MIXBA'AL,
Revista Metropolitana de Matemáticas.
Vol.16, No.1, páginas 89-231
Recibido 7-05-2025. Aceptado 31-07-2025.
www.doi.org/10.24275/uami/dcbi/mix/v16n1/luvato

USO DE LA HERRAMIENTA DE SIMULACIÓN MATLAB PARA SOLUCIÓN DE PROBLEMAS MATEMÁTICOS Y SUS APLICACIONES

LUIS A. VÁSQUEZ TOLEDO

Resumen

En estas notas se exponen con detalle y de manera autocontenida las herramientas básicas de la teoría de los espacios de Hilbert complejos de dimensión infinita, que son necesarias para un primer estudio, matemáticamente riguroso, de la mecánica cuántica. En la última parte incluimos una introducción axiomática a la mecánica cuántica y una breve exposición del modelo del oscilador armónico y de la teoría de estados cuánticos Gaussianos.

Introducción

Las herramientas de simulación, hoy día, juegan un papel indispensable en la vida académica y profesional en diversos aspectos (laboral, personal, familiar, de entretenimiento, de negocios, etc.). Una herramienta muy utilizada en la actualidad en distintas disciplinas es MATLAB. MATLAB es un programa comercial que permite realizar una gran variedad de cálculos matemáticos y técnicos, es un potente lenguaje diseñado para computación técnica. El nombre de MATLAB proviene de MATrix LABoratory, puede ser utilizado en computación matemática, modelado y simulación, análisis y procesamiento de datos, visualización y representación de gráficos, así como para el desarrollo de algoritmos.

MATLAB es ampliamente conocido y utilizado en universidades e institutos para el aprendizaje en cursos básicos y avanzados de matemáticas, ciencias y, especialmente, ingeniería. Recientemente la Universidad Autónoma Metropolitana adquirió la licencia con una suscripción de "Campus-Wide License de MATLAB" para todo el personal y alumnos. Está disponible para las plataformas Unix, Windows, macOS y GNU/Linux.

Una herramienta de simulación es un componente fundamental para el diseño, la implementación y el monitoreo de los sistemas, permiten predecir el comportamiento de diferentes eventos que se pueden presentar en el sistema, por lo que el objetivo primordial del presente manual está dirigido a estudiantes que utilizan MATLAB por primera vez y que no tienen conocimientos previos sobre programación. Este manual se puede utilizar como un libro de texto para estudiantes de los primeros cursos de ingeniería y ciencias básicas, así como para estudiantes avanzados o incluso para seminarios orientados al aprendizaje de MATLAB. El presente manual también pretende ser una referencia en cursos más avanzados de ciencias e ingeniería, donde MATLAB se utilice básicamente como una herramienta para la resolución de problemas.

En la unidad 1 "Fundamentos de MATLAB", el objetivo es aprender los fundamentos de: operaciones básicas, creación de funciones, etc. Al final de la sesión debemos estar en condiciones

de crear también ficheros guión ("script") y funciones sencillas.

En la unidad 2 "Gráficas con MATLAB" se adentra al alumno en el diseño de funciones que sirvan para mostrar gráficamente resultados. Las gráficas en el entorno de computación con MATLAB son una parte natural, es decir, son sencillas. La gráfica de los resultados de los cálculos puede efectuarse con algunos comandos. Esta unidad se escribió con la intención de ayudar al lector a hacer precisamente esto.

La unidad 3 "Aplicaciones en Comunicaciones Analógicas y Comunicaciones Digitales", se enfoca en mostrar aplicaciones que se utilizan en campos importantes en Telecomunicaciones. Algunas de las aplicaciones son: el teorema de muestreo, el proceso de cuantización y codificación, la modulación en amplitud, los códigos de línea, densidad espectral de potencia, etc. En este capítulo se desarrolla la parte teórica y el desarrollo matemática de las posibles aplicaciones.

En la unidad 4 "Series y Transformada de Fourier" se adentra al alumno al manejo de MATLAB para resolver series de Fourier y trasformada de Fourier para la representación de señales periódicas y las no periódicas determinísticas respectivamente. También se muestra la parte teórica y el desarrollo matemático de la serie y transformada de Fourier.

En la unidad 5 "Aplicaciones en Propagación y Desvanecimientos" se analizan y simulan los fenómenos presentes en la propagación de la señal en el espacio libre, como son canales con desvanecimientos lentos, canales con desvanecimientos rápidos, radio enlace, zonas de Fresnel, cálculo de altura de las torres, entre otros. Cabe mencionar que estos análisis pueden servir para los sistemas celulares 5G, ya que una de las propuestas para 5G es el uso de nuevas técnicas de acceso al medio, NOMA (acceso múltiple no ortogonal), en el cual uno de los requerimientos es calcular las ganancias de canal de cada usuario, para el cual es necesario el cálculo de los desvanecimientos como se realiza en esta unidad.

En la unidad 6 "Aplicaciones en Modulación y Telefonía", se tratarán ejemplos sobre modulación QAM y se calcula la probabilidad de bit erróneo, se simulan redes jerárquicas, utilizadas en la telefonía y se calcula la probabilidad de bloqueo.

En la unidad 7 "Aplicaciones en Sistemas Celulares", se enfoca en mostrar la simulación de celdas celulares, usuarios con movimientos en las celdas. Esto puede servir para analizar el comportamiento de los usuarios en una red 5G. También se muestra un ejemplo para NOMA en una red 5G.

Finalmente, MATLAB nos ofrece una herramienta llamada Simulink, la cual es un entorno de simulación multidominio y diseño basado en modelos mediante diagrama de bloques, esto se aborda en la unidad 8 "Introducción a Simulink". En la **Tabla 1** se presentan las UEA's en las que se pueden aplicar los temas desarrollados en este manual.

Tabla 1: Unidades Enseñanza-Aprendizaje Licenciatura.

División	**CBI**	
Clave	**UEA**	**Plan de estudios**
2130039	Cálculo Integral	Licenciatura en Matemáticas
2131084	Temas Selectos de Matemáticas Aplicadas II	
2131106	Matemáticas Discretas	
2131110	Diseño de Experimentos	
2131145	Probabilidad I	
2131167	Simulación	
2150008	Introducción a la Programación	

Índice

1. Fundamentos de MatLab **6**
1.1. Ventana de comandos . . . 6
1.2. Funciones matemáticas en MATLAB . . . 8
1.2.1. Variables de arreglo . . . 8
1.3. Instrucciones de control (sentencias) . . . 13
1.3.1. Sentencia if . . . 13
1.3.2. Sentencia for . . . 13
1.3.3. Sentencia while . . . 14
1.3.4. Sentencia break . . . 14
1.4. Funciones que facilitan los cálculos . . . 15
1.4.1. Máximo y mínimo . . . 15
1.4.2. Sumatoria . . . 16
1.4.3. Ordenar . . . 16
1.4.4. Variables Aleatorias . . . 17
1.5. Creación de archivos M . . . 17
1.6. Crear archivos M de función . . . 18
1.6.1. Función que devuelve una sola variable . . . 19
1.6.2. Función que devuelve múltiples variables . . . 20
1.6.3. Función que utiliza otra función . . . 20
1.7. Cómo utilizar MATLAB con señales de tiempo continuo . . . 21

2. Gráficas con MATLAB **22**
2.1. Gráficas simples . . . 23
2.2. Gráficas con marcas . . . 24
2.3. Colores y tipos de líneas . . . 25
2.4. Eje y retícula . . . 26
2.4.1. Retícula . . . 28
2.5. Gráficas polares . . . 28
2.6. Gráficas logarítmicas y semilogarítmicas . . . 29
2.7. Múltiples curvas . . . 30
2.7.1. Retención . . . 31
2.7.2. Leyenda . . . 34
2.7.3. Figura . . . 35
2.8. Subgráficas . . . 35
2.9. Versión tridimensional de una gráfica . . . 36

3. Aplicaciones en Comunicaciones Analógicas y Digitales **37**
3.1. Conceptos básicos de señales . . . 37
3.1.1. Corrimiento de una señal en el tiempo . . . 39
3.1.2. Inversión de una señal en el tiempo . . . 40
3.1.3. Escalamiento de una señal en el tiempo . . . 42
3.1.4. Función escalón unitario y rampa unitaria . . . 44
3.1.5. El impulso . . . 48
3.1.6. Señales periódicas . . . 51
3.1.7. Señales pares e impares . . . 56
3.2. Teorema de Muestreo . . . 59
3.3. Cuantización . . . 65

3.4. Conversión Analógica a Digital (PCM) . . . 65
3.5. Modulación AM y FM . . . 68
3.6. Densidad Espectral de Potencia . . . 69
3.7. Código AMI . . . 75

4. Serie y Transformada de Fourier **77**
4.1. Serie de Fourier . . . 77
4.1.1. Representación de señales en términos de sus componentes de frecuencia . 77
4.1.2. Serie trigonométrica de Fourier . . . 81
4.1.3. Serie exponencial compleja . . . 89
4.2. Transformada de Fourier . . . 95
4.3. Transformada inversa de Fourier . . . 98

5. Aplicaciones en propagación y desvanecimientos **98**
5.1. Patrón de Radiación . . . 98
5.2. Desvanecimientos . . . 100
5.3. Diseño de radioenlace . . . 103
5.3.1. Pérdida en el espacio libre . . . 105
5.3.2. Cálculo de zonas de Fresnel . . . 106
5.3.3. Cálculo de la altura de las antenas en un radioenlace . . . 108
5.3.4. Uso de antenas repetidoras . . . 109
5.3.5. Método de diseño . . . 109

6. Aplicaciones en Modulación y Telefonía **116**
6.1. 4QAM . . . 116
6.2. MPSK . . . 120
6.3. Telefonía . . . 120

7. Aplicaciones en Sistemas Celulares **123**
7.1. Celdas celulares . . . 124
7.2. Usuarios en las celdas celulares . . . 125
7.3. Esquema NOMA para 5G . . . 129

8. Introducción a SIMULINK **134**
8.1. Ingresar a SIMULINK . . . 135
8.2. Entorno de trabajo de SIMULINK . . . 135
8.3. Muestreo y Cuantización . . . 136

Referencias **143**

1. Fundamentos de MatLab

"Fundamentos de MatLab" es una sección que será útil al discente como un tutorial para aquellos que van iniciando su aprendizaje en **MATLAB**. Antes de iniciar a leer este material, recomendamos al alumnado instalar el software de **MATLAB** en su ordenador y, posterior a ello, abrir la ventana de comandos. La ventana de comandos es el espacio principal donde se ingresan los comandos, aquí el alumnado escribe instrucciones y puede ver los resultados al concluir las operaciones. La ventana de comandos en **MATLAB** es como la que se muestra en la **Figura 1**.

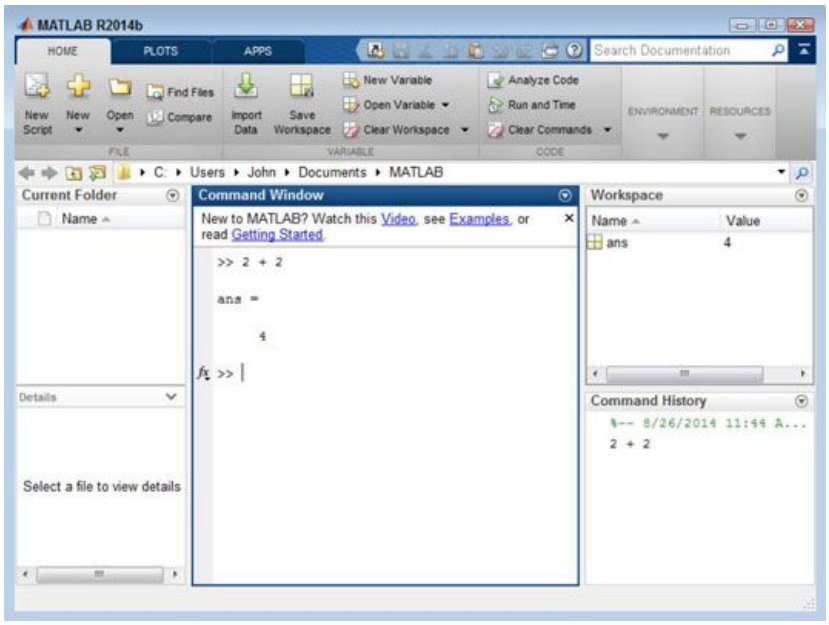


Figura 1: Ventana de comando en **MATLAB**.

1.1. Ventana de comandos

Una vez descargado e instalado el sofrware **MATLAB**, la instrucción para abrir **MATLAB** en un ordenador con sistema operativo *Unix* sería tecleando:

```
>matlab
```

Para abrir el sofrware **MATLAB** en *Macintosh* o *Windows*, haga clic en el icono de **MATLAB**. Una vez iniciado el software **MATLAB** aparece la ventana de comandos (ver **Figura 1**), en la cual aparece el siguiente símbolo>>. Seguido de este símbolo, se teclean los *"comandos"* que se explican en esta sección.

Si se desea salir de **MATLAB**, el procedimiento es similar para *Unix*, *Macintosh* o *Windows*, y se debe teclear la siguiente instrucción:

```
>>quit
```

Una de las instrucciones que se utiliza con mayor frecuencia en **MATLAB**, es la instrucción `help`. Esta instrucción despliega una explicación de los comandos y, en algunos casos, incluso se incluyen ejemplos que ayudan al alumnado a entender de manera sencilla y fácil cada uno de los comandos bajo cuestión. Un ejemplo sería el siguiente:

```
help log
```

Al teclear esta instrucción vemos que *log* significa logaritmo natural, y la función logaritmo base 10 se denotara como log_{10}.

Una de las principales ventajas que nos ofrece **MATLAB** radica en la flexibilidad para la gestión de las variables. Es decir, del hecho que no es necesario declarar los nombres de las variables, ni el tipo de variable, ni tampoco es necesario declarar la dimensión de un arreglo. Todo esto es debido a que los nombres de las variables en **MATLAB** no son diferentes para las variables enteras, reales y complejas, es decir, cualquier variable puede adoptar valores reales, complejos o enteros.

Como ya se mencionó, aunque no es necesario declarar los nombres de las variables, hay que tener mucho cuidado con las variables que se emplean, ya que algunas variables en el entorno de **MATLAB** tienen asignados ciertos valores, nombres de funciones o nombres de comandos.

Una forma rápida y sencilla de verificar si el nombre de una variable es válido en **MATLAB** sería ejecutándolo desde la ventana de comando. Es decir, que un ejmeplo de lo que acabamos de comentar sería introducir la instrucción `x=9`, si ésta es aceptada, generará una respuesta como la que se muestra a continuación:

```
>> x =
9
```

De lo contrario, si se intenta asignar un valor a la palabra `end`, es decir, si se introduce `end = 4`,se producirá un error, dado que `end` es una palabra reservada del lenguaje. Ahora, si se utilizan los comandos *sin* y *cos* como nombres de variables, es decir, que no tengan relación con las funciones trigonométricas cosenoidales, como las que se muestran a la derecha...

```
>> sin = 3
>> cos = sin * 2
```

...las operaciones continuarán ejecutándose, sin embargo, las funciones trigonométricas `sin` y `cos` dejarán de estar disponibles si han sido sobrescritas como nombres de variables, a menos que se utilice el comando `clear` o se reinicie el entorno de **MATLAB**, es decir, adquieren nuevamente la función `sin` y `cos`. Las variables i y j están reservadas para denotar el valor imaginario unitario, $\sqrt{-1}$; por tanto, si un cálculo incluye variables complejas es recomendable evitar el uso de i y j como variables definidas por el usuario con el fin de evitar conflictos o resultados incorrectos.

En la **Tabla 2** se presentan ejemplos de los nombres de variables reservados que tienen significado especial **(Gilat, 2008)**.

Tabla 2: Variables especiales.

Variable	Significado	Valor
eps	Épsilon	$2.2204e^{-16}$
pi	π	3.14159...
i y j	Unidad imaginaria	$\sqrt{-1}$
inf	Infinito	∞
NaN	No es número	
date	Fecha	
flops	Contador de operaciones de punto flotante	
nargin	Número de argumentos de entrada de una función	
nargout	Número de argumentos de salida de una función	

1.2. Funciones matemáticas en MATLAB

MATLAB cuenta con numerosas funciones matemáticas, que resuelven tareas especificas. En la **Tabla 3** se muestran algunas de estas funciones específicas **(Gilat, 2008)**.

Tabla 3: Variables especiales.

Función T.	Comentarios	Función T.	Comentarios
$sin(x)$		$\sqrt{(}x)$	Raíz cuadrada de x
$cos(x)$		$real(x)$	Parte real del valor complejo x
$tan(x)$		$imag(x)$	Parte imaginaria del valor complejo x
$asin(x)$		$conj(x)$	Conjugado complejo x
$acos(x)$		$round(x)$	Redondear al entero más cercano
$atan(x)$	$-\pi/2 \geq atan(x) \geq \pi/2$	$fix(x)$	Redondear al valor real hacia cero
$atan2(y,x)$	$-\pi \geq atan(y,x) \geq \pi$	$floor(x)$	Redondear x hacia $-\infty$
$sinh(x)$		$ceil(x)$	Redondear x hacia $+\infty$
$cosh(x)$		$sign(x)$	$+1$ si $x > 0$; -1 si $x < 0$
$tanh(x)$		$rem(x,y)$	Residuo al dividir: $x - y * fix(x/y)$
$asinh(x)$		$exp(x)$	Base exponencial e
$acosh(x)$		$log(x)$	Logaritmo base e
$abs(x)$	Valor absoluto de x	$log10(x)$	Logaritmo base 10
$angle(x)$	Ángulo de fase de un valor complejo		

1.2.1. Variables de arreglo

En **MATLAB**, se distinguen dos clases principales de variables tipo arreglo:

(a) Unidimensional

(b) Bidimensional

Variable de arreglo unidimensional. Los arreglos unidimensionales pueden adoptar la forma de una fila o una columna, lo que los vincula directamente con los conceptos de vectores fila y columna, respectivamente. En este contexto, un arreglo en forma de fila equivale a un *vector fila*, y uno en forma de columna corresponde a un *vector columna*.

En el entorno **MATLAB**, para definir los elementos de un *vector fila*, se separa cada uno de los elementos ya sea con un simple espacio o con una coma (`,`), y estos elementos se colocan entre corchetes (`[]`), como se indica en el siguiente ejemplo:

$$>> x = [1\ \ 3\ \ 5\ \ 7\ \ 9]$$

Que, como ya se mencionó antes, es lo mismo que tener:

$$>> x = [1, 3, 5, 7, 9]$$

Ahora veamos un ejemplo de la variable de arreglo tipo *vector fila*, en este se puede observar que la variable de arreglo tipo *vector fila "x"*, contiene los primeros 5 números impares. Sabiendo esto, se puede definir la la variable de arreglo tipo *vector fila* con un *incremento* o *decremento* fijo como se presenta a continuación:

$$>> x = 1 : 2 : 9$$

Aquí, el primer valor es el *límite inferior* del arreglo, el segundo valor nos indica el *incremento*, que en este ejemplo en específico es un incremento de 2 en 2, y el tercer valor es el *límite superior* del arreglo. Una vez mencionado esto, a continuación se presenta lo que da como resultado en el entorno:

$$x = 1\ \ 3\ \ 5\ \ 7\ \ 9$$

Una vez que ya tenemos la variable de arreglo tipo *vector fila*, si requerimos conocer y desplegar (imprimir) un elemento en particular de este arreglo, lo que debemos teclear es la instrucción *x (subíndice)*. Aquí, el *"subíndice"* indica el elemento seleccionado del arreglo.

Ejemplo, si insertamos la instrucción $x(2)$, los arreglos se enumeran de 1 a n, estamos solicitando que se despliegue el valor contenido en la posición número dos del arreglo, a lo que el sistema mostrará en pantalla lo siguiente:

$$ans = 3$$

En el entorno **MATLAB**, para definir los elementos de un *vector columna* es muy similar a la de una variable de arreglo tipo *vector fila*, excepto que aquí, para separar cada uno de los elementos del arreglo se emplea el delimitador de punto y coma (`;`). A continuación se presenta un ejemplo de cómo se debe de declarar una variable de arreglo tipo *vector columna*:

$$>> x = [1; 3; 5; 7; 9]$$

Una vez declarado la variable de arreglo tipo *vector columna* se presenta a continuación lo que se produce en el entorno:

$$x = \begin{matrix} 1 \\ 3 \\ 5 \\ 7 \\ 9 \end{matrix}$$

Podemos observar que al escribir la instrucción `x=[1;3;5;7;9]`, los elementos del arreglo se despliegan en pantalla como se observa arriba. De lo contrario, si no se desea la impresión de los elementos del arreglo se debe colocar al final de la instrucción el signo de punto y coma (`;`) como se presenta a continuación:

$$>> x = [1; 3; 5; 7; 9];$$

En el entorno de **MATLAB** podemos procesar operaciones básicas entre variables de arreglo como son:

(a) Suma

(b) Resta

(c) Multiplicación

(d) División

Si x y z son arreglos del mismo tipo (*vector fila* o *vector columna*) y, tienen la misma longitud (es decir, la misma cantidad de elementos), entonces x y z se pueden sumar, restar, multiplicar o dividir empleando los operadores aritméticos de arreglos **(Gilat, 2008)**.

$$\begin{matrix} y = z + x \\ y = z - x \\ y = z. * x \\ y = z./x \end{matrix}$$

Ejemplo 1.1 Ejemplo de suma entre arreglos

$>> x = [4, 2, 8, 12, 6, 4];$
$>> y = [2, 1, 4, 6, 3, 2];$
$>> z = x + y;$

$>> z$

$z = 6\ 3\ 12\ 18\ 9\ 6$

■

Ejemplo 1.2 Ejemplo de resta entre arreglos

$>> x = [4, 2, 8, 12, 6, 4];$
$>> y = [2, 1, 4, 6, 3, 2];$
$>> z = x - y;$

$>> z$

$z = 2\ 1\ 4\ 6\ 3\ 2$

■

Ejemplo 1.3 Ejemplo de multiplicación entre arreglos

$>> x = [4, 2, 8, 12, 6, 4];$
$>> y = [2, 1, 4, 6, 3, 2];$
$>> z = x. * y;$

$>> z$

$z = 2\ 1\ 4\ 6\ 3\ 2$

■

Ejemplo 1.4 Ejemplo de división entre arreglos

$>> x = [4, 2, 8, 12, 6, 4];$
$>> y = [2, 1, 4, 6, 3, 2];$
$>> z = x./y;$

$>> z$

$z = 2\ 1\ 4\ 6\ 3\ 2$

■

Los operadores $.*$ y $./$ son operadores asignados para realizar la multiplicación y la división de arreglos respectivamente. El operador matemátio que realiza la potenciación de arreglos se ilustra a continuación:

$w = y.\hat{}2;$

Aquí se observa que se coloca un punto (.) antes del operador "^". En este caso, w se convierte en un vector de la misma longitud que y.

En algunas ocasiones, se requiere agregar elementos a los arreglos ya sea al principio o al final. Por ejemplo, consideremos el siguiente arreglo $x = [1, 3]$, al cual, se le desea agregar un elemento de valor 5 al final del arreglo. Para realizar esta acción, se debe escribir la instrucción es la siguiente:

$$>> x = [x, 5]$$

Ahora, si lo que se desea es agregar el elemento al principio del arreglo la instrucción sería la siguiente:

$$>> x = [5, x]$$

Por otro lado, si x es una variable de arreglo tipo *vector columna* basta con cambiar el signo de coma (`,`) por el signo de punto y coma (`;`). Si necesitamos conocer el número de elementos que contiene un arreglo se debe usar el comando `length( )`. Por ejemplo, si la variable de arreglo tipo *vector fila* es $x = [1, 3]$, la instrucción que se debe escribir para conocer el número de elementos que contiene el arreglo es `length(x)`, lo que despliega:

$$ans = 2$$

Variable de arreglo bidimensional. Las variables de arreglo bidimensional, que es lo mismo que una matriz en **MATLAB**, se puede definir con la siguiente instrucción:

$$b = [1, 2, 3; 4, 5, 6; 7, 8, 9]$$

La instrucción anterior despliega lo siguiente:

$$b = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$$

Una vez que se declara un arreglo bidimensional se puede acceder a un elemento en particular de este, por ejemplo, si se desea acceder al elemento de valor 4, el cual esta ubicado en la fila 2 y columna 1, entonces la instrucción para acceder a este elemento sería la siguiente:

$$b(2, 1)$$

En el manejo de *arreglos bidimensionales*, es posible referenciar una *fila* o *columna* completa mediante el operador de dos puntos (`:`). Ejemplificando lo anterior, la instrucción $b(1, :)$ denota

la primera *fila* de la matriz b, mientras que $b(:,3)$ hace referencia a su tercera *columna*.

Nota Ambos casos se interpretan como estructuras vectoriales unidimensionales.

1.3. Instrucciones de control (sentencias)

En esta sección se describirán las estructuras para las sentencias `if`, `for`, `while` y `break`. Antes de adentrarnos en las sentencia, a continuación presentaremos los operadores *mayor que*, *menor que*, *igual o mayor que*, *igual o menor que* y *diferente de* respectivamente **(Gilat, 2008)**:

$>$ $<$ $\geq$ $\leq$ $\sim=$

1.3.1. Sentencia if

La sentencia `if` siempre debe de terminar con la palabra reservada `end`, la sintaxis para esta sentencia es la siguiente:

$$\begin{aligned} &if(condición)\\ &\quad > Operaciones\\ &end \end{aligned}$$

La estructura condicional `if` puede complementarse también con las sentencias `else` o `elseif`, y estas servirán para evaluar múltiples condiciones. Es importante destacar que el operador `elseif` puede utilizarse sucesivamente y sin limitaciones, permitiendo así una verificación escalonada de tantos casos como sean necesarios. A continuación, se presenta un ejemplo de la estructura de varias sentencias `elseif` y `else`:

$$\begin{aligned} &\quad if(condición\ 1)\\ &\quad\quad > Operación1\\ &elseif(condición\ 2)\\ &\quad\quad > Operación2\\ &elseif(condición\ 3)\\ &\quad\quad > Operación3\\ &\quad\quad\quad ...\\ &\quad else\\ &\quad\quad > OperaciónN\\ &\quad end \end{aligned}$$

1.3.2. Sentencia for

En la estructura de la sentencia `for` siempre debe terminar con `end`. La sintaxis para una estructura de la sentencia `for` es la siguiente:

$$for\ variable = límite_{inferior} : incremento : límite_{superior}$$
$$> Operaciones$$
$$end$$

Se observa que la variable se inicializa con el valor de *límite inferior*, y la ejecución se detendrá cuando la variable alcance el valor del *límite superior*, esta se incrementará por pasos dependiendo del valor de la variable *incremento*.

1.3.3. Sentencia while

En la estructura de una sentencia `while` esta siempre se debe terminar con la palabra reservada `end`. La sintaxis para una estrcutura de la sentencia `while` es la siguiente:

$$while(condición)$$
$$> Operaciones$$
$$end$$

1.3.4. Sentencia break

La función de una sentencia `break` es la de terminar la ejecución de un ciclo `for` o `while`. Esta sentencia se emplea de la siguiente manera:

$$for\ variable = límite_{inferior} : incremento : límite_{superior}$$
$$if(condición), break, end$$
$$> Operaciones$$
$$end$$

Al ejecutar los comandos, **MATLAB** memoriza las variables utilizadas. Los valores de estos comandos permanecen en la memoria hasta que uno finaliza el programa de **MATLAB** o hasta que se borran las variables, ya sea de manera individual o de manera general. Si empleamos el comando `clear`, lo que sucederá es que borrará todas las variables guardadas, por otro lado, si solo se desea borrar algunas variables en específico, sus nombres se indican después de la palabra `clear`.

A continuación, se muestra el ejemplo para borrar una variable en específico:

$$>> clear\ \ x$$

Cuando utilizamos el comando `clc` lo que sucederá será que borraremos sólo la ventana de comandos, a continuación se presenta la sitaxis para este comando:

```
>> clc
```

1.4. Funciones que facilitan los cálculos

En está sección se describirán algunas funciones que nos ayudan a realizar cálculos y nos ahorran el desarrollo de líneas de código, así como tiempo **(Gilat, 2008)**.

1.4.1. Máximo y mínimo

Uno de los cálculos más utilizados en el entorno de **MATLAB** es encontrar el valor *máximo* o *mínimo* de un arreglo. Por ejemplo, consideremos un arreglo x como el que se muestra a continuación:

```
>> x = [2, 1, 4, 6, 3, 2];
```

Si se requiere hallar el elemento con valor *máximo* de este arreglo, se deberían escribir las instrucciones que se muestran a la derecha.

```
>> x = [2, 1, 4, 6, 3, 2];
>> y = x(1);
>> for  k = 1 : length(x)
        if  x(k) > y
            y = x(k);
        else
            y = y;
        end
    end
```

Pero, en **MATLAB** está configurado el comando `max(variable)`. El cual, al ser aplicado a la variable x, lo que se desplegará al ejecutar dicho comando, será el valor *máximo* del arreglo, quedando almacenado en la variable y de la siguiente manera:

```
>> x = [2, 1, 4, 6, 3, 2];
>> y = max(x);
```

Comparando el resultado aplicando `max(x)` con el código anterior, se observa que utilizando las herramientas que ofrece **MATLAB** nos facilita y nos ahorra líneas de código. Analogamente, para conocer o desplegar el elemento de valor mínimo de un arreglo, podemos utilizar el comando `min`, como se muestra a continuación:

```
>> x = [2, 1, 4, 6, 3, 2];
>> y = min(x);
```

1.4.2. Sumatoria

Otro de los comandos que podemos encontrar en el entorno de **MATLAB**, es el comando *sum(variable)*, el cual realiza la suma discreta de los elementos de un vector o matriz x. Para los vectores tanto de fila como de columna, `sum` calcula la suma total de los elementos. Si x es una matriz, se calcula la suma discreta de cada columna y como resultado se obtiene un vector tipo fila, el cual estará formado por las sumas de todas las columnas. En resumen, la manera de realizar la suma varía según la estrucutra de los datos.

- Para **vectores** (tanto fila como columna), devuelve un escalar con la suma total de todos sus elementos.
- En el caso de **matrices**, opera a nivel columna, calculando la suma de cada columna y retornando un vector fila compuesto por estas sumas parciales.

A continuación se presentan ejemplos de lo antes descrito:

Ejemplo 1.5 Ejemplo de sumatoria de un vector

$>> sum([2, 1, 5]);$

$>> ans =$

8

■

Ejemplo 1.6 Ejemplo de sumatoria de una matriz

$>> sum([2, 1, 5; 9, 8, 5]);$

$>> ans =$

$11 \quad 9 \quad 10$

■

1.4.3. Ordenar

La función `sort` que se encuentra en el entorno de **MATLAB** ordena los elementos contenidos dentro de un vector de manera ascendente. El argumento puede ser un *vector fila*, un *vector columna* o una *matriz*. Su comportamiento varía según el tipo de entrada como se describe:

- Para vectores (fila o columna): La función ordena todos sus elementos.
- Para matrices: Se realiza el ordenamiento de manera independiente en cada una de las columnas.

