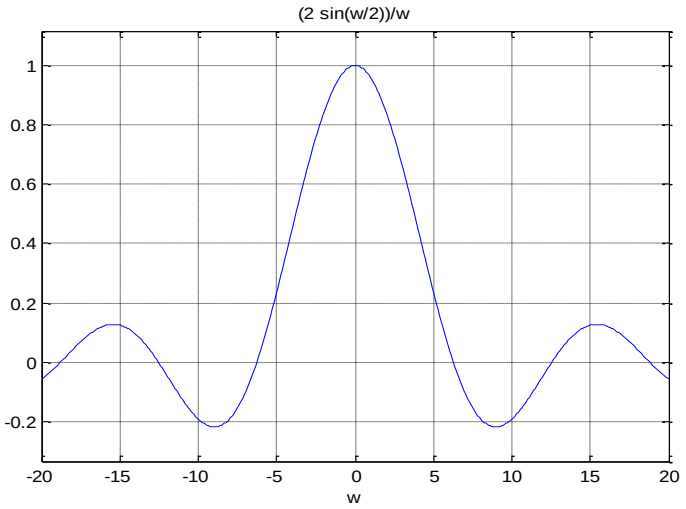
Código para el cálculo de la transformada de Fourier de la función impulso cuadrado.	**Observación/lectura de resultados**
ezplot((2*sin((T*w)/2))/w,[-20 20]); grid on; `%Gráficamos la expresión obtenida en el command window`	

Figura 4. 18: *Gráfica de la solución encontrada*
Fuente: El autor

Preguntas
1. ¿Cuál es el código necesario para calcular la transformada de Fourier del impulso rectangular usando la función **fft**?
2. Repetir el proceso de la práctica mostrada para una función escalón unitario.
3. Para una señal aleatoria. ¿Es posible; obtener su transformada de Fourier? Explique.

CONCLUSIONES

4.3. Simulación # 3: Modulación y Demodulación de Amplitud (AM) *Simulink*

Esta práctica constará de dos apartados, los cuales tienen un fin común el cual es, mostrar el proceso para modular y demodular una señal en amplitud.

El primer apartado será simular el proceso de Modulación AM de doble banda lateral; el segundo apartado constará del proceso de Demodulación de la misma.

4.3.1. Modulación (AM)

Realizar la simulación de la modulación de amplitud (AM) cuyo comportamiento obedece al concepto teórico que indica que una señal portadora variará su amplitud de manera proporcional a la amplitud de la señal modulante y hace énfasis en utilizar una señal portadora con una frecuencia superior a la del mensaje.

TRABAJO PREPARATORIO

- Consultar las operaciones aritméticas factibles entre señales sinusoidales de distintas frecuencias.
- Consultar las diferencias entre los varios tipos de modulación AM (hacer referencia por lo menos 3 de ellas)
- Con la ayuda de ***Matlab/Simulink*** realizar operaciones aritméticas entre sinusoides y mostrar sus resultados.
- Usando ***Simulink***; diseñar y gráficar sinusoides con diferentes amplitudes y frecuencias.
- Con la ayuda de los bloques de ***Simulink*** gráficar el espectro en frecuencia de una señal sinusoidal.
- Consultar en la librería **EXTRAS** de ***Simulink*** el bloque *"power spectral density"*; como colocar los parámetros de manera que el resultado mostrado sea el esperado.

TRABAJO PRÁCTICO
Sustento teórico
Conceptos:

- **Banda base:** Representa la banda de frecuencias que contiene la señal de interés o mensaje.
- **Moduladora:** Este es el nombre con el que se conoce a la señal que representa el mensaje enviado desde el transmisor ***m(t)***; generalmente para modulación analógica esta señal tiene frecuencia inferior a la de la portadora.
- **Portadora:** Señal ***c(t)*** de alta frecuencia, que se modifica de acuerdo a los parámetros de una señal moduladora ***m(t)***.

- **Señal modulada:** La señal modulada ***s(t)*** es el resultado de la interacción entre la señal moduladora ***m(t)*** y la portadora ***c(t)***.

La portadora siempre tendrá la siguiente forma; ***c(t) = Ac*cos*(2π*fc*t + φc)***, para modulación analógica, donde:

- **Ac=** Amplitud de la señal portadora
- **fc=** Frecuencia de la señal portadora
- ***φc=*** Fase de la señal portadora

La señal modulada resulta entonces:

$$s(t) = m(t)c(t)$$

$$s(t) = Ac*m(t)*cos(2\pi f_c t + \varphi_c)$$

Con ayuda de los bloques de ***Simulink*** para modelado; diseñar un sistema que simule un modulador/demodulador de amplitud y mostrar sus resultados usando los bloques de visualización del mismo.

Objetivos

- Realizar un modulador AM convencional o (DSB-SC) y comprobar la ventaja que presenta este tipo de modulación en la recepción de la señal de información.
- Observar el espectro en el tiempo de la señal modulada en AM
- Observar el espectro en frecuencia de la señal modulada en AM

Librerías utilizadas

- ***Simulink***/Sources
- ***Simulink***/Sinks
- ***Simulink***/Math operators
- ***Simulink***/Extras

Desarrollo:

Generar una señal AM de doble banda lateral y portadora suprimida y visualizar sus resultados.

Para generar un modulador AM de doble banda lateral con portadora suprimida, un esquema posible es el siguiente:

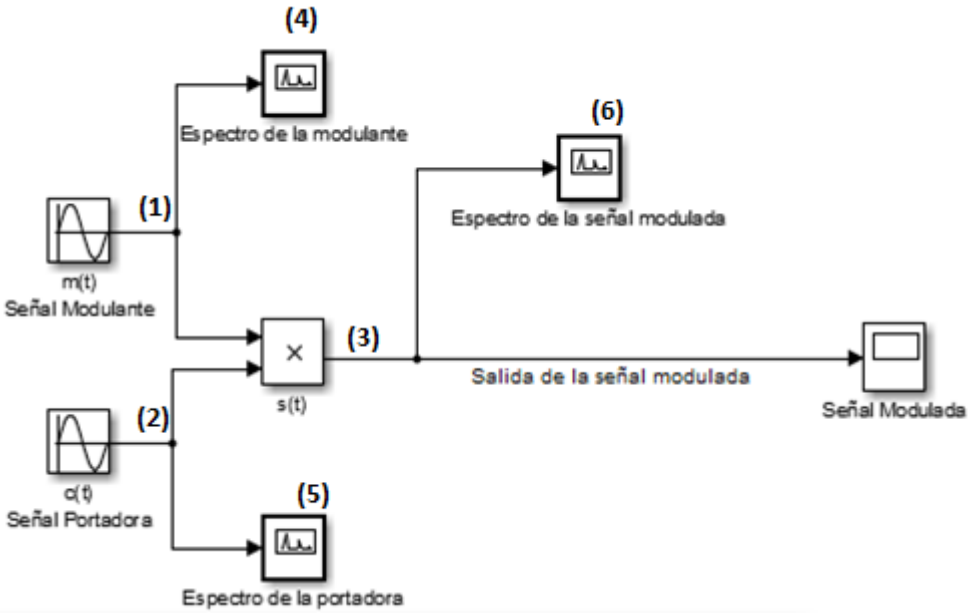


Figura 4. 19*:* Modulador AM Doble banda lateral con portadora suprimida
Fuente: El autor

El Apéndice B contiene los bloques con los parámetros que fueron configurados para obtener el resultado mostrado.

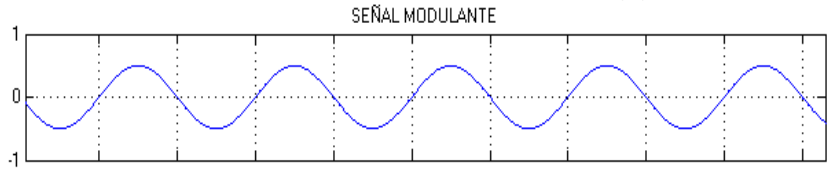
Mensaje:	**Observación/lectura de resultados** (1)
• Tipo: sinusoidal • Frecuencia: 50 rad/s • Amplitud cero pico: 0.5	

Figura 4. 20: Señal modulante
Fuente: El autor

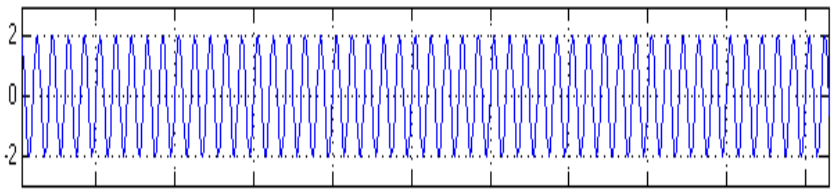
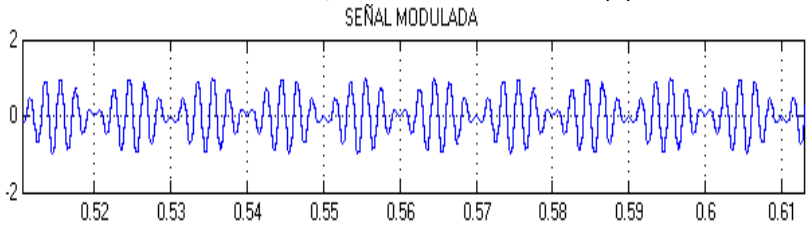
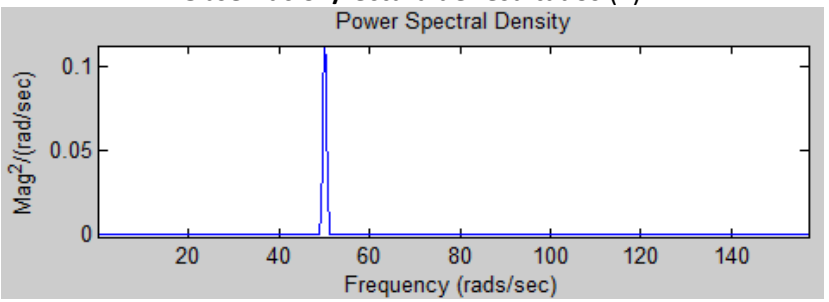
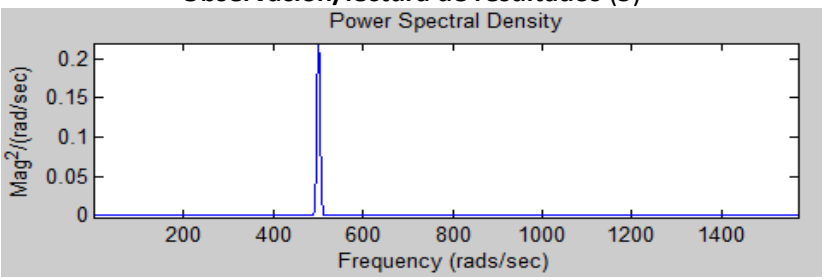
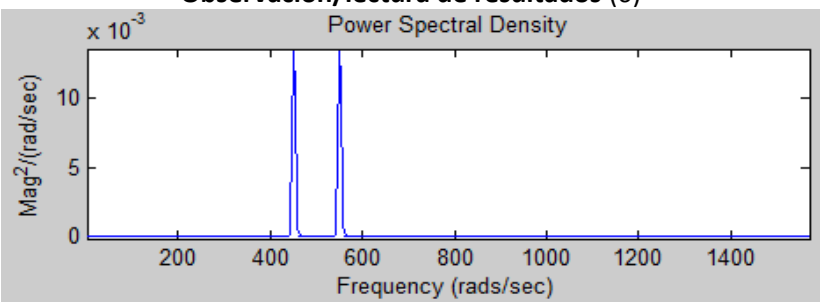
Portadora:	**Observación/lectura de resultados** (2)
• Tipo: sinusoidal • Frecuencia: 500rad/s • Amplitud cero pico: 2	

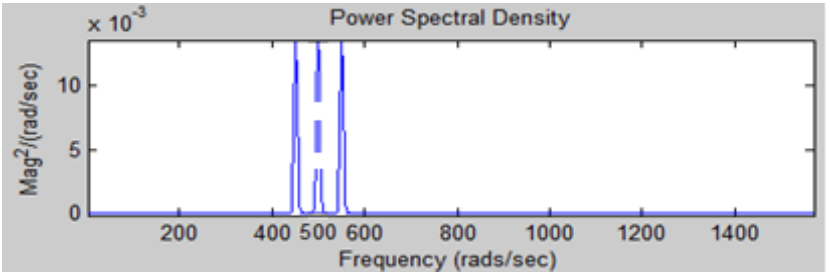
Figura 4. 21: Señal portadora
Fuente: El autor

Señal AM: $s(t) = Ac*m(t)*\cos(2\pi f c\ t + \varphi c)$	**Observación/lectura de resultados** (3)  **Figura 4. 22:** Señal modulada Fuente: El autor

Espectro de la señal modulante: El espectro se centra en la frecuencia fundamental de la señal modulante, es decir; 50 rad/seg	**Observación/lectura de resultados** (4)  **Figura 4. 23:** Espectro señal modulante Fuente: El autor

Espectro de la señal portadora: El espectro se centra en la frecuencia fundamental de la señal portadora, es decir; 500 rad/seg	**Observación/lectura de resultados** (5)  **Figura 4. 24:** Espectro señal portadora Fuente: El autor

Espectro de la señal modulada: El espectro muestra: **Banda lateral inferior:** $fc - fm$=450 rad/seg **Banda lateral superior:** $fc + fm$=550 rad/seg	**Observación/lectura de resultados** (6)  **Figura 4. 25:** Espectro señal modulada Fuente: El autor

Espectro de la señal modulada: Este sería el espectro de la señal de no usarse portadora suprimida.	**Observación/lectura de resultados**  **Figura 4. 26:** Espectro señal modulada Fuente: El autor

Preguntas
1. Observar el espectro en frecuencia de la señal modulada. ¿Cómo se puede calcular el ancho de banda de la señal transmitida?
2. Observar el espectro en frecuencia de la señal modulada. ¿Qué representan los dos impulsos que se avistan?
3. ¿Que interpretación tiene el espectro en frecuencia de la señal modulada? Explique .
4. Rediseñar el circuito del modulador y represente la ecuación de la modulación AM $s(t) = Ap(1 + m(t) * \cos(2 * \text{pi} * \text{fc})$

CONCLUSIONES

4.3.2. Demodulación AM

La demodulación AM estándar se consigue multiplicando la señal portadora por la señal modulada y aplicando un filtro pasa-bajo a la salida de esta interacción entre las señales mencionadas.

TRABAJO PREPARATORIO

- Consultar el proceso necesario; para detectar la señal de información en AM de Doble banda lateral con portadora suprimida.
- Con ayuda de ***Simulink*** modelar el diseño de un filtro pasa-bajo y mostrar su respuesta en frecuencia.
- A que refiere el **orden de un filtro**?

TRABAJO PRÁCTICO

Sustento teórico

Conceptos:
Si se conoce la frecuencia de la señal de información, se puede colocar la frecuencia de corte del filtro pasa bajo de manera tal; que no se eliminen las frecuencias de interés y podamos tener la señal enviada.
El proceso de detección de la modulación AM es prácticamente sencillo, se trata de hacer un circuito multiplicador y aplicar un filtrado pasa bajo.

Objetivos

- Observar la importancia de fijar la frecuencia de corte de un filtro.
- Comprobar la ventaja que tiene utilizar la modulación AM de doble banda lateral en el momento de recuperación de la señal.
- Observar el proceso requerido para la recuperación de la señal.