A continuación, se presentan ejemplos de aplicación:

Ejemplo 1.7 Ejemplo de ordenar de un vector

$>> sort([2, 1, 5]);$

$>> ans =$

$1 \quad 2 \quad 5$

■

Ejemplo 1.8 Ejemplo de ordenar de una matriz

$>> sum([9, 1, 5; 2, 8, 4]);$

$>> ans =$

$$\begin{matrix} 2 & 1 & 4 \\ 9 & 8 & 5 \end{matrix}$$

■

1.4.4. Variables Aleatorias

Una de las ramas de las ciencias matemáticas más utilizadas en el análisis de sistemas de comunicaciones es la probabilidad, debido a que es necesario caracterizar los fenómenos que ocurren en el sistema. Y más específico, el estudio de la distribución uniforme. En el entorno de **MATLAB**, existe una herramienta para generar variables aleatorias distribuidas uniformemente (`rand`), y esta se logra con el comando $rand(n)$, donde n especifica el tamaño de la matriz de los números aleatorios que se deben generar. Si $n = 1$, se devuelve un solo número aleatorio que está entre 0 y 1. Si lo que se desea es generar una variable aleatoria con distribución normal (Gaussiana), el comando utilizado es `randn`.

1.5. Creación de archivos M

Hasta ahora, el trabajo se ha limitado a la ventana de comandos, ya que esta es apropiada si se ejecutan instrucciones breves, si no hay que teclear mucho código o si se desea hacer el llamado de alguna función. De no ser así, conviene que el usuario escriba un archivo *"M" de subrutina* o un archivo *"M" de función*, ya que permite guardar el archivo y, posteriormente, pueden realizarse correcciones tantas veces se necesite.

Para elaborar un archivo *"M"* en Macintosh o Windows, se deben seguir los siguientes pasos:

(a) Clic en el menú `File` (Archivo) en la parte superior de la ventana de comandos.

(b) En `New`, aparecerá una ventana nueva.

(c) Para guardar el archivo haga clic en `Save AS` del menú `File`.

(d) Guarde el archivo con el siguiente formato, `nombre.m` (ejemplo: `funcion.m`).

(e) El archivo puede ejecutarse desde la ventana de comando escribiendo el nombre del archivo sin la extensión seguido de `Enter` **(Gilat, 2008)**.

Para crear un archivo *"M"* en sistemas Unix inicie **MATLAB** desde el directorio destino donde se almacenarán los archivos. Abra simultáneamente un editor de texto en esa misma ubicación empleando cualquier programa de edición disponible. Modifique el contenido en la ventana del editor y guarde los cambios manteniendo la sesión activa. Asigne al archivo la extensión obligatoria `.m`. Después, nos dirigimos a la ventana de **MATLAB** y ejecutamos el código ingresando el nombre del archivo sin la extensión según lo indicado por **(Gilat, 2008)**.

El símbolo `%` dentro de archivos *"M"* denota comentarios, y todo el texto posterior a este caracter en la línea será ignorado durante la ejecución. Estos comentarios incorporados estratégicamente mejoran la documentación del código al explicar el propósito de cada una de las

variables y estructuras que conforman el archivo.

Ejemplo 1.9 Ejemplo de creación de un archivo *"M"* empleando el comando *rand*

Como ya se comentó en la sección anterior, el comando *rand* genera variables aleatorias con distribución uniforme entre 0 y 1, es decir, genera un número aleatorio ente estos dos números.

Si lanzamos una moneda legal, la probabilidad de que salga cara o salga cruz es la misma, es decir, de 0.5. Desarrolle el código que simule la acción de lanzar una moneda y que calcule la probabilidad de que salga cara.

Para dar solución a este problema, utilizaremos la definición de probabilidad relativa, la cual nos dice que para calcular la probabilidad de un evento, el evento se tiene que repetir un número grande de veces y, además, contabilizar los eventos que salen favorables. Una vez realizado esto, para el cálculo de la probabilidad sólo se toma el promedio, es decir, dividir el número de eventos favorables que se contabilizaron entre eventos totales.

A continuación presentamos el código que determina la probabilidad de que salga cara al lanzar una moneda.

Código 3.1.

Código

```
k=0;
t=10000;
for i=1:t
x=rand(1);
if x <= 0.5
k=k+1;
end
end
P=k/t
```

■

Nota Una de las grandes ventajas de los archivos *"M"* es que dentro de estos se pueden mandar a llamar a otros archivos *"M"*. Ahora a estos archivos se denominan archivos *"M" de función*.

1.6. Crear archivos M de función

Para desarrollar funciones propias en el entorno de **MATLAB**, las cuales pueden ser equivalentes a las subrutinas y funciones de otros lenguajes, se deben crear archivos *"M"* **(Gilat, 2008)**.

Para desarrollar funciones personalizadas en **MATLAB** equivalentes a subrutinas o funciones de otros lenguajes de programación se deben implementar archivos *"M"*. Estos archivos *"independientes"* contienen la definición completa de las funciones y operan como unidades modulares autocontenidas, y como se presenta en **(Gilat, 2008)**, las funciones almacenadas en archivos *"M"* externos cumplen el mismo rol estructural que las subrutinas en otros entornos de programación.

1.6.1. Función que devuelve una sola variable

En los sistemas de comunicaciones se requiere convertir señales contínuas a señales discretas en el tiempo, y este proceso se realizada midiendo la señal en momentos periódicos del tiempo. A este proceso se le conoce como *muestreo* y, en efecto, en el entorno de **MATLAB** podemos encontrar esta función a través del comando *sampling*. Consideremos la siguiente ecuación:

$$f(x) = \frac{sen(x)}{x}$$

Suponiendo que el archivo *"M"* se guarda como *ec_a.m*, su código es el siguiente:

Código

Código 3.2.

```
1 function y=ec_a(x)
2 y=sin(x)./x;
```

Podemos observar del **código 3.2** que el nombre del archivo *"M"* es idéntico al nombre de la *función* (que aparece a la derecha del signo de igual). En el archivo *"M"* se utilizan los operadores aritméticos de arreglos, así el argumento x puede ser un número escalar, un arreglo tipo vector o una matriz. Una vez que se guarda *ec_a.m* como archivo *"M"*, este se puede mandar a llamar desde la ventana de comandos o incluso dentro de otro archivo *"M"*. Explicado lo anterior, si escribimos la siguiente instrucción:

```
>> y = ec_a(20)
```

El entorno de **MATLAB** desplegará lo siguiente:

```
y =
    0.0456
```

En otro ejemplo, si ahora trabajamo con una matriz, como la que se presenta a continuación:

```
>> x = [10, 20; 5, 30];
>> y = ec_a(x)
```

El resultado también será una matriz como la que se presenta abajo:

```
>> y =;
   -0.0544   0.0456
   -0.1918   - 0.0329
```

1.6.2. Función que devuelve múltiples variables

Continuando con el tema de funciones, podemos mencionar que una función puede devolver más de una variable. Por ejemmplo, supongamos una función que tiene que evaluar la media y la desviación estándar de una serie de datos y, para devolver las dos variables, utilizaremos un vector en el miembro izquierdo del enunciado de la función como se puede observar abajo:

Código 3.3.

```
1 function [media, dvstd] = ec_b(x)
2 n=length(x);
3 media = sum(x)/n;
4 dvstd = sqrt(sum(x.^2)/n - media.^2);
```

Para utilizar esta función, el miembro derecho del enunciado de llamada también debe ser un vector. El código anterior debe guardarse como *ec_b.m*. Entonces:

$$>> x = [1\ \ 5\ \ 3\ \ 4\ \ 6\ \ 5\ \ 8\ \ 9\ \ 2\ \ 4];$$
$$>> [m, d] = ec_b(x)$$

El **Código 3.3** entonces generará y almacenará en las variables *"m"* y *"d"* el valor de estas dos operaciones como se puede observar a continuación:

$$m = \quad 4.7000$$
$$s = \quad 2.3685$$

1.6.3. Función que utiliza otra función

En secciones previas, se había comentado que el argumento de una función puede ser el nombre de otra función. Por ejemplo, supongamos la siguiente ecuación:

$$ec_c = f(a) + 2f(b) + f(c)$$

donde

$f(x)$ es la función que se nombraría en el argumento.

Ejemplo 1.10 Ejemplo de una función que utiliza otra función
El siguiente código ilustra una función ec_c.m que calcula la ecuación anterior:

Código

Código 3.4.

```
function mp = ec_c(nombre_f, a, b, c)
mp = (feval(nombre_f, a)+2*feval(nombre_f, b)+feval(
    nombre_f, c));
```

En el código anterior, nombre_f (una variable de cadena) es el nombre de la función $f(x)$. Si $f(x)$ es la función seno, nombre_f será sin. feval (nombre_f) es un comando de **MATLAB** *que evalúa la función llamada nombre_f para el argumento x. Por ejemplo, $y = feval('sin', x)$ equivale a $y = sin(x)$.*

Si queremos evaluar ec_a para la función definida por ec_c (el ***Código 3.4*** *se guardó como ec_c, se guardó como archivo "M") con $a = 10$, $b = 20$ y $c = 30$, entonces, el comando:*

$$>> A = ec_c('ec_a', 10, 20, 30)$$

Produce

0.0040

El número de argumentos de entrada y de salida de feval debe coincidir con el formato de la función nombre_f. A partir de este tema todos los códigos se consideran que son creados en archivos "M". ■

1.7. Cómo utilizar MATLAB con señales de tiempo continuo

Una señal de tiempo continuo $x(t)$, dada por una expresión matemática puede definirse y graficarse a través del entorno de **MATLAB**. Para tener una mejor comprensión de los términos, se sugiere que el alumnado se familiarice con los comandos básicos del entorno de **MATLAB** revisando diferentes tutoriales o libro de texto.

Un ejemplo de una señal continua en el entorno de **MATLAB** puede ser representada como $x(t)$:

$$x(t) = e^{-0.1t}\sin(\frac{2}{3}t)$$

Una gráfica de $x(t)$ **vs.** t, para un intervalo de valores de t, puede generarse mediante **MATLAB**. Por ejemplo, para un intervalo r entre 0 y 30 segundos, con incrementos de 0.1 segundos, las instrucciones de **MATLAB** para generar $x(t)$ son:

```
t = 0:0.1:30;
x = exp(-.1*t).*sin(2/3*t);
plot(t,x)
axis([0 30 -1 1])
grid
xlabel('Time (sec)')
ylabel('x(t)')
```

En el código, los valores de tiempo para los que x se grafica, se almacenan como elementos en el

vector t. Cada una de las expresiones `exp(-.1*t)` y `sin(2/3*t)` crea un vector con elementos iguales a los de la expresión evaluada en los valores de tiempo correspondientes. Los vectores resultantes deben multiplicarse, elemento por elemento, para definir el vector x. Como se ve en el comando `x=exp(-.1*t).*sin(2/3*t)`, las operaciones elemento por elemento necesitan un punto antes del operador. Entonces, mediante el comando `plot(t,x)`, x se grafica contra t.

El comando `axis` se utiliza para sobrescribir los valores predeterminados (por lo general, los predeterminados son aceptables, y este comando no se necesita). Es importante destacar que el uso del comando `axis` varía según la versión de **MATLAB** que se utilice. La gráfica resultante de $x(t)$ aparece en la **Figura 2**. Observe que la gráfica que genera **MATLAB** es en forma de caja, y los ejes son etiquetados como se muestra.

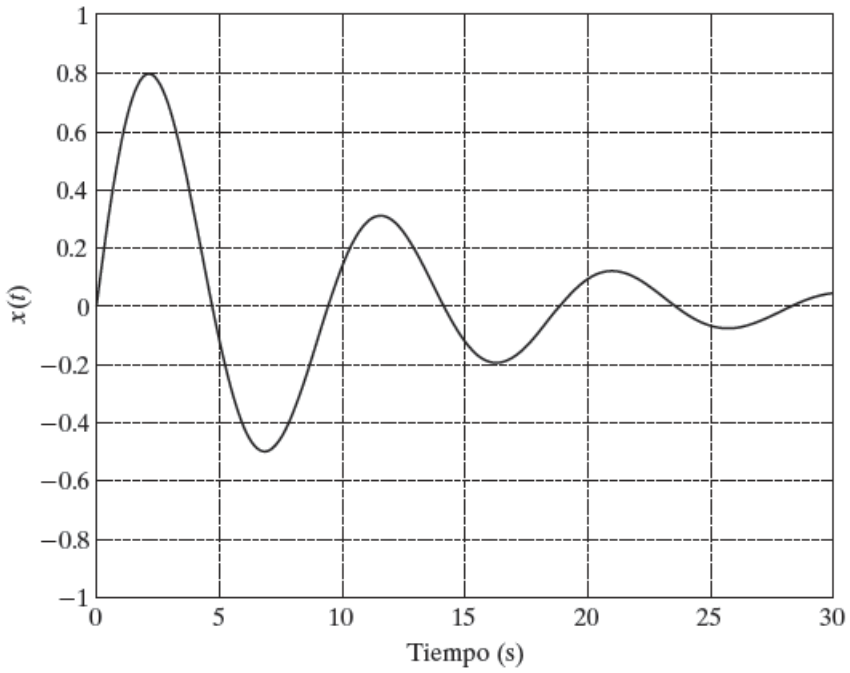


Figura 2: Gráfica en MATLAB para la señal $x(t) = e^{-0.1t}\sin(\frac{2}{3}t)$.

Es importante tomar en cuenta que, cuando generemos gráficas de señales de tiempo continuo con **MATLAB**, el incremento en el escalón de tiempo debe elegirse lo suficientemente pequeño para generar una gráfica suave. Si el incremento se elige demasiado grande (para una señal dada), entonces cuando los valores de la señal se conecten mediante líneas rectas (en la generación por computadora de la gráfica), el resultado será que la gráfica se verá dentada. Para ver este efecto, invitamos al lector a que ejecute de nuevo el programa anterior, utilizando un incremento de tiempo de 1 segundo para graficar $x(t) = e^{-0.1t}\sin(\frac{2}{3}t)$.

2. Gráficas con MATLAB

En el entorno de **MATLAB** las ecuaciones matemáticas se pueden expresar en relación de una, dos, tres o más dimensiones, sin embargo, entender estas ecuaciones sin la ayuda de gráficas resulta muy complicado. Es por esto que el empleo de gráficas en la ingeniería en general es muy importante, y más en aquellas áreas de estudios científicos y de ingeniería, e incluso hoy en día

en las áreas médicas **(Gilat, 2008)**.

En los lenguajes disponibles en las décadas pasadas resulta complicado presentar los resultados con gráficas. Las gráficas en el entorno de computación con **MATLAB** son una parte natural, es decir, son sencillas, y esto conlleva a que los resultados obtenidos de diferentes cálculos, puedan desplegarse con sencillos comandos. Analizar las ecuaciones matemáticas a través de gráficas es una forma atractiva y eficiente de aprender matemáticas. En esta sección se presentarán los temas más relevantes sobre las gráficas en **MATLAB** con la intención de ayudar al alumnado a adquirir los comandos básicos del entorno.

2.1. Gráficas simples

Suponga que se desea graficar lo siguiente:

$$\{x(t_0), x(t_1), x(t_2), x(t_3), ..., x(t_N), \}$$

En este punto, lo primero que hay que realizar es adecuar tanto t como y en arreglos idénticos, es decir, convertirlos en arreglos de tipo *fila* o de *columna* de la misma longitud. El comando a emplear para desplegar el gráfico es `plot`, la manera de emplearse este comando se puede observar en el **código 4.1**, el cual produce la gráfica de la **Figura 3 (Gilat, 2008)**.

Código

Código 4.1.

```
1 t=0:0.05:2;
2 y=cos(2*pi*t);
3 plot(t,y)
```

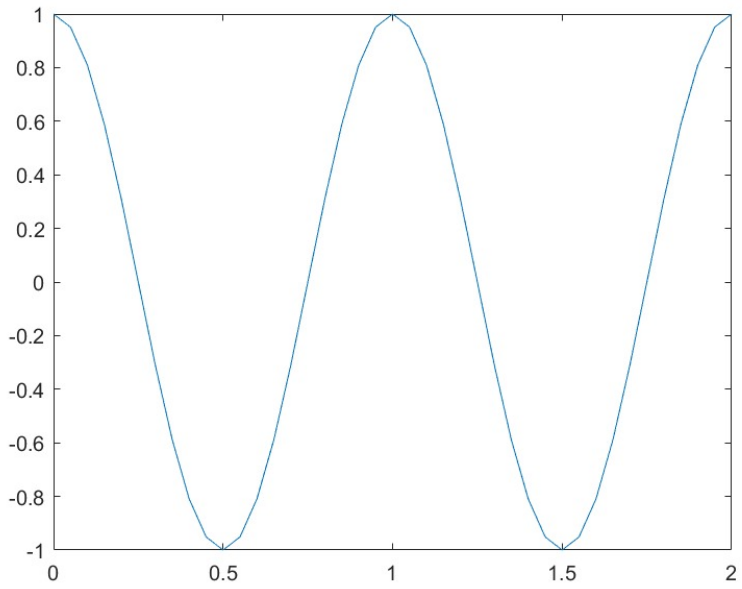


Figura 3: Gráfica usando plot sin asignar etiquetas a los ejes.

Para agregar etiquetas en los ejes se utilizan los comandos `xlabel` y `ylabel`. Para agregar un título en la gráfica se utiliza el comando `title`, esto se muestra a continuación en el siguiente **código 4.2**, el cual produce la gráfica de la **Figura 4**.

Código 4.2.

Código

```
t=0:0.05:2;
y=cos(2*pi*t);
plot(t,y)
xlabel('t ')
ylabel('y ')
title('Grafica del coseno')
```

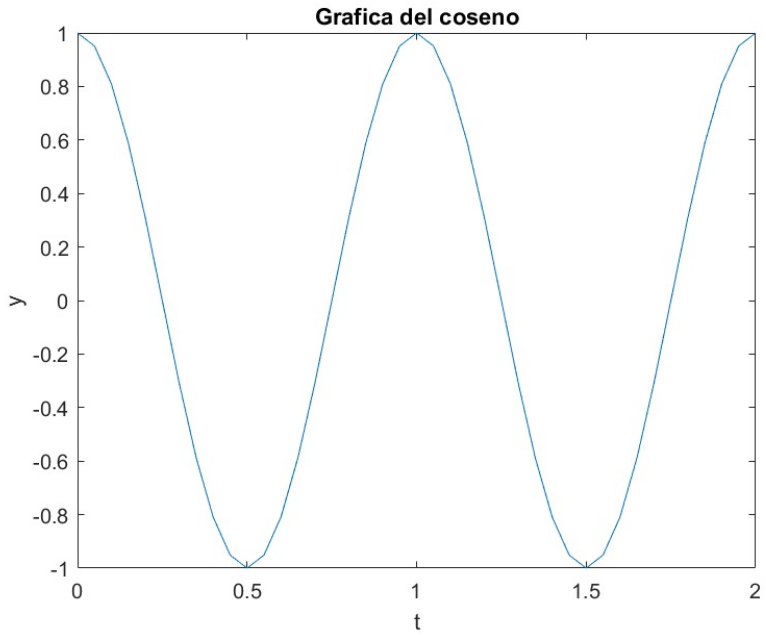


Figura 4: Gráfica de la **Figura 3** con etiquetas en los ejes.

2.2. Gráficas con marcas

En las gráficas del entorno de **MATLAB** se pueden representar los datos de manera puntual, es decir, colocar marcas en cada uno de los valores sin que estos estén conectados entre sí por líneas. En la **Tabla 4** se presentan algunos de los tipos de marcas o letras que se pueden utilizar. Si se desea graficar con un solo tipo de marca, lo que se debe hacer es colocar el símbolo después de las coordenadas en los argumentos de `plot`. La gráfica producida por el **código 4.3** se muestra en la **Figura 5** **(Gilat, 2008)**.

Código 4.3.

Código

```
t=0:0.05:2;
y=cos(2*pi*t);
plot(t,y, '+ ')
xlabel('t ')
ylabel('y ')
title('Grafica del coseno con marcas')
```

Tabla 4: Marcas para gráficas.

Tipo de marca	Símbolo
Punto	·
Más	+
Estrella	*
Círculo	○
Marca x	x

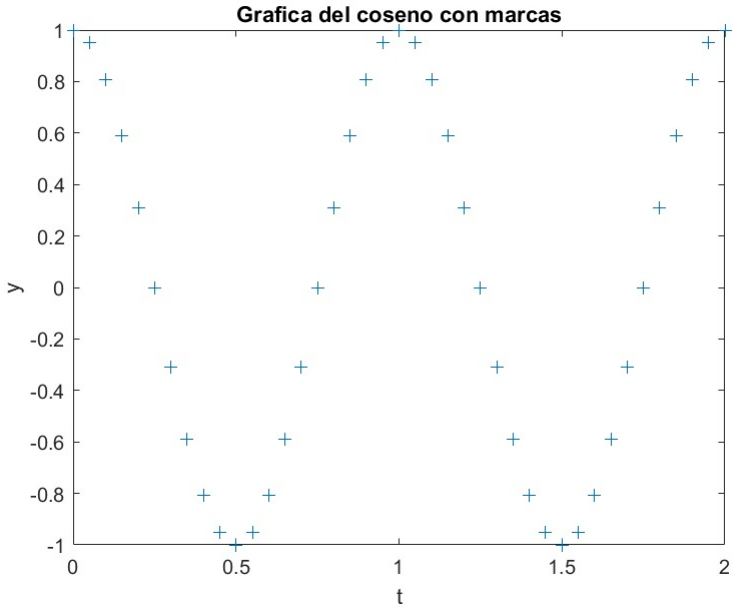


Figura 5: Gráfica con marcas.

2.3. Colores y tipos de líneas

En relación a las líneas con las que se unen los valores en las gráficas de **MATLAB**, en la **Tabla 5** se pueden observar cuatro tipos diferentes de líneas **(Gilat, 2008)**. Se hace notar que el tipo de línea por omisión es el continuo, y si se desea desplegar alguna gráfica con algún tipo de línea en particular, se debe especificar la marca de línea después de las coordenadas como se presenta en el ejemplo siguiente:

```
plot(x,y,'–')
```

En términos de *color* las opciones de pueden observar en la **Tabla 6**. Para asignar un color específico a una gráfica se debe colocar el símbolo del color, así como se realizó para el tema de la *marca*. Si es el mismo proceso que para los tipos de línea, en el argumento del comando `plot`, debemos escribir de la siguiente manera:

Tabla 5: Tipos de líneas.

Tipo de línea	Símbolo
Continua	$-$
Guiones	$--$
Punteada	:
Guiones y puntos	$-.$

```
plot(x,y,'g')
```

Tabla 6: Colores de líneas.

Color de línea	Símbolo
Rojo	r
Amarillo	y
Magenta	m
Turquesa	c
Verde	g
Azul	b
Blanco	b
Negro	k

Tanto el tipo de línea como el color se pueden combinar en una gráfica de **MATLAB**, y en el ejemplo siguiente se puede observa como es la sintaxis de una gráfica con marcas $+$ en color verde:

```
plot(x,y,'+g')
```

Grafica los datos con marcas + en color verde.

2.4. Eje y retícula

El valor mínimo y el valor máximo de los ejes coordenados, las marcas de la escala, así como los valores de las coordenadas en las marcas de escala el entorno de **MATLAB** los asigna de manera automática. Sin embargo, es posible modificar la forma del marco, así como los valores mínimos y valores máximos de las coordenadas con el comando `axis` **(Gilat, 2008)**.

Para modificar la forma del entorno figura, a una forma "cuadrada", se tiene que colocar la siguiente sintaxis:

```
axis('square')
```

Si se desea que los ejes de coordenadas y las marcas de escala se omitan, se debe colocar la siguiente instrucción:

axis('off')

Si ahora colocamos la instrucción `axis('on')`, lo que sucederá, será que se cancela la instrucción previa que fue (`axis('off')`). El valor máximo y el valor mínimo de las coordenadas en una gráfica se pueden especificar con la siguiente instrucción:

axis($[x_{min}, x_{max}, y_{min}, y_{max}]$)

Las líneas que se salgan de estos límites serán recortados. El comando `axis` se utiliza después de `plot` para poder modificar el área de visualización. Por ejemplo, en el **código 4.4** que realiza la gráfica del coseno (**Figura 6**), si queremos sólo graficar la parte del eje x con un rango de entre [0.75, 1.25] y del eje y con un rango de entre [0 1], se deben escribir las siguientes instrucciones:

Código 4.4.

Código

```
t=0:0.05:2;
y=cos(2*pi*t);
plot(t,y, '+ ')
axis([0.75, 1.25, 0, 1])
xlabel('t ')
ylabel('y ')
title('Grafica del coseno utilizando axis')
```

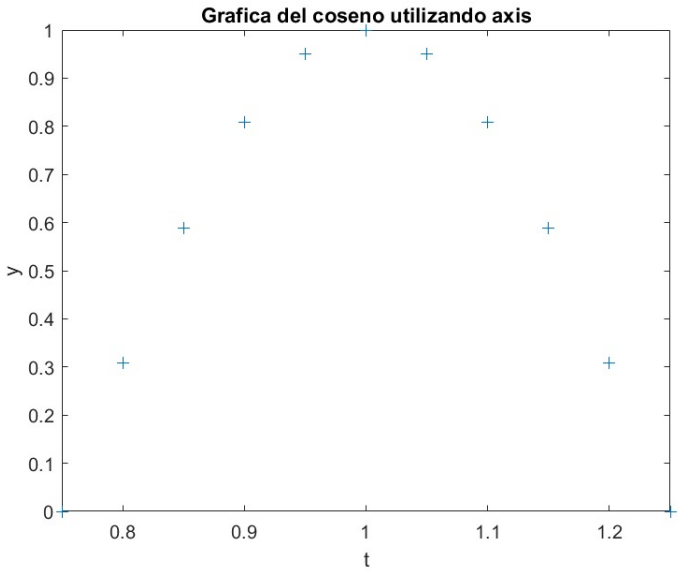


Figura 6: Uso de axis.

2.4.1. Retícula

En las gráficas del entorno de **MATLAB** se puede agregar una retícula a la gráfica con el comando `grid on`. Por otro lado, la acción que realiza el comando `grid off` es la de eliminar la retícula en la gráfica **(Gilat, 2008)**. Si se utiliza de manera simple el comando `grid`, la acción que se realizará será la de activar y desactivar la retícula alternadamente. El **código 4.5** es un ejemplo del empleo del comando `grid on`, el cual produce la gráfica de la **Figura 7**.

Código

Código 4.5.

```
t=0:0.05:2;
y=cos(2*pi*t);
plot(t,y)
grid on
xlabel('t ')
ylabel('y ')
title('Grafica del coseno con reticula')
```

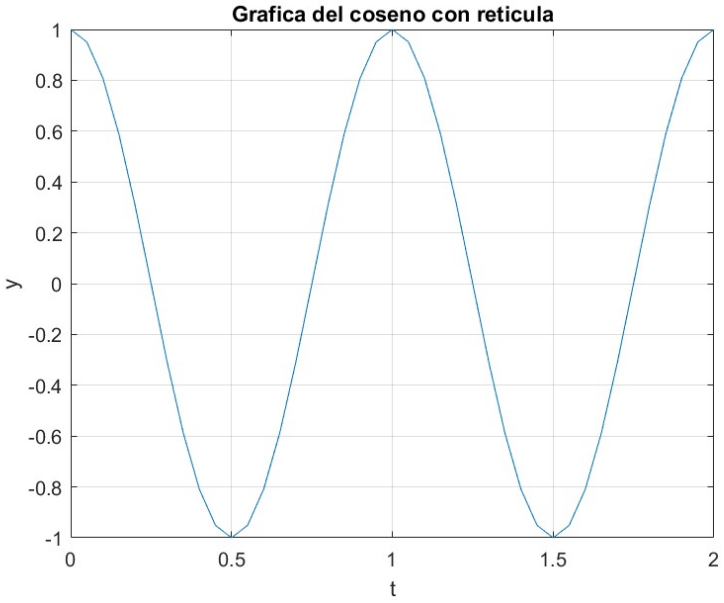


Figura 7: Gráfica usando retícula.

2.5. Gráficas polares

Uno de los diferentes tipos de gráficos en el entorno de **MATLAB** son las gráficas *polares*. Para realizar la gráfica de una función en *coordenadas polares* se emplea el comando `polar` **(Gilat, 2008)**. El **código 4.6** es un ejemplo del empleo del comando `polar`, el cual produce la gráfica de la **Figura 8**.

Código 4.6.

Código

```
t=0:0.05:pi+0.01;
y=sin(3*t).*exp(-0.3*t);
polar(t,y)
title('Grafica polar')
grid
```

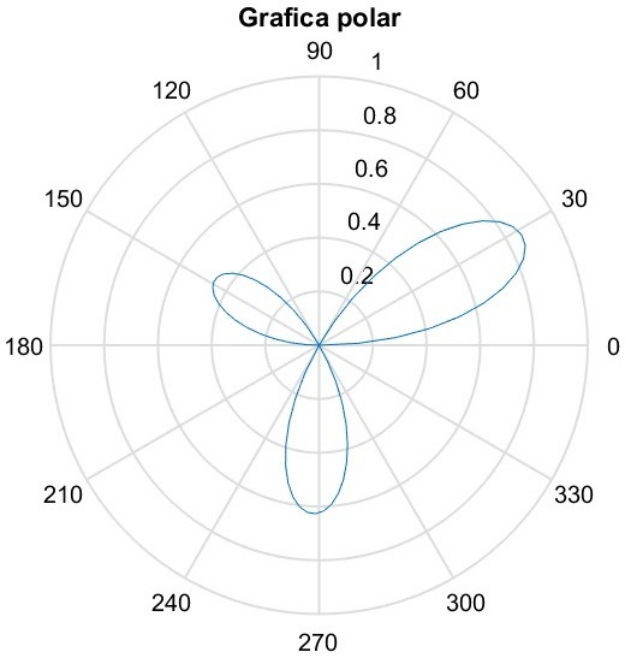


Figura 8: Gráfica usando el comando `polar`.

2.6. Gráficas logarítmicas y semilogarítmicas

En el área de ingeniería, la escala logarítmica es muy útil tanto en cálculos matemáticos como en la generación de gráficas. En el entorno de **MATLAB**, las funciones pueden graficarse en una escala log-log con el comando `loglog` como se muestra en el **código 4.7**. Este código produce la gráfica de la **Figura 9 (Gilat, 2008)**.

Código 4.7.

Código

```
t=0.1:0.1:3;
x=exp(t);
y=exp(t.*sinh(t));
loglog(x,y)
title('Grafica log-log')
xlabel( 'x ')
ylabel( 'y ')
grid
```

En la **Figura 9** se observa que, tanto el eje horizontal como el eje vertical, están en escala logarítmica. Si queremos cambiar esta característica, que sólo uno de los ejes esté en escala logarítmica, por ejemplo el eje y, el comando utilizado es `semilogy`, como se muestra en el siguiente **código 4.8**. El resultado de aplicar esta instrucción se puede observar en la gráfica de la **Figura**

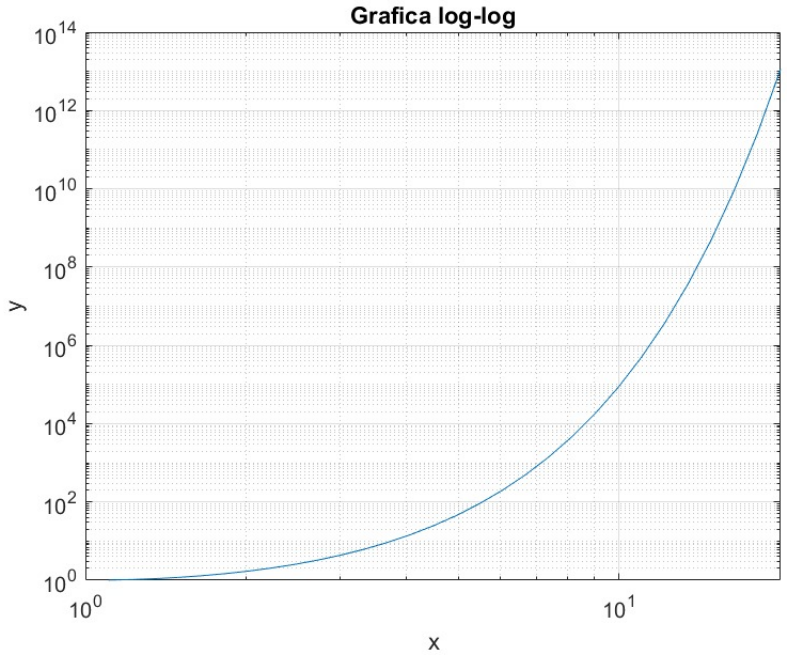


Figura 9: Gráfica en escala logarítmica.