Librerías utilizadas

- ***Simulink***/Sources
- ***Simulink***/Sinks
- ***Simulink***/Math operators
- ***Simulink***/Extras
- ***Simulink***/DSP System toolbox

Desarrollo:
Obtener el mensaje a partir de una señal AM de doble banda lateral y portadora suprimida generada y visualizar sus resultados.
Para obtener un demodulador de AM, un esquema posible es el siguiente:

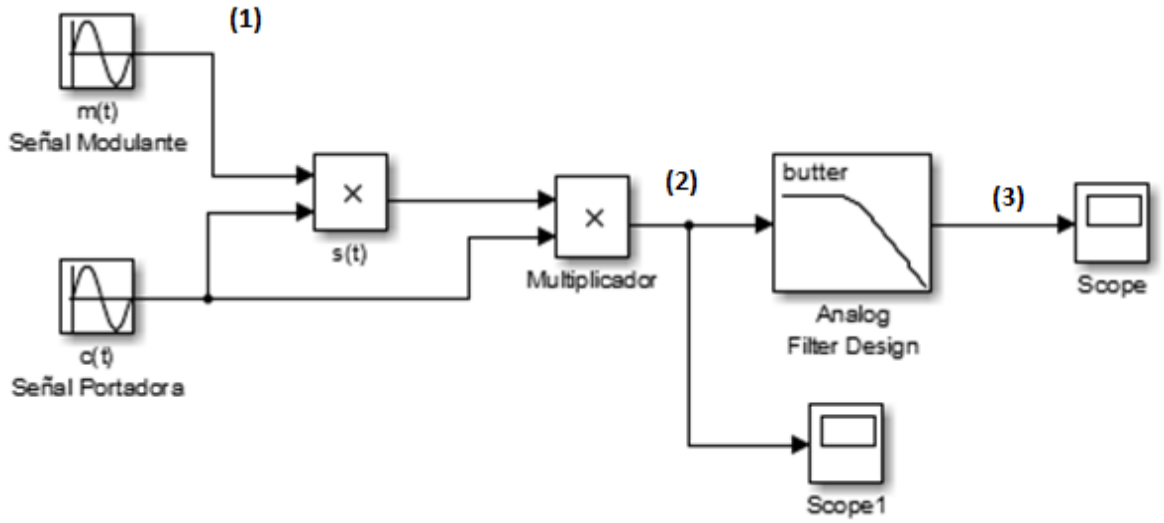
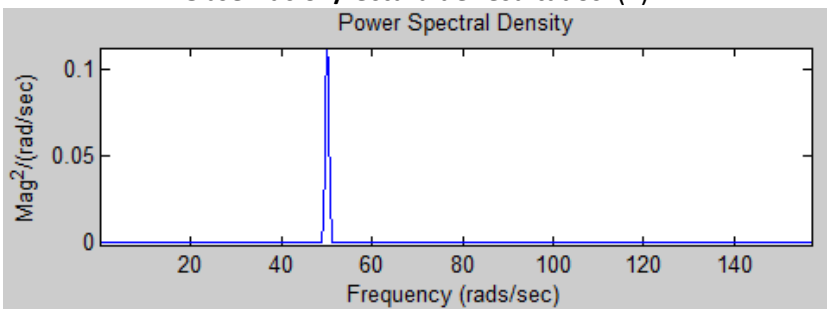
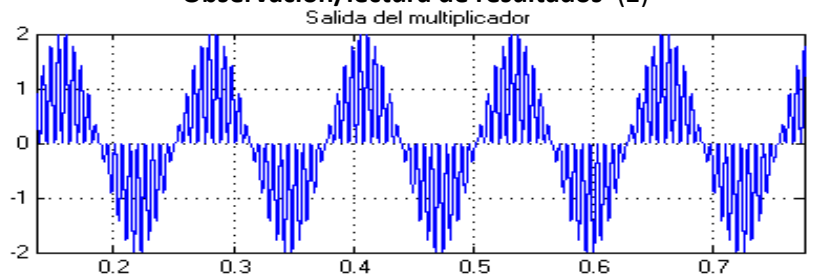


Figura 4. 27: Modulador/Demodulador AM (DBLPS)
Fuente: El autor

El Apéndice B contiene los bloques con los parámetros que fueron configurados para obtener el resultado mostrado.

Mensaje: La frecuencia del mensaje se encuentra centrada en 50 rad/seg.	**Observación/lectura de resultados** (1)  **Figura 4. 28:** Espectro señal modulante Fuente: El autor

1er paso para la Demodulación: $U(t)=s(t)*\cos(2\pi f_c t + \varphi_c)$	**Observación/lectura de resultados** (2)  **Figura 4. 29:** Detector de envolvente Fuente: El autor

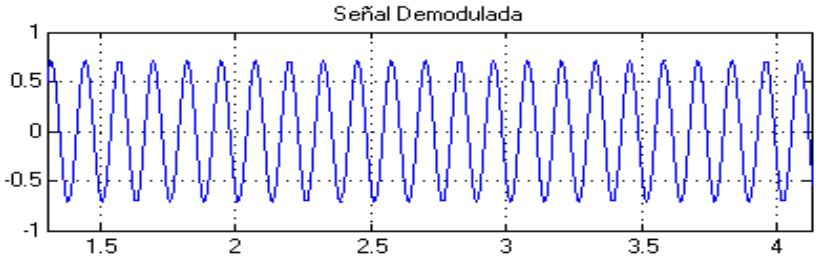
Mensaje:	**Observación/lectura de resultados** (3)
El mensaje obtenido tiene una amplitud proporcional a al mensaje enviado.	

Figura 4. 30: Salida del filtro pasabajo

Fuente: El autor

Preguntas
1. Observar el espectro en el tiempo de la señal Demodulada. ¿Por qué la señal recibida no es una réplica exacta de la señal enviada? Explique.
2. ¿Cuál debe ser la frecuencia de corte del filtro pasa bajos sí se coloca un mensaje con frecuencia de 100Hz?
3. Duplique el orden del filtro pasa bajos. ¿Qué ocurre? Explique.

CONCLUSIONES

4.4. Simulación # 4: Modulación y Demodulación de Frecuencia (FM) *Simulink*.

Esta práctica constará de dos apartados; el primero contendrá lo relacionado a un modulador FM y el segundo será el que contenga lo relacionado a la demodulación de la misma.

4.4.1. Modulación FM

En esta práctica simularemos la modulación FM. El comportamiento de este tipo de modulación obedece al concepto teórico que indica que la frecuencia de la señal portadora varía de manera proporcional a la amplitud de la señal modulante.

TRABAJO PREPARATORIO

- En ***Simulink***, consulte el bloque *Saturation* sus Configuraciónes y aplicaciones.
- En ***Simulink*** librería Communications Systems, consulte el bloque *FM Modulator*.
- Consultar que es un PLL y la utilidad de este tipo de circuitos en la modulación FM.
- En ***Simulink*** consultar; el bloque *Math Function*.
- Consultar los bloques Derivator e Integrator de la librería *Maths* de ***Simulink***.
- Consulte como modelar ecuaciones usando ***Simulink***

TRABAJO PRÁCTICO
Sustento teórico
Conceptos:

Se obtiene una señal FM regida bajo la siguiente ecuación:

$$y(t) = A * \cos *[w_0 * \mathrm{t} + 2\pi * \Delta\mathrm{fm} \int m(x) * dx]$$

De donde:

- $m(x)$ es la señal moduladora y está normalizada $\max[m(x)] = 1$ por lo que Δfm es la máxima desviación de frecuencia en la modulación.
- A*cos representa la señal portadora con frecuencia instantánea variante en el tiempo.

La variación del ángulo debe ser lineal ya que es un requisito para obtener una señal FM. Si se integra primero la señal de información f(t) y se usa para

modular una portadora en frecuencia el resultado es una onda modulada en fase, cualquier variación en fase produce una variación en frecuencia.

Según el teorema de Armstrong al integrar la señal moduladora m(t) y usarla para modular en fase a una señal portadora, se obtiene una señal modulada en frecuencia.

Objetivos

- Realizar un modulador FM indirecto.
- Observar el espectro en el tiempo de la señal modulada en FM
- Tener la capacidad de representar ecuaciones de un sistema, mediante los bloques de ***Simulink***.
- Conocer la ventaja de aplicar el método indirecto de modulación FM, en cuanto a complejidad se refiere.

Librerías utilizadas

- ***Simulink***/Sources
- ***Simulink***/Sinks
- ***Simulink***/Math operators
- ***Simulink***/Comonly used blocks
- ***Simulink***/Continuous

Desarrollo

Generar un modulador FM indirecto.

Para generar un modulador FM usando el método indirecto, un esquema posible es el siguiente:

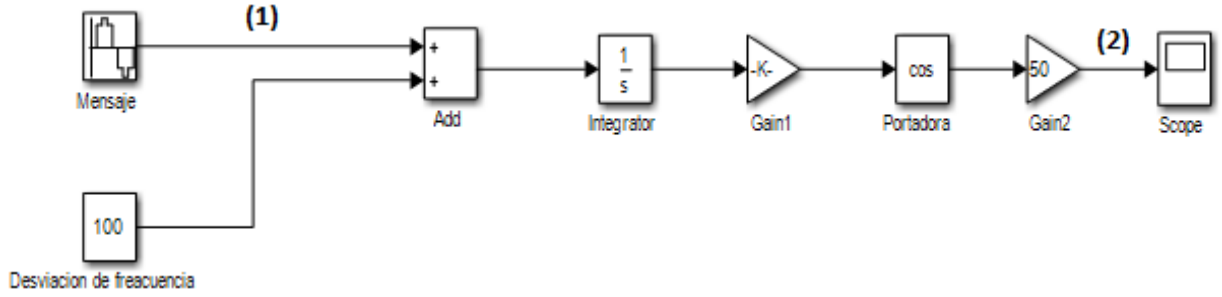


Figura 4. 31: Modulador FM indirecto

Fuente: El autor

El Apéndice B contiene los bloques con los parámetros que fueron configurados para obtener el resultado mostrado.

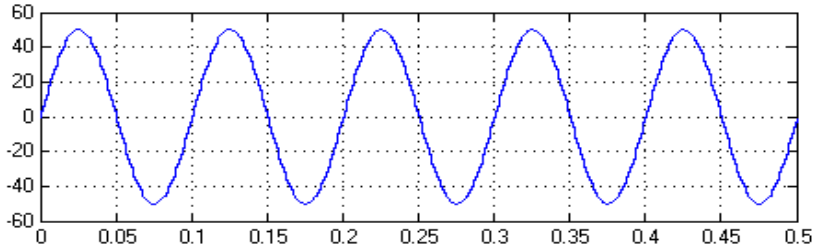
Mensaje	**Observación/lectura de resultados** (1)
• Tipo: sinusoidal • Frecuencia: 2*pi*10 rad/s • Amplitud cero pico: 50	

Figura 4. 32: Señal modulante
Fuente: El autor

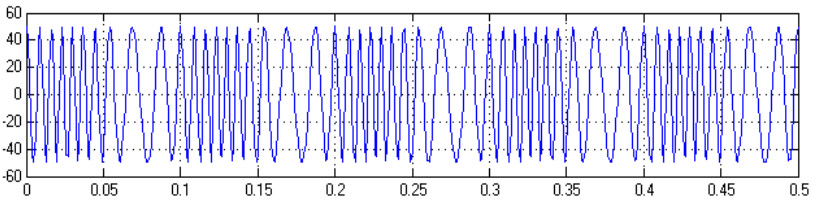
Señal modulada	**Observación/lectura de resultados** (2)
$y(t) = A * \cos *[w_0 * \mathrm{t} + 2\pi * \Delta\mathrm{fm} \int m(x) * dx]$	

Figura 4. 33: Señal modulada en FM
Fuente: El autor

$$(t) = A * \cos *[w_0 * \mathrm{t} + 2\pi * \Delta\mathrm{fm} \int m(x) * dx]$$

Preguntas
1. ¿Cuál es el ancho de banda de la señal modulada en FM?.
2. Coloque el bloque de Power Spectral Density a la salida de la salida de la señal modulada. ¿Qué valor debe colocarse en el parámetro **"sample time"** del bloque?
3. Variar la amplitud de la señal modulante a 200, ¿Qué efecto tiene en frecuencia de la señal modulada?

CONCLUSIONES

4.4.2. Demodulación FM

TRABAJO PREPARATORIO

- En ***Simulink*** usando la librería Communications Systems, consulte el bloque *FM Demodulator*.
- Consultar que es un PLL y la utilidad de este tipo de circuitos en la demodulación FM.
- Consulte como realizar la demodulación FM sin usar PLL.

TRABAJO PRÁCTICO

Sustento teórico
Conceptos:
Para recuperar el mensaje utilizaremos el proceso inverso es decir; se debe derivar la señal para recuperar la señal en fase, esto porque aplicamos el método indirecto para generar una señal FM; caso contrario deberíamos usar un PLL para detectar los cambios de fase y poder obtener el Mensaje enviado.

Librerías utilizadas
- ***Simulink***/Sources
- ***Simulink***/Sinks
- ***Simulink***/Math operators
- ***Simulink***/Comonly used blocks
- ***Simulink***/Continuous
- ***Simulink***/Discontinuities
- ***Simulink***/Definied functions
- ***Simulink***/DSP System toolbox

Desarrollo

Recuperar la señal modulante a partir del modulador FM indirecto generado anteriormente.
Un posible esquema para el Demodulador FM es presentado a continuación:

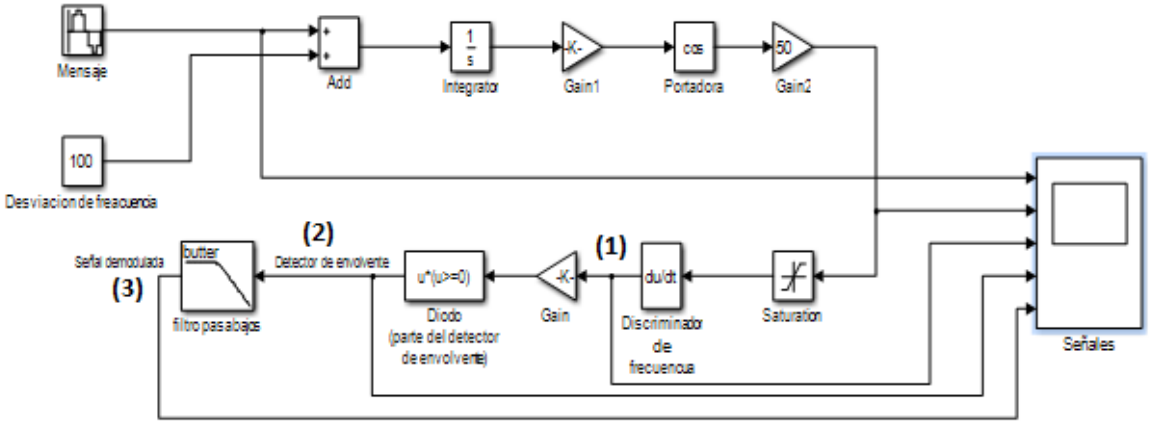


Figura 4. 34: Modulador/Demodulador FM indirecto

Fuente: El autor

El Apéndice B contiene los bloques con los parámetros que fueron configurados para obtener el resultado mostrado.

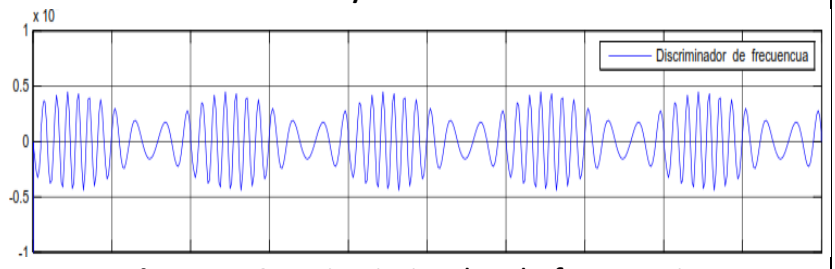
Discriminador de frecuencia Se obtiene una señal modulada en AM, con lo que ya conocemos a partir de este paso. ¿Cómo podríamos recuperar la señal aplicando filtrado pasa bajos?	**Observación/lectura de resultados** 

Figura 4. 35: Discriminador de frecuencia

Fuente: El autor

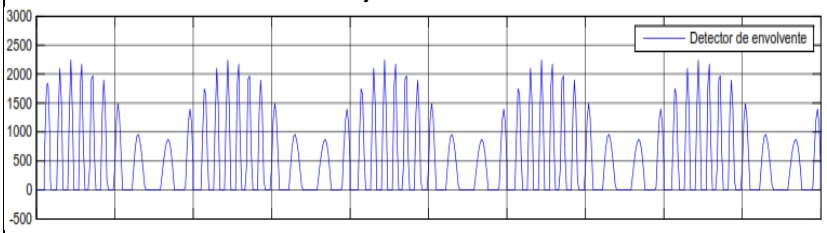
Detector de envolvente Conservamos solamente la parte positiva de la señal.	**Observación/lectura de resultados** 

Figura 4. 36: Detector de envolvente

Fuente: El autor

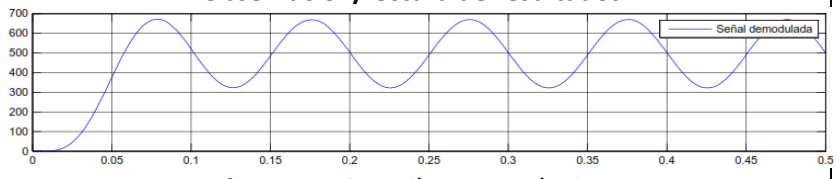
Señal demodulada La señal recuperada se encuentra oscilando por encima del umbral del voltaje del mensaje enviado.	**Observación/lectura de resultados** 

Figura 4. 37: Filtro Pasabajos

Fuente: El autor

Preguntas
1. ¿Qué tipo de señal muestra la salida del Discriminador de frecuencia? Explique este fenómeno.
2. ¿Qué utilidad pudo comprobar en el bloque Gain?
3. ¿Qué otros componentes se necesita para obtener la señal del mensaje enviado replicado en el receptor de manera exacta?