10.

Código 4.8.

```
t=0.1:0.1:3;
y=exp(t.*t);
semilogy(t,y)
title('Grafica semilogy')
xlabel( 't ')
ylabel( 'y ')
grid
```

Código

De forma similar, si lo que se desea es el eje x en escala logarítmica el comando utilizado es `semilogx`.

2.7. Múltiples curvas

En el entorno de **MATLAB** si se desea desplegar dos o más curvas con una sola gráfica empleando el comando `plot`, se debe escribir todos los conjuntos de coordenadas repetidamente con el comando `plot`. En el siguiente **código 4.9**, se presenta es un ejemplo de lo antes explicado, el cual produce la gráfica de la **Figura 11** **(Gilat, 2008)**.

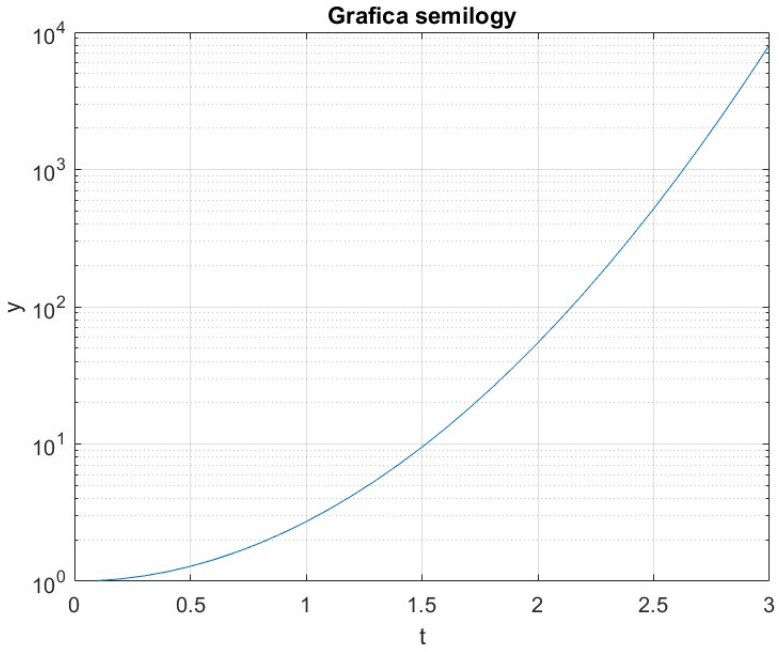


Figura 10: Gráfica en escala semilogarítmica.

Código 4.9.

Código

```
t=0:0.05:2;
y=cos(2*pi*t);
z=sin(2*pi*t);
plot(t,y,t,z)
xlabel('t ')
ylabel('y, z ')
title('Grafica del coseno y seno ')
grid
```

El entorno de **MATLAB** elegirá automáticamente diferentes tipos y colores de línea para las diferentes curvas contenidas dentro de una misma gráfica. Aunque, el usuario puede especificar el color de la línea, el tipo de línea e incluso el tipo de marca para representar los datos, y esto es posible indicándolo después de cada par de coordenadas. A continuación presentamos un ejemplo de cómo se debería colocar la sintaxis:

```
plot(x,y,'-',x,z,'*')
plot(x,y,':',x,z,'*g')
plot(x,y,'r',x,z,'y')
```

2.7.1. Retención

Hasta este punto de las notas hemos graficado las curvas en una sola operación con el comando `plot`. Sin embargo, frecuentemente se tiene la necesidad de añadir alguna curva a una gráfica que ya se desplegó. Esta necesidad puede ser solventada con el comando `hold on`. En el siguiente **código 4.10**, se presentan las instrucciones para resolver esta necesidad de añadir una curva,

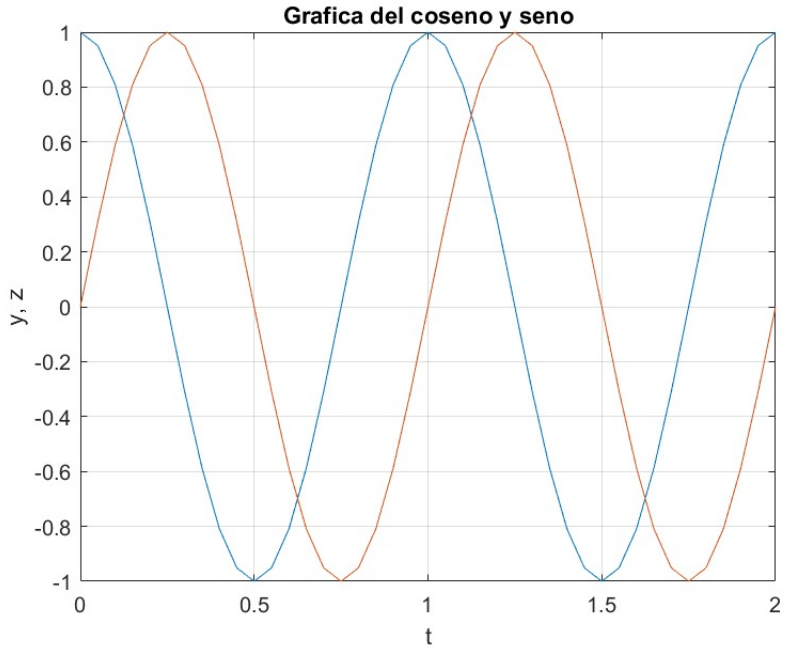


Figura 11: Múltiples curvas.

el código despliega la gráfica de la **Figura 12** **(Gilat, 2008)**.

Código 4.10.

Código

```
t=0:0.05:2;
y=cos(2*pi*t);
plot(t,y)
hold on
z=sin(2*pi*t);
plot(t,z, '- - ')
xlabel('t ')
ylabel('y(-), z( '- - ')
title('Grafica del coseno y seno ')
grid
```

Una vez que se ha agregado el comando `hold on`, la gráfica se mantiene en la pantalla e incluso si se ejecuta otro código. Debido a esto, se requiere insertar el comando `hold off` tanto al principio como al final para que las nuevas gráficas agregadas a los ejes borren las gráficas existentes y restablezcan todas las propiedades de los ejes. El siguiente **código 4.11** es un ejemplo del uso del comando `hold on`.

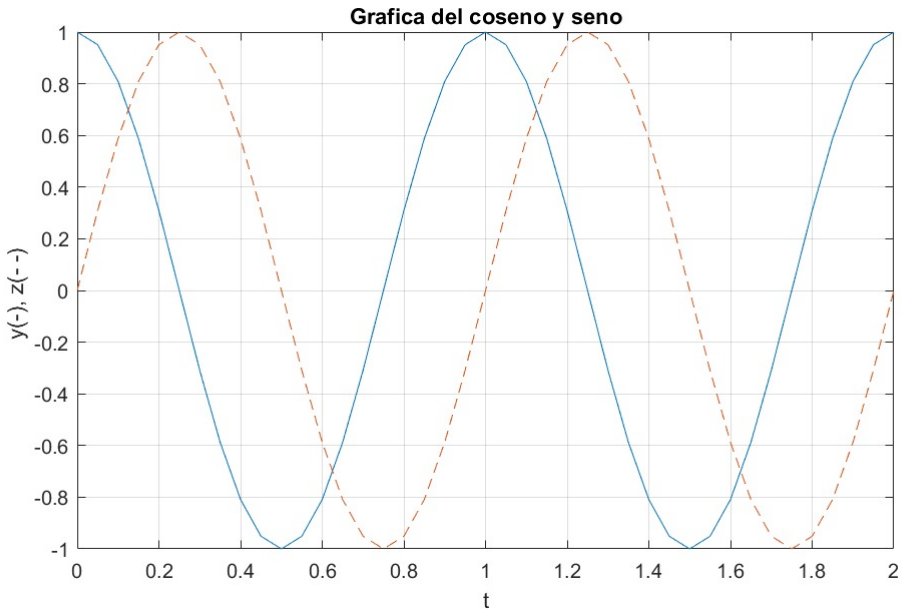


Figura 12: Múltiples curvas.

Código 4.11.

Código

```
clear; clf; hold on
t=0:0.05:2;
y=cos(2*pi*t);
plot(t,y) hold on
z=sin(2*pi*t);
plot(t,z, '- - ') hold on
xlabel('t ') ylabel('y(-), z(- -) ')
title('Grafica del coseno y seno ')
grid
```

Cuando graficamos diferentes curvas empleando el comando `hold on`, se recomienda especificar los valores mínimos y máximos de los ejes coordenados con el comando `axis`, ya que si no se realiza esta especificación, los límites se determinarán por default con base en los datos de la primera curva.

El comando `hold on` es un comando que resulta indispensabble cuando se está trabajando una gráfica que tarda mucho en desplegarse. Por este motivo, los comandos para modificar los parámetros de las figuras tales como:

1. Los ejes,
2. El mapa de color
3. Los ángulos de perspectiva
4. El eje de color, entre otros,

se pueden modificar después de haber desplegado una figura, sin embargo, cada vez que se ejecute un comando nuevo se dibujará la figura completa. Para reducir el tiempo, se sugiere emitir todos

los comandos de los parámetros antes de graficar, retener la figuran con el comando `hold on`, y luego el comando de `plot`.

2.7.2. Leyenda

En las gráficas del entorno de **MATLAB** se pueden agregar leyendas a estas, utilizando el comando `legend`. El siguiente **código 4.12** es un ejemplo de cómo se utiliza el comando `legend`, el cual produce la gráfica de la **Figura 13** **(Gilat, 2008)**.

Código 4.12.

Código

```
t=0:0.05:2;
y=cos(2*pi*t);
plot(t,y)
hold on
z=sin(2*pi*t);
plot(t,z, '- - ')
legend( 'cos ', 'sin ', 2)
xlabel('t ')
ylabel('y(-), z( '- - ')
title('Grafica del coseno y seno ')
grid
```

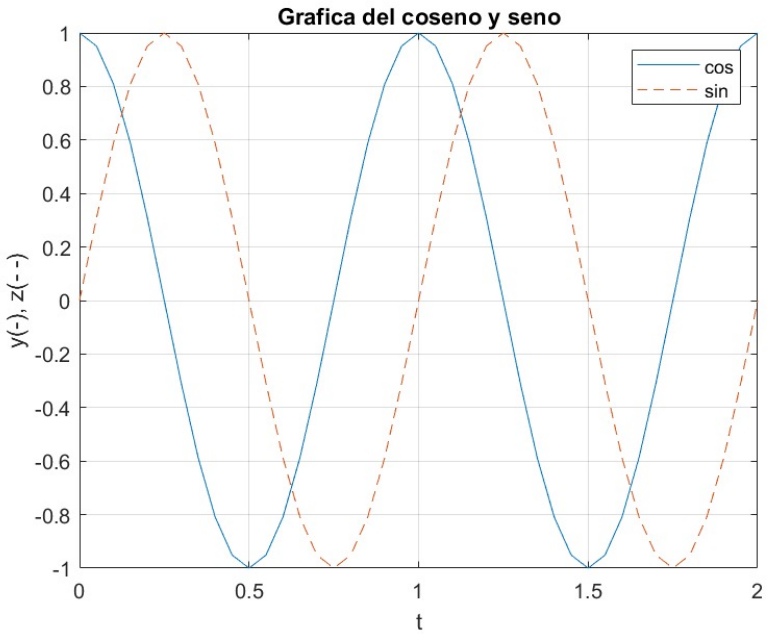


Figura 13: Gráfica con legend.

El primer parámetro es nombre o etiqueta para la primera gráfica, el segundo parámetro corresponde a la segunda gráfica, el tercer parámetro corresponde a la posición de la leyenda.

2.7.3. Figura

Es posible abrir múltiples ventanas de gráficos con `figure`. Si sólo se utiliza una ventana, no hay necesidad de usar este comando, pero si se requiere trabajar en más de una ventana de gráficos se pueden abrir con este comando. Si se requiere volver a activar una ventana de algún gráfico que ya se había desplegado, se puede utiliar el comando `figure(n)`. Este comando reactivará la `n-ésima` ventana, donde n es el número secuencial de figura que aparece en la parte superior de la ventana de gráficos. Las propiedades de las figuras, como su tamaño, ubicación, mapa de color asignado, etc., pueden modificarse.

2.8. Subgráficas

Con el comando `subplot` podemos desplegar diferentes gráficas en una misma ventana de figura, es decir, distribuir en m por n gráficas en una sola ventana de figura. La sintaxis la presentamos a continuación **(Gilat, 2008)**:

```
subplot(m,n,k)
```

Nota Los argumentos m, n y k son números enteros.

En esta sintaxis, tanto m como n, representan un arreglo de m por n gráficas, por lo que k es el número secuencial de la gráfica (cuadrante). El siguiente **código 4.13**, es un ejemplo del empleo del comando `subplot`, el cual produce la gráfica de la **Figura 14**.

Con el comando `subplot` podemos trazar un arreglo vertical de dos gráficas con las siguientes instrucciones:

```
subplot(2,1,1),plot(...
```

```
subplot(2,1,2),plot(...
```

De forma similar, trazamos una fila de dos gráficas con:

```
subplot(1,2,1),plot(...
```

```
subplot(1,2,2),plot(...
```

Código 4.13.

Código

```
clear; clf
t=0:0.05:2;
x=cos(2*pi*t);
y=sin(2*pi*t);
z=(cos(2*pi*t)).^2;
w=(sin(2*pi*t)).^2;
subplot(2,2,1)
plot(t,x)
xlabel('t ')
ylabel('x ')
title('Subgrafica 2,2,1 ')
grid on
subplot(2,2,2)
plot(t,y)
xlabel('t ')
ylabel('y ')
title('Subgrafica 2,2,2 ')
grid on
subplot(2,2,3)
plot(t,z)
xlabel('t ')
ylabel('z ')
title('Subgrafica 2,2,3 ')
grid on
subplot(2,2,4)
plot(t,w)
xlabel('t ')
ylabel('w ')
title('Subgrafica 2,2,4 ')
grid on
```

2.9. Versión tridimensional de una gráfica

El comando `plot3` es un comando para generar gráficas en la versión tridimensional de `plot`. Para el comando `plot3`, todas las reglas y comandos que ya explicamos para `plot` se pueden aplicar también. El siguiente **código 4.14**, es un ejemplo de cómo desplegar una gráfica 3D, el cual produce la gráfica de la **Figura 15** **(Gilat, 2008)**.

Código 4.14.

Código

```
clear, clf
t=0:0.1:20;
r=exp(-0.2*t);
th=pi*t*0.5;
z=t;
x=r.*cos(th);
y=r.*sin(th);
plot3(x,y,z)
hold on
plot3([1,1], [-0.5,0], [0,0])
xlabel( 'X ');
ylabel( 'Y ');
zlabel( 'Z ');
```

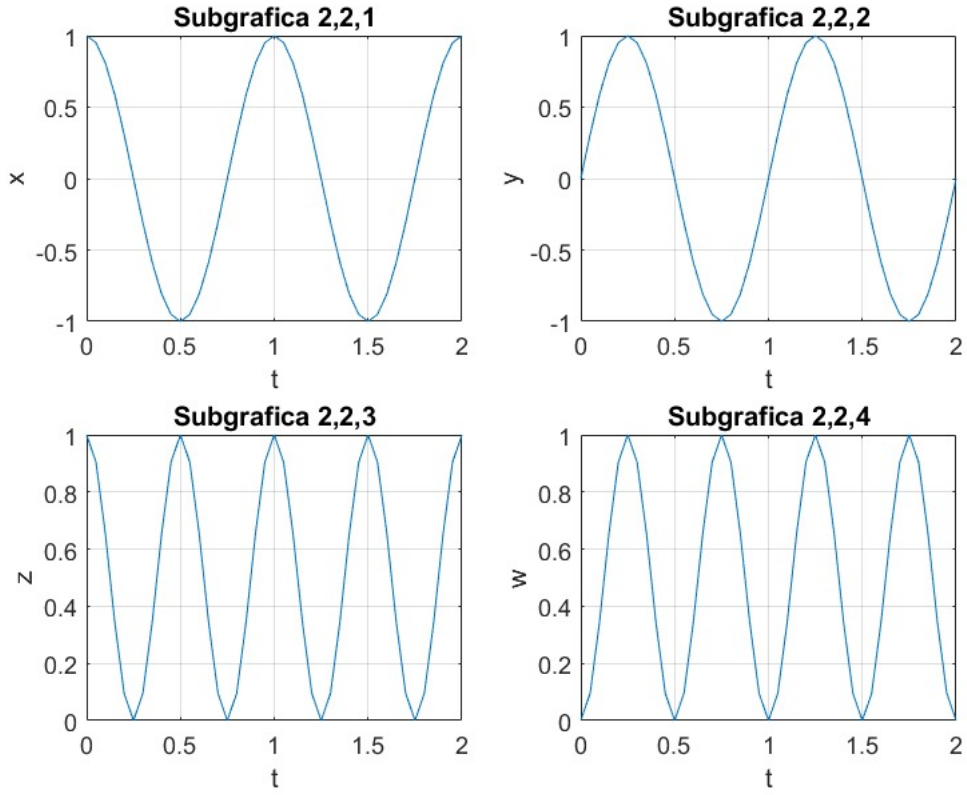


Figura 14: Gráfica usando subplot.

También puede utilizarse el comando `axis` como se presenta a continuación.

axis($[x_{min}; x_{max}; y_{min}; y_{max}; z_{min}; z_{max}]$)

3. Aplicaciones en Comunicaciones

Dos de los campos más importantes para un ingeniero en telecomunicaciones son las comunicaciones analógicas y comunicaciones digitales, ya que son la base y son fundamentales en la formación en su profesión. En esta sección nos enfocaremos en mostrar algunas aplicaciones que se utilizan para dichos campos, como son el muestreo de las señales, la cuantización, la modulación, la representación de las señales digitales mediante códigos de líneas, entre otros.

3.1. Conceptos básicos de señales

Una señal $x(t)$ se considera una función, la cual, se asume que tendrá valores *reales* o *escalares* de la variable independiente (t). Cuando hablamos de un término con valor *real* significa que cualquier valor que tome t (fijo), la señal $x(t)$ resultará en un número real. Cuando a t se le asignan estos valores ($\Re$), se dice que es una *variable de tiempo continuo* y, por lo tanto, la señal $x(t)$ también se considera como una *señal de tiempo continuo*, o en su defecto, una *señal analógica* como se estudia en **(Oppenheim, Willsky, & Nawab, 1998)**. Existen infinidad de señales

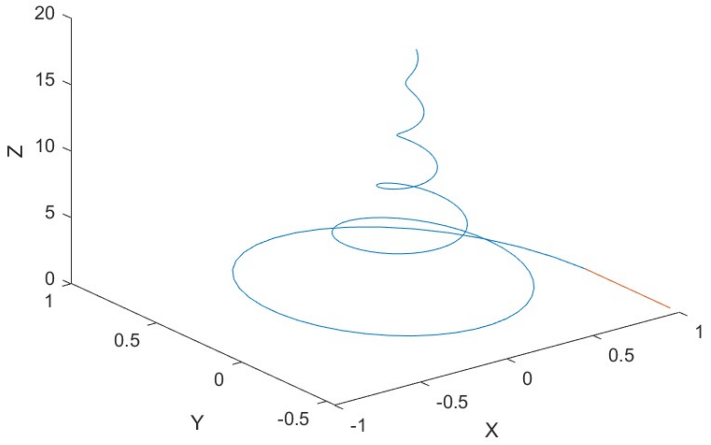


Figura 15: Gráfica en 3D.

analógicas, sin embargo los ejemplos más comunes de este tipo de señales los podemos asociar con señales de voltaje, señales que representan ondas de corriente de un circuito eléctrico, la voz, señales de audio, ondas musicales, representación de la velocidad de un cuerpo en movimiento, cualquier tipo de fuerza mecánica, cualquier señal bioeléctrica (electrocardiogramas (ECG) o electroencefalogramas (EEG)), flujos de líquidos, etc. Por naturaleza, las señales continúas $x(t)$ no siempre son fáciles de representar a través de una función matemática, por lo que es muy complejo intentar darle esta representación. Por ejemplo, si consideramos una señal de voz de duración de 50 ms como la que se observa en la **Figura 16**, aquí se escucha la pronunciación de la palabra *should*, la cual quedá representada por estas variaciones cuando transciende de la *"sh"* a *"u"*.

Esta expresión, al ser complicada de expresar, las variaciones de amplitud al pronunciar la palabra no se especifican en forma matemática, sino que se establecen como un conjunto de muestras tal y como lo menciona **(Kamen & Heck, 2008)**. Analizando nuevamente la **Figura 16**, $x(t)$ que es la señal de la voz, puede representarse a través de un conjunto de muestras como se observa a continuación.

Consideremos a $x(t)$, la cual expresa la señal de voz de la **Figura 16**. Esta señal puede representarse mediante el conjunto de muestras como se presenta a continuación:

$$\{x(t_0), x(t_1), x(t_2), x(t_3), ..., x(t_N),\}$$

En donde:

$x(t_i)$	valor de la señal cuando es evaluada en un t_i, donde i toma valores de $0, 1, 2, ...,$ hasta N.
$N+1$	será el número de puntos muestreados.

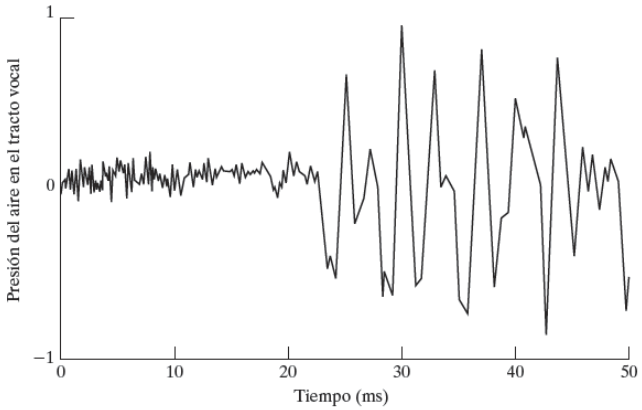


Figura 16: Segmento de un diálogo de la palabra *should*.

Para representar cualquier señal $x(t)$ en un conjunto de muestras como las presentadas arriba, se le realiza a la señal $x(t)$ el proceso de muestreo. Además, de poder con esto obtener la representación matemática de una señal, las señales también pueden analizarse en términos de su *"contenido de frecuencias"* o *"espectro en frecuencia"*. Cuando hablamos de representar a una señal en términos de su espectro en frecuencia podemos utilizar la transformada de Fourier, la cual se abordará en la siguiente sección (**Sección 4**).

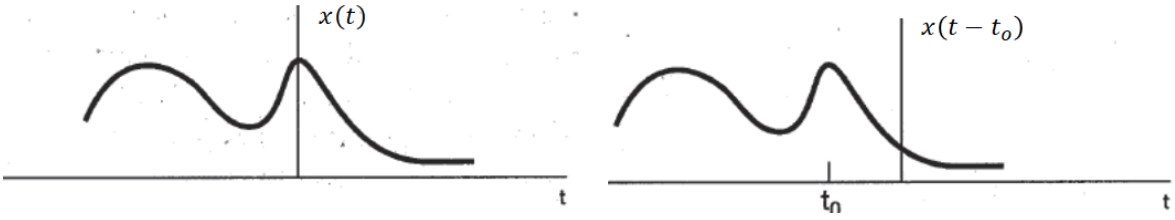
Una señal puede presentar una transformación de su variable independiente como **(Kamen & Heck, 2008)**:

- Corrimiento de tiempo
- Inversión de tiempo
- Escalamiento de tiempo

3.1.1. Corrimiento de una señal en el tiempo

Una señal $x(t)$ puede presentar corrimiento de tiempo si esta señal se desplaza en el eje del tiempo, es decir, representa una versión de $x(t)$ desplazada como se pude observar en la **Figura 17 (Kamen & Heck, 2008)**.

El valor de t_0 puede ser negativo o positivo.



(a) Señal original ($x(t)$) (b) señal desplazada ($x(t_0)$)

Figura 17: Corrimiento en tiempo de una señal.

$x(t - t_0)$

t_0 positivo ⇒ Retardada

t_0 negativo ⇒ Adelantada

Ejemplos de corrimiento de una señal en el tiempo pueden ser las señales de radar, sonar, procesamiento de señales sísmicas, etc.

Ejemplo 3.1 Corrimiento de una señal en el tiempo

El siguiente **código 5.1** realiza el corrimiento de una señal en el tiempo como se puede ver en la **Figura 18**.

Código 5.1.

```
%% Desplazamiento
t1=-1:0.1:0;x1=[0*t1];
t2=0:0.01:1;x2=[1+0*t2];
t3=1:0.01:2;x3=[-t3+2];
t4=2:0.01:3;x4=[0*t4];
t=[t1 t2 t3 t4];x=[x1 x2 x3 x4];
plot(t,x,'r', 'LineWidth', 2, 'DisplayName','Senal original');
grid on;
hold on;
%% Desplazamiento-Retraso por 3 unidades
to=3;
plot(t+to,x,'k', 'LineWidth', 2,'DisplayName','Senal retrasada');
%% Desplazamiento-Adelanto por 3 unidades
xlabel ('t', 'FontName', 'Times', 'Fontsize', 14);
ylabel ('Amplitud', 'FontName', 'Times', 'Fontsize', 14);
title ('Desplazamiento de una seal', 'FontName', 'Times', 'Fontsize', 16);
plot(t-to,x,'g', 'LineWidth', 2,'DisplayName','Senal adelantada');
```

Código ■

3.1.2. Inversión de una señal en el tiempo

Una señal $x(t)$ puede presentar inversión en el tiempo si esta señal se invierte en $t = 0$, es decir, representa una versión de $x(t)$ invertida $(x(-t))$ como se pude observar en la **Figura 19 (Kamen & Heck, 2008)**.

Ejemplos de la inversión de una señal en el tiempo pueden ser una grabación de audio pero que se reproduce en sentido contrario.

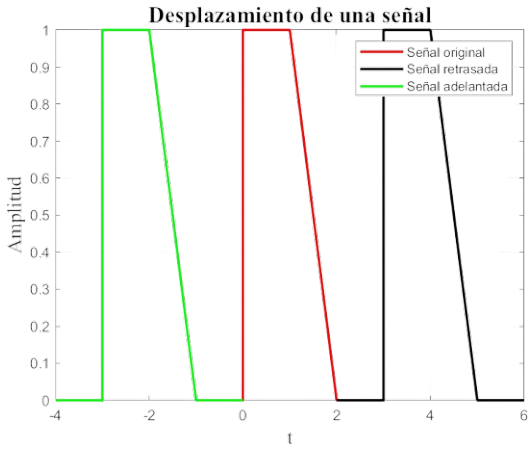
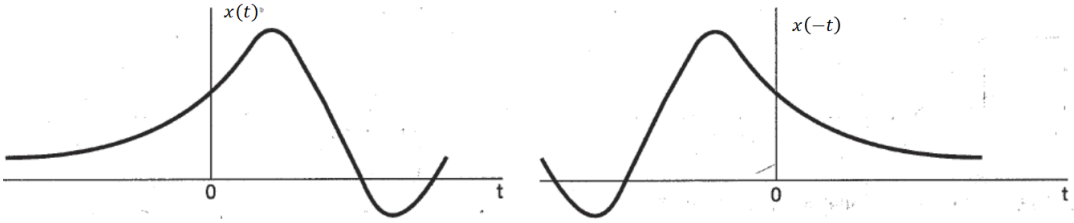


Figura 18: Corrimiento en el tiempo de una señal.



(a) Señal original ($x(t)$) (b) señal invertida ($x(-t)$)

Figura 19: Inversión en tiempo de una señal.

Ejemplo 3.2 Inversión en tiempo de una señal

El siguiente **código 5.2** realiza la inversión en tiempo de una señal como se puede ver en la **Figura 20**.

Código 5.2.

```
%% Inversion
t1=-1:0.01:0;x1=[0*t1];
t2=0:0.01:1;x2=[1+0*t2];
t3=1:0.01:2;x3=[-t3+2];
t4=2:0.01:3;x4=[0*t4];
t=[t1 t2 t3 t4];x=[x1 x2 x3 x4];
plot(t,x,'r','LineWidth',2,'DisplayName','Senal
    original');
grid on;
hold on;
%% Inversion
t_inv=-t;
xlabel('t','FontName','Times','Fontsize',14);
ylabel('Amplitud','FontName','Times','Fontsize',
    14);
title('Inver. de una senal','FontName','Times','
    Fontsize',16);
plot(t_inv,x,'k','LineWidth',2,'DisplayName','
    Senal invertida');
```

Código ■

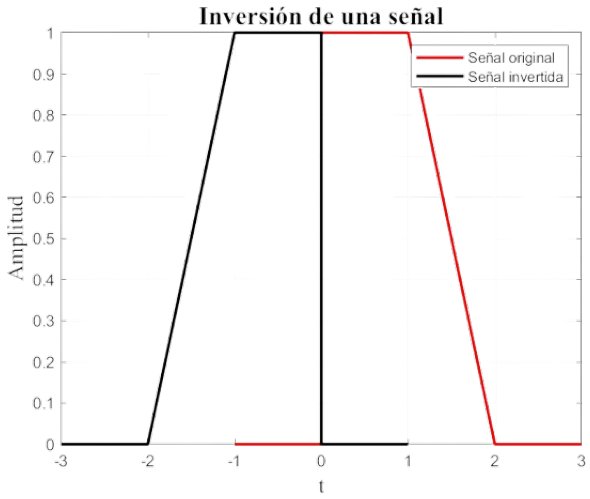


Figura 20: Inversión en el tiempo de una señal.

3.1.3. Escalamiento de una señal en el tiempo

Una señal $x(t)$ puede ser escalada en tiempo si esta señal se modifica por un factor n que afecte la variable independiente (t), es decir, una señal cuyo argumento se vea multiplicado o dividido por este factor n como se pude observar en la **Figura 21 (Kamen & Heck, 2008)**.

Ejemplos del escalamiento en tiempo de una señal puede ser una grabación de audio, es decir, para la señal $x(nt)$ la grabación de audio será n veces más rápida, lo contrario para para la señal $x(t/n)$, donde la grabación de audio será n veces más lenta.

Ejemplo 3.3 Expansión en el tiempo de una señal

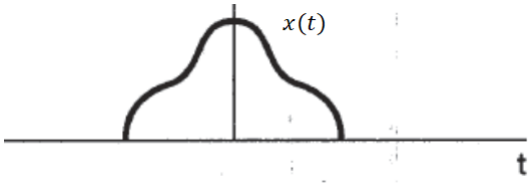
El siguiente **código 5.3** realiza la expansión en tiempo de una señal como se puede ver en la **Figura 22**.

Código 5.3.

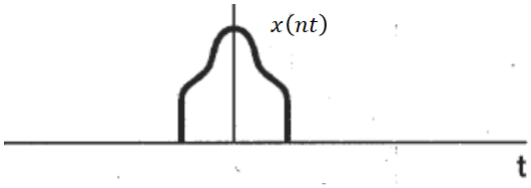
Código

```
t1=-1:0.01:0;x1=[0*t1];
t2=0:0.01:1;x2=[1+0*t2];
t3=1:0.01:2;x3=[-t3+2];
t4=2:0.01:3;x4=[0*t4];
t=[t1 t2 t3 t4];x=[x1 x2 x3 x4];
plot(t,x,'r', 'LineWidth', 2, 'DisplayName','Senal
    original');
grid on;
hold on;
%% Expans. n=2
n=2;
t_compr=n*t;
xlabel ('t', 'FontName', 'Times', 'Fontsize', 14);
ylabel ('Amplitud', 'FontName', 'Times', 'Fontsize',
    14);
title ('Expans.', 'FontName', 'Times', 'Fontsize',
    16);
plot(t_compr,x,'k', 'LineWidth', 2, 'DisplayName','
    Senal expandida');
```

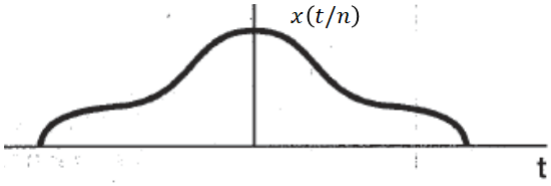
■



(a) Señal original ($x(t)$)



(b) Señal ($x(nt)$)



(c) Señal ($x(t/n)$)

Figura 21: Escalamiento en tiempo de una señal.

Ejemplo 3.4 Compresión en el tiempo de una señal

El siguiente **código 5.4** realiza la compresión en tiempo de una señal como se puede ver en la **Figura 23**.

Código 5.4.

Código

```
t1=-1:0.01:0;x1=[0*t1];
t2=0:0.01:1;x2=[1+0*t2];
t3=1:0.01:2;x3=[-t3+2];
t4=2:0.01:3;x4=[0*t4];
t=[t1 t2 t3 t4];x=[x1 x2 x3 x4];
plot(t,x,'r', 'LineWidth', 2, 'DisplayName','Senal
    original');
grid on;
hold on;
%% Compres n=2
n=2;
t_compr=t/n;
xlabel ('t', 'FontName', 'Times', 'Fontsize', 14);
ylabel ('Amplitud', 'FontName', 'Times', 'Fontsize',
    14);
title ('Compres. de una senal', 'FontName', 'Times',
    'Fontsize', 16);
plot(t_compr,x,'k', 'LineWidth', 2, 'DisplayName','
    Senal comprimida');
```

■

En la siguiente sección, se presentan algunas señales de tiempo continuo, y las cuales se pueden

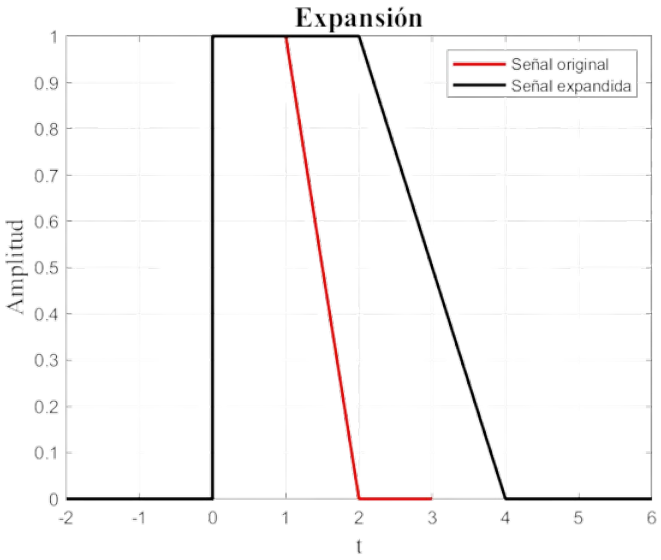
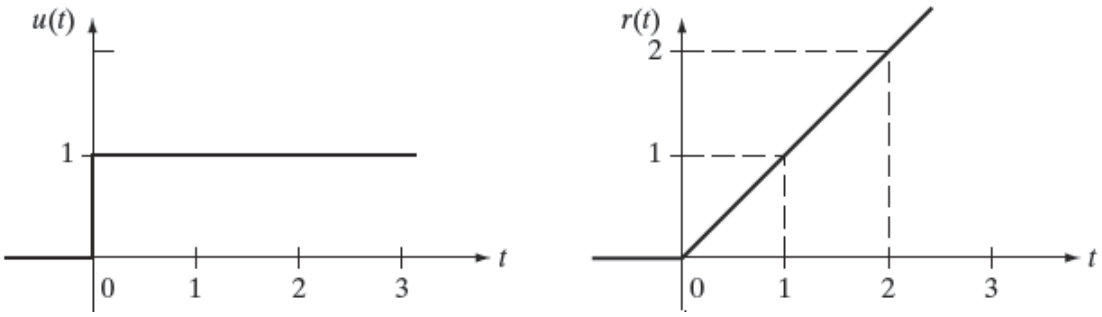


Figura 22: Expansión en el tiempo de una señal.

expresar en forma matemática.

3.1.4. Función escalón unitario y rampa unitaria

Las señales de tiempo continuo básicas e indispensables en los cursos de **MATLAB** son la función escalón unitario y la función rampa unitaria. La función escalón unitario se representa como $u(t)$ y la función rampa unitaria se representa como $r(t)$. La función escalón unitario se puede observar en la **Figura 24(a)** y la función rampa unitaria se puede observar en la **Figura 24(b)** **(Kamen & Heck, 2008)**.



(a) (b)

Figura 24: Funciones: (a) escalón unitario y (b) rampa unitaria.

La representación matemática para una función escalón unitario ($u(t)$) se presenta a continuación:

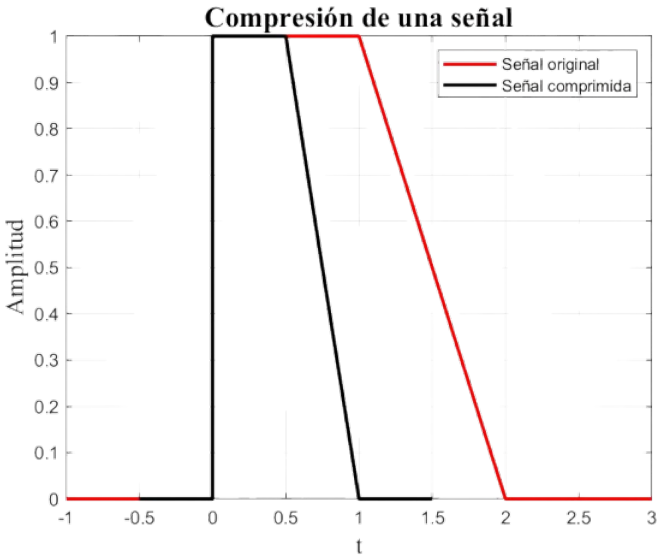


Figura 23: Compresión en el tiempo de una señal.

$$u(t) = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0 \end{cases} \tag{1}$$

Para este material, la función *escalón unitario* se define con una amplitud de $u(t)$ es igual a 1 para toda $t \geq 0$.

Nota En algunos casos, dependiendo la bibliografía consultada, la función de escalón unitario está definido como $u(0) = 1$, mientras que en otros está definido como $u(0) = 0$.

Si la variable K se considera como un número real diferente de cero, entonces, $Ku(t)$ es la función escalón tomando como valor de amplitud el valor que se le asigne a K para valores mayores o iguales a cero $(t \geq 0)$. Consideremos una señal de tiempo continuo $x(t)$, y si a esta señal le aplicamos la operación de producto con una función escalón, es decir, $x(t)u(t)$, el resultado será igual a $x(t)$ para los valores que tome t mayores o iguales a cero $(t \geq 0)$. Para cuando t tome valores menores a cero $(t < 0)$, el resultado del producto será cero. En conclusión, realizar la multiplicación una señal $x(t)$ por una función escalón $(u(t))$, resultará en eliminar cualquier valor diferente de cero de $x(t)$ para $t < 0$.

La representación matemática para una función rampa unitaria $(r(t))$ se presenta a continuación:

$$r(t) = \begin{cases} \mathrm{t}, & \mathrm{t} \geq 0 \\ 0, & \mathrm{t} < 0 \end{cases} \tag{2}$$

En esta representación matemática podemos observar que para $t \geq 0$, la pendiente de $r(t)$ es de valor 1 ($m = 1$), por lo que se considera que $r(t)$ tiene *"pendiente unitaria"*. Derivado de lo anterior, es que a $r(t)$ se le conoce como la *"función rampa unitaria"*. Si nuevamente, la variable K es un número escalar (número real) cualquiera diferente de cero, la función rampa $Kr(t)$, tendrá una pendiente m que tomará los valores de K para $t \geq 0$.

Analizando estas dos funciones, podemos observar que la función rampa unitaria $r(t)$ se puede representar como la integral de la *"función escalón unitario"* $u(t)$ como se presenta a continuación:

$$r(t) = \int_{-\infty}^{t} u(\lambda) d\lambda \tag{3}$$

Nota Caso inverso, podemos identificar a la 1ra. derivada de $r(t)$ respecto a t es igual a $u(t)$, excluyendo a $t = 0$, donde la derivada de $r(t)$ no está definida.

Ejemplo 3.5 Función escalón unitario

El siguiente **código 5.5** grafica la *"función escalón unitario"*, y la gráfica que despliega este código se observa en la **Figura 25**.

Código 5.5

Código

```
%% Funcion escalon
t = -5:.001:5;
uo = heaviside(t);
plot(t,uo,'LineWidt',3,'Color','r');
axis([-5 5 -1 2])
hold on;
plot([-5 5],[0 0])
plot([0 0],[-1 2])
grid on;
title('Grafica de la se al escalon unitario u_{0}(t)')
xlabel('Variable independiente t')
ylabel('u(t)')
```

■

Ejemplo 3.6 Suma de escalones unitarios

El siguiente **Código 5.6** grafica la suma de diferentes escalones unitarios como se puede observar en la **Figura 26**.

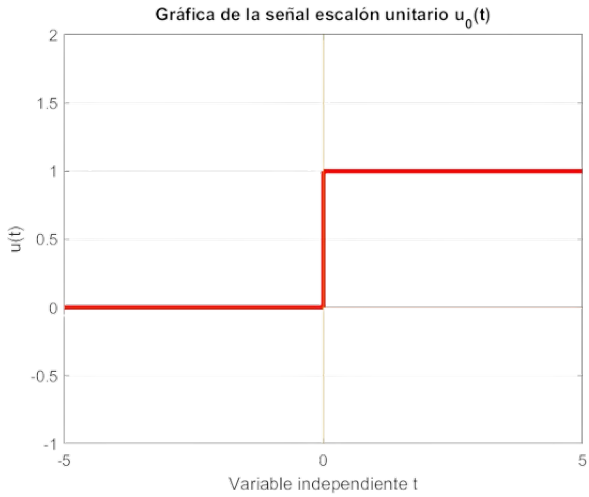


Figura 25: Gráfica de la función escalón unitario.

Código 5.6.

Código

```
%% Suma de escalones unitarios "x(t)=u(t + 1) 2u (t
    -1)+u(t-3)"
t = -2:.001:2;
f = heaviside(t+1)-1*heaviside(t-1)+1*heaviside(t-3)
    ;
plot(t,f,'LineWidt',3,'Color','r');
grid on
title('Combinacion de senales escalon unitario')
xlabel('Variable independiente t')
ylabel('u(t)')
```

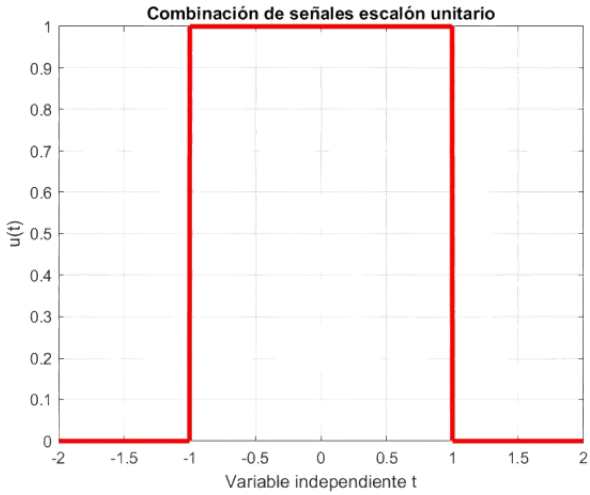


Figura 26: Suma de escalones unitarios del **ejemplo 3.6**.

Ejemplo 3.7 Suma de escalones unitarios

El siguiente **Código 5.7** grafica la suma de diferentes escalones unitarios como se puede observar en la **Figura 27**.

Código

Código 5.7.

```
f = heaviside(t+1)-heaviside(t-1)+2*heaviside(t-3)
    -2*heaviside(t-5)+2*heaviside(t-7)-2*heaviside(t
    -9);
plot(t,f,'LineWidt',3,'Color','r');
grid on
title('Combinacion de se ales escalon unitario')
xlabel('Variable independiente t')
ylabel('u(t)')
```

■

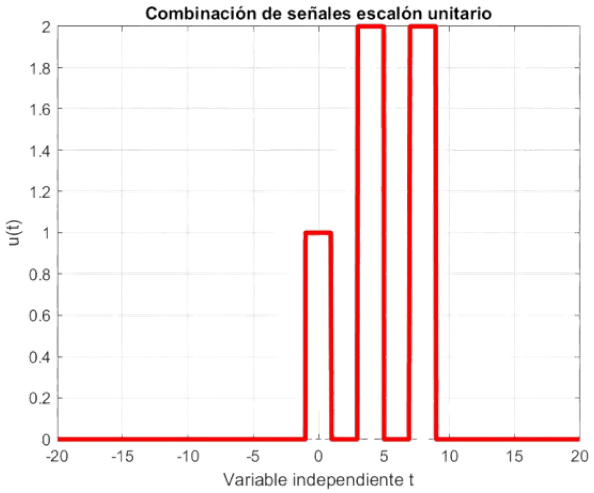


Figura 27: Suma de escalones unitarios del **ejemplo 3.7**.

Ejemplo 3.8 Función rampa

El siguiente **Código 5.8** grafica una señal rampa como se puede observar en la **Figura 28**.

Código

Código 5.8.

```
t = -1:.001:1;
Rampa =t.*heaviside(t);
plot(t,Rampa,'LineWidt',3,'Color','r');
grid on;
title('Grafica de la se al rampa unitaria r(t)')
xlabel('Variable independiente t')
ylabel('r(t)')
```

■

3.1.5. El impulso

Un *impulso unitario* se representa como $\delta(t)$, y es también conocido como una *función delta* o *distribución de Dirac*. Su definición la podemos consultar más a detalle en **(Kamen & Heck, 2008)**, y su representación matemática es como se presenta a continuación:

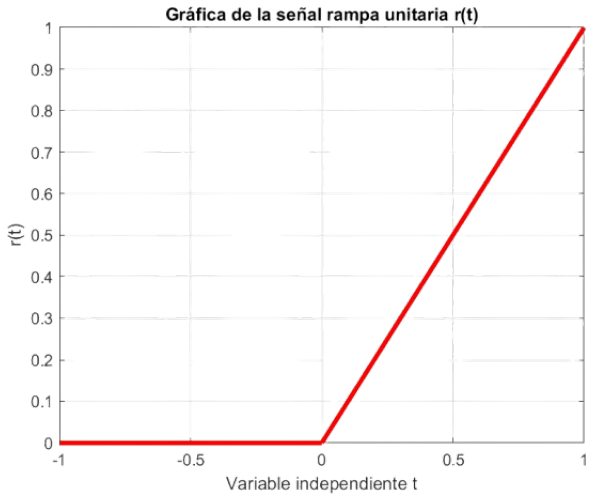
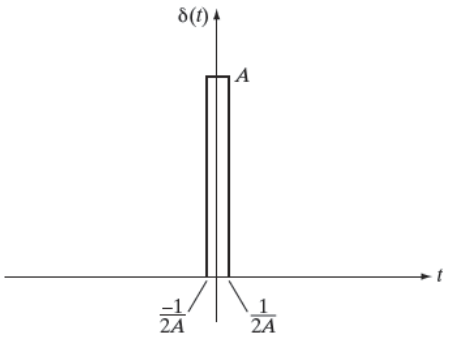


Figura 28: Señal rampa.

$$\delta(t) = 0, \quad t \neq 0$$

$$\int_{-\varepsilon}^{\varepsilon} \delta(\lambda)d\lambda, \quad \text{para cualquier número real} \varepsilon > 0 \tag{4}$$

Una de las principales condiciones establece que $\delta(t) = 0$ para todo $t \neq 0$. También, se establece que el área bajo el impulso es 1, por lo que $\delta(t) = 1$. Cabe señalar que cuando se evalúa el impulso en un $t = 0$ ($\delta(0)$), la función no está definida (en particular, $\delta(0)$ no es igual a infinito). La función impulso unitario la podemos aproximar empleando un pulso centrado en cero, con A de amplitud y duración de $1/A$. Siendo así, A tomaría un valor positivo muy grande. La interpretación de pulso para $\delta(t)$ la presentamos en la **Figura 29**.



Considerando lo anterior, asignaremos valores reales positivos a la variable K, $K\delta(t)$ será el impulso con un área igual a K, y estará definida entonces como:

$$K\delta(t) = 0, \quad t \neq 0$$

Figura 29: Interpretación de pulso para $\delta(t)$.

$$\int_{-\varepsilon}^{\varepsilon} K\delta(\lambda)d\lambda = K, \quad \text{para cualquier número real } \varepsilon > 0$$

En la **Figura 30** se muestra la representación gráfica de $K\delta(t)$, en donde *"(K)"* hace referencia al área del impulso $(K\delta(t))$.

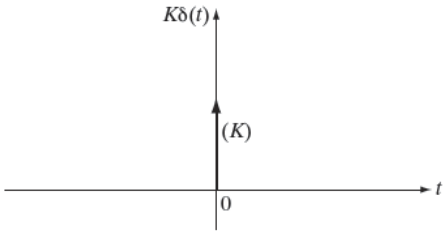


Figura 30: Impulso $K\delta(t)$.

También podemos representar a la función escalón unitario $u(t)$ como la integral del impulso unitario $\delta(t)$. En la **ecuación 5** podemos observar esta interpretación:

$$u(t) = \int_{-\infty}^{t} \delta(\lambda)d\lambda, \quad \text{para toda t, excepto t=0} \tag{5}$$

Para demostrar la **ecuación 5**, debemos analizar primero que para valores menores a cero $(t < 0)$, se cumple que:

$$\int_{-\infty}^{t} \delta(\lambda)d\lambda = 0, \quad \text{debido a que } \delta(t) = 0, \text{para toda } t < 0$$

Considerando valores mayores a cero $(t > 0)$,

$$\int_{-\infty}^{t} \delta(\lambda)d\lambda = \int_{-t}^{t} \delta(\lambda)d\lambda = 1, \quad \text{ya que } \int_{-\varepsilon}^{\varepsilon} \delta(\lambda)d\lambda = 1, \text{para cualquier } \varepsilon > 0$$

Ejemplo 3.9 **Generar la función impulso unitario en Matlab**
En este ejemplo, se presentará un intervalo de 0 a 9, en el cual graficaremos un impulso en la posición 5. El siguiente **Código 5.9** ejemplifica el concepto de función delta, en el cual se desea graficar:

$$\delta_{[n-n_0]} = \begin{cases} 1; & n = n_0 \\ 0; & n \neq n_0 \end{cases}$$

En el intervalo de $n_1 < n_0 < n_2$ (en donde $n_1 = 0$ y $n_2 = 9$), declararemos la función presentada en el siguiente código:

Código 5.9.

Código

```
function [x,n]=impulso(n0,n1,n2) % Genera x[n]=delta(
    n-no); n1<=n<=n2
n=[n1:n2];
x=[(n-n0)==0];
```

Una vez que tenemos definida nuestra función "impulso", la utilizaremos dentro del siguiente **Código 5.10** esto con el fin de implementar una función arbitraria $T = \delta_{[n-5]}$ en el intervalo

antes mencionado, lo que generará la gráfica de la **Figura 31**.

Código 5.10.

Código

```
%% Graficar el impulso en 5
n=[0:9];
T=impulso(5,0,9);
grid on
stem(n,T)
xlabel('n')
ylabel('x[n]')
title('Impulso unitario en la posicion 5')
```

■

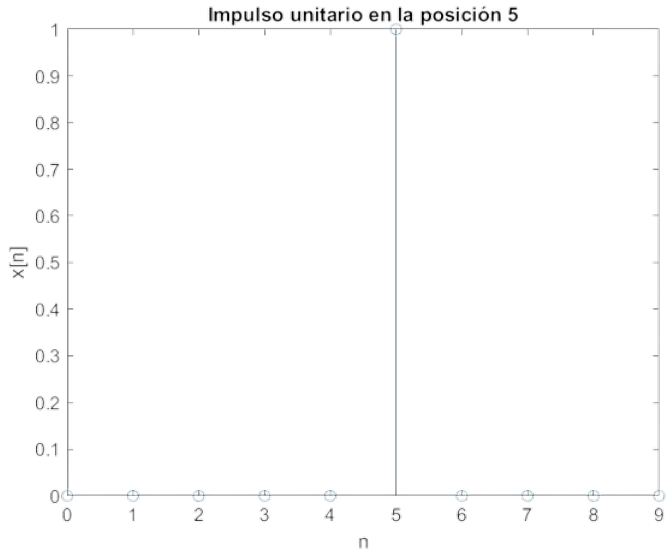


Figura 31: Ejemplo de función delta en la posición 5.

Ejemplo 3.10 Generar dos impulsos unitarios en Matlab

En este ejemplo, se generará y graficará una secuencia sobre un intervalo de -5 a 5 para:

$$x_{[n]} = 2\delta_{[n+2]} - \delta_{[n-4]}; \quad -5 \leq n \leq 5$$

Código 5.11.

Código

```
n=[-5:5];
x=2*impulso(-2,-5,5)-impulso(4,-5,5);
stem(n,x);
xlabel('n')
ylabel('x[n]')
title('Grafica de 2 impulsos unitarios')
```

El código generará la gráfica de la **Figura 32**. ■

3.1.6. Señales periódicas

Para explicar una señal periódica, consideremos que T tomará valores reales positivos fijos. Para que una señal sea considerada continua y periódica con periodo T, $x(t)$ debe cumplir **(Kamen**

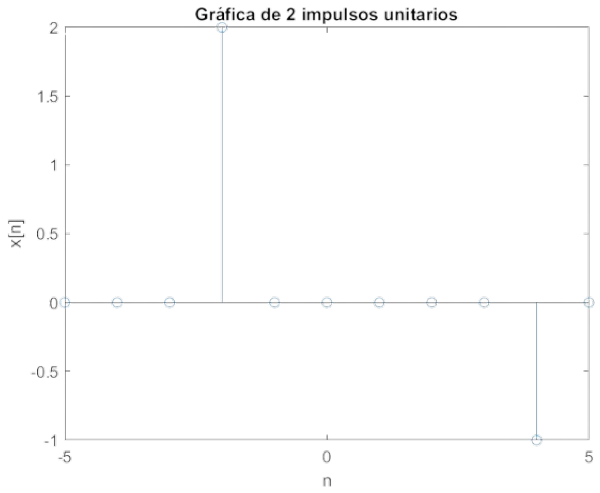


Figura 32: Gráfica de 2 impulsos uinitarios.

& Heck, 2008):

$$x(t+T) = x(t), \quad \text{para toda t, } -\infty < t < \infty \tag{6}$$

Analizando la **ecuación 6** podemos resumir que si $x(t)$ es periódica con un periodo T, también lo será con un periodo qT, donde q tomará cualquier valor entero positivo. De aquí se desprende un nuevo concepto que es el *periodo fundamental* (T_o). Este concepto nos refiere al número positivo más pequeño de T, para el cual la **ecuación 7** se cumple.

$$x(t) = A\cos(\omega t + \theta), -\infty < t < \infty \tag{7}$$

donde

A es la amplitud.
ω es la frecuencia [rad/s].
θ es la fase [rad].

La frecuencia f en hertz [Hz] es:

$$f = \omega/2\pi \tag{8}$$

Las señales periódicas más relevantes bajo estudio son las señales senoidales. Para comprobar que la señal senoidal dada por la **ecuación 7** es periódica, analizamos que para cualquier valor de la variable de tiempo t se cumple lo siguiente:

$$A\cos\left[\omega\left(t+\frac{2\pi}{\omega}\right)\right] = A\cos\left(\omega t + 2\pi + \theta\right) = A\cos\left(\omega t + \theta\right)$$

Con esto, comprobamos que la señal senoidal es periódica con periodo $T = 2\pi/\omega$ y, también, $2\pi/\omega$ es el *periodo fundamental* T_o. La señal senoidal $A\cos(\omega t + \theta)$ se observa en la **Figura 33**,

en este caso θ toma valores entre $-\pi/2$ y 0 ($-\pi/2 < \theta < 0$).

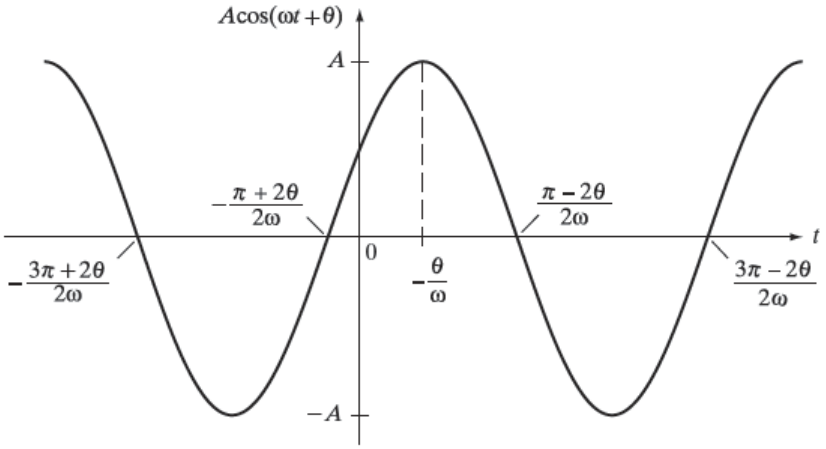


Figura 33: Sinusoide $x(t) = A\cos(\omega t + \theta)$ con $-\pi/2 < \theta < 0$.

Si ahora consideramos que $\theta = -\pi/2$, entonces observamos que nuestra señal coseno defasado $-\pi/2$ se convierte en una señal seno con argumento ωt ($\sin(\omega t)$) como se observa en la siguiente expresión:

$$x(t) = A\cos(\omega t + \theta) = A\sin(\omega t)$$

Cuando trabajamos con señales periódicas en ocasiones se requiere trabajar con la suma de estas, y surge la incógnita de saber si la suma de dos señales periódicas resultará también en una señal periódica. Entonces, consideremos ahora dos señales periódicas definidas como $x_1(t)$ y $x_2(t)$, con sus respectivos *periodos fundamentales* T_1 y T_2. La pregunta problema a resolver:

¿Es periódica la suma $x_1(t) + x_2(t)$?;

es decir,

¿Existe un número positivo T, tal que satisfaga lo siguiente?

$$x_1(t+T) + x_2(t+T) = x_1(t) + x_2(t) \quad \text{para toda t?} \tag{9}$$

Concluimos que la **ecuación 9** se satisface si y sólo si la relación entre los periodos fundamentaes T_1/T_2 se logra escribir también como una relación de dos números enteros, por ejemplo q y r (q/r). Una vez explicado lo anterior, podemos observar que si se cumple que $T_1/T_2 = q/r$, relacionamos estos conceptos como $rT_1 = qT_2$, y como ya se mencionó que tanto r como q son números enteros, concluimos que la señal $x_1(t)$ y la señal $x_2(t)$ son también señales periódicas con periodo rT_1. Es por eso que la expresión de la **ecuación 9** se sigue satisfaciendo con $T = rT_1$.

Adicional, si los valores que toman r y q son coprimos, es decir, r y q no tienen factores enteros comunes diferentes de 1, entonces $T = rT_1$ es el *periodo fundamental* de la suma de ambas señales $x_1(t) + x_2(t)$.

Ejemplo 3.11 Suma de señales periódicas

Considerando la señal periódica $x_1(t) = \cos(\pi t/2)$ y la señal periódica $x_2(t) = \cos(\pi t/3)$, expresamos que las señales son periódicas con periodos fundamentales de $T_1 = 4\ s$ y $T_2 = 6\ s$, respectivamente, como se observa a continuación:

$$\frac{T_1}{T_2} = \frac{4}{6} = \frac{2}{3}$$

Resultando en que si $q = 2$ y $r = 3$, se concluye que la suma entre ambas señales $(x_1(t) + x_2(t))$ es periódica, con un *periodo fundamental* de $rT_1 = (3)(4) = 12$ segundos. El siguiente **Código 5.12** presenta la sintaxis del procedimiento antes explicado y este genera la gráfica que se presenta en la **Figura 34**.

Código

Código 5.12

```
T=12;
t=0:0.001:T;
x1=cos((pi*t)/(2));
x2=cos((pi*t)/(3));
y=x1+x2;
plot(t,y,'r')
xlabel('t','FontName','Times','Fontsize',14);
ylabel('Amplitud','FontName','Times','Fontsize',
    14);
grid on;
hold on;
title('Suma de dos senales periodicas','FontName',
    'Times','Fontsize',16);
```

■

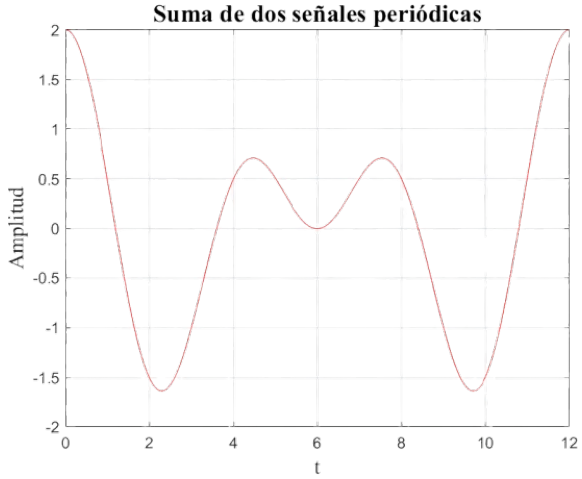


Figura 34: Suma de dos señales periódicas.

Nota

El *periodo fundamental* T_0 de una señal $x(t)$ es el valor positivo más pequeño de T. Una

señal $x(t)$ que no cumple con ser periódica se define como una señal *aperiódica*.

Ejemplo 3.12 Señal periódica

Consideremos la siguiente señal periódica $x(t)$:

$$x(t) = \cos\left(\frac{t}{3}\right)$$

Como primer paso calcularemos el *periodo fundamental* de la señal $x(t)$, seguido graficaremos la señal con periódica con duración de 4 ciclos. El siguiente **Código 5.13** presenta la sintaxis para calcular el *periodo fundamental*, y con este dato grafica la señal en 4 ciclos como se observa en la **Figura 35**.