CONCLUSIONES

4.5. Simulación # 5: Transmisión de señales en sistemas básicos de telecomunicaciones.

Esta práctica constará de dos apartados, que tienen como fin mostrar el proceso para enviar una señal desde un transmisor y recibirla en otro extremo llamado receptor, usando las técnicas de modulación disponibles.

4.5.1. Transmisión de una señal

Para transmitir una señal, se necesita realizar un proceso que involucra una adecuación en la misma, de manera que se pueda enviar por un canal de comunicaciones. En este proceso, intervienen las técnicas de modulación, las cuales pueden ser análogas o digitales y ya que la tendencia de la mayoría de transmisiones actuales involucra el envío de señales digitales; se desarrollan varias técnicas para lograr obtener señales de este tipo, sobre todo si estas, tienen características de naturaleza análogas, como por ejemplo la voz humana.

TRABAJO PREPARATORIO

- Consultar la teoría de muestreo de Nyquist.
- De la librería ***Sources***, consultar el bloque de ***Simulink*** llamado Zero-Order hold.
- Consultar acerca del efecto aliasing.
- Consulte la referencia de niveles de cuantización.

TRABAJO PRÁCTICO

Sustento teórico
Conceptos:

Existen diversas combinaciones y técnicas para transportar una señal. La técnica empleada dependerá del tipo de aplicación para el que este destinado la señal de información. Las técnicas de modulación han ido evolucionando, de tal manera, que podemos encontrar diversas categorías para los moduladores, entre los que destacan los moduladores de señal en banda base, banda ancha, etc.

La tendencia de las transmisiones actuales requieren el uso de señales digitales, por esa razón la señal debe ser tratada previamente a entrar al modulador y ser enviada. Entre los requerimientos necesarios para enviar una señal de información encontramos: la codificación y el muestreo, siendo

este último el proceso necesario para cambiar una señal de característica análoga a digital.

El Transmisor contiene un generador de señales, el cual puede ser sinusoidal y actuará como mensaje. Este ingresa a la etapa de muestreo que se encarga de volver la entrada análoga en una equivalente digital, para lo cual según el criterio de Nyquist la tasa mínima de muestreo debe ser igual a 2 veces la frecuencia máxima de la señal muestreada. Los niveles de voltaje de la señal digital obtenida pueden ser codificados para que cada uno de ellos represente un símbolo, a esto se le conoce como cuantización. En el proceso de cuantización dependiendo de los niveles a codificar, tendremos señales binarias o señales de mucha más complejidad conocidas como señales M-arias, donde M es el número de bits que se usará para codificar cada nivel de la señal, y viene definido por la siguiente formula: $\boldsymbol{M = log_2 L}$; donde L es el número de niveles empleados para cuantizar la señal. En este caso veremos el caso más simple que sería la generación de una señal binaria, es decir la presencia de voltaje representará un 1L y la ausencia de la misma representara un 0L. Luego de esto, la señal pasa a un modulador y finalmente se envía.

En un sistema básico de telecomunicaciones puede no ser considerada la presencia del ruido, esto supondría que lo que se tiene es una transmisión en condiciones ideales y serviría como introducción hacia el análisis de circunstancias más complejas.

Librerías utilizadas

- ***Simulink***/Sources
- ***Simulink***/Sinks
- ***Simulink***/Extras
- ***Simulink***/Communications system toolbox
- ***Simulink***/Discrete
- ***Simulink***/DSP system toolbox

Desarrollo

Un posible esquema para representar un sistema transmisor es presentado a continuación:

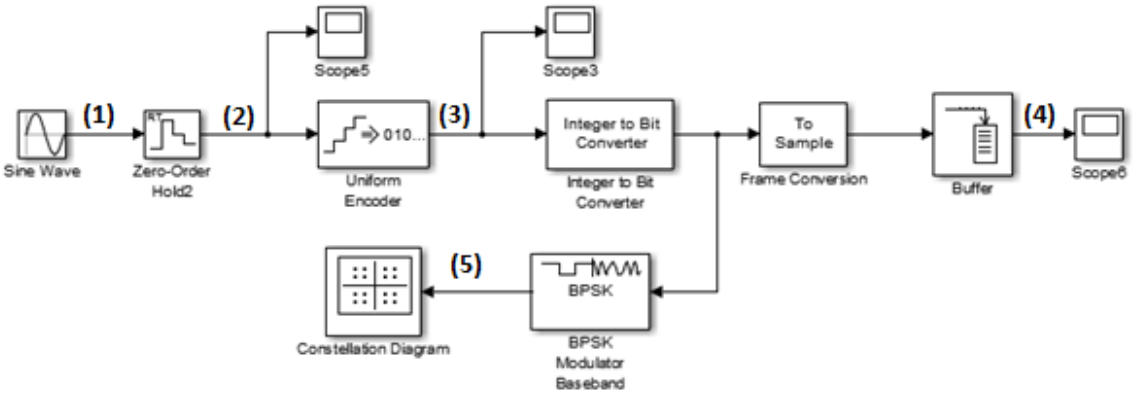


Figura 4. 38: Transmisión de una señal y modulación en BPSK
Fuente: El autor

El Apéndice B contiene los bloques con los parámetros que fueron configurados para obtener el resultado mostrado.

El diagrama implementado nos devuelve los siguientes resultados:

Mensaje

- Tipo: sinusoidal
- Frecuencia: 2*pi*10 rad/s
- Amplitud cero pico: 2

Observación/lectura de resultados (1)

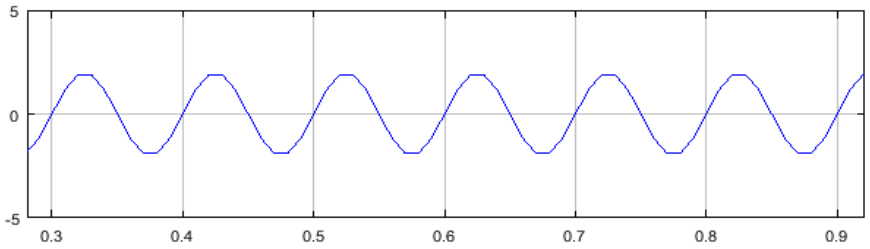


Figura 4. 39: Señal moduladora
Fuente: El autor

Muestreo:

La frecuencia mínima de muestreo para la señal de entrada será:

$$fs = 2 * fmax$$

$$fs = 2 * 10 = 20$$

Observación/lectura de resultados (2)

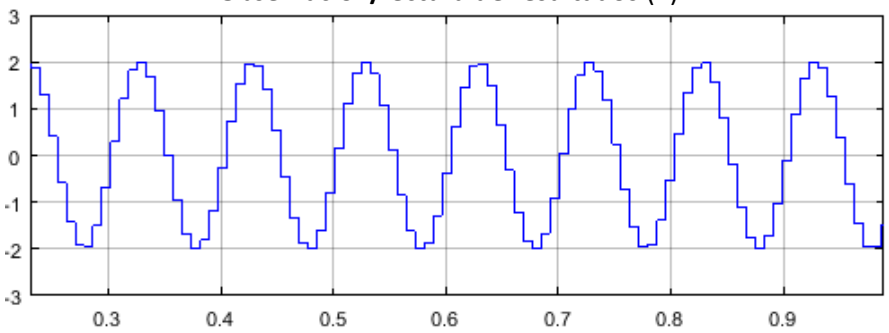
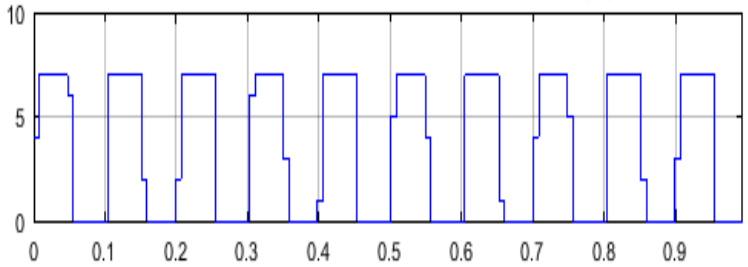
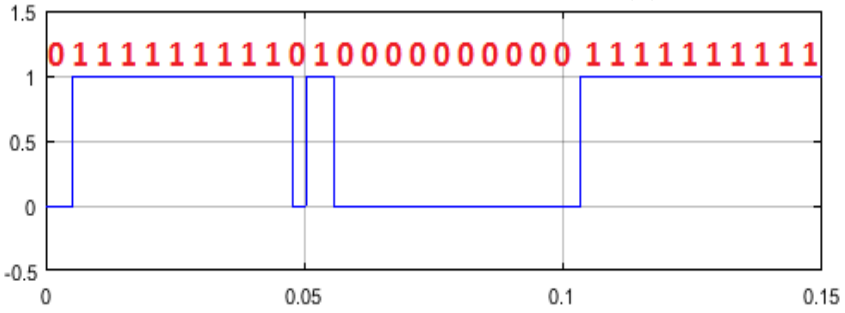
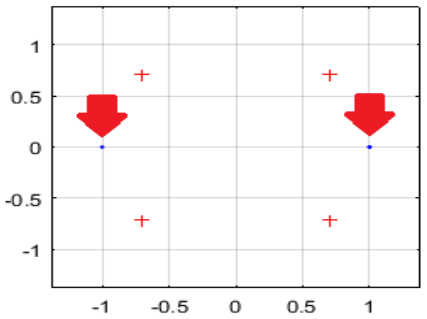


Figura 4. 40: Señal muestreada
Fuente: El autor

Cuantizador:	**Observación/lectura de resultados** (3)
Señal obtenida con 8 niveles para representar los voltajes de la señal muestreada Teniendo una señal discreta en amplitu y en tiempo con valores para la amplitud desde 000 hasta 111 Para los 8 niéveles de cuantización.	 **Figura 4. 41:** Señal cuantificada Fuente: El autor

Codificador:	**Observación/lectura de resultados** (4)
Señal binaria codificada obtenida con 8 niveles Teniendo una señal binaria.	 **Figura 4. 42:** Señal codificada Fuente: El autor

Modulación BPSK	**Observación/lectura de resultados** (5)
Se transmiten 2 bits y la constelación nos muestra que se encuentran espaciados lo suficiente como para evitar el aliasing o interferencia entre símbolos.	 **Figura 4. 43:** Constelación de bits en BPSK Fuente: El autor

Preguntas
1. ¿Qué beneficio se obtiene al volver una señal análoga digital? Explique,
2. ¿Por qué con la frecuencia mínima de muestreo, la señal muestreada no tiene una forma legible? Explique.

3. Modificar la frecuencia de muestreo y observar los resultados para explicar. ¿Qué sucede al aumentar la frecuencia del muestreador?
4. ¿Por qué mientras más aumenta la frecuencia de muestreo la señal de salida, se parece más a la original?

CONCLUSIONES

4.5.2. Recepción de una señal

Como en todos los tipos de modulación antes estudiados, para obtener la señal original se debe realizar el proceso inverso al realizado en el Modulador. En este caso se hará todo el proceso inverso realizado en el transmisor de manera que se obtenga una réplica casi exacta de la señal original en el receptor.

TRABAJO PREPARATORIO

- De la librería ***System Comunications Toolbox***, consultar el bloque de ***Simulink*** llamado *BPSK Demodulator*.
- Consultar en que consiste la Demodulación coherente.
- Consultar de la librería ***Utility Blocks*** el bloque Integer to bit converter
- Consultar como realiza el proceso de Demodulación en esquemas de modulación de pulsos banda base.

TRABAJO PRÁCTICO
Sustento teórico
Conceptos:

Se asume que a la entrada del demodulador se tiene una sucesión de bits, por lo que solamente se debe hacer la detección de los mismos mediante un decodificador y suavizar su forma haciéndolos pasar por un filtro pasabajas, es decir haciendo que pierdan su forma rectangular de manera que se pueda recuperar la señal sinusoidal.

Hay que recalcar, que en este esquema sugerido no interviene el ruido, por lo que se diría que la señal se envía en condiciones ideales y los pulsos no se ven alterados en su forma.

El esquema sugerido se muestra a continuación.

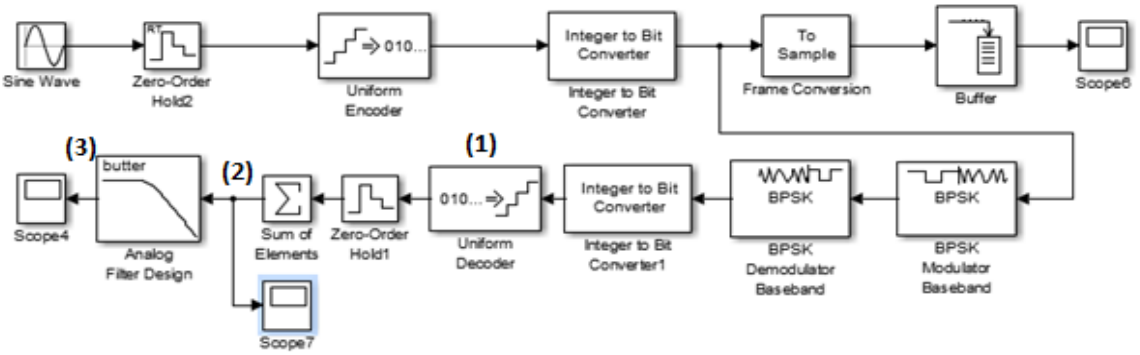
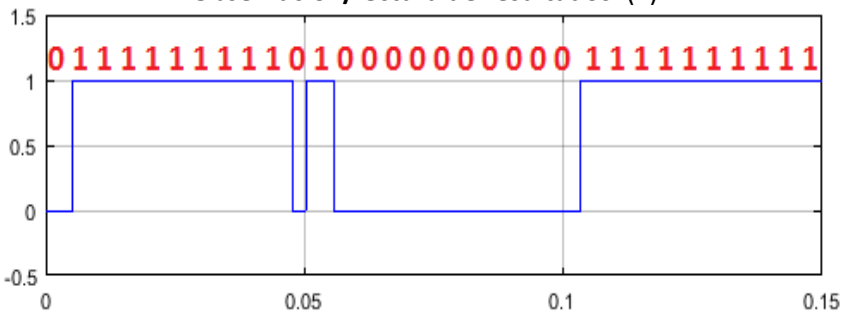
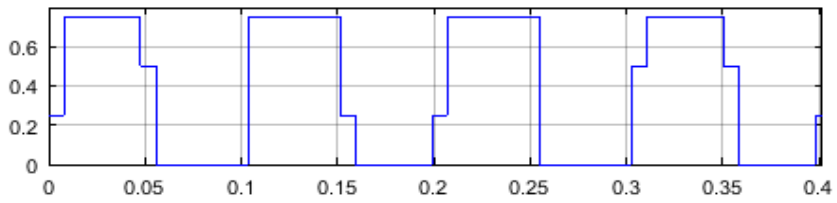


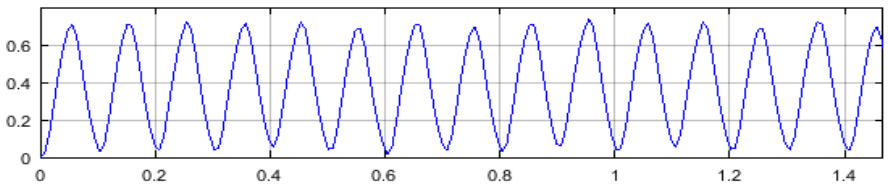
Figura 4. 44: Recepción de una señal y demodulación en BPSK o 2QAM

Fuente: El autor

El Apéndice B contiene los bloques con los parámetros que fueron configurados para obtener el resultado mostrado.

Demodulación 2-QAM Señal que llega al receptor luego de pasar por el demodulador BPSK.	**Observación/lectura de resultados** (1)  **Figura 4. 45:** Señal Recibida Fuente: El autor

Decodificador: Señal obtenida a la salida del Decodificador modulada en 2-QAM.	**Observación/lectura de resultados** (2)  **Figura 4. 46:** Señal Decodificada Fuente: El autor

Filtro pasabajos: La salida del filtro pasabajos muestra una señal sinusoidal	**Observación/lectura de resultados** (3)  **Figura 4. 47:** Señal Recuperada Fuente: El autor

Preguntas
1. Observando la señal a la salida del Demodulador. Explique; ¿Por qué la señal recibida no es una réplica exacta de la señal enviada?
2. ¿A qué frecuencia debe centrarse el filtro análogo?

3. Observando en el diagrama la sección del demodulador. ¿Cuál es la función del Zero Order-Hold? Explique.

CONCLUSIONES

CAPITULO V

CONCLUSIONES

- El estudiante que desea iniciarse en ***Matlab*** encontrará toda la ayuda necesaria gracias a la facilidad y flexibilidad que el software presenta, además de que puede encontrar gran contenido de ejemplos del uso de comandos y bloques de ***Simulink*** en el sitio oficial del fabricante del programa (***www.Mathworks.com***), con esto se garantiza un entorno de aprendizaje, actualización e investigación permanente.
- Siendo ***Matlab/Simulink*** un entorno de computación práctico con posibilidad de mostrar resultados de forma inmediata mediante el empleo de gráficas, el estudiante será capaz de asimilar y analizar de manera criteriosa la teoría que lo introduce en el campo de las Telecomunicaciones. Además, conforme vaya adquiriendo experiencia en este entorno de programación, ser capaz de desarrollar posibles soluciones a otros tipos de problemas que se presentan en el transcurso de la carrera de Tecnología en Electrónica y Telecomunicaciones, de esta manera estará colaborando a las futuras generaciones.
- Mediante el aprendizaje del Software ***Matlab/Simulink*** orientado a los sistemas básicos de telecomunicaciones, el estudiante se centra simplemente en asimilar lo correspondiente a su carrera, lo que le dará una optimización en su meta profesional creando un amplio juicio en los procedimientos que han de llevarse a cabo cuando se modela uno de estos sistemas.
- Al tratarse de un software en constante crecimiento, los egresados en Tecnología en Electrónica y Telecomunicaciones, estarán en plena capacidad de proponer, ampliar y actualizar el manual de prácticas que se ha propuesto, además de presentar otros posibles diseños y soluciones para los diagramas presentados.

RECOMENDACIONES

- Revisar todas las materias en las que sea posible la adaptabilidad al software ***Matlab/Simulink***, de manera que los estudiantes en Tecnología en Electrónica y Telecomunicaciones encuentren en este software, una herramienta actualizada con la cual puedan trabajar a lo largo de sus estudios.

- El personal docente debe ampliar los espacios dedicados al debate que generan las prácticas de laboratorios, con esto el enriquecimiento y fortalecimiento de los conocimientos del estudiante serán resultados positivos en el futuro.

- Impartir la materia de manera técnica, de manera que el estudiante asimile lo que se le está difundiendo en la teoría con ejemplos reales, y con esto se satisfaga el aprendizaje basado en contenidos innovadores en el laboratorio de la materia Telecomunicaciones I.

BIBLIOGRAFÍA

- Dolores M. Etter. *Solución de problemas de ingeniería con **MATLAB** Segunda edición (1997).*
- Moore, Holly. *Matlab para Ingenieros Primera edición (2007).*
- Amos Gilat. ***Matlab** una introducción con ejemplos prácticos Segunda edición (2005).*
- Steven T. Karris. *Introdution to **Simulink** with Engineering Applications Primera edición(2006).*
- Steven T. Karris. *Signals and systems with **Matlab** computing and **Simulink** Modeling Tercera edición (2007).*
- Luis F. Chaparro. *Signals and Systems using **Matlab** Primera edición (2011).*
- Wayne Tomasi. *Sistemas de Comunicaciones Electrónicas Segunda edición (2006).*
- William Stallings*. Comunicaciones y Redes de Computadores Quinta edición. (2008). Amos Gilat. Matlab (2009), Recuperado el 28 de septiembre del 2011.*
- José E. Briceño M. *Principios de las comunicaciones Tercera Edición (2005).*
- Luis Montoto. *Fundamentos físicos de la información y las comunicaciones (2004)*.
- Enrique Sanchis. *Fundamentos y Electrónica de las comunicaciones (2004)*.
- MathWorks. *Symbolic Math Toolbox User´s Guide.*
- Simon Haykin. *Sistemas de comunicación Primera edición (2001).*
- José E. Briceño M. *Transmisión de Datos* (2005)
- Bernard Sklar. *Digital comunications Fundamental and Applications Segunda edición (2005)*

REFERENCIAS WEB

- www.books.google.com/books?id=dAvjPaUFdAMC&printsec=frontcover&dq=M
- TYHrPOfd0QH11MzPBQ&sa=X&oi=book_result&ct=result&resnum=1&ved=0C
- *https://books.google.com.ec/books?isbn=9681857526*www.books.google.com/books?id=45HN_RhzdkYC&printsec=frontcover&dq=fu
- https://books.google.com.ec/books?id=njgqBAAAQBAJ&pg=PA1&dq=symbolic+math+toolbox+matlabwww.books.google.com/books?id=kZKsps2_F4YC&printsec=frontcover&dq=fun

- https://books.google.com.ec/books?id=2rpuFsX5fUMC&printsec=frontcover&dq=Matlab+computing+and+simulink+modeling&hl=es-419&sa=X&ei=H9FrVYT3B4WdgwSWgIGACg&ved=0CBwQ6AEwAA#v=onepage&q=Matlab%20computing%20and%20simulink%20modeling&f=false

ANEXOS

ANEXO A – FUNCIONES Y COMANDOS DE *MATLAB*[82]

ANEXO B – CONFIGURACIÓN DE LOS BLOQUES DE SIMULINK USADOS EN LAS SIMULACIONES DEL CAPÍTULO 4

ANEXO C – RESPUESTAS DE LAS SIMULACIONES DEL CAPÍTULO 4

82

http://www.icm.espol.edu.ec/jornadas/14/archivos/Diapositivas/Rodr%C3%ADguezLuis/Rodr%C3%ADguezLuis_***Matlab***_como_soporte_para_c%C3%A1lculos_en_Estad%C3%ADstica.pdf

ANEXO A

FUNCIONES Y COMANDOS DE *MATLAB*

1. Símbolos especiales en *MATLAB*

[ ]	para definir vectores y matrices
( )	para definir precedencia en expresiones y para subíndices
,	para separar elementos de un vector use comas o espacios
;	para separar filas y para evitar mostrar contenido de variables
%	para iniciar un comentario (programas y funciones)
...	para continuar un comando en la siguiente línea

2. CÁLCULO NUMÉRICO

2.1 Formatos de exhibición de números en la pantalla

```
>> format long                  muestra 14 decimales
>> x=exp(2)                     un ejemplo para visualizar
    x =
        7.38905609893065
>> format bank                  formato para 2 decimales
>> x
   x =
        7.39
>> format rat                   notación racional (fracciones)
>> x
    x =
        2431/329
>> format short e               notación científica
>> x
    x =
         7.3891e+00

>> format long e                notación científica con 14 decimales
>> format +                     muestra signos +,  , -
>> format short                 4 decimales  (MATLAB lo usa por omisión)
>> format compact               suprime líneas adicionales en la salida
>> format loose                 inserta líneas en blanco en la salida(recomendado)
>> format hex                   formato hexadecimal
>> vpa(sqrt(2), 20)             variable precision arithmetic
        ans =                       (muestra la raíz cuadrada de 2 con 20
                                     dígitos
          1.4142135623730950488
>> format short                 regrese al formato normal de MATLAB
```

2.2 Operadores aritméticos

```
+ - * / \ ^ ( )          ^  se usa para potenciación
                         /  es división a la derecha
                            \  es división a la izquierda (para
                                        matrices)
>> help ops              listar los operadores y caracteres especiales
```

2.3 Funciones matemáticas

```
>> help elfun            listar las funciones matemáticas elementales
        Trigonometric
        . sin      -
        Sine.
        sinh       - Hyperbolic
        sine. asin       - Inverse
        sine.
        asinh      - Inverse hyperbolic sine.
          . . .
```

2.4 Operadores relacionales y lógicos

< <= > >= == ~= & | ~ los tres últimos corresponden a: ∧ ∨ ¬
== representa al símbolo **=**
~= representa al símbolo ≠

>> t=sin(2) < 0.8 & log(2) > 0.5 el resultado es un valor lógico (0 o 1)

2.5 Símbolos numéricos especiales

```
>> 2/0
     Inf                    es el símbolo ∞
>> 0/0
     NaN                    significa "Not A Number" (valor indeterminado)
>> pi                       contiene la constante π
>> eps                      es la precisión del tipo real en MATLAB
>> realmin                  el menor número real en MATLAB
>> realmax                  el mayor número real en MATLAB
```

2.6 Manejo de números complejos

i representa al símbolo $\sqrt{-1}$

```
>> x = 3+2i                 asignar un número complejo
>> t = 2*x + 3 - 5i         operación con números complejos
   t =
      9.0000 - 1.0000i
>> y =exp(x)                el resultado también es complejo
   y =
     -8.3585 +18.2637i
>> y = log(-2)              el referencial de MATLAB son los complejos
   y =
      0.6931 + 3.1416i
```

2.6.1 funciones para números complejos

conj, real, imag, abs, angle, complex

```
>> z=3+2i;
>> t=conj(z)                obtener el conjugado
```

3 VARIABLES

- No requieren ser declaradas
- Su tipo depende del valor asignado
- Pueden ser redefinidas
- Sensible al tipo de letra (mayúsculas o minúsculas)
- **ans** es la variable por omisión provista por ***MATLAB***
- ***MATLAB*** realiza la asignación de memoria a variables durante la ejecución.

```
>> x=3                      x es de tipo real
>> x='mensaje'              ahora x es de tipo literal (use comillas simples)
>> syms x                   x se redefine a tipo símbolo
>> x=[2 7 4]                x es ahora un vector un vector
>> x=2+3i                   x es de tipo complejo
>> x                        muestre el contenido actual de la variable
>> whos x                   muestre el tipo actual de la variable
>> disp(x)                  muestre solamente el contenido
>> x=input('¿dato?');       ingrese un valor para una variable desde el teclado
>> exp(x)/3
>> ans                      la variable ans contiene el último resultado
```

>> y=2*ans la puede usar

4 ALGUNOS COMANDOS DEL SISTEMA OPERATIVO

>> help general lista de comandos

>> who lista las variables en uso

>> whos lista las variables en uso y su descripción

>> cd c:***MATLAB***\work **cd** cambia la ruta del directorio actual lista

>> dir el contenido del directorio actual También se

lo puede hacer con las opciones de la barra

4.1 Comandos especiales

>> clock fecha hora, vea su uso con help.

>> format rat para visualizar la fecha con mas claridad

5 CADENAS DE CARACTERES

>> x='Matematica'; asignación de una cadena (use comillas simples)

>> x(4) manejo de un carácter de la cadena, use un indice

En *MATLAB* los índices se escriben entre

paréntesis y son numerados desde 1

>> t=x(2:5); manejo de una subcadena, use: **nombre(inicio: final)**

>> n=length(x) longitud de la cadena

>> c=strcat(x, t) concatenación de cadenas

>> help strfun listar las funciones para cadenas

6 VECTORES Y MATRICES

```
>> x=[3, -1, 4, 7, -2]
>> x=[3  -1  4  7  -2]
>> x(2)=5
>> y=x(2: 4)
   y =
      -1    4    7
>> t=[3; -1; 4; 5]
   t =
      3
     -1
      4
      5
>> t=x'
```

asignación directa de un vector fila
puede separar con **comas** o con **espacios**
manejo de un componente del vector.
En *MATLAB* los índices se escriben entre paréntesis y son numerados desde 1
para asignar parte de un vector use **(inicio: final)**
para asignar un vector columna use **;**
para obtener la transpuesta de un vector use **'**
x' es la transpuesta del vector **x**

```
>> y = [3, x, -6, 7]
   y =
      3    3   -1    4    7   -2   -6    7
>> y = 2:1:10
   y =
      2    3    4    5    6    7    8    9   10
```

puede asignar un vector usando otro vector
puede asignar un vector mediante una secuencia
En *MATLAB* las secuencias se escriben: valor inicial : incremento : valor final
si el incremento es 1 puede omitirlo

```
>> y=[2, 5, 4, ...
      7, -3]
>> x=[3, 5, 2, 0]
>> y=2*x
>> y=exp(x)
```

Para continuar en la siguiente línea use **...**
Escribir la continuación de la línea anterior
puede realizar operaciones escalares
o crear vectores con funciones

```
>> a = [6  3 ; 5  1]
   a =
      6    3
      5    1
>> a(2,1)
```

asignación directa de una matriz 2x2
separe elementos con espacios o comas
separe filas con punto y coma
manejo de los componentes de una matriz con índices numerados desde 1: **(fila, columna)**

```
>> a=[2, -3; 5, 1; 0, 7]
>> x=[7, 3]
>> a=[x; x]
>> b=[5, 6]
>> c=[a; b]
>> d=[a, b']
>> x=c(1, :)
>> x=c(: ,1)
>> c(:,2)=[ ]
```

una matriz 3x2
una matriz 2x2
c es una matriz aumentada 3x2
c es una matriz aumentada 2x3
asigne a **x** la primera fila de **c**
asigne a **x** la primera columna de **c**
elimine la segunda columna de **c**

6.1 Matrices especiales

```
>> a=ones(3)
>> a=ones(3,5)
>> a=zeros(4,5)
>> a=eye(5)
>> a=magic(4)
>> a=hilb(5)
```

matriz 3x3 iniciada con unos
matriz 3x5 iniciada con unos
matriz 4x5 iniciada con ceros
matriz identidad 5x5
cuadrado mágico 4x4
matriz de Hilberth 5x5

```
>> x=[2, 5, 3, 7];
>> a=vander(x)
>> a=[ ]
```

un vector
matriz de Vandermonde 4x4 usando un vector
matriz nula

6.2 Una matriz puede componerse con otras matrices

```
>> a = rand(3);          matriz 3x3 con números aleatorios
>> b = [5 3 9];          vector de tres componentes
>> e = diag(b);          matriz 3x3 con b en la diagonal
   e =
      5    0    0
      0    3    0
      0    0    9
>> c=eye(3);             matriz identidad 3x3
>> d=zeros(3);           matriz con ceros 3x3
>> t=[a e; c d]          matriz compuesta 9x9
```

6.3 Editor de vectores y matrices

En la ventana **workspace** puede activar el editor de arreglos, similar a una hoja electrónica, con el cual puede modificar con facilidad las dimensiones y el contenido de vectores y matrices.