Código 5.13

Código

```
T=6*pi;
m0=1;
t=0:0.1:m0*T;
y=cos(t/3);
plot(t,y,'r', 'DisplayName','Senal original')
grid on;
hold on;
m1=2;t1=m0*T:0.1:m1*T;y1=cos(t1/3);
plot(t1,y1,'b', 'DisplayName','Senal 2do periodo')
m2=3;t2=m1*T:0.1:m2*T;y2=cos(t2/3);
plot(t2,y2,'g', 'DisplayName','Senal 3er periodo')
m3=4;t3=m2*T:0.1:m3*T;y3=cos(t3/3);
plot(t3,y3,'k', 'DisplayName','Senal 4to periodo')
xlabel ('t', 'FontName', 'Times', 'Fontsize', 14);
ylabel ('Amplitud', 'FontName', 'Times', 'Fontsize',
    14);
title ('Senal 4*T', 'FontName', 'Times', 'Fontsize',
    16);
```

■

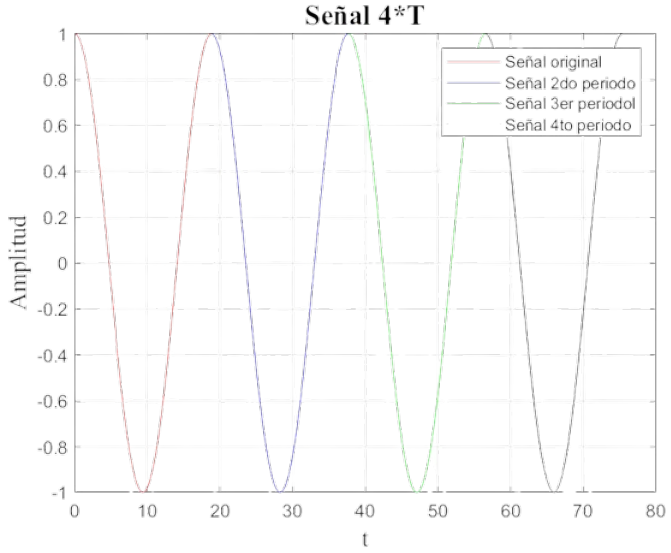


Figura 35: Ejemplo de una señal periódica.

3.1.7. Señales pares e impares

Una señal periódica puede clasificarse como una señal *par* o *impar*. Esta característica está relacionada directamente con la simetría que presentan las señales en función con la inversión de tiempo t.

Señales par. ■

Una señal periódica $x(t)$ se clasifica como una *señal par* si ésta es idéntica a su contraparte invertida en el tiempo t, es decir, de forma visual debe cumplir que su reflejo sea respecto del origen **(Oppenheim et al., 1998)**. Graficamente podemos observar esta definición en la **Figura 36**

$$x(t) = x(-t)$$

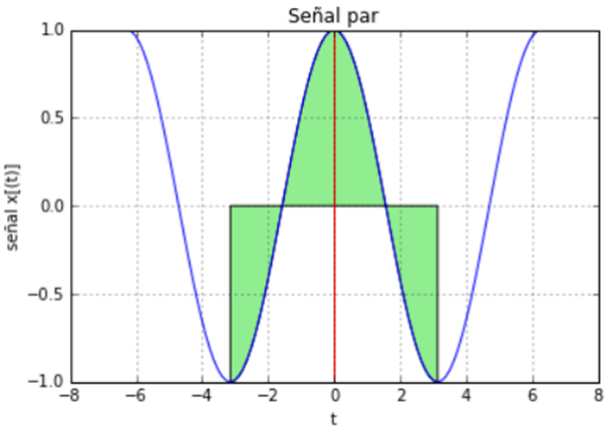


Figura 36: Señal par.

Señales impar. ■

Una señal periódica $x(t)$ se clasifica como una *señal impar* si al ser evaluada en $t = 0$ el valor de la función también es 0, también debe cumplir la condición $x(t) = -x(-t)$ para todos los valores de t **(Oppenheim et al., 1998)**. Graficamente podemos observar esta definición en la **Figura 37**

$$x(t) = -x(-t)$$

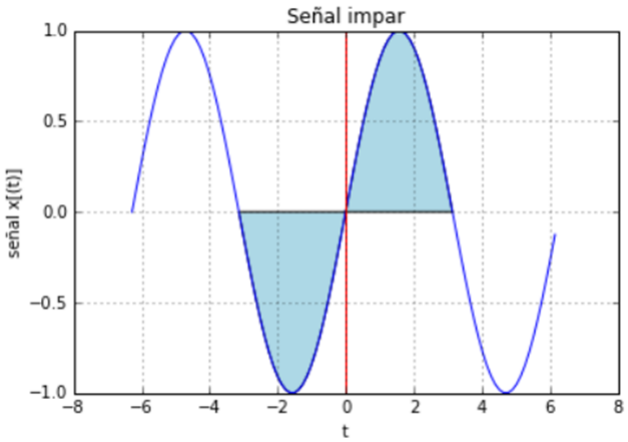


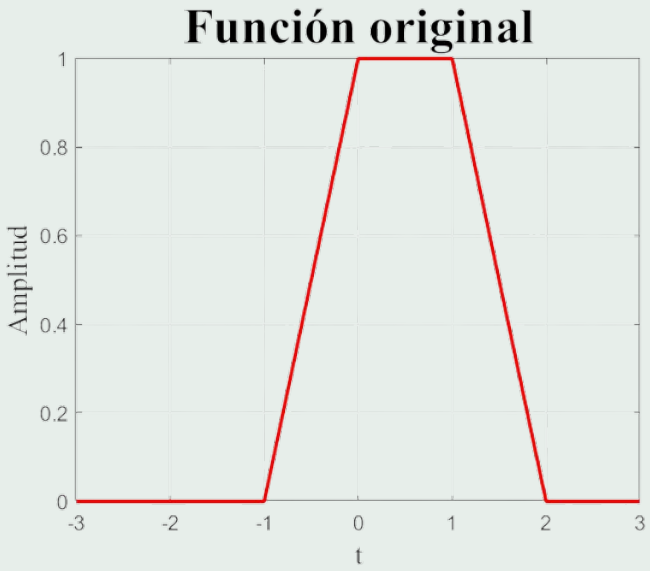
Figura 37: Señal impar.

Algo importante que se debe mencionar es que *cualquier señal* se puede descomponer en la suma de dos señales, una de estas señales será la que corresponde a una *señal par* y la otra la que corresponde a una *señal impar*.

Parte PAR
$$x(t)_{PAR} = \frac{1}{2}[x(t) + x(-t)]$$

Parte IMPAR
$$x(t)_{IMPAR} = \frac{1}{2}[x(t) - x(-t)]$$

Ejemplo 3.13 Señales pares e impares
Consideremos una señal $x(t)$, de la cual se debe obtener su parte par y su parte impar. Posterior se debe realizar la comprobación mediante un código de que la suma de $x(t)_{PAR}$ y $x(t)_{IMPAR}$ resulta la señal original.



La señal $x(t)$ se genera con el siguiente **Código 5.14**:

Código

Código 5.14.
```
function x = PAR(t)
n = length(t);
x = zeros(1,n);
i = find(t < -3);
x(i) = 0; %1
i = find(t > -3);
x(i) = 0;% 2
i = find(t > -1);
x(i) = t(i)+1 ; % 3
i = find(t > 0 );
x(i) = 1 ; % 4
i = find(t > 1 );
x(i) = (-t(i)+2) ; % 5
i = find(t > 2 );
x(i) = 0; % 6
return;
```

Una vez que se genera la señal, se utiliza el siguiente **Código 5.15** para contestar el ejemplo.

Código 5.15.

Código

```
%% Senales Pares e Impares
t=-3:0.1:3;
x=PAR(t);
subplot(2,3,1);
plot(t,x,'r', 'LineWidth', 2);
xlabel ('t', 'FontName', 'Times', 'Fontsize', 14);
ylabel ('Amplitud', 'FontName', 'Times', 'Fontsize',
    14);
grid on;
hold on;
title ('Func. original', 'FontName', 'Times', '
    Fontsize', 24);
xp1=0.5*PAR(t);
subplot(2,3,2);
plot(t,xp1,'c','LineWidth', 1.5);
xlabel ('t', 'FontName', 'Times', 'Fontsize', 14);
ylabel ('Amplitud', 'FontName', 'Times', 'Fontsize',
    14);
grid on;
hold on;
title ('Senal 0.5x(t)', 'FontName', 'Times', '
    Fontsize', 24);
tt=-t;
xp2=0.5*PAR(tt);
subplot(2,3,3);
plot(t,xp2,'G','LineWidth', 1.5);
xlabel ('t', 'FontName', 'Times', 'Fontsize', 14);
ylabel ('Amplitud', 'FontName', 'Times', 'Fontsize',
    14);
grid on;
hold on;
title ('Senal 0.5x(-t)', 'FontName', 'Times', '
    Fontsize', 24);
xp=xp1+xp2;
subplot(2,3,4);
plot(t,xp,'b','LineWidth',4);
xlabel ('t', 'FontName', 'Times', 'Fontsize', 14);
ylabel ('Amplitud', 'FontName', 'Times', 'Fontsize',
    14);
grid on;
hold on;
title ('Senal Par', 'FontName', 'Times', 'Fontsize',
    24);
xi=xp1-xp2;
subplot(2,3,5);
plot(t,xi,'m','LineWidth',4);
xlabel ('t', 'FontName', 'Times', 'Fontsize', 14);
ylabel ('Amplitud', 'FontName', 'Times', 'Fontsize',
    14);
grid on;
hold on;
title ('Senal Impar', 'FontName', 'Times', 'Fontsize'
    , 24);
%% Senal original
X_original=xp+xi;
subplot(2,3,6);
plot(t,X_original,'k','LineWidth',4);
xlabel ('t', 'FontName', 'Times', 'Fontsize', 14);
ylabel ('Amplitud', 'FontName', 'Times', 'Fontsize',
    14);
grid on;
hold on;
title ('Func. original', 'FontName', 'Times', '
    Fontsize', 24);
```

Las gráficas que se desprenden de correr el código anterior son las presentadas en la **Figura 38**. ■

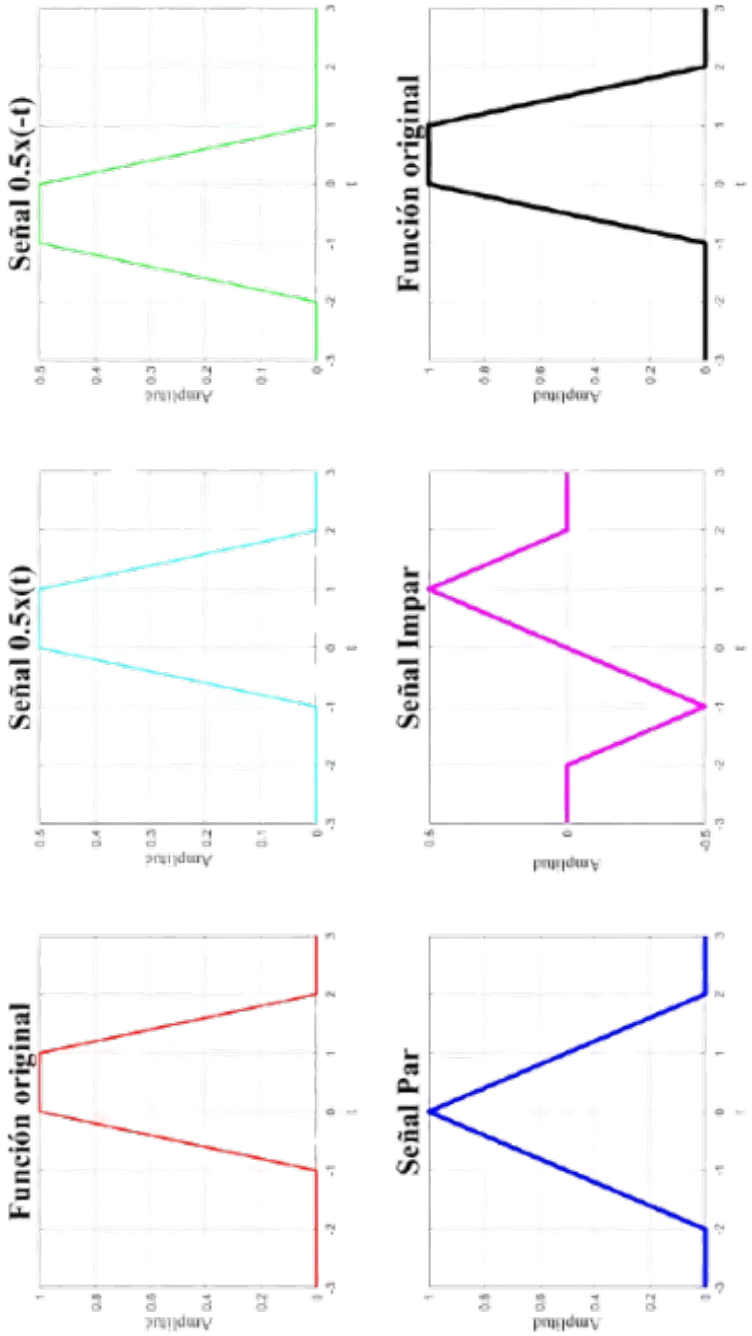


Figura 38: Parte par y parte impar de la señal $x(t)$, así como la comprobación de que la suma de $x(t)_{PAR}$ y $x(t)_{IMPAR}$ resulta la señal original.

3.2. Teorema de Muestreo

Uno de los procesos de digitalización de una señal es el muestreo, el *teorema de muestreo* de una señal establece que, una señal analógica que está limitada en su banda con una frecuencia fundamental f_0, puede ser muestreada (discretizar la señal en tiempo) sin que exista pérdida de su información siempre y cuando la frecuencia para muestrear f_s sea superior que la relación de $2f_0$. Entonces, estas muestras determinan únicamente a la señal, y esta puede ser reconstruida sin distorsión y sin error a partir de dichas muestras, **(Proakis, Salehi, Zhou, & Li, 1994)**.

Ejemplo 3.14 Diseño de una función de MATLAB que dibuje la señal analógica, el tren de impulso y la señal muestreada.

Existen varios tipos de muestreo, el más común es el *muestreo ideal*, el cual consiste en multiplicar la señal analógica por un tren de impulsos de amplitud unitaria, y período $ts = 1/f_s$, es decir, tiempo de muestreo. Al realizar esta multiplicación obtenemos a la salida solo muestras de nuestra señal en cada t_s segundos, este proceso se muestra en la **Figura 39**.

Diseñar una función de MATLAB que dibuje la señal analógica, el tren de impulso y la señal muestreada, cumpliendo el teorema de muestreo.

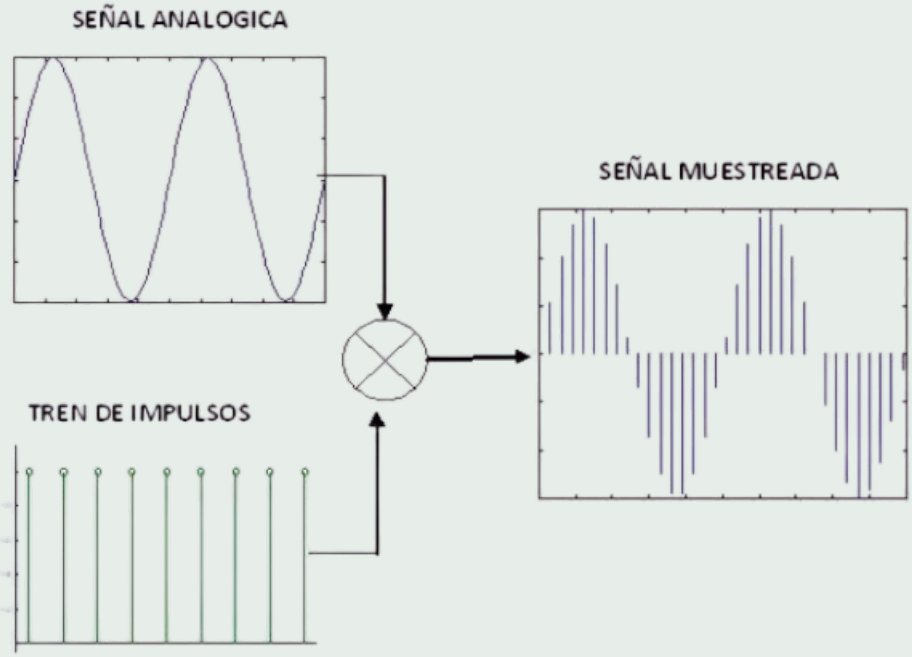


Figura 39: Muestreo Ideal.

Código 5.16.

Código

```
fo=2;
fs=30;
A=3;
t=0:0.01:1;
y=A*sin(2*pi*fo*t);
plot(t,y)
xlabel('t')
ylabel('f(t)')
title('Senal Analogica')
grid on
figure(2)
for ts=0:(1/fs):1
aux1=[ts ts];
aux2=[0 1];
plot(aux1,aux2,'k')
hold on
end
xlabel('ts')
ylabel('Amplitud')
title('Tren de Impulsos')
grid on
axis([-0.2 1.2 -0.2 1.2])
figure(3)
for ts=0:(1/fs):1
aux3=[ts ts];
f=A*sin(2*pi*fo*ts);
aux4=[0 f];
plot(aux3,aux4,'k')
hold on
xlabel('ts')
ylabel('f(ts)')
title('Senal Muestreada')
grid on
xlabel('ts')
ylabel('f(ts)')
title('Señal Muestreada')
grid on
```

El **código 5.16** produce las gráficas de la **Figura 40**, **Figura 41** y **Figura 42**.

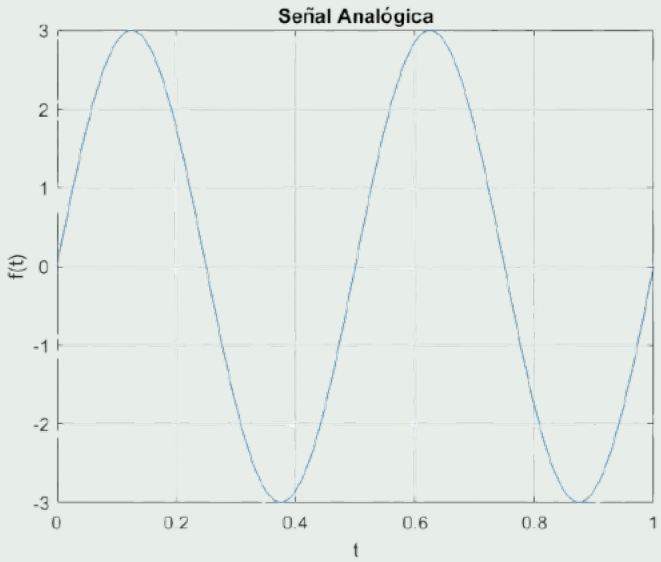


Figura 40: Señal analógica.

Se observa en el código que la señal muestreada solo tiene valores cada t_s segundos, también se observa que se utilizan variables auxiliares nombradas como $aux1$, $aux2$, $aux3$ y $aux4$, las cuales se utilizan para la gráfica de los impulsos, esto se puede omitir utilizando el comando `stem` que nos sirve para la gráfica de secuencias discretas, dado que el muestreo es la discretización de la señal en el tiempo. Esto se muestra en el **código 5.17**, y el cual produce las gráficas de la **Figura 43**, **44** y **45**.

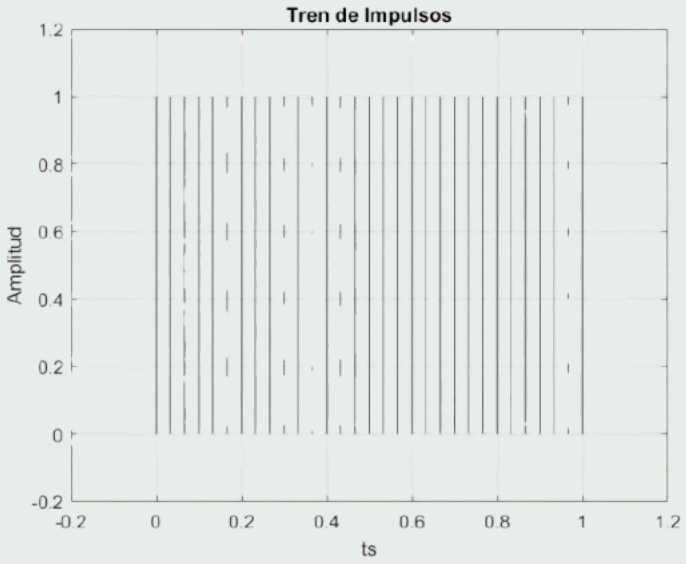


Figura 41: Tren de impulsos.

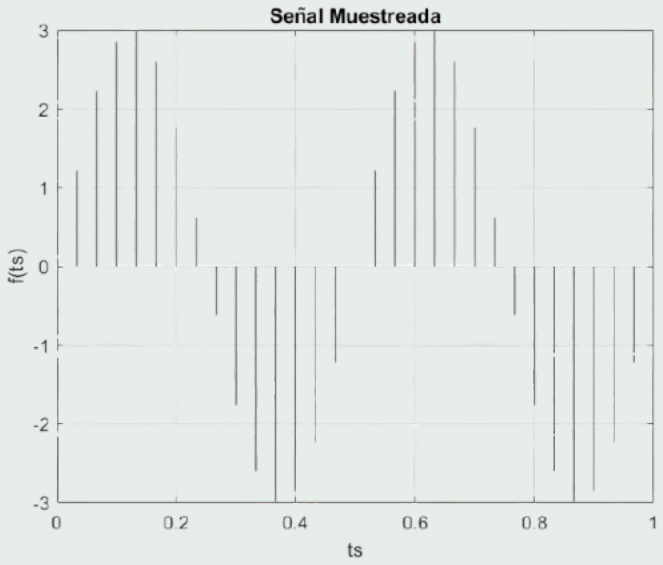


Figura 42: Señal muestreada.

Código 5.17

Código

```
fo=2;
fs=30;
A=3;
t=0:0.01:1;
y=A*sin(2*pi*fo*t);
plot(t,y)
xlabel('t')
ylabel('f(t)')
title('Senal Analogica')
grid on
figure(2)
ts=0:(1/fs):1;
n=length(ts);
tren=ones(n);
stem(ts,tren,'k')
xlabel('ts')
ylabel('Amplitud')
title('Tren de Impulsos')
grid on
axis([-0.2 1.2 -0.2 1.2])
figure(3)
ts=0:(1/fs):1;
f=A*sin(2*pi*fo*ts);
stem(ts,f,'k')
xlabel('ts')
ylabel('f(ts)')
title('Senal Muestreada')
grid on
```

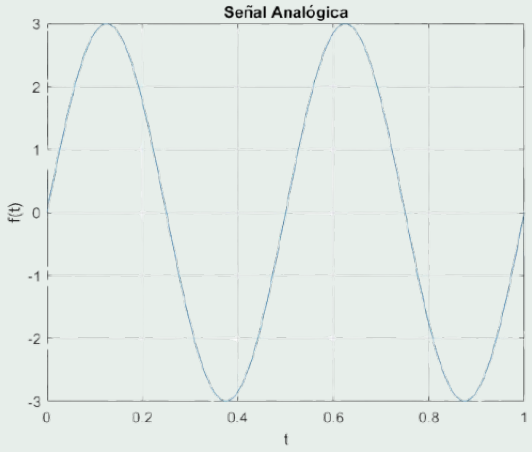


Figura 43: Señal Analógica.

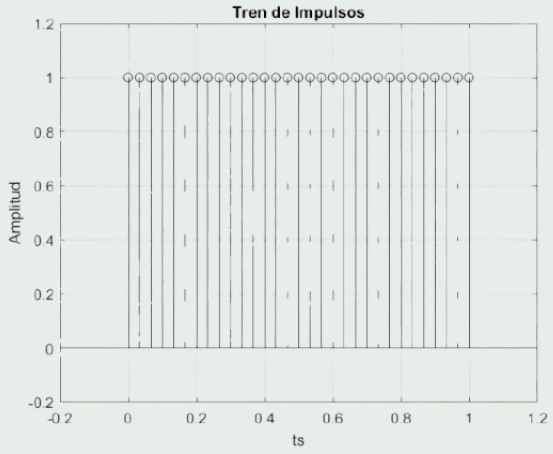


Figura 44: Tren de Impulsos.

Figura 45: Señal Muestreada.

3.3. Cuantización

Una vez obtenida la señal muestreada, el siguiente proceso para digitalizar la señal es hacer que la señal tome valores de amplitudes de un conjunto finito de valores, a este proceso se le conoce como cuantización **(Haykin, 2002.)**.

Ejemplo 3.15 Escribir un código que dibuje una señal cuantificada

Para este ejemplo, podemos utilizar la señal muestreada obtenida anteriormente, y utilizar el comando `stairs` para unir los puntos. El **código 5.18** produce las gráfica de la **Figura 46**.

Código 5.18.

Código

```
fo=2; % Frecuencia de la se al
fs=30; % Frecuencia de muestreo
A=3; % Amplitud de la se al
ts=0:(1/fs):1; % Tiempo de Muestreo
f=A*sin(2*pi*fo*ts);
stairs(ts,f)
xlabel('ts')
ylabel('f(ts)')
title('Se al Cuantificada')
grid on
```

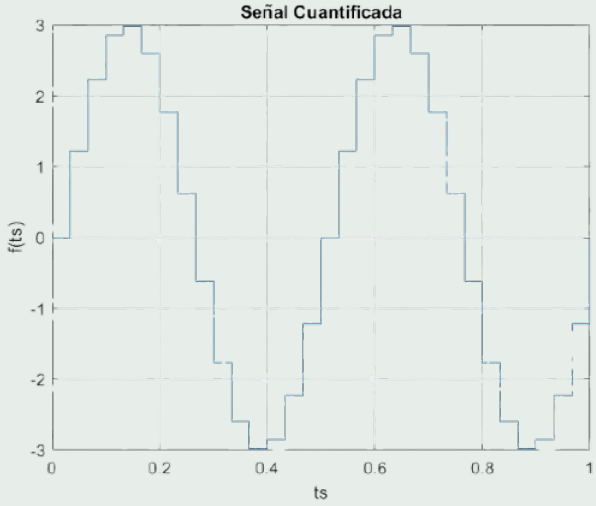


Figura 46: Cuantización.

■

3.4. Conversión Analógica a Digital (PCM)

El **código 5.19** convierte una señal analógica de N componentes a una señal digital de B bits, es decir, muestrea, cuantiza (con L niveles de cuantización, con $L = 2^B$) y codifica la señal para producir una *señal PCM*. Al ingresar los parámetros de la **Figura 47**, el programa genera las gráficas de la **Figura 48** y **49**, así como los resultados presentados en la **Figura 50**.

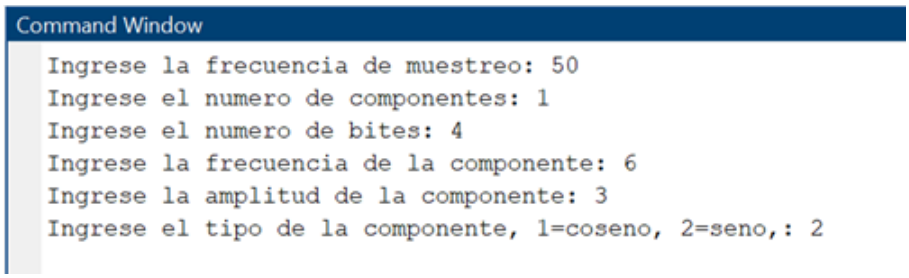


Figura 47: Parámetros .

Código 5.19

Código

```
fs=input('Ingrese la frecuencia de muestreo: ');
ts=1/fs;
N=input('Ingrese el numero de componentes: ');
t=ts/10:ts/10:1;
suma=0;
B=input('Ingrese el numero de bits de cada nivel: ');
for k=1:N
    f=input('Ingrese la frecuencia de la componente: ')
       ;
    A=input('Ingrese la amplitud de la componente: ');
    ti=input('Ingrese el tipo de la componente, 1=
       coseno, 2=seno,: ');
    if ti==1
        y=A*cos(2*pi*f*t);
    else
        y=A*sin(2*pi*f*t);
    end
    suma=suma+y;
end
plot(t,suma,'k')
title('Senal Analogica')
maxsum=max(suma);
minsum=min(suma);
D=maxsum+(-1*(minsum));
L=2^B;
Delta=D/L;
Deltamed=[Delta/2];
Cuantificacion=[0];
Valor=[0];
b=1;
a=0;
figure(2)
for k2=ts/10:ts/10:N
    if mod(k2,ts)==0
        z(b)=suma(b);
        a=a+1;
        xy(a)=z(b);
    else
        z(b)=0;
    end
    aux1=[k2,k2];
    aux2=[0,z(b)];
    plot(aux1,aux2,'r')
    hold on
    b=b+1;
end
```

Código

```
%% Cont...
cuant=[0];
aux=maxsum/(L/2);
for c=1:L
    if c==1
        cuant(c)=maxsum-(aux/2);
    else
        cuant(c)=cuant(c-1)-aux;
    end
end
k3=L-1;
    for k4=1:L/2
        for k5=1:length(xy)
            ls=cuant(k4)+aux/2;
            li=cuant(k4)-aux/2;
            if xy(k5)<=ls && xy(k5)>=li
                Cuantificacion(k5,:)=[k3];
            end
        end
        k3=k3-1;
    end
k6=0;
    for k7=L/2+1:L
        for k8=1:length(xy)
            ls=cuant(k7)+aux/2;
            li=cuant(k7)-aux/2;
            if xy(k8)<=ls && xy(k8)>=li
                Cuantificacion(k8,:)=[k6];
            end
        end
        k6=k6+1;
    end
Muestreo=transpose(xy);
Niveles=transpose(Cuantificacion);
Codificacion=dec2bin(Niveles,B);
T = table(Muestreo,Cuantificacion,Codificacion)
```

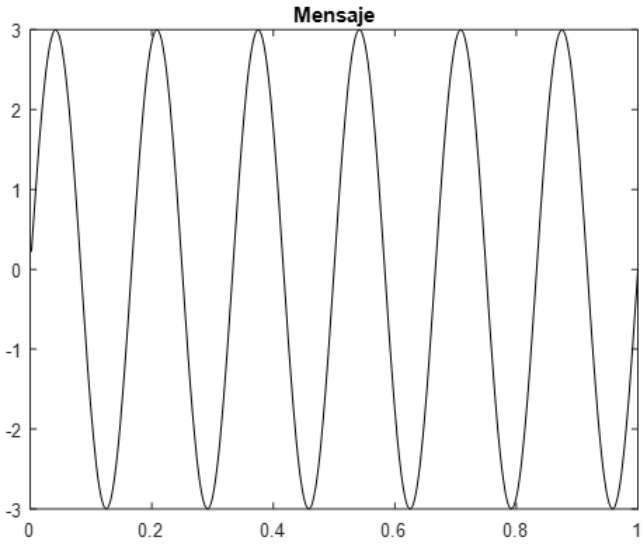


Figura 48: Señal analógica.

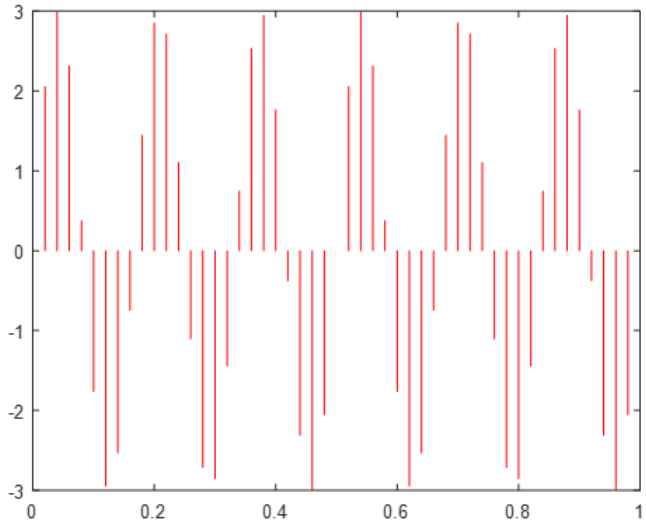


Figura 49: Señal muestreada.