6.4 Elementos de vectores y matrices pueden manejarse con otro vector o matriz

```
>> x=[ 8 7 9 5 6];
>> p=[2 4 1];            vector para direccionar al vector  x
>> t=x(p)                t  contiene los elementos 2, 4 y.1 del vector x
>> a=[4 7 3; 5 7 8; 6 0 9];
>> p=[1 3];              vector para direccionar las filas de la matriz  a
>> q=[2 3];              vector para direccionar las columnas de la matriz a
>> t=a(p, q)             t  contiene las filas 1 y 3, columnas 2 y 3 de a
```

6.5 Operaciones con matrices

```
>> a=[3, 2; 1, 4];
   a =
      3
      2
      1
      4
>> b=[8, 6; 5, 7];


>> c=a'                  transpuesta
   de a c =
      3
      1
      2
      4
>> c=2*a                 producto de un escalar por matriz
   c =
      6    4
      2    8
>> c=a+b                 suma de matrices
   c =
     11    8
      6   11
>> c=a*b                 producto de matrices
   c =
     34   32
     28   34
>> c=a.*b                producto elemento por elemento de matrices
   c =                          para operar elemento a elemento use un
   punto
     24   12                    antes del operador
      5
     28
```

>> c=a^2 matriz al cuadrado, equivale a: **a*a**
>> c=a.^2 **cada elemento** de la matriz **a**, elevar al cuadrado**>>**
c=a==b compare igualdad entre matrices (de igual tamaño)

el resultado es una **matriz binaria** (ceros y unos)
>> c=a~=b compare si dos matrices no son iguales
el resultado es una **matriz binaria** (ceros y unos)
>> c=a>3 compare si cada elemento de **a** es mayor a 3
el resultado es una **matriz binaria** (ceros y unos)

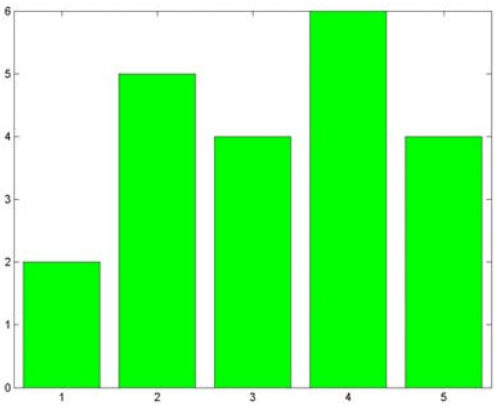
6.6 Funciones para operar con matrices

```
>> x=[-2, 0, 6, 5];          un vector para los ejemplos
>> a=[1, 2, 3; 4, 5, 6;7, 8, 9];   una matriz para los ejemplos
>> n=length(x)               longitud del vector x
>> [n,m]=size(a)             tamaño de la matriz a: el resultado es un vector
>> n                         número de filas: 3
>> m                         número de columnas: 3
>> isempty(a)                chequea si un vector o matriz está vacío
>> any(x)                    determina si el vector contiene algún valor no cero
>> any(a)                    igual que arriba, pero por columnas de la matriz
>> t=find(x)                 obtiene índices de elementos del vector no ceros
>> t=find(x>3)               obtiene los índices de cada elemento > 3
>> [f,c]=find(a)             obtiene los índices de filas y columnas de la matriz
                             cuyos elementos son no ceros

>> t=dot(x, x)               producto punto entre dos vectores
>> k=rank(a)                 rango de a
>> t=trace(a)                traza de a
>> d=det(a)                  determinante de a
>> b=inv(a)                  inversa de a
>> h=norm(a, 1)              norma de columna de la matriz a
>> h=norm(a, inf)            norma de fila de la matriz a
>> h=norm(x, inf)            norma de fila o columna del vector x
>> c=cond(a)                 número de condición de la matriz a
>> t=diag(a)                 vector con la diagonal de la matriz a
>> t=diag(x)                 matriz con x en la diagonal
>> t=rot90(a)                rote a 90 grados (sentido opuesto al reloj)
>> t=fliplr(a)               voltee horizontalmente la matriz a
>> t=tril(a)                 obtenga la matriz triangular inferior de a
>> t=triu(a)                 obtenga la matriz triangular superior de a
>> b=[5,-1; 3, 4; 2, 7];
>> b=reshape(b, 2, 3)        reconfigura la matriz b de 3x2 a 2x3

>> [t,s]=lu(a)               descomposición triangular de a en las matrices
>> t                         triangulares t y s tales que t*s es igual que a
>> s
>> t*s                       se obtiene la matriz a
>> t=cov(a)                  matriz de covarianza de a
>> e=eig(a)                  valores propios de a
>> p=poly(a)                 polinomio característico de a
>> r=roots(ans)              valores propios de a
>> help matfun               liste las funciones para matrices
```

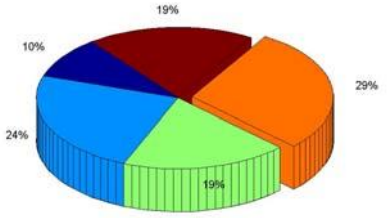
6.7 Funciones adicionales para manejo de datos con vectores y matrices

```
>> x=[2, 5, 4, 6, 4];        un vector
>> a=[5,-1; 3, 4; 2, 7];     una matriz
>> t=max(x)                  el mayor valor del vector x
   t =
      6
>> v=max(a)                  el mayor valor por columnas de la matriz a
   v =
      5     7
>> t=sum(x)                  suma de componentes
>> v=sum(a)                  suma de componentes por columnas
```

>> t=prod(x)	producto escalar
>> v=prod(a)	producto escalar por columnas
>> t=cumsum(x)	suma acumulada
>> v=cumsum(a)	suma acumulada por columnas
>> t=cumprod(x)	producto acumulado
>> v=cumprod(a)	
>> t=mean(x)	media aritmética
>> v=mean(a)	
>> t=median(x)	mediana
>> v=median(a)	
>> t=std(x)	desviación estándar
>> v=std(a)	
>> t=sort(x)	ordenamiento ascendente
>> v=sort(a)	
>> t=dsort(x)	ordenamiento descendente
>> bar(x)	diagrama de barras



>> bar(a)	
>> hist(x)	histograma
>> stairs(x)	dibuja x mediante escalones
>> pie(x)	gráfico tipo pastel
>> pie3(x)	pastel en relieve
>> v=[0,0,0,1,0]	vector para extraer sectores del pastel
>> pie3(x,v)	gráfico tipo pastel 3-d con un sector separado



7 GENERACIÓN DE NÚMEROS ALEATORIOS

>> x=**rand**	genera un número aleatorio entre 0 y 1
>> a=**rand**(5)	genera una matriz 5x5 con números aleatorios
>> b=**rand**(4,5)	genera una matriz 4x5 con números aleatorios
>> d=fix(rand*10)+1	transfomación para obtener un entero aleatorio entre 1 y 10

8 INGRESO DE PUNTOS DESDE LA PANTALLA CON EL MOUSE

```
>> ezplot('sin(x)');          ejemplo para tomar puntos desde un gráfico
>> grid on
>> [x,y]=ginput(5);           ingrese 5 puntos desde la pantalla .
                              Presione el botón del mouse para
                              ingresar cada punto
>> x                          observe las abscisas
>> y                          y las ordenadas ingresadas
>> plot(x, y, 'o')            grafique los puntos ingresados
```

9 POLINOMIOS

```
>> a=[2, -3, 0, 5],           define el polinomio  2x^3 – 3x^2 + 5
>> y=polyval(a,4)             evaluación del polinomio con un valor
>> x=roots(a)                 obtenga un vector con raíces (reales y complejas)
>> t=polyval(a, x(1))         verifique una raíz
>> p=poly(x)                  producto de todas las raíces
>> b=[3, 4, -2];              define el polinomio  3x^2+ 4x - 2
>> c=conv(a,b)                producto de polinomios
>> [c, r]=deconv(a,b);        división de polinomios
>> c                          cociente
>> r                          residuo
>> x=[2 3 5 7 8];             abscisas de puntos (x,y)
>> y=[3.2 4.1 5.8 6.4 6.3];   ordenadas de los puntos
>> z=3.2;                     valor para interpolar, z puede ser un vector
>> u=interp1(x,y,z,'linear')  resultado de la interpolación lineal
>> u=spline(x,y,z)            interpolación con un trazador cúbico
>> a=polyfit(x, y, 2);        polinomio de mínimos cuadrados de grado 2
>> a                          el vector a  contiene los coeficientes
```

10 MANEJO SIMBÓLICO

```
>> syms x;                    definición de variable tipo simbólico
>> 2*x+3*x                    suma algebraica
>> a=[x 5; 3*x 4];            matriz con elementos símbolos
>> t=inv(a)                   su inversa también contiene símbolos
>> f=3*x^2+5*x;               definición simbólica de una función
>> t=factor(f)                factorar la expresión
>> s=expand(t)                expandirla
>> e=taylor(exp(x))           expansión con la serie de Taylor
>> limit(sin(x)/x)            obtencíon de límites de funciones
>> syms y;
>> f=2*x^3+3*y^2              una función de dos variables
>> g=diff(f,x)                derivada parcial
>> u=int(f,x)                 integrar en x
>> f='2*t+1';                 definición de una función en forma literal
>> t=3;
>> y=eval(f)                  evaluación de la función
```

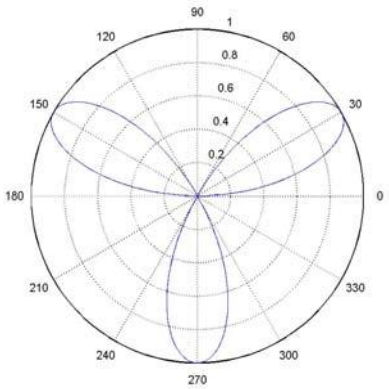
11 FUNCIONES ESPECIALES PARA MEDIR EFICIENCIA DE ALGORITMOS

```
>> tic;                       Inicia cronómetro
>> toc;                       muestra el tiempo transcurrido

>> tic; a=inv(rand(500, 500)); toc
                              tiempo utilizado en invertir una matriz 500x500
```

12 GRÁFICACIÓN

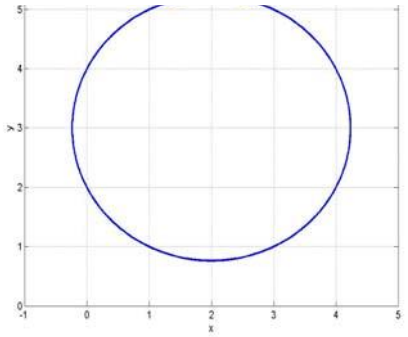
12.1 Gráfico de funciones de una variable

```
>> f='exp(x)-3*x';          función para el ejemplo (use comillas simples)
>> ezplot(f)                función básica para gráficar f en [-2π, 2π]
>> ezplot(f, [0, 2])        función básica para gráficar f en un dominio dado
>> grid on                  colocar cuadrículas en el dibujo

>> x=[0: 0.1: 2*pi];        puntos para evaluar alguna función
>> y=sin(x);                puntos de la función seno
>> plot(x,y);               función para gráficar la función con línea contínua
>> plot(x,y,'o')            gráfico con puntos. Puede elegir: o . * + x --
>> plot(x,y,'r')            cambiar a color rojo. Puede elegir  r,b,y,m,g,w,k
>> plot(x,y,'og')           grafique con círculos verdes.
>> grid on                  colocar cuadrículas en el dibujo

>> title('seno de x')       incluya un título en el gráfico
>> gtext('seno de x')       posicione el texto en el gráfico con el mouse
>> xlabel('X')              rotule el eje horizontal
>> ylabel('Y')              rotule el eje vertical

>> c=[0, 2*pi, -2, 2]       defina la región para el gráfico
>> axis(c)

>> hold on                  superponer siguientes gráficos
>> hold off                 deshabilitar opción anterior
>> clf                      borrar el gráfico

>> figure(1)                puede tener varias figuras abiertas
                            cada una en una ventana rotulada con 1, 2, ...
>> subplot(2,3,1)           puede dividir una figura en subgráficos.
                            Ej. en 2 filas y 3 columnas. Activando el gráfico 1
>> clf(1)                   borra el gráfico 1
>> clf                      borre todos los gráficos
>> x=[0:0.1:10};
>> y=exp(x);
>> semilogx(x,y)            gráficar en escalas logarítmicas
>> semilogy(x,y)
>> loglog(x,y)              doble logarítmica
>> grid on

>> a=0:0.01:2*pi;
>> r=sin(3*a);              'rosa' de 3 pétalos
>> polar(a, r);             grafique en coordenadas polares
```

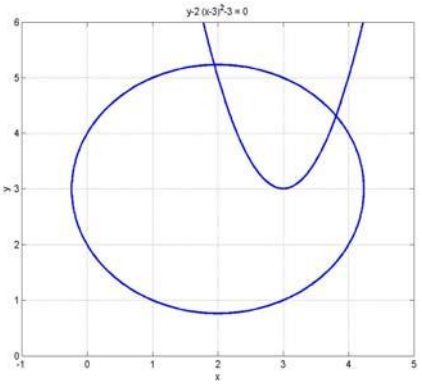


12.2 Gráfico de funciones implícitas y ecuaciones con dos variables

```
>> f='(x-2)^2+(y-3)^2-5';
>> ezplot(f,[-1,5,0,6])        Gráficar f en el dominio  -1≤x≤5, 0≤y≤6
>> grid on;                    Colocar cuadrículas
```



```
>> hold on;                    Superponer el siguiente gráfico:
>> g='y-2*(x-3)^2-3';          una parábola y=2(x-3)^2-3  en el mismo dominio
>> ezplot(g,[-1,5,0,6])
```



12.3 Gráfico de funciones definidas en forma paramétrica

```
>> ezplot('sin(t)','cos(t)',[-pi,pi]);                  Gráficar x=x(t), y=y(t) en -π≤t≤π
>> ezplot('sin(3*t)*cos(t)','sin(3*t)*sin(t)',[0,pi]);  Una rosa de 3 pétalos
```

12.4 Editor de gráficos

Después que el gráfico ha sido realizado puede utilizar las facilidades del editor de gráficos para cambiar las propiedades de las figuras: color, tipo, etc. También puede realizar estadísticas básicas y ajuste de curvas. Adicionalmente puede insertar directamente en el gráfico texto, líneas, flechas, rótulos, etc.

Para habilitar el editor de gráficos seleccione el botón **tools** en la barra de opciones del gráfico y luego elija **edit plot**. Para realizar estadísticas básicas y ajuste de curvas, elija respectivamente **Data Statistics** y **Basic Fitting**

Ejercicio. Obtenga y grafique el polinomio de interpolación, la recta de mínimos cuadrados y el trazador cúbico para un conjunto de datos dados

```
>> x=[1 2 4 5 7];              cinco puntos (x, y) para el ejemplo
>> y=[5 3 6 7 4];
>> plot(x,y,'o')               grafique los datos con círculos
>> grid on                     poner cuadrículas
>> hold on                     superponer los siguientes gráficos

>> a=polyfit(x,y,4);           polinomio de interpolación, 5 puntos: grado 4
>> a                           coeficientes a(1)x^4 + a(2)x^3 + a(3)x^2 + ...
>> z=[1: 0.1: 7];              puntos para evaluar el polinomio
```

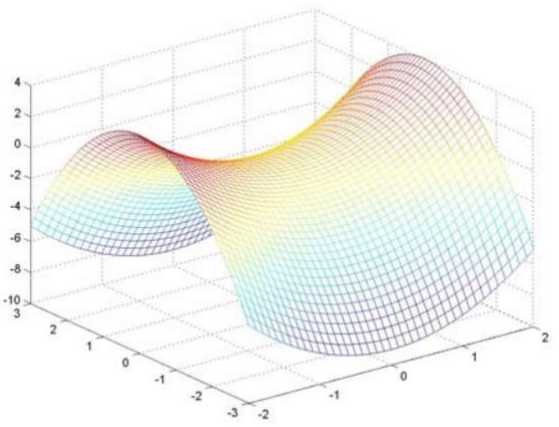
```
>> p=polyval(a,z);          evalúe el polinomio con z  obtenga puntos p
>> plot(z,p)                grafique el polinomio de interpolación

>> b=polyfit(x,y,1);        recta de mínimos cuadrados (grado 1)
>> b                        coeficientes de la recta:  b(1)x + b(2)
>> t=[1 7];                 puntos extremos de la recta (abscisas)
>> q=polyval(b,t);          obtenga las ordenadas respectivas de la recta
>> plot(t,q,'r')           grafique la recta en color rojo

>> s=spline(x,y,z);         evalúe con z  el trazador cúbico y obtenga  s
>> plot(z,s,'g')            grafique el trazador cúbico con verde
>> hold off                 deshabilite la superposición de gráficos
```

12.5 Gráfico de funciones de dos variables

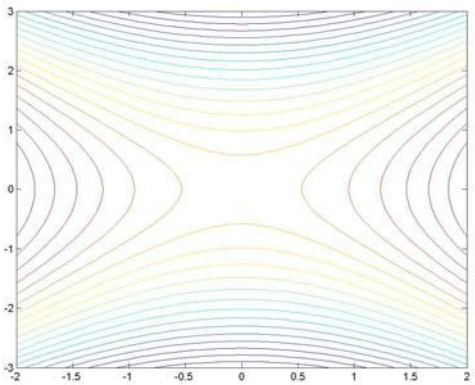
```
>> a=[1 3 2; 5 3 7; 4 5 2];  una matriz 3x3
>> mesh(a);                  gráficar los elementos como puntos sobre el plano.
```

El siguiente ejemplo es una referencia para gráficar funciones de dos variables
Gráficar **$z = x^2 - y^2$, $-2 \le x \le 2$, $-3 \le y \le 3$**

```
>> x=-2:0.1:2;              dominio de la función para el ejemplo
>> y=-3:0.1:3;
>> [u,v]=meshgrid(x,y);      u, v: matrices q' contienen cada par ordenado  x,y
>> z=u.^2 - v.^2;           puntos de la función  z = x^2 – y^2
>> mesh(x, y, z)            gráfico de malla
```



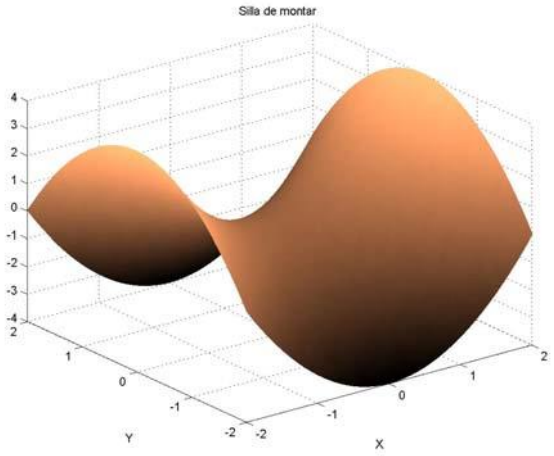
```
>> contour(x, y, z, 20)     gráfico de contorno con 20 curvas de nivel
```



```
>> surfc(x, y, z)           gráfico de superficie y contorno
>> surf(x, y, z)            gráfico de superficie
>> xlabel('X')              rotulación de eje x
```

```
>> ylabel('Y')          rotulación de eje y; también puede usar zlabel
>> title('Silla de montar')   título para el gráfico
>> colormap copper;     color del gráfico; también: gray, jet, pink
>> shading interp;      suavizado del gráfico
```

Gráfico final



Adicionalmente puede usar las opciones del editor de gráficos para editar la figura, rotar, cambiar la perspectiva, insertar títulos, etc.

12.6 Para insertar el gráfico en un documento

Si desea insertar el gráfico elaborado con ***MATLAB*** en un documento, usualmente escrito en **WORD**, puede seguir el siguiente procedimiento:

1) Elija en la barra de opciones del gráfico el botón **File** y luego la opción **Export**
2) Elija una carpeta para almacenar el gráfico y un nombre para el gráfico.
3) Guarde el gráfico con tipo .**jpg**
4) Copie el gráfico almacenado y péguelo en el documento, en el lugar elegido y reduzca el tamaño hasta encuadrarlo en el texto.

13 FUNCIONES PARA ESPECIALES PARA ANÁLISIS NUMÉRICO

13.1 Raíces de ecuaciones no lineales

```
>> f='exp(x)-pi*x';
>> x=solve(f)
>> x=eval(x)                    cambia la solución simbólica a real
        x =
          0.5538                resultados de MATLAB
          1.6385
>> x=fzero(f,2)                 solución de una ecuación con un valor inicial
        x =
          1.6385                resultado de MATLAB
>> x=fzero(f,[1,2])             solución usando un rango para la raiz
        x =
          1.6385                resultado de MATLAB
```

13.2 Raíces de sistemas de ecuaciones no lineales

Resolver el sistema: $a^2 + ab - b = 3$, $a^2 - 4b = 5$

```
>> [a,b] = solve('a^2 + a*b - b = 3','a^2 - 4*b = 5');
>> a=eval(a)                    para expresar la solución en forma real
   a =
      -1.0000                   resultados entregados por MATLAB
       1.8284
```

```
          -3.8284

>> b=eval(b)                          resultados entregados por MATLAB
    b =
         -1.0000
         -0.4142
          2.4142
```

13.3 Integración

```
>> f = 'exp(x)-pi*x';
>> v = int(f)                         integración analítica
    v =
       exp(x)-1/2*pi*x^2
>> r = eval(int(f, 0, 2))             integración entre límites
    r =
        0.1059
>> g = 'x*exp(-x)';
>> r = int(g, 0, Inf);                integral impropia
    r =
         1
```

13.4 Diferenciación

```
>> u = diff(f)                        diferenciación con una variable
    u =
       exp(x)-pi
>> f = 'x*exp(x+y)';
>> u = diff(f,'x')                    diferenciación con dos variables
    u =
       exp(x+y)+x*exp(x+y)
```

13.5 Ecuaciones diferenciales ordinarias de primer orden

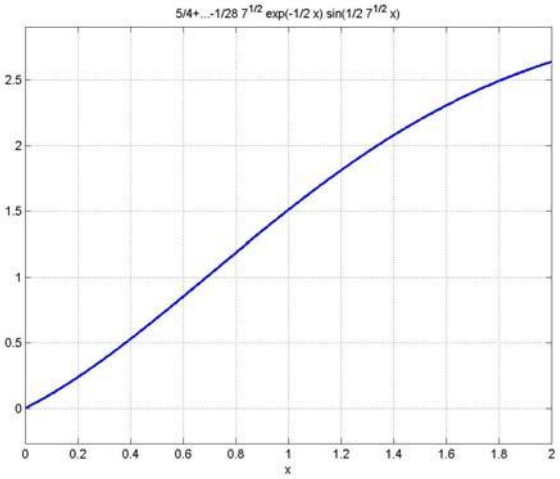
Resolver la ecuación $y' = (x - y)/x$, $y(0) = 0$

```
>> y=dsolve('Dy=(x-y)/x','y(0)=0','x')
>> ezplot(y,0,2);
>> grid on
```

13.6 Ecuaciones diferenciales ordinarias de segundo orden con cond. en el inicio

Resolver la ecuación $y'' + y' + 2y - x - 3 = 0, y(0) = 0, y'(0) = 1$

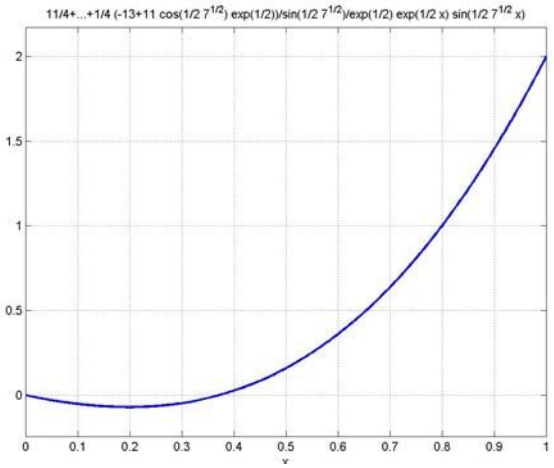
```
>> y=dsolve('D2y+Dy+2*y-x-3=0','y(0)=0,Dy(0)=1','x')
y =                                                    Solución calculada
5/4+1/2*x-5/4*exp(-1/2*x)*cos(1/2*7^(1/2)*x)-1/28*7^(1/2)*exp(-1/2*x)*sin(1/2*7^(1/2)*x)

>> ezplot(y,0,2), grid on
```



13.7 Ecuaciones diferenciales ordinarias de segundo orden cond. en los bordes

Resolver la ecuación $y'' - y' + 2y - 5x - 3 = 0, y(0) = 0, y(1) = 2$

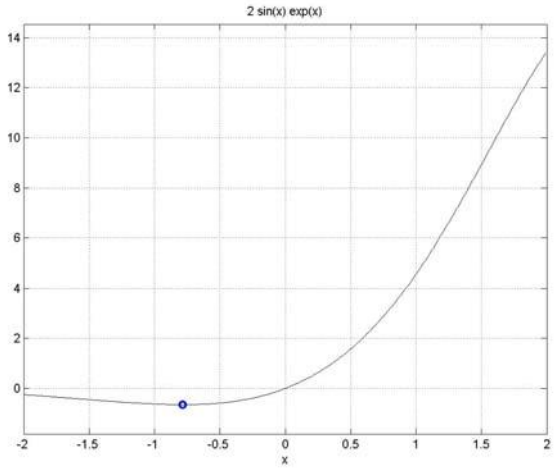
```
>> y=dsolve('D2y-Dy+2*y-5*x-3=0','y(0)=0,y(1)=2','x');
>> ezplot(y, [0, 1])
```



13.8 Optimización

Encontrar un mínimo local de $f(x) = 2\text{sen}(x)e^x$, $-4 \leq x \leq 4$

```
>> f='2*sin(x)*exp(x)';
>> [x,y]=fminbnd(f,-2,2)
x =
  -
0.7854 y
=
  -0.6448
>> ezplot(f,-2,2), grid on
>> hold on
>> plot(x,y,'o');
```



ANEXO B

CONFIGURACIÓN DE LOS BLOQUES DE SIMULINK USADOS EN LAS SIMULACIONES DEL CAPÍTULO 4

Simulación 4.3.1 (Modulación AM)
Configuración de los Bloques:

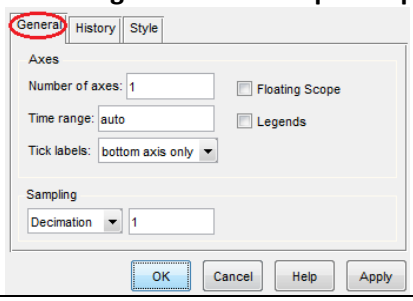
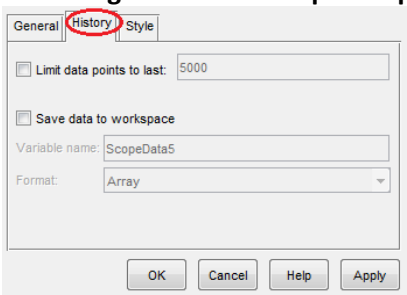
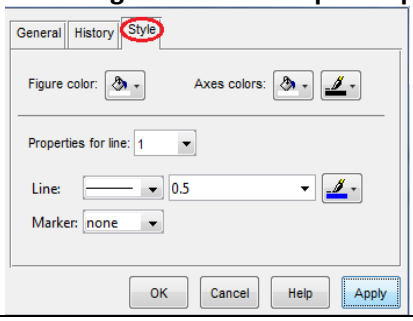
Sine Wave:	**Configuración del bloque Sine Wave (Señal Modulante)**
	Parameters • **Sine type:** Time based • **Time(t):** Use simulation time • **Amplitude:** 0.5 • **Bias:** 0 • **Frecuency (rad/seg):** 50 • **Phase (rad):** 0 • **Sample time:** 0 • **Interpret vector parameters as a 1-D:** check

Sine Wave:	**Configuración del bloque Sine Wave (Señal Portadora)**
	Parameters • **Sine type:** Time based • **Time(t):** Use simulation time • **Amplitude:** 2 • **Bias:** 0 • **Frecuency (rad/seg):** 500 • **Phase (rad):** 0 • **Sample time:** 0 • **Interpret vector parameters as a 1-D:** check

Power Spectral Density:	**Configuración del bloque Power espectral density**
	Parameters • **Length of buffer:** 128 • **Number of points for fft:** 2048 • **Ploter after how many points:** 128 • **Sample time:** 1/(50)

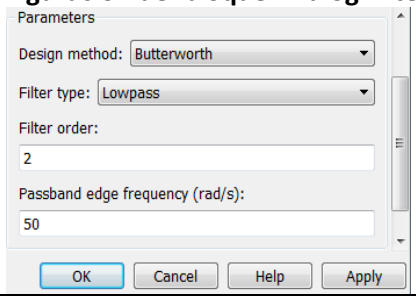
Power Spectral Density:	**Configuración del bloque Power espectral density**
	Parameters • **Length of buffer:** 128 • **Number of points for fft:** 2048 • **Ploter after how many points:** 128 • **Sample time:** 1/(500)

Product	**Configuración del bloque Product**
	Main • **Number of inputs:** 2 • **Multiplication:** Element-Wise.* • **Sample time (-1 for inherted):** -1

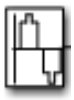
Scope	**Configuración del bloque Scope**
	
Scope	**Configuración del bloque Scope**
	
Scope	**Configuración del bloque Scope**
	

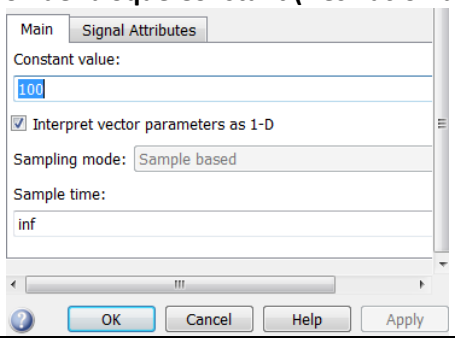
Nota: Todos los bloques **Scope**, fueron configurados de la misma manera.

Simulación 4.3.2 (Demodulación AM)
Configuración de los Bloques:

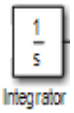
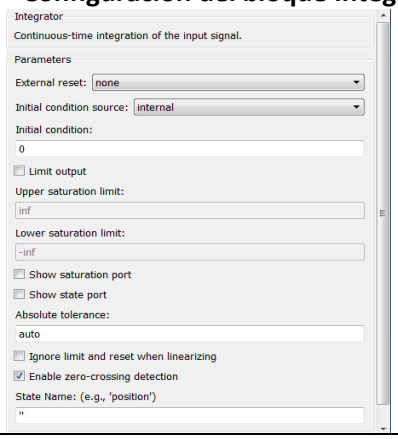
Analog filter Design:	**Configuración del bloque Analog Filter Design**
	

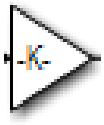
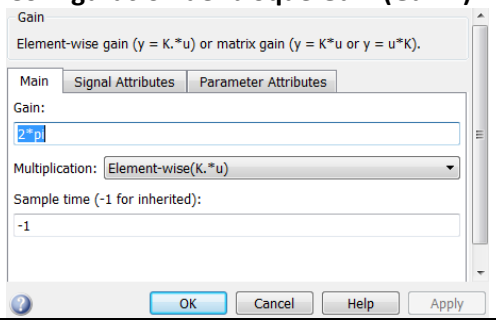
Simulación 4.4.1 (Modulación FM)
Configuración de los Bloques:

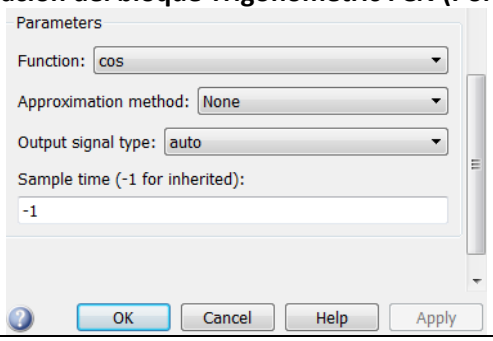
Sine Wave:	**Configuración del bloque Sine wave (mensaje)**
	Parameters • **Sine type:** Time based • **Time(t):** Use simulation time • **Amplitude:** 50 • **Bias:** 0 • **Frecuency (rad/seg):** 2*pi*10 • **Phase (rad):** 0 • **Sample time:** 1/1000 **Interpret vector parameters as a 1-D:** check

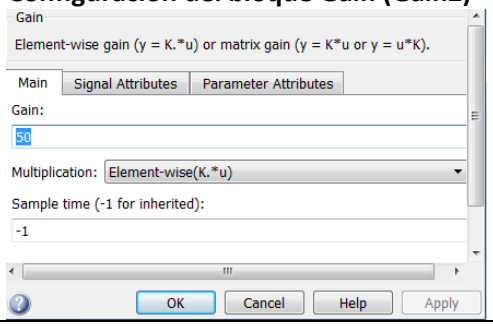
Constant:	**Configuración del bloque Constant (Desviacion de frecuencia)**
	

Adder:	**Configuración del bloque Adder (Desviacion de frecuencia)**
	Main • **List of signs:** ++ • **Sample time (-1 for inherted):** -1

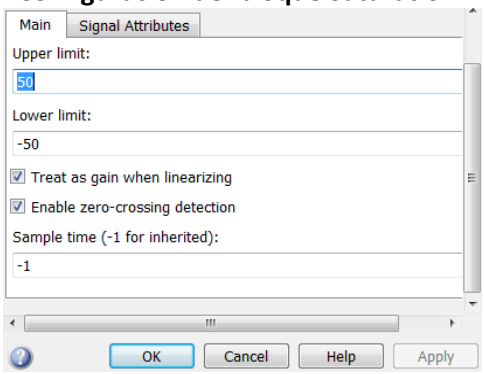
Integrator:	**Configuración del bloque Integrator**
	

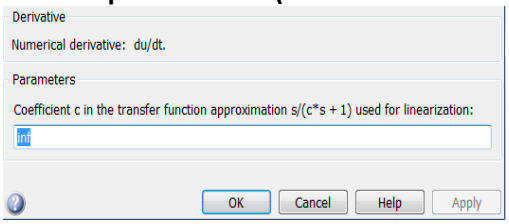
Gain:	**Configuración del bloque Gain (Gain1)**
	

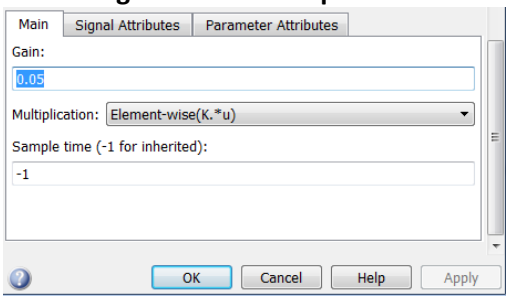
Trigonometric Fcn:	**Configuración del bloque Trigonometric FCN (Portadora)**
	

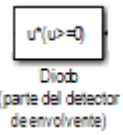
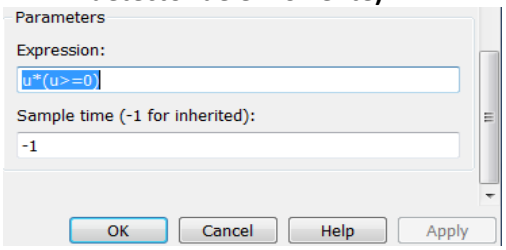
Gain:	**Configuración del bloque Gain (Gain2)**
	