Muestreo	Cuantificacion	Codificacion
2.0536	13	1101
2.9941	15	1111
2.3115	14	1110
0.376	9	1001
-1.7634	4	0100
-2.9469	7	0111
-2.533	6	0110
-0.74607	1	0001
1.4453	11	1011
2.8532	15	1111
2.7145	15	1111
1.1044	10	1010
-1.1044	2	0010
-2.7145	7	0111
-2.8532	7	0111
-1.4453	3	0011
0.74607	9	1001

Figura 50: Tabla de resultados.

3.5. Modulación AM y FM

La modulación es el proceso de modificar una o más propiedades de una señal llamada portadora, de mayor frecuencia que la señal de mensaje, para transmitir información. Básicamente, se trata de adaptar la señal de mensaje a la señal portadora, logrando una transmisión eficiente a través de un medio. Dos de las modulación analógicas mas utilizadas son:

Modulación por Amplitud (AM): La amplitud de la señal portadora cambia dependiendo de la señal de mensaje.

Modulación por Frecuencia (FM): La frecuencia de la señal portadora cambia según la señal de mensaje.

El siguiente **código 5.20** calcula y dibuja las señales moduladas en amplitud y en frecuencia, así como produce las gráficas de la **Figura 51**, **52** y **53**.

Código 5.20

Código

```
fo=2; A=3; t=0:0.005:1; f=A*sin(2*pi*fo*t);
am=ammod(f,fo,4); fm=fmmod(f,fo,50,1);
plot(t,f) grid on;
xlabel('x') ylabel('y')
title('Senal Analogica')
figure(2)
plot(t,am)
xlabel('x') ylabel('y') title('Senal AM')
figure(3)
plot(t,fm)
xlabel('x') ylabel('y') title('Senal FM')
```

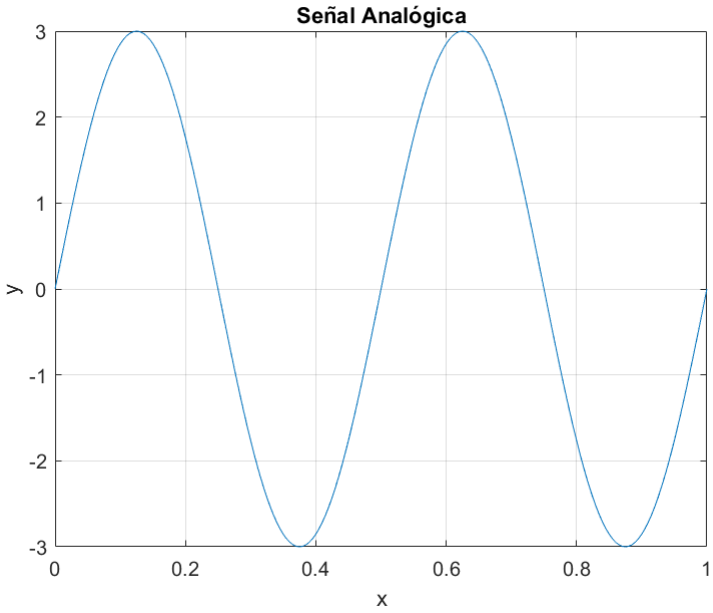


Figura 51: Señal Analógica.

Se utiliza el comando `ammod`, el cual tiene como parámetros de entrada la señal, la frecuencia (f) de la señal y la frecuencia de la portadora. Si se desea saber más sobre este comando, en la ventana de comandos escriba `help ammod`, de la misma manera para `fmmod`.

3.6. Densidad Espectral de Potencia

El siguiente **código 5.21** muestra la densidad espectral de potencia para una señal modulada en amplitud (AM), haciendo uso de la transformada de Fourier. Se consideran varios esquemas de la modulación en amplitud:

- Modulación de Amplitud de Doble Banda Lateral con portadora Suprimida (AM-DSB-SC)
- Modulación de Amplitud de Doble Banda Lateral con portadora (AM-DSB-C)
- Modulación de Amplitud de Banda Lateral Única (AM-SSB)

Código 5.21

Código

```
Am=input('Ingrese la amplitud del mensaje: ');
fm=input('Ingrese la frecuencia del mensaje: ');
Ac=input('Ingrese la amplitud de la portadora: ');
fc=input('Ingrese la frecuencia de la portadora:');
fs=input('Ingrese la frecuencia de muestreo: ');
m=input('Ingrese el indice de modulación: ');
t=0:1/fs:2;
x=Am*cos(2*pi*t*fm);
y=Ac*cos(2*pi*t*fc);
z=Ac*(1+m*x).*cos(2*pi*t*fc);
subplot(4,1,1)
plot(t,x)
grid on;
xlabel ('Tiempo [s]'); ylabel ('Amplitud'); title 'Mensaje
    ';
subplot(4,1,2)
plot(t,y)
grid on;
xlabel ('Tiempo [s]');
ylabel ('Amplitud');
title ('Portadora');
subplot(4,1,3)
plot(t,z)
grid on;
xlabel ('Tiempo [s]'); ylabel ('Amplitud');
title ('Modulacion AM de doble banda lateral con portadora
    ');
N = 10*length(t);
fnorm = fs.*(-N/2:(N/2)-1)/N;
subplot(4,1,4)
t_f=fft(z,N);
plot(fnorm,abs(t_f))
grid on;
xlabel ('Frecuencia [Hz]'); ylabel ('Amplitud');
title ('Espectro AM doble banda lateral con portadora');
w=amdemod(z,fc,fs);
z2=x.*y;
figure(2)
subplot(4,1,1)
plot(t,x)
grid on;
xlabel ('Tiempo [s]');
ylabel ('Amplitud');
title 'Mensaje';
subplot(4,1,2)
plot(t,y)
grid on;
xlabel ('Tiempo [s]');
ylabel ('Amplitud');
title 'Portadora';
subplot(4,1,3)
plot(t,z2)
grid on;
xlabel ('Tiempo [s]');
ylabel ('Amplitud');
title ('Modulacion AM de doble banda lateral con portadora
     suprimida');
```

Código

```
%% Cont...
subplot(4,1,4)
t_f=fft(z,N);
plot(fnorm,abs(t_f))
grid on;
xlabel ('Frecuencia [Hz]'); ylabel ('Amplitud');
title ('Espectro AM doble banda lateral con portadora');
w=amdemod(z,fc,fs);
z2=x.*y;
figure(2)
subplot(4,1,1)
plot(t,x)
grid on;
xlabel ('Tiempo [s]');
ylabel ('Amplitud');
title 'Mensaje';
subplot(4,1,2)
plot(t,y)
grid on;
xlabel ('Tiempo [s]');
ylabel ('Amplitud');
title 'Portadora';
subplot(4,1,3)
plot(t,z2)
grid on;
xlabel ('Tiempo [s]');
ylabel ('Amplitud');
title ('Modulacion AM de doble banda lateral con
   portadora suprimida');
subplot(4,1,4)
t_f2=fft(z2,N);
plot(fnorm,abs(t_f2))
grid on;
xlabel ('Frecuencia [Hz]');
ylabel ('Amplitud');
title ('Espectro AM doble banda lateral con portadora
   suprimida');
w2=amdemod(z2,fc,fs);
figure(3)
y3 = ssbmod(x,fc,fs);
x3 = ssbdemod(y3,fc,fs);
subplot(2,1,1);
plot(t,y3);
grid on;
xlabel ('Tiempo [s]');
ylabel ('Amplitud');
title ('Modulacion AM de banda lateral unica');
grid on;
xlabel ('Tiempo [s]');
ylabel ('Amplitud');
title ('Modulacion AM de banda lateral unica');
subplot(2,1,2)
t_f3=fft(y3,N);
plot(fnorm,abs(t_f3))
grid on;
xlabel ('Frecuencia [Hz]');
ylabel ('Amplitud');
title ('Espectro AM banda lateral unica');
```

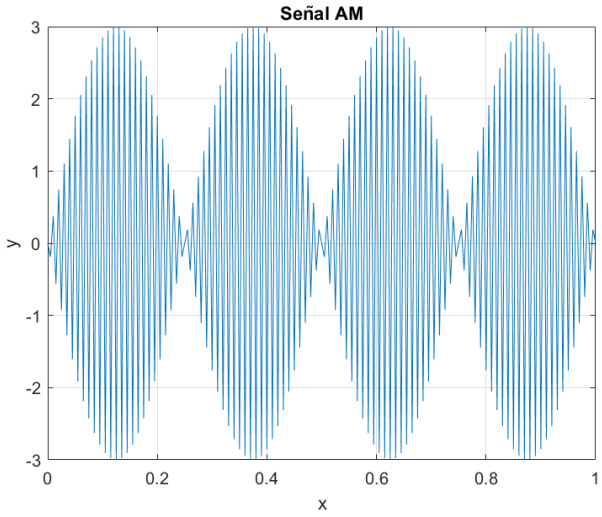


Figura 52: Modulación AM.

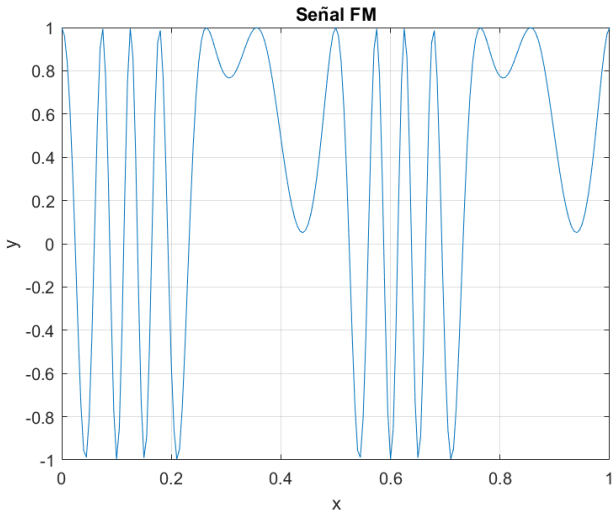


Figura 53: Modulación FM.

Utilizando los valores de entrada que se muestran en la **Figura 54**, el código produce las gráficas de la **Figura 55-57**.

```
Command Window
  Ingrese la amplitud del mensaje: 5
  Ingrese la frecuencia del mensaje: 8
  Ingrese la amplitud de la portadora: 8
  Ingrese la frecuencia de la portadora:70
  Ingrese la frecuencia de muestreo: 1000
  Ingrese el indice de modulación: 0.8
```

Figura 54: Valores de entrada del **código 5.21**.

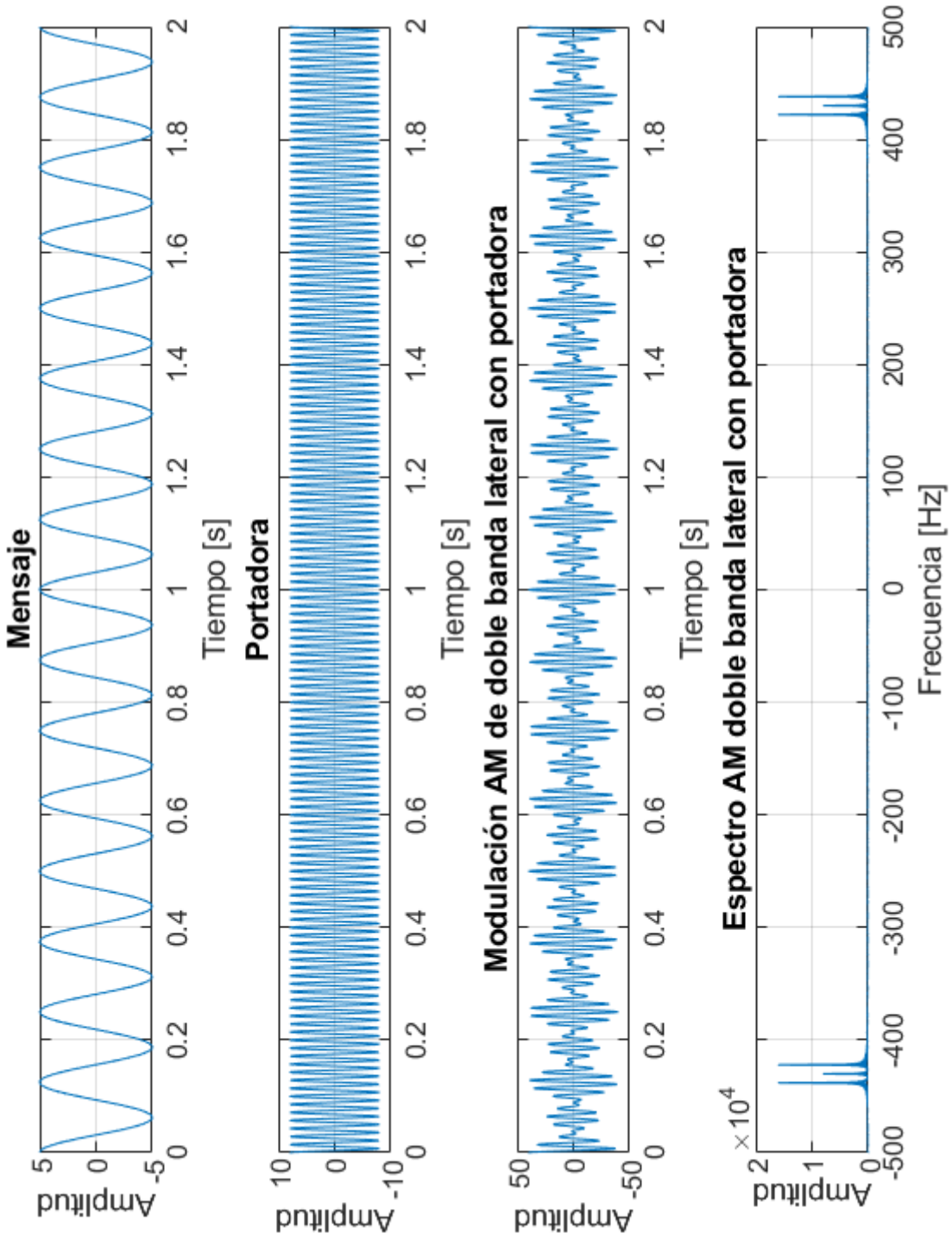


Figura 55: Modulación de Amplitud de Doble Banda Lateral con portadora.

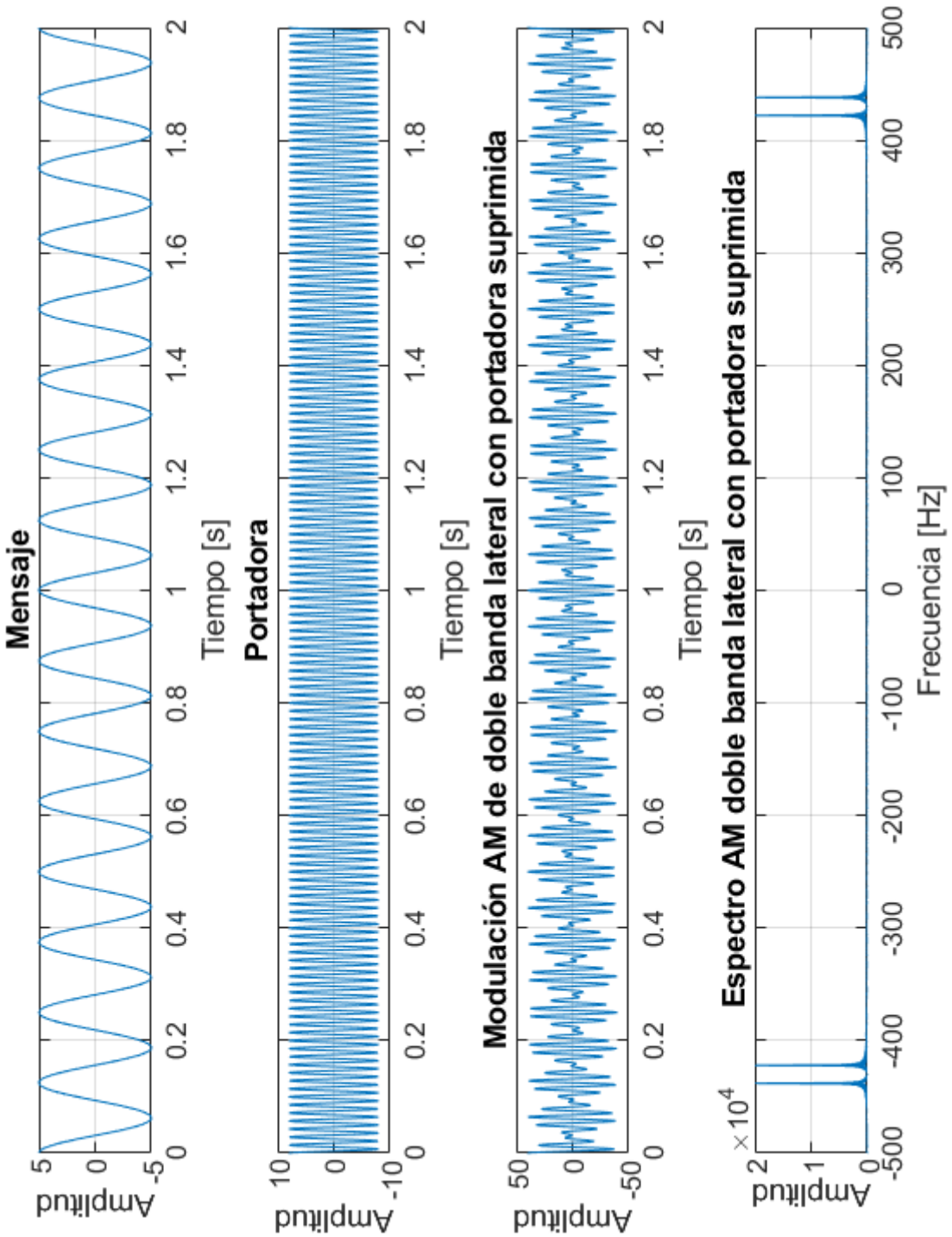


Figura 56: Modulación de Amplitud de Doble Banda Lateral con portadora Suprimida.

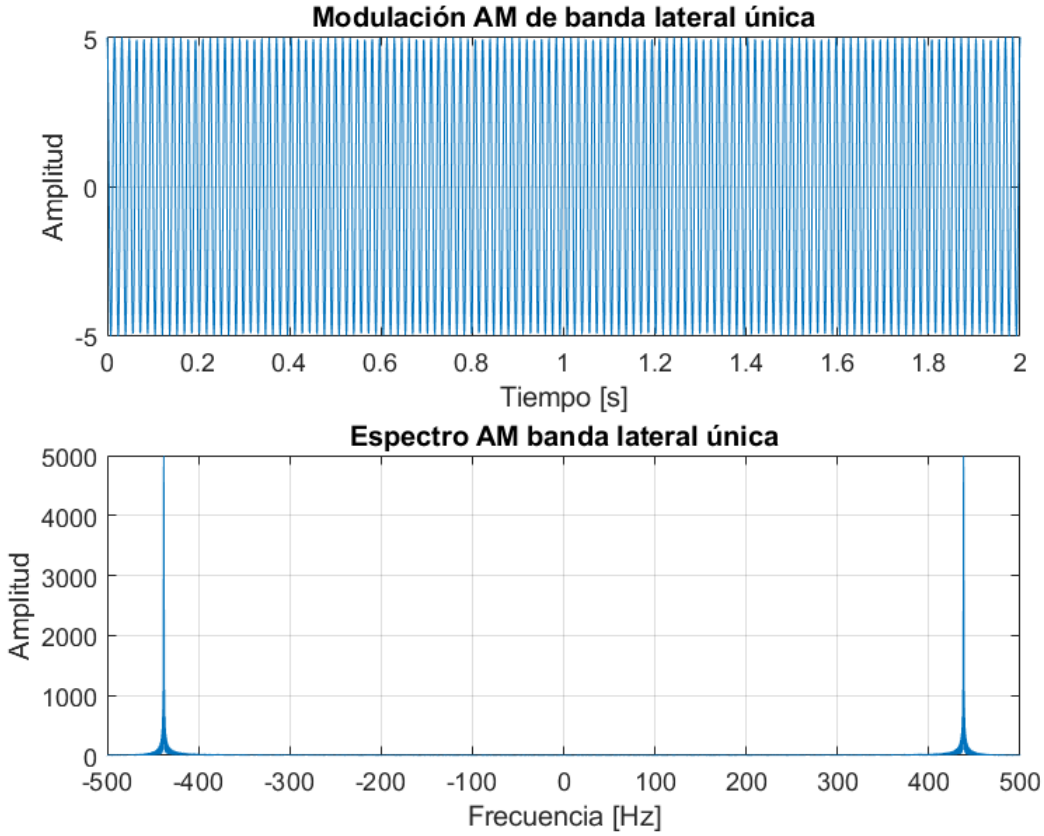


Figura 57: Modulación de Amplitud de Banda Lateral Única.

3.7. Código AMI

Las señales digitales son secuencias de pulsos de unos $(1')$ y ceros $(0')$, aquí cada uno de los pulsos representa un elemento de la señal. Estos pulsos son representados por ceros $(0')$ y unos $(1')$. Los *códigos de línea* surgen bajo la necesidad de tener que transmitir señales digitales a través de diversos medios de transmisión. En otras palabras, son una forma de representación de las secuencias de unos $(1')$ y ceros $(0')$ **(Proakis et al., 1994)**.

El código de línea **AMI**, del acrónimo proveniente de *Alternate Mark Invesiones*, y que indica una inversión alternada, es decir, va alternando los valores en donde la secuencia vale 1, a un valor de amplitud positivo $(+A)$ y un valor de amplitud negativo $(-A)$, y los ceros quedan igual sin sufrir cambio alguno, esto se puede observar en la **Figura 58**.

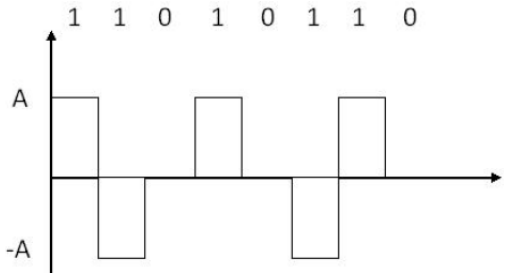


Figura 58: Código AMI.

El siguiente **código 5.22** convierte una secuencia a código AMI y produce la gráfica de la **Figura 59**, así como imprime las siguientes líneas:

```
La secuencia de entrada es:
     1     1     0     0     1     0     1     1     0     1     1     0     0     1

La secuencia AMI es:
     5    -5     0     0     5     0    -5     5     0    -5     5     0     0    -5
```

Código 5.22

Código

```
else
aux(k)=0;
end
end
disp('La secuencia de entrada es: ')
disp(sec)
disp('La secuencia AMI es: ')
disp(aux)
t=0:n-1;
stairs(t,aux)
axis([-1 14 -6 6])
grid on
xlabel('Periodo') ylabel('Amplitud') title('Codigo AMI
    ')
A=5;
sec=[1 1 0 0 1 0 1 1 0 1 1 0 0 1];
b=0;
n=length(sec);
for k=1:n
if sec(k)==1
b=b+1;
if mod(b,2)==0
aux(k)=-A;
else
aux(k)=A;
end
```

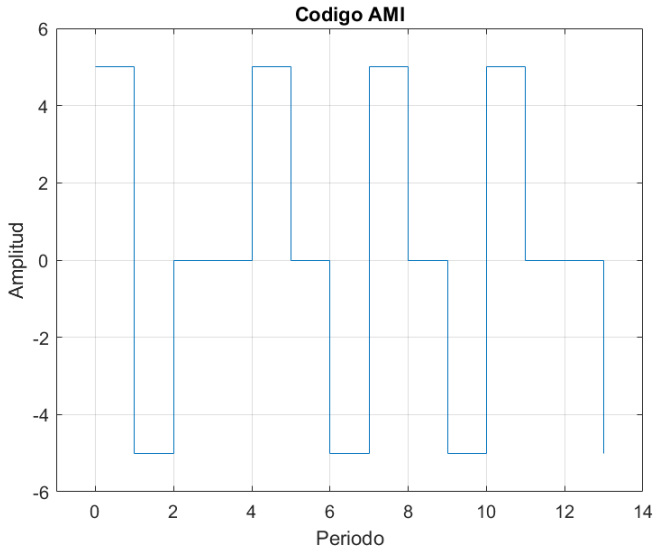


Figura 59: Código AMI.

4. Serie y Transformada de Fourier

4.1. Serie de Fourier

En la ingeniería, y en especial en las áreas de comunicaciones y electrónica, las *series de Fourier* se convierten en un tema relativamente importante y esencial, ya que son una herramienta matemática que nos ayudan en el análisis de señales periódicas. Como bien se dice, al ser una herramienta matemática, esta nos ayuda al análisis de diferentes señales a través de su descomposición en una suma infinita de funciones cosenoidales más simples. Antes de introducirnos directamente en el tema se recomienda que el alumnado estudie la representación de señales en términos de sus componentes de frecuencia.

4.1.1. Representación de señales en términos de sus componentes de frecuencia

Cuando comenzamos a trabajar con señales, uno de los conceptos fundamentales que se abordan es referente al *contenido de frecuencia* de una señal. La manera de generar el contenido de frecuencia para una gran cantidad de señales se puede lograr si dividimos la señal en sus componentes de frecuencia, las cuales están dadas por señales cosenoidales **(Oppenheim et al., 1998)**.

Consideremos la siguiente señal $x(t)$, la cual es una señal continua en el tiempo (t) y definida por una suma finita de señales cosenoidales:

$$x(t) = \sum_{n=1}^{N} A_n \cos(\omega_n t + \theta_n), \quad -\infty < t < \infty \tag{10}$$

De la **ecuación 10** definimos:

- N toma valores de enteros positivos.
- A_n representan las amplitudes de las funciones sinusoidales (las cuales asumimos como no negativas)
- ω_n representan las frecuencias (en rad/s) de las sinusoides
- θ_n representan las fases de las sinusoides.

Las frecuencias de la señal definida por la **ecuación 10** son ω_1 y sus componentes de frecuencia son las sinusoides $A_n \cos(\omega_n t + \theta_n)$. La señal definida en la **ecuación 10** está caracterizada por tres parámetros principales, que son: las frecuencias ω_1, ω_2, ..., ω_N, las amplitudes A_1, A_2, ..., A_N, y las fases θ_1, θ_2, ..., θ_N, **(Kamen & Heck, 2008)**.

Como se menciono anteriormente, la señal se puede caracterizar en función de los tres parámetros que la componen. Un caso particular, son las amplitudes A_1, A_2, ..., A_N, definen los pesos relativos de cada componente en cada frecuencia de la señal. Estas amplitudes caracterizan la *"forma"* de la señal, como se muestra en el siguiente ejemplo.

Ejemplo 4.1 Suma de sinusoides

Considere la señal continua dada por

$$x(t) = A_1\cos(t) + A_2\cos(4t + \pi/3) + A_3\cos(8t + \pi/2)+, \quad -\infty < t < \infty \tag{11}$$

Se puede observar que los parámetros de la señal son los siguientes; frecuencias de 1, 4, 8 rad/s, amplitudes A_1, A_2, A_3, y fases $0, \pi/3, \pi/2$ rad. Queremos mostrar que la forma de la señal se define por las amplitudes A_1, A_2, A_3.

Código

Código 6.1.

```
t = 0:20/400:20;
w1 = 1; w2 = 4; w3 = 8;
A1 = input('Input the amplitude A1 for w1 = 1: ');
A2 = input('Input the amplitude A2 for w2 = 4: ');
A3 = input('Input the amplitude A3 for w3 = 8: ');
x = A1*cos(w1*t)+A2*cos(w2*t+pi/3)+A3*cos(w3*t+pi/2);
```

Para alcanzar el propósito antes descrito, utilizamos los comandos de **MATLAB** presentados en la página anterior para generar $x(t)$, para distintos valores de A_1, A_2, y A_3. Mediante los comandos generamos las gráficas **MATLAB** de $x(t)$ para los tres casos $A_1 = 0.5$, $A_2 = 1$, $A_3 = 0$; $A_1 = 1$, $A_2 = 0.5$, $A_3 = 0$; y $A_1 = 1$, $A_2 = 1$, $A_3 = 0$. En la **Figura 60** se muestran los resultados. En los tres casos, sólo aparecen las componentes de frecuencia de 1 rad/s y 4 rad/s.

Primer caso: La componente de frecuencia 4 rad/s, se observa que sobresale en la **Figura 60(a)**, debido a que su amplitud es el doble en comparación con la componente de frecuencia 1 rad/s.

Segundo caso: caso contrario al caso 1, la componente dominante es la de frecuencia 1 rad/s, su amplitud es el doble en comparación con la componente de frecuencia 4 rad/s. Esto se muestra en la **Figura 60(b)**.

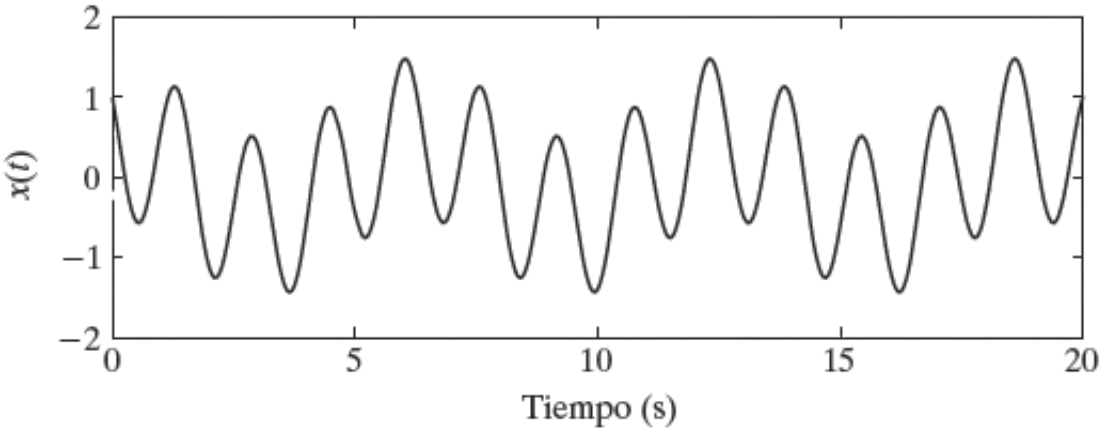
Tercer caso: No existe componente dominante, ambos tienen la misma amplitud, como se muestra en la **Figura 60(c)**.

Ejecutamos de nuevo el programa de **MATLAB** con los siguientes valores $A_1 = 0.5$, $A_2 = 1$, $A_3 = 0.5$; $A_1 = 1$, $A_2 = 0.5$, $A_3 = 0.5$; y $A_1 = 1$, $A_2 = 1$, $A_3 = 1$. ■

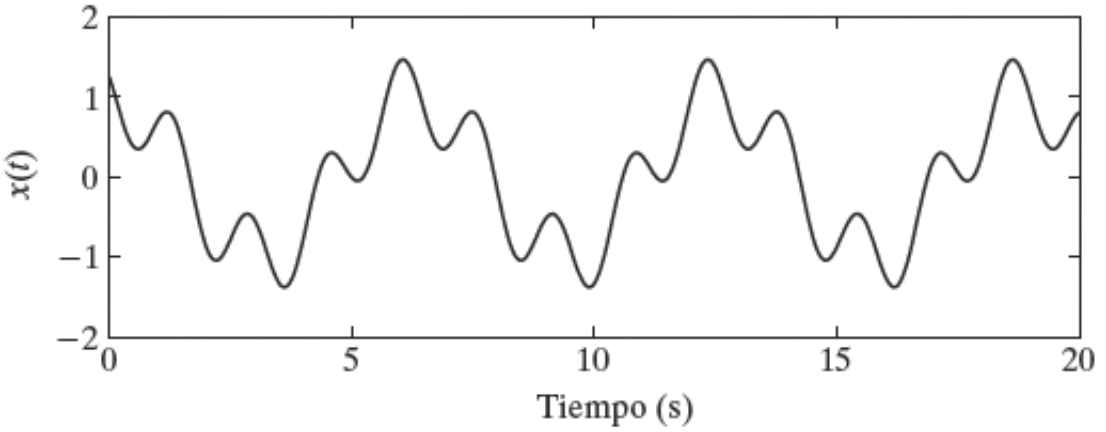
En los tres casos, están presentes todas las componentes. En el primer caso la componente dominante es la de 4 rad/s. En el segundo caso la componente dominante es la de 1 rad/s, y en el último caso todas las componentes tienen la misma amplitud. Estos casos se muestran en la **Figura 61**.

Considerando, nuevamente, la señal de la **ecuación 10**. Sea ω la *frecuencia variable*. Se puede graficar las amplitudes A_n vs ω. La señal $x(t)$ esta definida para valores finitos de frecuencias, por lo tanto, la gráfica tendrá un número finitos de valores en las frecuencias ω_n.

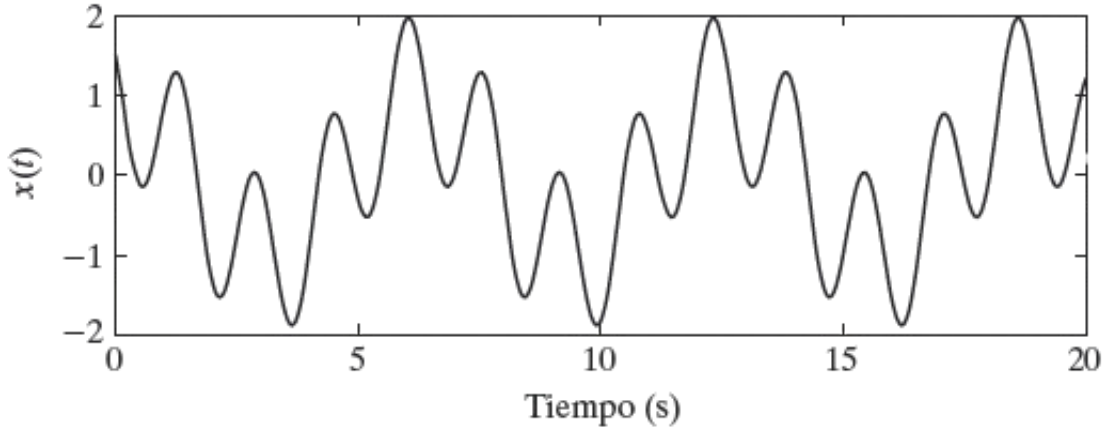
La gráfica que se muestra se conoce como *espectro de línea* o *espectro de amplitud* de la señal $x(t)$. El objetivo de este tipo de graficas es mostrar las magnitudes relativas de todas las componentes de frecuencia que integran la señal. Como ejemplo, podemos considerar la señal del **ejemplo 4.1**.



(a)

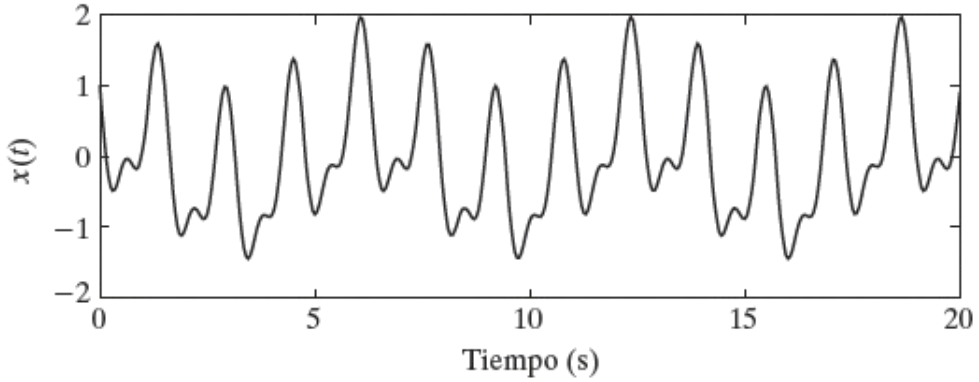


(b)

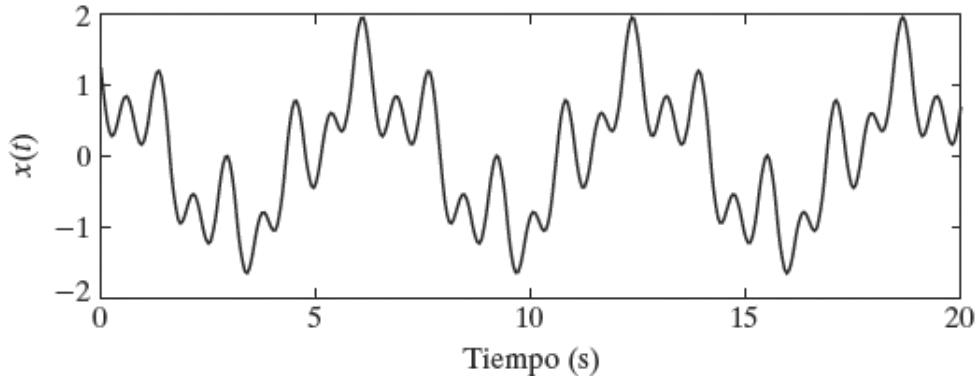


(c)

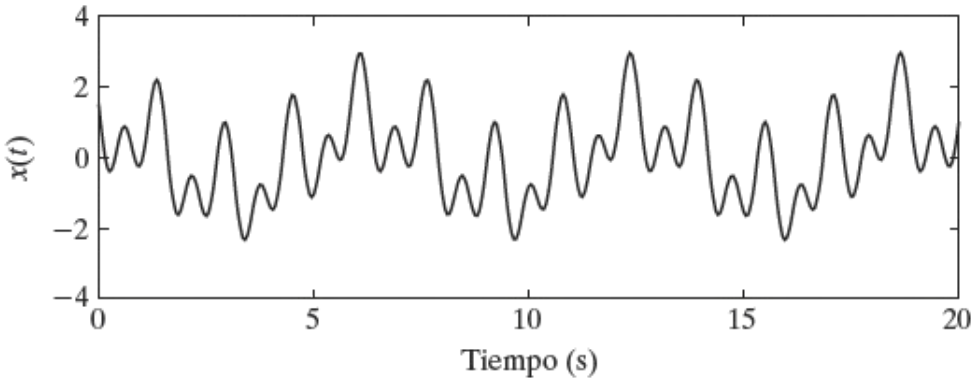
Figura 60: Gráficas de $x(t)$ para (a) $A_1 = 0.5$, $A_2 = 1$, $A_3 = 0$; (b) $A_1 = 1$, $A_2 = 0.5$, $A_3 = 0$; y (c) $A_1 = 1$, $A_2 = 1$, $A_3 = 0$.



(a)



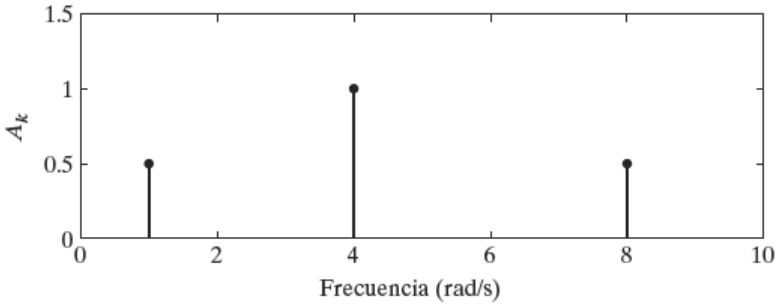
(b)



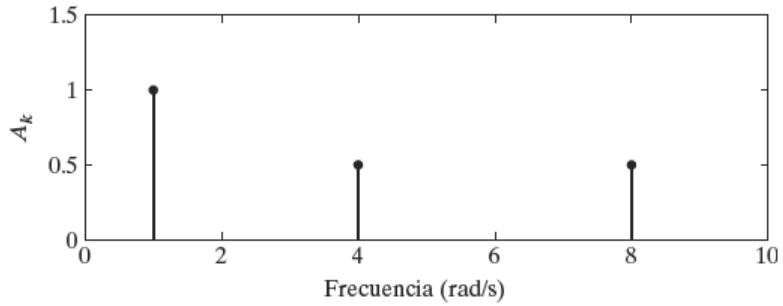
(c)

Figura 61: Gráficas de $x(t)$ para (a) $A_1 = 0.5$, $A_2 = 1$, $A_3 = 0.5$; (b) $A_1 = 1$, $A_2 = 0.5$, $A_3 = 0.5$; y (c) $A_1 = 1$, $A_2 = 1$, $A_3 = 1$.

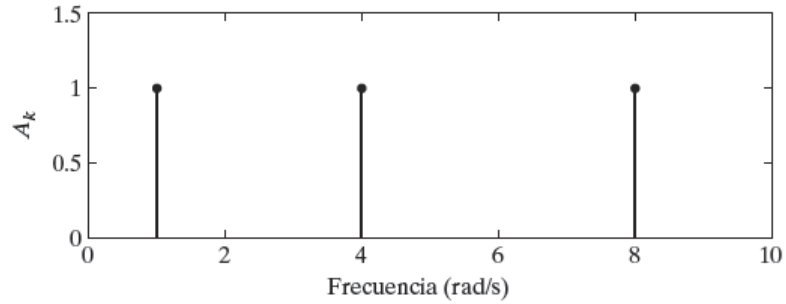
En la **Figura 62** se presenta el espectro de amplitud, de todas las versiones de la señal de la **Figura 61**. Existe una correspondencia entre la **Figura 62** y la **Figura 61**. La **ecuación 10** también contiene un *espectro de fase*, que consiste en graficar la fase θ_n vs la frecuencia ω. En la **Figura 63** se muestra un ejemplo de este tipo de espectro, para la **ecuación 11**.



(a)



(b)



(c)

Figura 62: Espectros de amplitud

4.1.2. Serie trigonométrica de Fourier

La serie de Fourier es una sucesión infinita que representa a una función o señal periódica, que también es continua por partes. Estas herramientas matemáticas, son la base fundamental para el análisis de Fourier, que sirven para estudiar las señales periódicas, mediante la desintegración de la señal en componentes de una suma infinita de funciones trigonométricas (senos y cosenos) **(Oppenheim et al., 1998)**.

Una señal $x(t)$ es periódica si cumple con la ecuación **ecuación 12**. Donde T se conoce como periodo, y es número real positivo.

$$x(t+T) = x(t), \quad -\infty < t < \infty \tag{12}$$

El número más pequeño para el cual se cumple la **ecuación 12** se conoce como periodo funda-

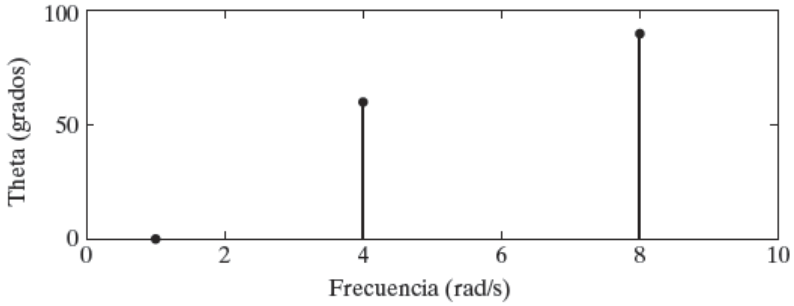


Figura 63: Espectro de fase de la señal $x(t)$ definida por **ecuación 11**.

mental T. En la **Figura 64** se muestra un ejemplo para $T = 2$, para un tren de pulsos.

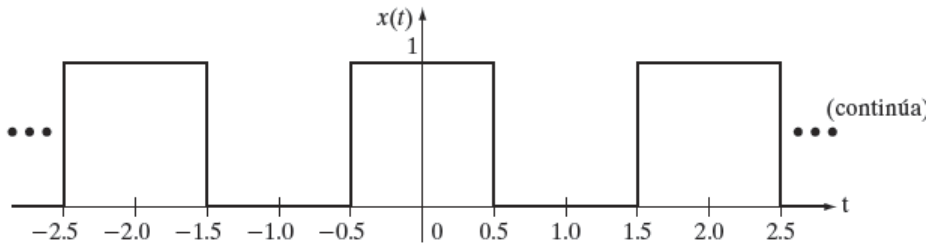


Figura 64: Señal periódica con periodo fundamental $T = 2$.

Si $x(t)$ es una señal periódica, se puede escribir como una suma de señales senoidales, como se muestra en la **ecuación 13**.

$$x(t) = a_0 + \sum_{n=1}^{\infty} \left[a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)\right], \quad -\infty < t < \infty \tag{13}$$

Donde a_0, a_n y b_n se conocen como coeficientes, y ω_0 es la *frecuencia fundamental* (en rad/s) dada por $\omega_0 = 2\pi/T$.

Los coeficientes a_n y b_n se calculan como se muestran en las siguientes ecuaciones.

$$a_n = \frac{2}{T}\int_0^T x(t)\cos(n\omega_o t)dt, \quad n = 1, 2, ... \tag{14}$$

$$b_n = \frac{2}{T}\int_0^T x(t)\sin(n\omega_o t)dt, \quad n = 1, 2, ... \tag{15}$$

Los coeficientes a_n y b_n, se pueden integrar en cualquier periodo completo, como se muestra:

$$a_n = \frac{2}{T}\int_{-T/2}^{T/2} x(t)\cos(n\omega_o t)dt, \quad n = 1, 2, ...$$

El coeficiente a_0 representa la componente de cd de la señal, dada por:

$$a_0 = \frac{1}{T}\int_0^T x(t)dt \tag{16}$$

La representación de la **ecuación 13** es conocida como *serie trigonométrica de Fourier*. Y a cada componente de la suma se conoce como armónicos. Observe que la frecuencia de cada componente de la suma son múltiplos de $n\omega_0$.
La serie trigonométrica de Fourier se puede mostrar como una *forma de cosenos con fase*

$$x(t) = a_0 + \sum_{n=1}^{\infty} A_n \cos(n\omega_0 t + \theta_n), \quad -\infty < t < \infty \tag{17}$$

donde

$$A_n = \sqrt{a_n^2 + b_n^2}, \quad n = 1, 2, ... \tag{18}$$

y

$$\theta_n = \begin{cases} \tan^{-1}\left(-\dfrac{b_n}{a_n}\right), & n = 1, 2, ..., \text{cuando } a_n \geq 0 \\ \pi + \tan^{-1}\left(-\dfrac{b_n}{a_n}\right), & n = 1, 2, ..., \text{cuando } a_n < 0 \end{cases} \tag{19}$$

La **ecuación 17** puede ser un caso particular de la **ecuación 10**, considerando que la suma sea infinita ($N = \infty$) y agregando la componente cd de la señal, a_0.

Es importante concluir que la serie de Fourier muestra que toda señal periódica, que cumple con las *condiciones de Dirichlet* **(Moya, 2008)**, se puede escribir como una suma de señales sinusoides.

1. $x(t)$ es integrable en su totalidad en cualquier periodo; es decir,

$$\int_a^{a+T} |x(t)|dt < \infty \quad \text{para cualquier } a$$

2. En un periodo la señal $x(t)$, contiene un número finito de mínimos y máximos.
3. En un periodo la señal $x(t)$, sólo tiene un número finito de discontinuidades.

Ejemplo 4.2 Tren de pulsos rectangulares

Dada la señal mostrada en la **Figura 64**. Se observa que la señal es periódica con $T = 2$, en consecuencia, la frecuencia fundamental es $\omega_0 = 2\pi/2$ rad/s. Calculando los coeficientes de la serie:

$$a_0 = \frac{1}{2}\int_0^2 x(t)dt = \frac{1}{2}\int_0^{0.5}(1)dt = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$$

$$\begin{aligned} a_n &= \frac{2}{2}\int_0^2 x(t)\cos(n\pi t)dt \\ &= \int_0^{0,5}\cos(n\pi t)dt + \int_{1,5}^2 \cos(n\pi t)dt \\ &= \frac{1}{\pi n}\sin(n\pi t)|_{t=0}^{t=0.5} + \frac{1}{\pi n}\sin(n\pi t)|_{t=1.5}^{t=2} \\ &= \frac{1}{\pi n}\left[\sin\left(\frac{n\pi}{2}\right) - \sin\left(\frac{3n\pi}{2}\right)\right] \\ &= \frac{1}{\pi n}\left[\sin\left(\frac{n\pi}{2}\right) - \sin\left(\frac{n\pi}{2}+n\pi\right)\right] = \frac{1}{\pi n}\left[2\sin\left(\frac{n\pi}{2}\right)\right], n = 1,2,... \end{aligned}$$

$$\begin{aligned} b_k &= \frac{2}{2}\int_0^2 x(t)\sin(k\pi t)dt = \int_0^{0.5}\sin(k\pi t)dt + \int_{1.5}^2 \sin(k\pi t)dt \\ &= -\frac{1}{\pi k}\cos(k\pi t)|_{t=0}^{t=0.5} - \frac{1}{\pi k}\cos(k\pi t)|_{t=1.5}^{t=2} = -\frac{1}{\pi k}\left[\frac{\pi}{2} - \frac{\pi}{2}\right] = 0, k = 1,2,... \end{aligned}$$

Con estos resultados, se puede escribir la señal $x(t)$ mediante su representación en serie de Fourier, como sigue:

$$x(t) = \frac{1}{2} + \frac{2}{\pi}\sum_{k=1}^{\infty}\frac{1}{k}\sin\left(\frac{k\pi}{2}\right)\cos(k\pi t), \quad -\infty < t < \infty \tag{20}$$

El resultado anterior, se puede reescribir para casos impares, debido a que la componente $\sin\left(\frac{k\pi}{2}\right)$ es cero para valores pares, quedando como sigue:

$$x(t) = \frac{1}{2} + \frac{2}{\pi}\sum_{\substack{k=1 \\ k_{impares}}}^{\infty}\frac{1}{k}\sin\left(\frac{k\pi}{2}\right)\cos(k\pi t), \quad -\infty < t < \infty \tag{21}$$

■

Simetría par o impar

Por definición, una señal $x(t)$ es una señal par si cumple con $x(t) = x(-t)$ para $\infty < t < \infty$. Por otro laso, es una señal impar si cumple con $x(t) = -x(-t)$ para $\infty < t < \infty$.

Sean $x(t)$ y $\nu(t)$ dos señales pares (o impares), se tiene que para cualquier constante $\eta > 0$, se cumple con:

$$\int_{-\eta}^{\eta} x(t)\nu(t)dt = 2\int_0^{\eta} x(t)\nu(t)dt \tag{22}$$

Sea $x(t)$ una señal par y $\nu(t)$ una señal impar, se cumple con:

$$\int_{-\eta}^{\eta} x(t)\nu(t)dt = 0 \tag{23}$$

Por lo tanto, dado que $\cos(n\pi t)$ es par y $\sin(n\pi t)$ es impar, si la señal $x(t)$ es par, la **ecuación 14** y la **ecuación 15** se reducen a

$$a_n = \frac{4}{T}\int_0^{T/2} x(t)\cos(n\omega_0 t)dt, \quad n = 1, 2, ... \tag{24}$$

$$b_n = 0, \quad n = 1, 2, ... \tag{25}$$

Por otro lado, si la señal $x(t)$ es impar, la **ecuación 14** y la **ecuación 15** se reducen a

$$a_n = 0, \quad n = 1, 2, ... \tag{26}$$

$$b_n = \frac{4}{T}\int_0^{T/2} x(t)\sin(n\omega_0 t)dt, \quad n = 1, 2, ... \tag{27}$$

Estás expresiones (**ecuación 24-27**), nos sirven para simplificar el cálculo de los coeficientes de las series de Fourier, según sea el caso, $x(t)$ par o impar. Para mostrar estas simplificaciones se muestra el siguiente ejemplo.

Ejemplo 4.3 Reducción por simetría

Considerando nuevamente la señal de la **Figura 64**. Se sabe que esta señal es par y, por lo que se pueden utilizar la **ecuación 24** y la **ecuación 25** para encontrar los coeficientes a_n y b_n de la serie de Fourier a_n y b_n.

- Primero, por simetría par de la señal $x(t)$, $b_n = 0, n = 1, 2, ...$
- Después, se calcula $a_n = 0$:

$$\begin{aligned} a_n &= \frac{4}{2}\int_0^1 x(t)\cos(n\pi t)dt = 2\int_0^{0.5}\cos(n\pi t)dt \\ &= \frac{2}{n\pi}\sin(n\pi t)|_{t=0}^{t=0.5} = \frac{2}{n\pi}\sin\left(\frac{n\pi}{2}\right), n = 1, 2, ... \end{aligned}$$

Los resultados coinciden con los encontrados en el ejemplo **4.2**, con esto se comprueba que usando la simetría se reduce el numero de cálculos.

■

Ejemplo 4.4 Ejercicio de MATLAB de serie de Fourier

Consideremos la función:

$$f(t) = 1 - \frac{t}{2}; \text{ para } -2 \leq x \leq 2 \tag{28}$$

Resolviendo los coeficientes de Fourier, se tiene $b_n = 2/n\pi$, con $T = 4$. El **código 6.2** genera la gráfica para la suma de los primeros N componentes de la serie de Fourier.

Código 6.2.

Código

```
x=-2:0.005:2;
suma=0;
N=50;
for n=1:N
b=2/(n*pi);
suma=suma+(b*sin(n*x*pi/2));
end
plot(x, suma)
grid on
xlabel('x')
ylabel('bn')
title('Serie de Fourier')
```

La gráfica que produce el **código 6.2** se muestra en la **Figura 65**. ■

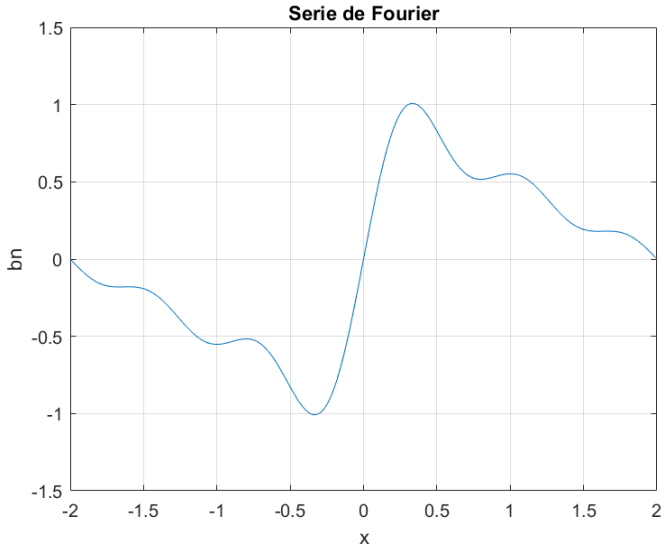


Figura 65: Serie de Fourier.

El fenómeno de Gibbs se genera por las discontinuidades que presenta la función, debido a que el resultado es una aproximación de la señal $x(t)$, dado que se toma para un valor finito de N. Gráficamente podemos observar este fenómeno en el **ejemplo 4.4**. Se presentan oscilaciones de la señal, el cual se puede reducir aumentando el número de elementos, por ejemplo para una valor de $N = 50$, se tiene el siguiente **código 6.3**, y el cual produce la gráfica de la **Figura 66**.

Ejemplo 4.5 **Reducción de las oscilaciones en la señal del ejemplo 4.4**

Código

Código 6.3.

```
x=-2:0.005:2;
suma=0;
N=50;
for n=1:N
b=2/(n*pi);
suma=suma+(b*sin(n*x*pi/2));
end
plot(x, suma)
grid on
xlabel('x')
ylabel('bn')
title('Serie de Fourier')
```

■

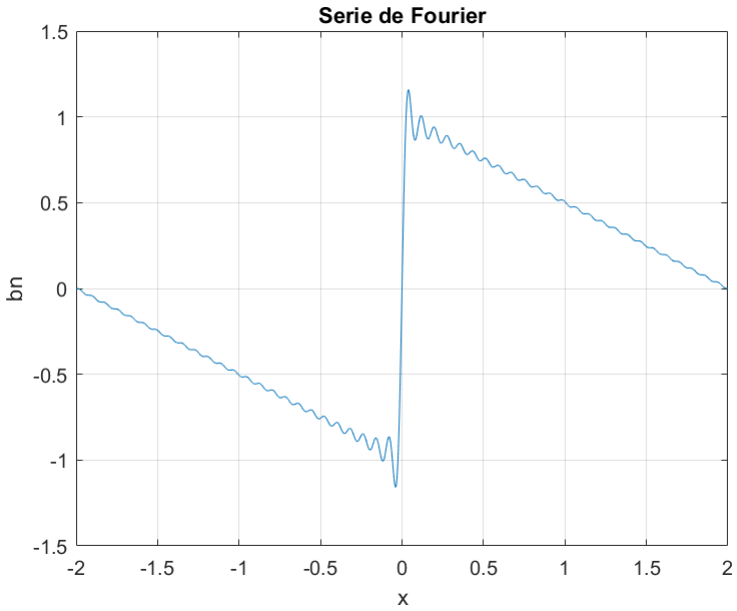


Figura 66: Gráfica de $x_N(t)$ cuando $N = 50$.

Veamos otro caso, tomando la señal $x(t)$ del ejemplo anterior del tren de pulsos, sea $x_N(t)$ la señal que define la suma finita, se tiene:

$$x_N(t) = \frac{1}{2} + \frac{2}{\pi} \sum_{\substack{k=1 \\ k_{impar}}}^{N} \frac{1}{k} \sin\left(\frac{k\pi}{2}\right) \cos(k\pi t), \quad -\infty < t < \infty$$

Por definición del teorema de Fourier (**ecuación 13**), $x_N(t)$ converge en $x(t)$ cuando $N \to \infty$. Por lo tanto, conforme N tiende a infinito, el error $|x_N(t) - x(t)|$ tiende a cero. El código siguiente, muestra estos resultados para distintos valores de N.

Código

```
t = 3:6/1000:3;
N = input('Number of harmonics = ');
a0 = 0.5;
w0 = pi;
xN = a0*ones(1, length(t));
for k = 1:2:N,
xN = xN + 2/k/pi*sin(k*pi/2)*cos(k*w0*t);
end
```

Dado que la señal es par, los coeficientes a_n se excluyen del ciclo para hacer el programa de **MATLAB** más eficiente.

Ahora, con $N = 3$, $x_N(t)$ se vuelve

$$x_3(t) = \frac{1}{2} + \frac{2}{\pi}\cos(\pi t) - \frac{2}{3\pi}\cos(3\pi t), \quad -\infty < t < \infty$$

Con $N = 3$, se obtiene la gráfica mostrada en la **Figura 67**.

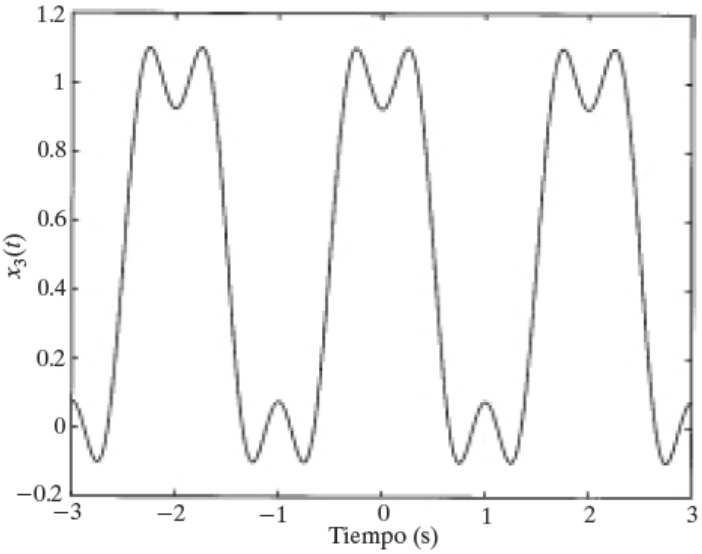


Figura 67: Gráfica de $x_N(t)$ cuando $N = 3$.

La **Figura 68** muestra los resultados para $N = 9$. Comparando la **Figura 67** y la **Figura 68** se observa que la aproximación para $N = 9$ presenta mejores resultados, dado que contiene mas armónicos. Al incrementar el valor de N se puede obtener una mejor aproximación de la señal $x(t)$. Por ejemplo, para $N = 21$ se obtiene la gráfica de la **Figura 69**.

Para $N = 45$ se obtiene el resultado mostrado en la , **Figura 70**. Se observa que los picos en las esquinas siguen apareciendo´, aún cuando N tiende a ∞ los picos estarán presentes (aproximadamente el 9 por ciento). Este fenómeno fue descubierto por Josiah Willard Gibbs, por lo que llevan su nombre y se conoce como fenómeno de Gibbs. Por lo tanto, para una señal $x(t)$ periódica, la aproximación por serie de Fourier presenta errores en las discontinuidades de $x(t)$ de hasta aproximadamente 9 %.

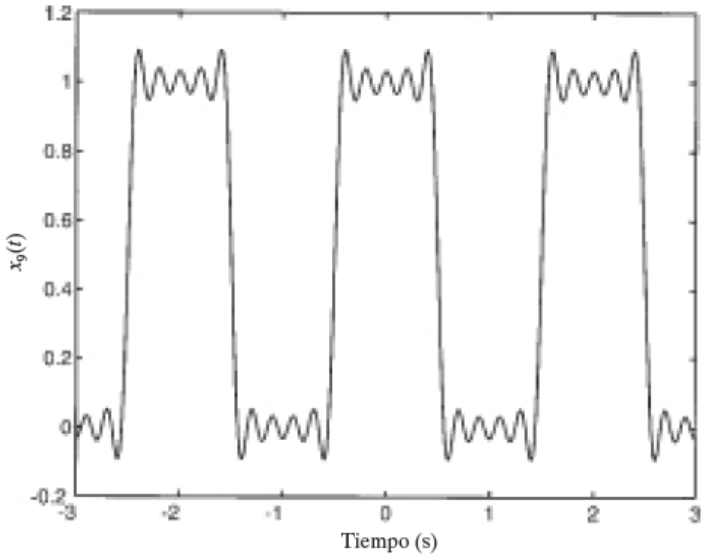


Figura 68: Aproximación $x_9(t)$.

4.1.3. Serie exponencial compleja

La serie trigonométrica de Fourier vista en la sección anterior (**ecuación 13**), se puede expresar de forma exponencial compleja, mediante:

$$\mathrm{x(t)}=\sum_{n=-\infty}^{\infty} c_n e^{jn\omega_0 t}, \quad -\infty < t < \infty \tag{29}$$

En la representación la **ecuación 29**, el coeficiente c_0 es un número real y los coeficientes c_n $(n \neq 0)$ son números complejos. ω_0 al igual que la sección anterior representa la frecuencua fundamental, $\omega_0 = 2pi/T$, donde T es el periodo fundamental.