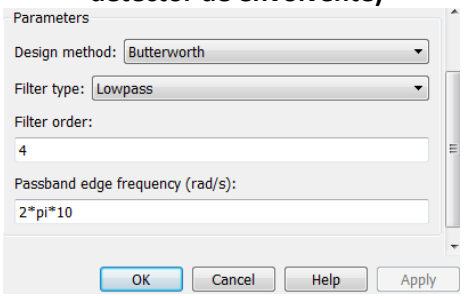
Simulación 4.4.2 (Demodulación FM)
Configuración de los Bloques:

Saturation:	**Configuración del bloque Saturation**
	

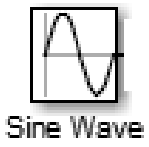
Derivator:	**Configuración del bloque Derivator (Discriminador de frecuencia)**
	

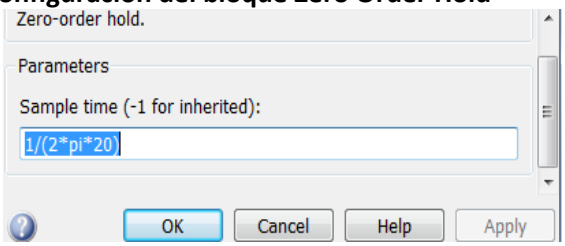
Gain:	**Configuración del bloque Gain**
	

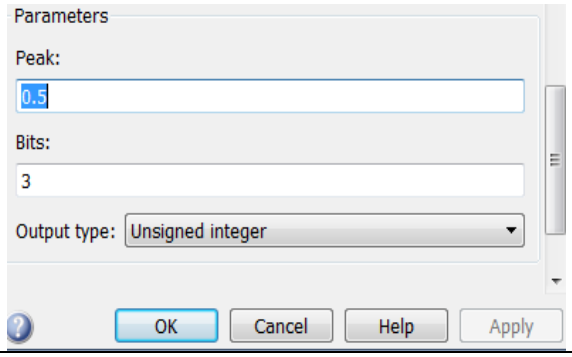
Function Block Parameters:	**Configuración del bloque Function Block Parameters(Diodo parte del detector de envolvente)**
	

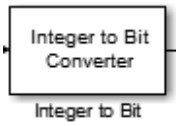
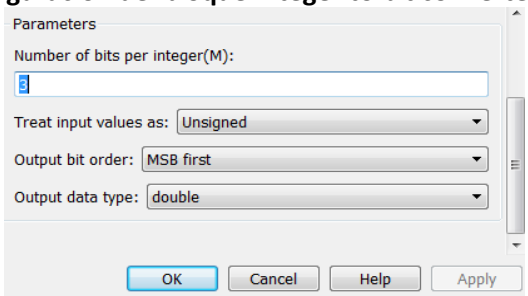
Function Block Parameters:	**Configuración del bloque Function Block Parameters(Diodo parte del detector de envolvente)**
	

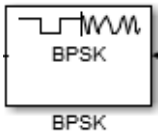
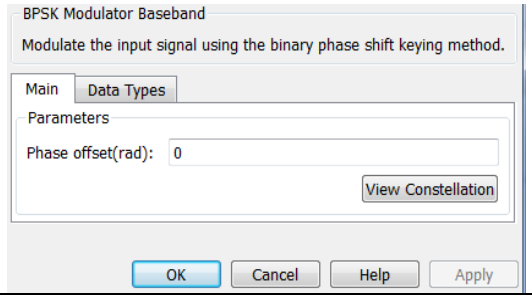
Simulación 4.5.1 (Transmisión de una señal)
Configuración de los Bloques:

Sine Wave:	**Configuración del bloque Sine Wave**
	Parameters • **Sine type:** Time based • **Time(t):** Use simulation time • **Amplitude:** 2 • **Bias:** 0 • **Frecuency (rad/seg):** 2*pi*10 • **Phase (rad):** 0 • **Sample time:** 0 **Interpret vector parameters as a 1-D:** check

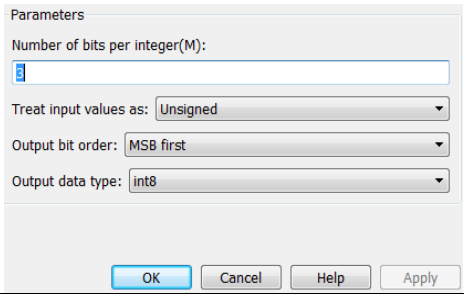
Zero-Order Hold:	**Configuración del bloque Zero Order Hold**
	

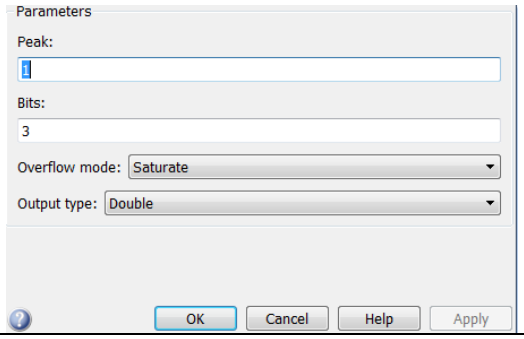
Uniform Encoder:	**Configuración del bloque Uniform Encoder**
 Uniform Encoder	

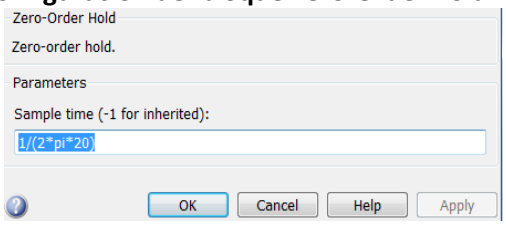
Integer to bit converter:	**Configuración del bloque Integer to bit converter**
 Integer to Bit Converter	

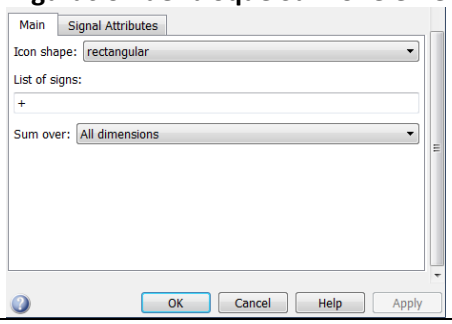
BPSK modulator:	**Configuración del bloque BPSK Modultator Baseband**
 BPSK Modulator Baseband	

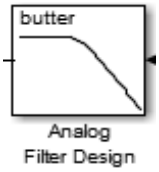
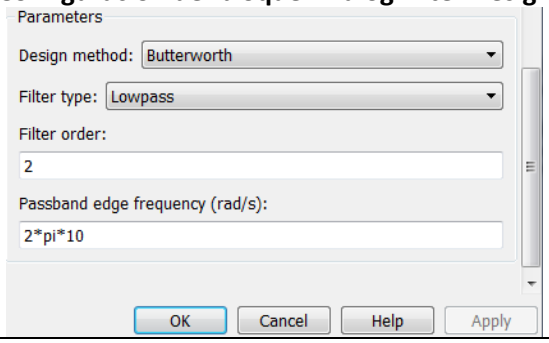
Simulación 4.5.2 (Recepción de una señal)
Configuración de los Bloques:

Integer to bit converter:	**Configuración del bloque Integer to bit converter1**
 Integer to Bit Converter1	

Uniform Decoder:	**Configuración del bloque Uniform Decoder**
	

Zero-Order hold:	**Configuración del bloque Zero-Order hold1**
	

Sum of elements:	**Configuración del bloque Sum of elements**
	

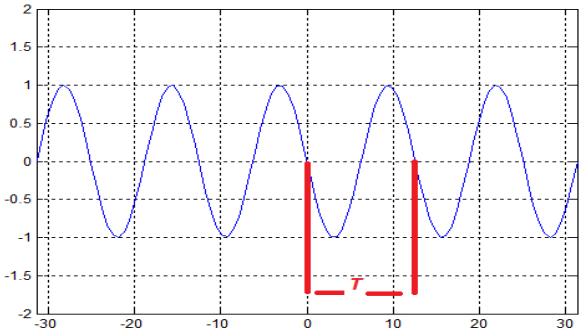
Analog Filter Design:	**Configuración del bloque Analog Filter Design**
	

ANEXO C

RESPUESTAS DE LAS SIMULACIONES DEL CAPÍTULO 4

Respuestas a las cuestiones del Capítulo 4

4.1.a. Generación de señales típicas en telecomunicaciones usando *Matlab*.

Preguntas
1. ¿Qué representa la variable fs en las señales tren de pulsos y diente de sierra? Explique.
Matlab utiliza una frecuencia de muestreo para poder graficar las señales que se generan, para lo cual considera el vector tiempo que crea el usuario. En las señales generadas, se utiliza un vector tiempo de 0 a 5 muestreado a una frecuencia de 1000 Hz. Con esta definición, fs representa la frecuencia de muestreo que usara ***Matlab***, para generar las señales diente de sierra y tren de pulsos respectivamente.
2. Observando en los resultados de las gráficas indique. ¿Cuál es el periodo que tienen las señales representadas?
Señal Continua: Por definición, el periodo es el número de veces que se repite el ciclo de una señal. En la señal continua dicha repetición no existe, por lo que el periodo es infinito. **Señal Seno:** ***y=sin(50*t)*** Debido a que la función ***linspace*** fue usada para crear el vector tiempo, el periodo de la señal queda definido como: $T = \frac{1}{f} * fs$ **Donde *'fs', es el número de puntos tomados en la función linspace.*** **Primero hay que encontrar *f* :** $w = 2 * \pi * f$ $=> \ 50 = 2 * \pi * f$ $=> \frac{50}{2 * \pi} = f \ \ => f = 7,9577$  $T = \frac{1}{f} * fs$ => $T = \frac{1}{7,9577} * 100$ $T = 12,566$

Señal Coseno:
El periodo es el mismo que para la señal Seno, ya que fueron usados los mismos datos.

Señal Tren de pulsos:
y=square(2*pi*t)

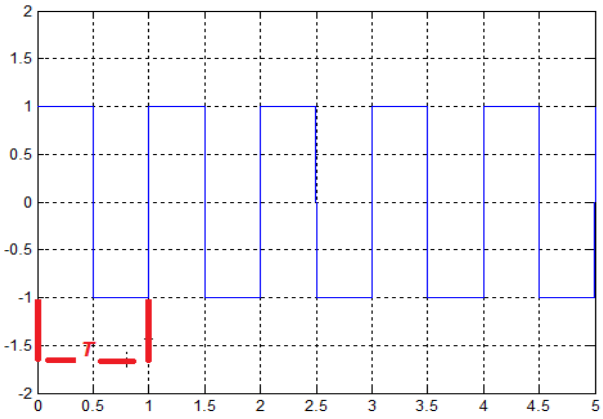
$$T = \frac{1}{f} * fs$$

$$w = 2 * \pi * f$$

$$=> \ 2 * \pi = 2 * \pi * f$$

$$=> \ f = 1$$

$$=> \ T = 1$$



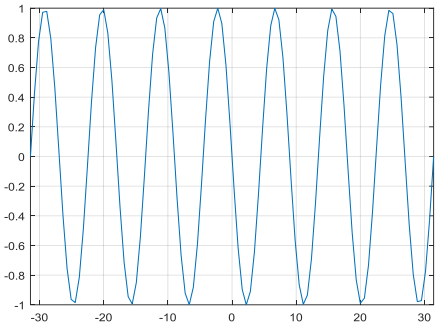
Señal diente de Sierra:
El periodo es el mismo que para la señal tren de pulsos, ya que fueron usados los mismos datos.

3. En la señal aleatoria. ¿Es posible calcular el periodo de la señal?
Explique.

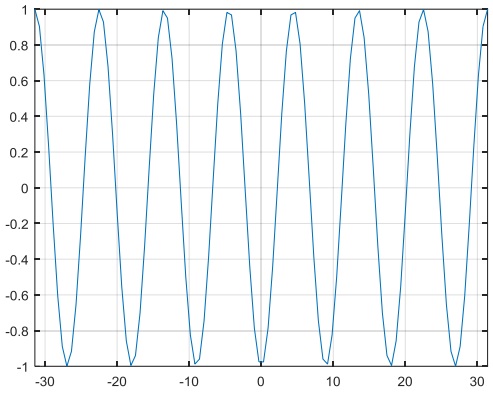
No es posible calcular el periodo de la señal aleatoria ya que no tiene una forma de onda uniforme en el eje temporal, por lo que no se puede identificar un ciclo de la misma.

4. Para las señales no aleatorias, variar su frecuencia y comprobar los resultados para una frecuencia mayor y para una frecuencia menor a la asignada.

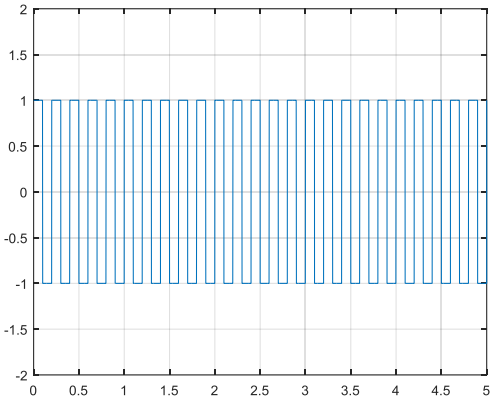
Señal Seno: Con una frecuencia mayor: ***y=sin(70*t)***	**Señal Seno:** Con una frecuencia menor: ***y=sin(30*t)***



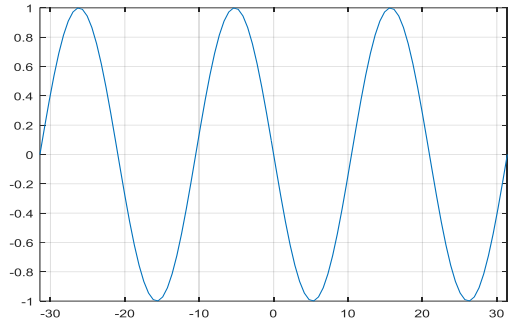
Señal Coseno:
Con una frecuencia mayor:
y=cos(70*t)



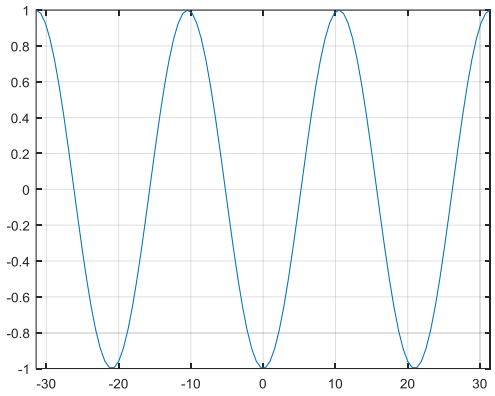
Señal tren de pulsos
Con una frecuencia mayor
y=square(10*pi*t)



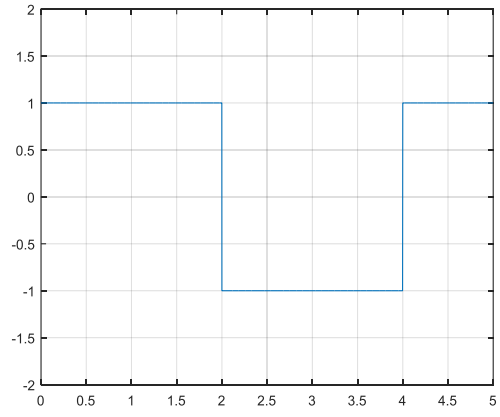
Señal diente de sierra
Con una frecuencia mayor
y=square(10*pi*t)



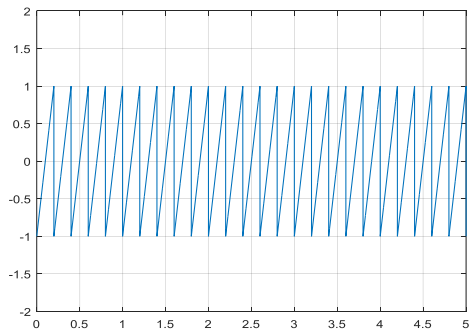
Señal Coseno:
Con una frecuencia mayor:
y=cos(30*t)

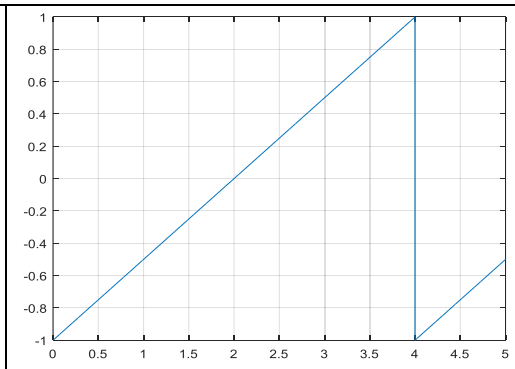


Señal tren de pulsos
Con una frecuencia menor
y=square(0.5*pi*t)



Señal tren de pulsos
Con una frecuencia menor
y=square(0.5*pi*t)



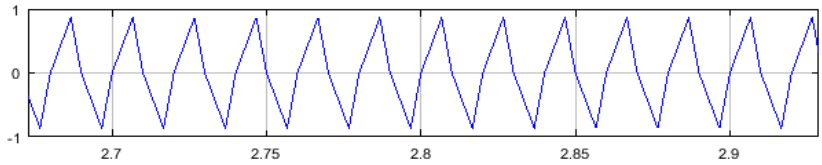


4.1.b. Generación de señales típicas en telecomunicaciones usando *Simulink*.

Preguntas

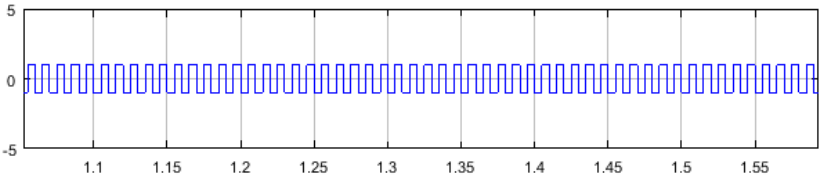
1. Varar el tiempo de simulación a 10s. ¿Qué ocurre con la representación gráfica de la función seno?

La señal se deforma en el intervalo muestreado (0-10s) de la escala tiempo debido a que la frecuencia de la señal aumenta.



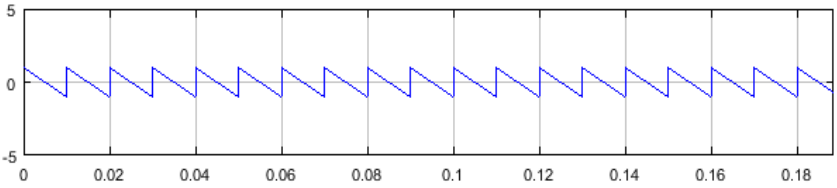
2. Repetir el proceso de la pregunta 1 para el resto de tipos de onda.
¿Qué es lo que ocurre?

Señal tren de pulsos



La frecuencia de la señal aumenta.

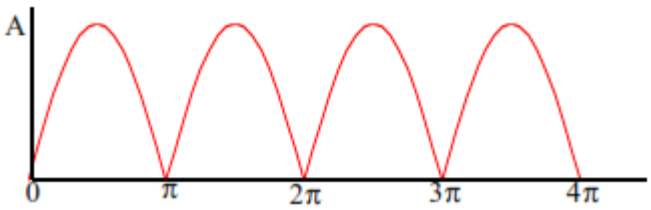
Señal diente de sierra



La frecuencia de la señal aumenta.
3. En la señal aleatoria. ¿Por qué no se altera su representación gráfica pese a haber cambiado el tiempo de simulación? Explique.
La señal aleatoria no se deforma debido a que no tiene un periodo definido, por lo que tampoco es posible determinar su frecuencia la cual está relacionada con el tiempo de simulación del software ***Simulink***.

4.2.1 Serie de Fourier

Preguntas
1. ¿Se puede usar la función *stem* para graficar variables simbólicas? Explique.
No es posible representar variables simbólicas utilizando la función stem, ya que esta función requiere de valores enteros para poder realizar gráficas.
2. ¿Cuál es el código para encontrar los coeficientes de Fourier para la forma de onda del rectificador de onda completa?




```
%CALCULO DE LOS COEFICIENTES DE FOURIER DE UNA SEÑAL PERIODICA
syms t A %Declaramos t y A como variables simbólicas
%Se hará el cálculo para 5 coeficientes de la Serie, ? n=1,2,3,4,5
n=1:5; %Creamos un vector "n" de 5 elementos
T=pi;%Periodo de la señal (se puede reemplazar)
wo=2*pi/(T);%Calculo de Wo
f1(t)=A*sin(t);%La función "A*sin(t)", depende del ejercicio a resolver
%Calculamos integrales definidas usando la función "int" de Matlab para
encontrar los términos de la serie
a0=int(f1(t),t,0,T) %Calculamos la integral definida de a0 respecto a "t"
en el intervalo "0 a T/2" para f1(t) y de "T/2 a T" para f2(t)
a0=(1*a0)/T;% sobrescribimos el valor encontrado de "a0", multiplicando
"a0" por el valor 1/T de la fórmula de "a0" justamente
disp('a0=') %Mostramos en el Command Window de Matlab la palabra a0
pretty(a0)%Se imprime la expresión simbólica "a0"con una mejor
presentación
```

```
an=int(f1(t)*cos(n*t*wo),t,0,T); % Calculamos la integral definida de an
respecto a "t" en el intervalo "0 a T"
an=(2*an/T);% sobrescribimos el valor encontrado de "an", multiplicando
"an" por el valor 2/T de la fórmula de "an" justamente
disp('an=') %Mostramos en el Command Window de Matlab la palabra an
pretty(an)% se imprime la expresión simbólica con una mejor presentación
bn=int(f1(t)*sin(n*t*wo),t,0,T); %Calculamos la integral definida de bn
respecto a "t" en el intervalo "0 a T"
bn=(2*bn/T);% sobrescribimos el valor encontrado de "bn", multiplicando
"bn" por el valor 2/T de la fórmula de "bn" justamente
disp('bn=') %Mostramos en el Command Window de Matlab la palabra bn
pretty(bn)% se imprime la expresión simbólica con una mejor presentación
```

3. Para una función aleatoria. ¿Es posible calcular su serie de Fourier? Explique.

No, no es posible calcular series de Fourier en señales no periódicas como lo son las señales aleatorias.

4.2.2 Transformada rápida de Fourier

Preguntas

1. ¿Cuál es el código necesario para calcular la transformada de Fourier del impulso rectangular usando la función **fft**?

```
fs=40; %Disminuyendo la frecuencia de muestreo se puede observar mejor la
fft
T=1/fs;
B=1; %Amplitud total del Pulso. NOTA: Si el pulso aumenta, la amplitud
%del Sinc disminuye
Bm=B/2; %Amplitud del pulso del origen hacia los extremos
t0=-5:
tf=5;
t1=t0:T:-Bm;
t2=-Bm:T:Bm;
t3=Bm:T:tf;
x1=zeros(1,length(t1));
x2=ones(1,length(t2));
x3=zeros(1,length(t3));
t=[t1 t2 t3]; %Concatenamos vectores
x=[x1 x2 x3];
subplot(4,2,3);
plot(t,x,'r')
axis([t0 tf 0 2])
title('Original Function')
xlabel('Time')
ylabel('Amplitude')
grid on;
%Fast Fourier Transform
X=fft(x);
le=length(X);
le=floor(le/2);
X=X(1:le);
```

```
f=(0:le-1)*(fs/2)/(le-1);
subplot(4,2,4),
plot(f,abs(X))
title('Fast Fourier Transform')
xlabel('Frequency')
ylabel('Amplitude')
grid on;
```

2. Repetir el proceso de la práctica mostrada para una función escalón unitario.

En este punto se puede volver a mostrar el código generado en la pregunta 1.

3. Para una señal aleatoria. ¿Es posible; obtener su transformada de Fourier? Explique.

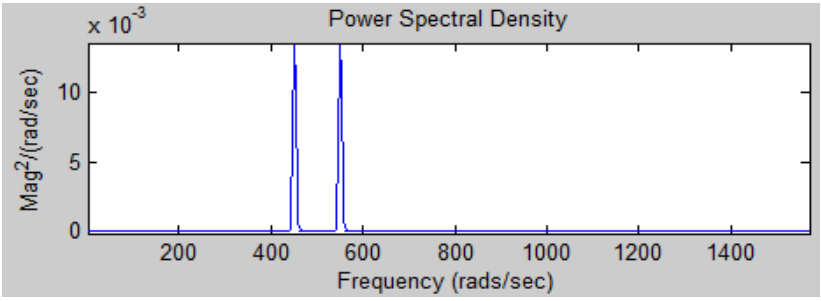
No, no es posible obtener la transformada de Fourier para una señal aleatoria ya que pese a no ser periódica esta existe en todos los puntos del eje temporal.

4.3.1 Modulación AM

Preguntas

4. Observar el espectro en frecuencia de la señal modulada. ¿Cómo se puede calcular el ancho de banda de la señal transmitida?

El ancho de banda de acuerdo al espectro mostrado, se puede calcular de la siguiente manera:



$$Bw = 550 - 450$$
$$Bw = 100$$

5. Observar el espectro en frecuencia de la señal modulada. ¿Qué representan los dos impulsos que se avistan?

Representan las bandas laterales que se generan producto de la traslación de espectros que genera la Modulación en Amplitud

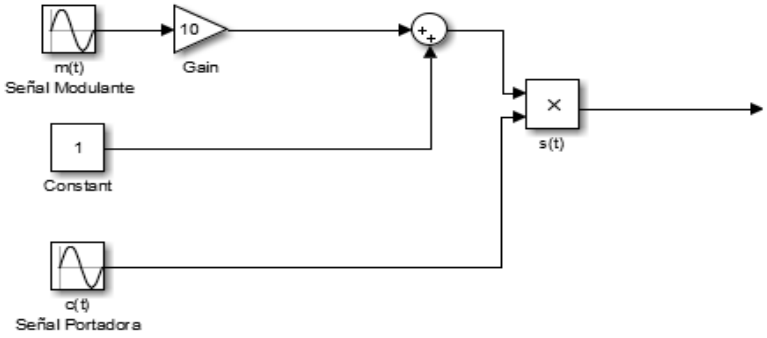
6. ¿Que interpretación tiene el espectro en frecuencia de la señal modulada? Explique .

La señal de información es trasladada a la frecuencia de la señal portadora generando dos bandas laterales.

5. Rediseñar el circuito del modulador y represente la ecuación de la modulación AM

$$s(t) = Ap(1 + m(t) * \cos(2 * \text{pi} * \text{fc})$$

Bloque **Gain**
Gain=10
Bloque **Constant**
Constant Value=1



4.3.2 Demodulación AM

Preguntas

1. Observar el espectro en el tiempo de la señal Demodulada.
¿Por qué la señal recibida no es una réplica exacta de la señal enviada?
Explique.

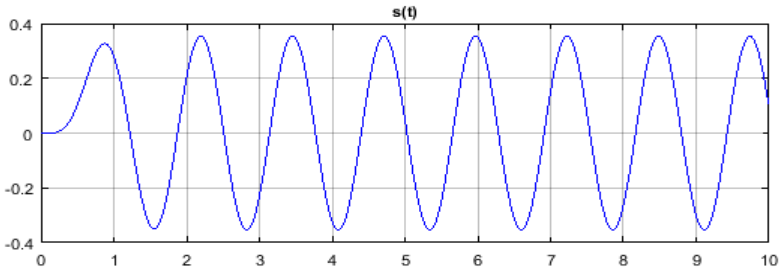
El hecho de que no se haya recuperado la señal de manera exacta, está relacionado al método de recepción que se utilizó. Ya que se utilizó el método de recepción no coherente.

2. ¿Cuál debe ser la frecuencia de corte del filtro pasa bajos sí se coloca un mensaje con frecuencia de 100Hz?

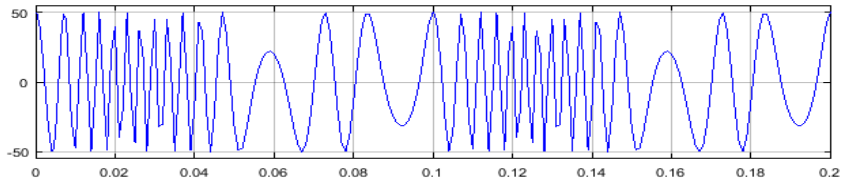
El filtro debe centrarse a una frecuencia igual a la del mensaje, es decir 100Hz

3. Duplique el orden del filtro pasa bajos. ¿Qué ocurre?
Explique.

La señal obtenida sufre deformación ya que el orden del filtro representa el grado de aceptación o rechazo de las frecuencias que atraviesan el filtro por arriba o debajo de la frecuencia fundamental del filtro.

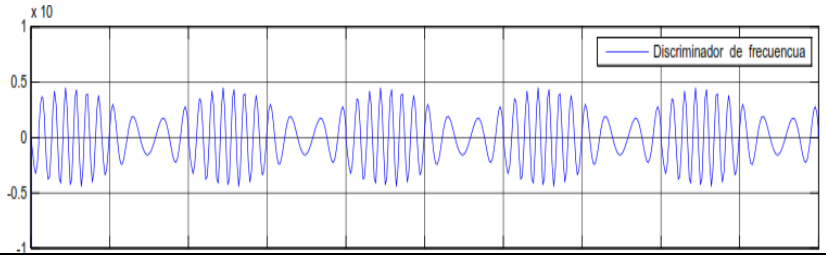


4.3.1 Modulación FM

Preguntas
1. ¿Cuál es el ancho de banda de la señal modulada en FM?.
El ancho de banda en FM se es teorícamente infinito y no es posible hacer un cálculo del mismo usando las herramientas de ***Matlab*** para señales generadas por el usuario.
2. Coloque el bloque de Power Spectral Density a la salida de la salida de la señal modulada. ¿Qué valor debe colocarse en el parámetro **"Sample time"** del bloque?
El parámetro ***"Sample time"*** debe colocarse ser 1/(tiempo de simulación) o inferior
3. Variar la amplitud de la señal modulante a 200, ¿Qué efecto tiene en frecuencia de la señal modulada?
La señal modulada en frecuencia tiene variaciones es decir aumento el índice de modulación FM o lo que es lo mismo su desviación en frecuencia. 

4.3.2 Demodulación FM

Preguntas
1. ¿Qué tipo de señal muestra la salida del Discriminador de frecuencia? Explique este fenómeno.
La señal que se muestra a la salida del Discriminador de frecuencia es una señal con variación de Amplitud

Este fenómeno ocurre debido a la estabilización en frecuencia que produce el **Discriminador**, con lo que se presentan cambios de amplitud en lugar de cambios de frecuencia 
2. ¿Qué utilidad pudo comprobar en el bloque Gain?
Es un bloque muy útil ya que ayuda a simular ecuaciones.
3. ¿Qué otros componentes se necesita para obtener la señal del mensaje enviado replicado en el receptor de manera exacta?
Se necesitan ecualizadores y amplificadores para obtener el mensaje de manera estable y exacta.

4.5.1 Transmisión de una señal

Preguntas
1. ¿Qué beneficio se obtiene al volver una señal análoga digital? Explique,
Tener una señal digital representa grandes beneficios, el que más destaca es tener de infinitos valores que teóricamente contiene una señal Análoga solamente 2, ya sea 0 o 1. Esto hace que sea más fácil poder recuperar las señales transmitidas.
2. ¿Por qué con la frecuencia mínima de muestreo, la señal muestreada no tiene una forma legible? Explique.
Se debe a que el criterio de Nyquist indica que frecuencia de muestreo debe ser mayor o igual a 2 veces la frecuencia máxima de la señal de muestra.
3. Modificar la frecuencia de muestreo y observar los resultados para explicar. ¿Qué sucede al aumentar la frecuencia del muestreador?
La señal se muestra mucho más parecida a la original conforme se aumenta la frecuencia de muestreo.
4. ¿Por qué mientras más aumenta la frecuencia de muestreo la señal de salida, se parece más a la original?
Esto se debe al teorema de muestreo de Nyquist, donde se indica que para tener una señal muestreada de manera satisfactoria, la frecuencia de muestreo debe ser $fs \geq 2 * fmax$ P

Por lo que mientras más alta sea la frecuencia de muestreo la señal de salida será más exacta.

4.5.2 Recepción de una Señal

Preguntas
1. Observando la señale a la salida del Demodulador. Explique; ¿Por qué la señal recibida no es una réplica exacta de la señal enviada?
Se debe a que en el receptor no se ha empleado un método de detección coherente.
2. ¿A qué frecuencia debe centrarse el filtro análogo?
El filtro debe centrarse a la frecuencia del mensaje antes de que este ingrese al muestreador.
3. Observando en el diagrama la sección del demodulador. ¿Cuál es la función del Zero Order-Hold? Explique.
Este bloque se encarga de convertir la señal digital a análoga a es decir, representa el bloque muestreador inverso.