Existen ecuaciones de equivalencias que relacionan los coeficientes de la serie exponencial compleja con los coeficientes de la serie trigonométrica de Fourier. Estas relaciones se calculan mediante las fórmulas

$$\begin{aligned} c_0 &= a_0 \\ c_n &= \frac{1}{2}(a_n - jb_n) \\ c_{-n} &= \frac{1}{2}(a_n + jb_n), n = 1, 2, ... \end{aligned} \tag{30}$$

Los coeficientes de la serie exponencial compleja, c_n, se pueden calcular utilizando la fórmula

$$\mathrm{c}_n = \frac{1}{T}\int_0^T x(t)e^{-jn\omega_0 t}dt, \quad n = 0, \pm 1, \pm 2, ... \tag{31}$$

Al igual que los coeficientes de la serie trigonométrica, los coeficientes de la serie exponencial, c_n, se pueden calcular integrando sobre cualquier periodo completo. Por ejemplo,

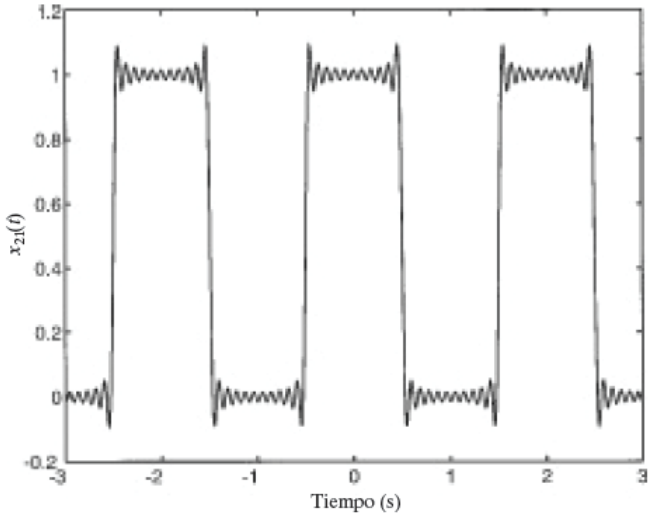


Figura 69: Aprovimación $x_{21}(t)$.

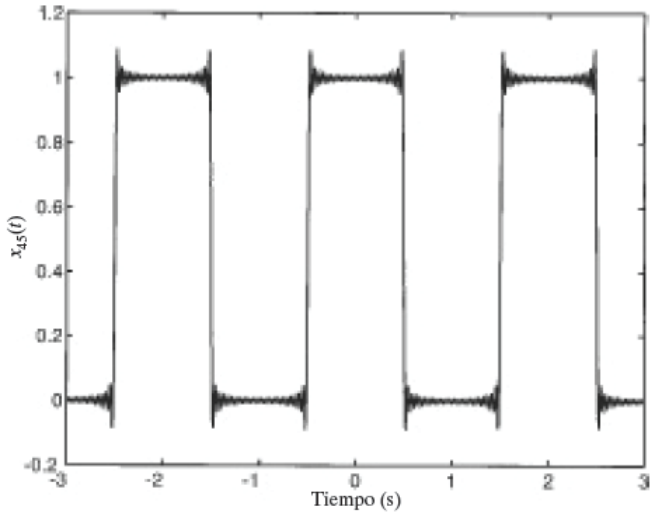


Figura 70: Aprovimación $x_{45}(t)$.

$$c_n = \frac{1}{T}\int_{-T/2}^{T/2} x(t)e^{-jn\omega_0 t}dt, \quad n = 0, \pm 1, \pm 2, ...$$

Si $x(t)$ es una señal periódica y está expresada mediante la serie exponencial compleja, entonces, es posible pasarla en forma de serie trigonométrica de Fourier utilizando las siguientes relaciones para los coeficientes

$$\mathrm{a}_0 = c_0 \qquad \begin{aligned} a_n &= c_n + c_{-n} \\ &= 2\Re(c_n) \end{aligned} \qquad \begin{aligned} b_n &= j\,(c_n - c_{-n}) \\ &= -2\Im(c_n), n = 1, 2, ... \end{aligned} \tag{32}$$

Las relaciones de la **ecuación 32** se deducen fácilmente de la **ecuación 30**. Cuando la señal $x(t)$ es par, los coeficientes de la serie exponencial son números reales y las ecuaciones se reducen a lo siguiente:

$$c_0 = a_0 \qquad c_n = \frac{1}{2}a_n \qquad c_{-n} = \frac{1}{2}a_n, n = 1, 2, ... \tag{33}$$

Cuando la señal $x(t)$ es impar de t, los coeficientes de la forma exponencial son números imaginarios (excepto para c_0) y las ecuaciones se reducen a lo siguiente:

$$c_0 = a_0 \qquad c_n = -j\frac{1}{2}b_n \qquad c_{-n} = j\frac{1}{2}b_n, n = 1, 2, ... \tag{34}$$

Ejemplo 4.6 Tren de pulsos rectangulares

Nuevamente, considerando la señal mostrada en la **Figura 64**. Con los resultados obtenidos en el ejemplo **4.2**, podemos calcular los coeficientes de la serie compleja de Fourier como sigue:

$$c_0 = \frac{1}{2}$$
$$c_n = \frac{1}{\pi n}\sin\left(\frac{n\pi}{2}\right), n = 0, \pm 1, \pm 3, \pm 5, ...$$

Por lo tanto, teniendo los coeficientes podemos expresar la serie exponencial compleja de Fourier de un tren de pulsos como sigue:

$$x(t) = \frac{1}{2} + \frac{1}{\pi}\sum_{\substack{n=-\infty \\ n_{impar}}}^{\infty}\frac{1}{k}\sin\left(\frac{n\pi}{2}\right)e^{jn\pi t}, \quad -\infty < t < \infty \tag{35}$$

■

Existe una manera sencilla de resolver las ecuaciones para el cálculo de los coeficientes de Fourier utilizando la herramienta de simulación MATLAB, para esto se puede puede utilizar `Symbolic Math Toolbox`. Un ejemplo se muestra en el siguiente código para la señal tren de pulsos rectangulares.

Código

```
k = 1:5;
syms ck t
ck = 0.5*int(exp(-j*k*pi*t),t,-0.5,0.5)
```

La instrucción `syms cn t` genera objetos simbólicos `cn` y `t`. La instrucción `int` integra el primer argumento respecto al segundo argumento `t`, desde -0.5 hasta 0.5. El resultado es un vector de `cn`.

Código

```
ck =[ 1/pi, 0, -1/3/pi, 0, 1/5/pi]
```

Serie de Fourier compleja y truncada

La señal $x_N(t)$ obtenida es un representación truncada de la serie trigonométrica de Fourier de la señal tren de pulsos rectangulares. Esta señal también se puede calcular truncando la forma de la serie exponencial de Fourier como sigue:

$$x_N(t) = \sum_{n=-N}^{N} c_n e^{jn\omega_0 t}$$

Utilizando **MATLAB** se puede calcular lo expresado en la ecuación anterior, como sigue:

Código

```
t = -3:6/1000:3;
N = input ('Number of harmonics = ');
c0 = 0.5;
w0 = pi;
xN = c0*ones (1, length(t));
for k = 1:N,
ck = 1/k/pi*sin(k*pi/2);
c_k = ck;
xN = xN + ck*exp(j*k*w0*t) + c_k*exp(-j*k*w0*t);
end
```

Utilizando el código anterior para los casos $N = 3, 9, 21$ y 45, obtenemos los mismos resultados mostrados en la **Figura 67** a la **Figura 70**.

Ejemplo 4.7 **Cálculo de la serie de Fourier de una señal acotada**

Expanda en una serie de Fourier la siguiente señal $f(t)$:

$$f(t) = \begin{cases} 0 & \text{para } -\pi < t < 0 \\ \pi - t & \text{para} 0 \leq t < \pi \end{cases} \tag{36}$$

El siguiente **código 6.4** calcula la serie de Fourier de la señal $f(t)$, y se genera la gráfica de la **Figura 71**.

Código 6.4.

Código

```
syms n t
A=pi;
T=2*pi;
wo=2*pi/T;
funcion=-t+A;
a0=int(funcion,0,pi)/(T/2);
an=int(funcion*cos(n*wo*t),0,pi)/(T/2);
bn=int(funcion*sin(n*wo*t),0,pi)/(T/2);
sum=0;
for k=1:6
 ank=subs(an,n,k);
 bnk=subs(bn,n,k);
 sum=sum+ank*cos(k*t)+bnk*sin(k*t);
end
suma=a0/2+sum;
xv=linspace(-pi,pi);
y=subs(suma,t,xv);
plot(xv,y,'*b','Linewidth',1)
grid on;
hold on;
t1=linspace(-(T/2),0);
t2=linspace(0,(T/2));
t=[t1,t2];
v=[(0*(-t1+A)) (-t2+A)];
plot(t,v,'r','Linewidth',3)
```

■

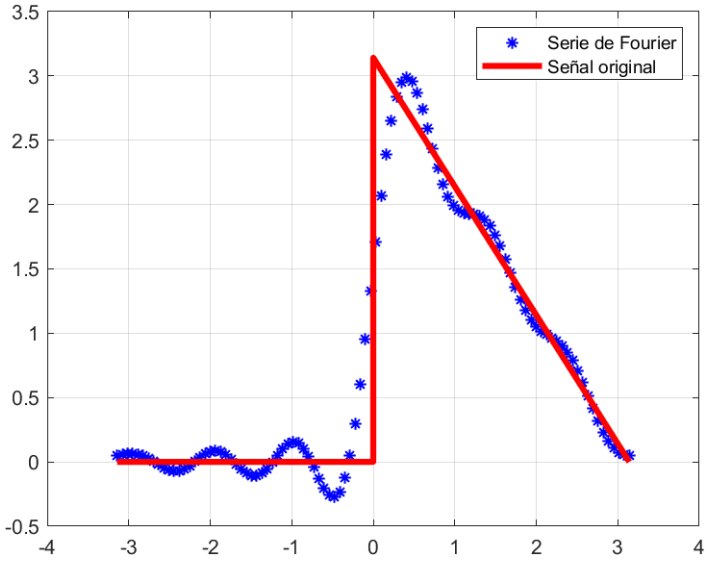


Figura 71: Cálculo de la serie de Fourier para la señal $f(t)$ del **ejemplo 4.7**.

Ejemplo 4.8 **Cálculo de la serie de Fourier de una señal cuadrada**

Calcular la serie de Fourier de la siguiente señal cuadrada $f(t)$:

$$f(t) = \begin{cases} 1 & \text{para} 0 < t < \pi \\ -1 & \text{para} \pi \leq t < 2\pi \end{cases} \tag{37}$$

El siguiente **código 6.5** calcula la serie de Fourier de la señal $f(t)$, y se genera la gráfica de la **Figura 72**.

Código 6.5.

Código

```
T=2*pi;
f=1/T;
wo=(2*pi)/(T);
A=1;
n=1:50;
t=0:0.01:2*pi;
for n=1:2:6
 for k=1:size(t,2)
 s(n,k)=(((4*A)/(pi)))*((sin(n*wo*t(k)))/n);
 end
end
for k=1:size(t,2)
 st(k)=sum(s(:,k));
end
st(1)=st(1)+1;
plot(t,st,'*b','Linewidth',1);
grid on;
hold on;
f_cuadrada=square(2*pi*f*t,50);
plot(t,f_cuadrada,'Linewidth',3);
xlabel('tiempo');
ylabel('Amplitud');
```

■

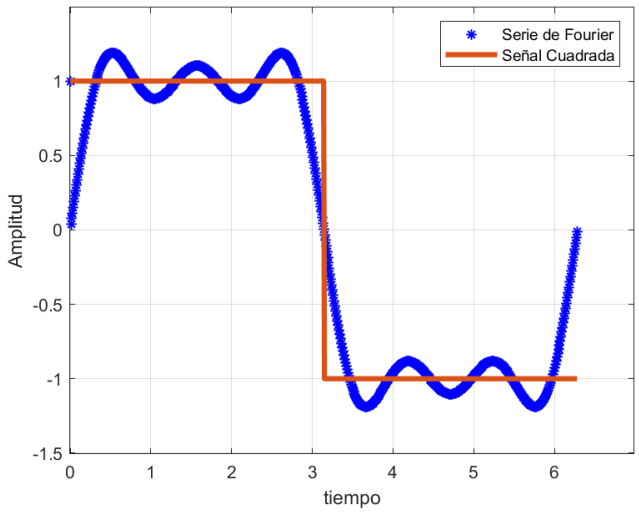


Figura 72: Cálculo de la serie de Fourier para la señal $f(t)$ del **ejemplo 4.8**.

Ejemplo 4.9 **Cálculo de la serie de Fourier de una señal diente de sierra**
Expanda en una serie de Fourier la siguiente señal $f(t)$:

$$f(t) = \begin{cases} At/\pi & \text{para} 0 < t < \pi \\ (At/\pi) - 2A & \text{para} \pi \leq t < 2\pi \end{cases} \tag{38}$$

El siguiente **código 6.6** calcula la serie de Fourier de la señal $f(t)$, y se genera la gráfica de la **Figura 73**.

Código 6.6.

```
syms n t
A=1;
T=2*pi;
wo=2*pi/T;
p=pi;
funcion1=(A*t)/pi;
funcion2=((A*t)/pi)-2*A;
a0=int(funcion1,0,pi)/(T/2)+int(funcion2,pi,2*pi)/(T
    /2);
an=int(funcion1*cos(n*wo*t),t,0,pi)/(T/2)+int(
    funcion2*cos(n*wo*t),t,pi,(2*pi))/(T/2);
bn=int(funcion1*sin(n*wo*t),t,0,pi)/(T/2)+int(
    funcion2*sin(n*wo*t),t,pi,(2*pi))/(T/2);
sum=0;
for k=1:40
 ank=subs(an,n,k);
 bnk=subs(bn,n,k);
 sum=sum+ank*cos(k*t)+bnk*sin(k*t);
end
suma=a0/2+sum;
xv=linspace(0,2*pi);
y=subs(suma,t,xv);
plot(xv,y,'*b','Linewidth',1)
grid on;
hold on;
t1=linspace(0,pi);
t2=linspace(pi,2*pi);
t=[t1,t2];
v=[(((A*t1)/pi)) (((A*t2)/pi)-2*A)];
plot(t,v,'r','Linewidth',2)
```

Código ■

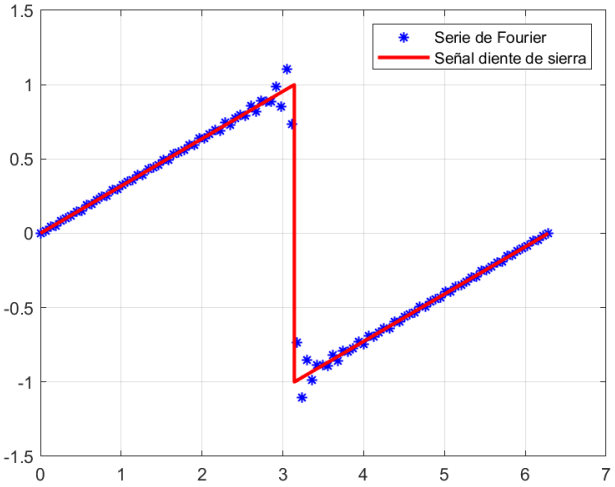


Figura 73: Cálculo de la serie de Fourier para la señal $f(t)$ del **ejemplo 4.9**.

4.2. Transformada de Fourier

Hemos visto que el análisis de una señal periódica se puede realizar mediante la serie de Fourier, esto permite representar a la señal en términos del contenido de frecuencia. Para las señales no periódicas existe una herramienta matemática que permite el análisis de las señales mediante sus componentes espectrales de frecuencia, esta herramienta es la transformada de Fourier.

Este análisis, mediante la transformada de Fourier, muestra que para las señales no periódicas las componentes existen o están definidas en todos los valores reales de la frecuencia ω, y no como en el caso de las señales periódicas que existe solo para valores discretos de ω, por lo tanto, para la transformada de Fourier no es un espectro de línea.

La transformada de Fourier de una señal $x(t)$ se denota por $X(\omega)$, se calcula como sigue:

$$\mathrm{X}(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt, \quad -\infty < t < \infty \tag{39}$$

donde ω es la variable que representa la frecuencia. Como se observa las señales se denotan con letras minúsculas y la transformada de Fourier con letras mayúsculas.

Por naturaleza de la **ecuación 39**, los valores de la transformada de Fourier pueden ser complejos. Por esta razón, los resultados se representan mediante la función magnitud $|X(\omega)|$ (espectro de magnitud) y la función ángulo $\angle X(\omega)$ (espectro de fase).

La transformada de Fourier $|X(\omega)|$ de una señal $x(t)$, existe si la señal $x(t)$ es completamente integrable, es decir, si la **ecuación 39** converge. Eso es:

$$\int_{-\infty}^{\infty} |x(t)|dt < \infty \tag{40}$$

El siguiente ejemplo es para una señal que no cumple con la condición de existencia de la transformada de Fourier.

Ejemplo 4.10 Señal constante

Considere la señal de cd o constante

$$x(t) = 1, \quad -\infty < t < \infty$$

La señal constante es físicamente no realizable, es decir no es una señal real, ya que es una señal diferente de cero todo el tiempo. La transformada de Fourier de la señal es:

$$\begin{aligned} X(\omega) &= \int_{-\infty}^{\infty} (1)e^{-j\omega t}dt = \lim_{T\to\infty} \int_{T/2}^{T/2} e^{-j\omega t}dt = \lim_{T\to\infty} -\frac{1}{j\omega}\left[e^{-j\omega t}\right]_{t=-T/2}^{t=T/2} \\ &= \lim_{T\to\infty} -\frac{1}{j\omega}\left[e^{(-\frac{j\omega T}{2})} - e^{(\frac{j\omega T}{2})}\right] \end{aligned} \tag{41}$$

Pero $e^{(\frac{j\omega T}{2})}$ no tiene un límite cuando $T \to \infty$ y, por lo tanto, la integral de la **ecuación 41** no converge. Esto indica que la señal no tiene transformada de Fourier, ya que al integrar la señal el área bajo la curva es infinita. ■

Ejemplo 4.11 Señal exponencial

Ahora considere la señal

$$x(t) = e^{-bt}u(t)$$

donde b es una número constante real, y $u(t)$ es la señal escalón unitario. La señal $x(t)$ toma el valor de la señal $u(t)$ cuando el número constante $b = 0$. La transformada de Fourier $X(\omega)$ de $x(t)$ está dada por

$$X(\omega) = \int_{-\infty}^{\infty} e^{-bt}u(t)e^{-j\omega t}dt$$

y debido a que $u(t) = 0$ para $t < 0$, $u(t) = 1$ para $t \geq 1$,

$$X(\omega) = \int_{0}^{\infty} e^{-bt}e^{-j\omega t}dt = \int_{0}^{\infty} e^{-(b+j\omega)t}dt$$

Evaluando la integral:

$$X(\omega) = -\frac{1}{b+j\omega}\left[e^{-(b+j\omega)t}\Big|_{t=0}^{t=\infty}\right]$$

Cuando $b \leq 0$ el límite superior $t = \infty$ no puede ser evaluado y, entonces, es este intervalo para b, no existe la transformada de Fourier de $x(T)$. Como se menciono anteriormente, $x(t) = u(t)$ cuando $b = 0$, lo que indica que la función escalón unitario no tiene transformada de Fourier. Cuando $b > 0$, $e^{-bt} \to 0$ cuando $t \to \infty$, entonces

$$\lim_{t\to\infty} e^{-(b+j\omega)t} = \lim_{t\to\infty} e^{-bt}e^{-j\omega t} = 0$$

La transforma de Fourier de $x(t)$ cuando $b > 0$ esta dad por

$$X(\omega) = -\frac{1}{b+j\omega}(0-1) = \frac{1}{b+j\omega}$$

y los espectros de amplitud y fase son

$$|X(\omega)| = \frac{1}{\sqrt{b^2+\omega^2}}$$

$$\angle X(\omega) = -\tan^{-1}\frac{\omega}{b}$$

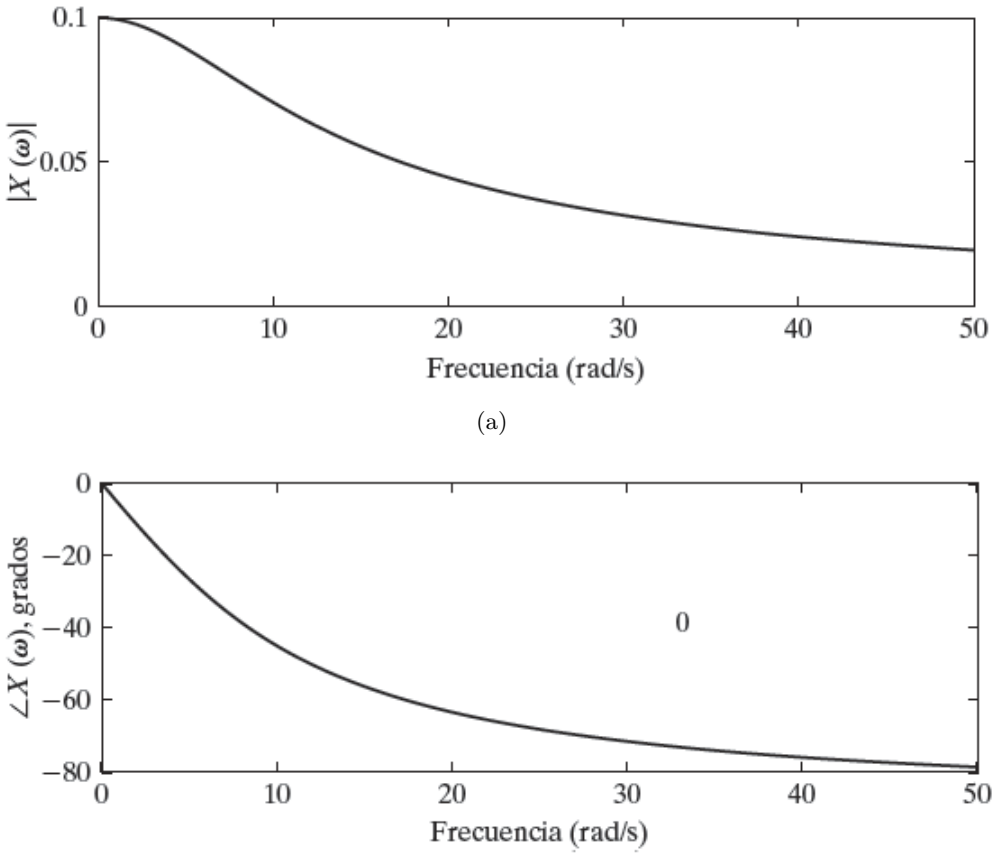
Como ejemplo, cuando $b = 10$, se generan los resultados para el espectro de amplitud $|X(\omega)|$ y de fase $\angle X(\omega)$ utilizando **MATLAB**, como se muestra en el codigo siguiente:

Código

Código 6.7.

```
w = 0:0.2:50;
b = 10;
X = 1./(b+j*w);
subplot(211), plot(w, abs (X));
subplot(212), plot(w, angle (X));
```

Las gráficas que se generan con el código se muestran en la **Figura 74**. ■



(a)

(b)

Figura 74: Gráficas de los espectros de (a) amplitud y (b) fase, de $x(t) = e^{-10t}u(t)$.

4.3. Transformada inversa de Fourier

Dada la transformada de Fourier $X(\omega)$ de la señal $x(t)$. Podemos obtener $x(t)$ a partir de $X(\omega)$ aplicando la transformada inversa de Fourier dada por

$$\mathrm{x(t)}= \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{j\omega t}d\omega \tag{42}$$

Como notación se utilizará la siguiente para los pares de transformadas

$$x(t) \leftrightarrow X(\omega)$$

Un ejemplo de pares de transformadas de Fourier es el siguiente

$$\mathrm{p}_\tau(t) \leftrightarrow \tau \arcsin\left(\frac{\tau\omega}{2\pi}\right) \tag{43}$$

Por la propiedad de dualidad de Fourier, la transformada de Fourier de la señal rectangular en el tiempo corresponde a una función *senc* en frecuencia, e inversamente, una función *senc* en frecuencia corresponde a una función rectangular en el tiempo.

5. Aplicaciones en propagación y desvanecimientos

En este capítulo se analizarán los fenómenos que se presentan y afectan al momento de realizar una transmisión de información en el espacio libre.

5.1. Patrón de Radiación

Al momento de hacer referencia a una transmisión en el espacio libre, lo primero que resalta es el término de *espacio libre*, el cual lo identificamos como el medio por el cual se realizará la propagación de las ondas electromagnéticas, y para que esta actividad se dé, se requiere del uso de antenas. Las antenas se pueden definir como dispositivos que pueden transmitir y/o recibir ondas electromagnéticas (**OEM**). Una antena puede ser caracterizada a través de diversos parámetros, y uno de los más esenciales e incluso de los más relevantes, es el patrón de radiación. El patrón de radiación de una antena se puede entender como una gráfica de la energía radiada vista desde afuera. Si consideramos un dipolo por ejemplo, el campo generado por este sería como el que se muestra en la siguiente ecuación **(Aguilar et al., 2002)**:

$$E = \frac{\cos\left(\beta\frac{l}{2} * \cos(\theta)\right) - \cos\left(\beta\frac{l}{2}\right)}{\sin(\theta)} \tag{44}$$

De la **ecuación 44** podemos describir los siguientes parámetros:

- β es la constante de fase $(2\pi/\lambda)$.
- l es la longitud del dipolo.
- θ es el ángulo medido sobre el eje z.

El **código 7.1** calcula y grafica el campo generado por un dipolo como se puede observar en la **Figura 75**.

Código 7.1.

Código

```
lambda=1;
beta=2*pi/lambda;
l=3*lambda/2; % longitud del dipolo
% angulos
fi=(0:.01:1)*2*pi;
teta=(0:.01:1)*pi;
E=abs((cos(beta.*l./2.*cos(teta))-cos(beta.*l./2))./sin
    (teta));
polar(teta,E)
```

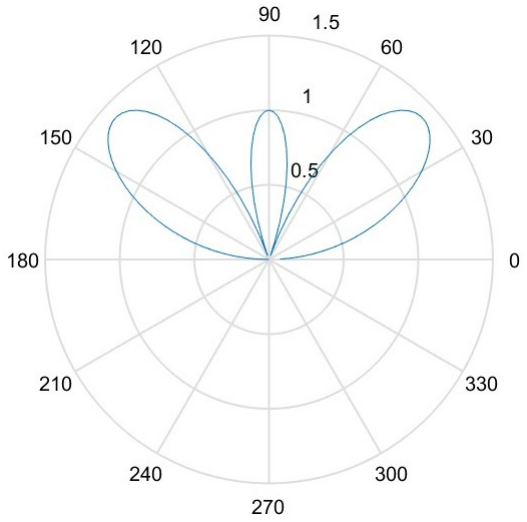


Figura 75: Patrón de radiación de un dipolo.

Se puede graficar este mismo campo en 3D, para realizar lo antes mencionado, se debe utilizar el siguiente **código 7.2**, el cual produce la gráfica de la **Figura 76**.

Código 7.2.

Código

```
lambda=1;
beta=2*pi/lambda;
l=3*lambda/2; % longitud del dipolo
% angulos.
fi=(0:.01:1)*2*pi;
teta=(0:.01:1)*pi;
E=abs((cos(beta.*l./2.*cos(teta))-cos(beta.*l./2))./sin
    (teta));
polar(teta,E)
[FI,TETA]=meshgrid(fi,teta); % vectores FI y TETA para
    graficar en 3D
% creacion del campo
E=abs((cos(beta.*l./2.*cos(TETA))-cos(beta.*l./2))./sin
    (TETA));
% cambio de coordenadas esfericas a rectangulares
X=E.*sin(TETA).*cos(FI);
Y=E.*sin(TETA).*sin(FI);
Z=E.*cos(TETA);
S=surface(X,Y,Z,(abs(E))); % Grafica de superficie
view(0,40) % Orientación de los ejes
```

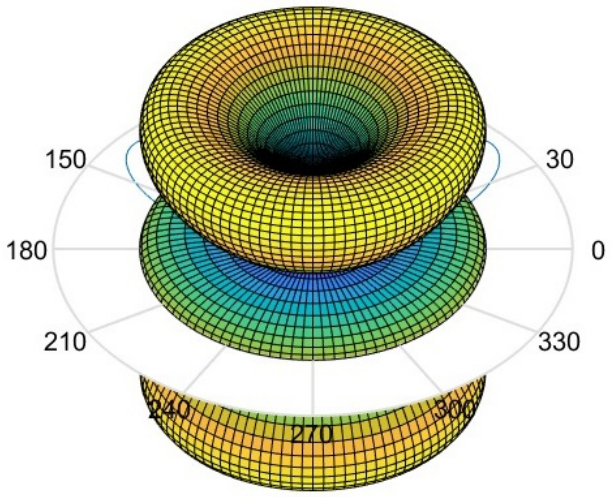


Figura 76: Campo del dipolo en 3D.

5.2. Desvanecimientos

Cuando el medio de transmisión es el canal inalámbrico se presentan diversos fenómenos que disminuyen la intensidad de la señal recibida, estos fenómenos son conocidos como desvanecimientos, los cuales se clasifican en *desvanecimientos lentos* y *desvanecimientos rápidos* **(Stüber & Stèuber, 2012)**.

Los *desvanecimientos rápidos* se dividen según el ambiente de propagación, es decir, el transmisor y receptor pueden tener o no línea de vista. Generalmente no se tiene línea de vista y cuando se tiene este ambiente de propagación la variable aleatoria que caracteriza estos desvanecimientos sigue una distribución de *Rayleigh*, es decir el canal de desvanecimiento es un *canal de Rayleigh*.

Cuando se tiene línea de vista, la variable aleatoria es una variable con *distribución Rice*. Los *desvanecimientos lentos* por lo general se modelan con una variable aleatoria con *distribución Log-normal* **(Stüber & Stèuber, 2012)**. El siguiente **código 7.3** genera un *canal Rayleigh* (ver **Figura 77**) y un *canal Log-Normal* (ver **Figura 78**).

Código 7.3.

Código

```
sigma=1; media=0; N=1000;
x1=randn(N,1)*sigma + media;
x2=randn(N,1)*sigma + media;
x3=randn(N,1)*sigma + media;
Z=sqrt(x1.^2 + x2.^2); % Calculo de V.A. Rayleigh
W=exp(x3); % Calculo de V.A. Log-Normal
plot(10*log10(Z)) % Grafica del canal Rayleigh en dB
xlabel('x') ylabel('Potencia (dB)')
grid on
figure(2)
plot(10*log10(W)) % Grafica del canal Log-Normal en dB
xlabel('x') ylabel('Potencia (dB)')
grid on
```

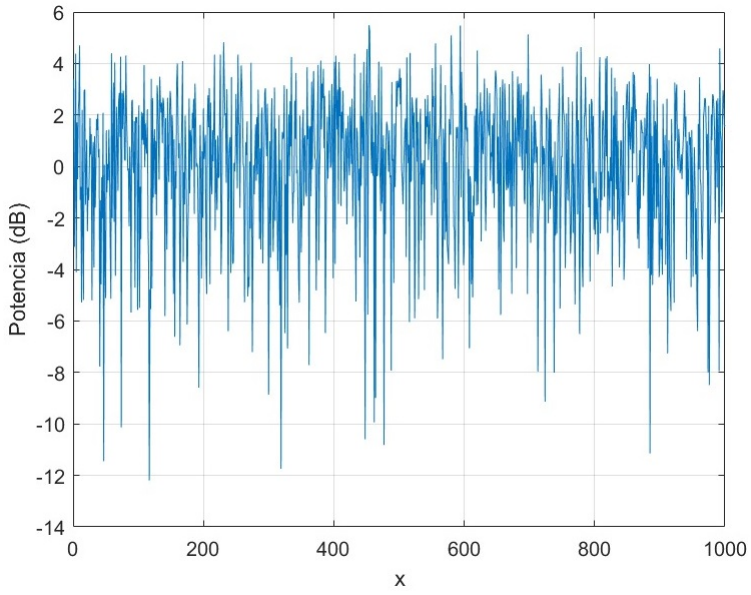


Figura 77: Canal Rayleigh.

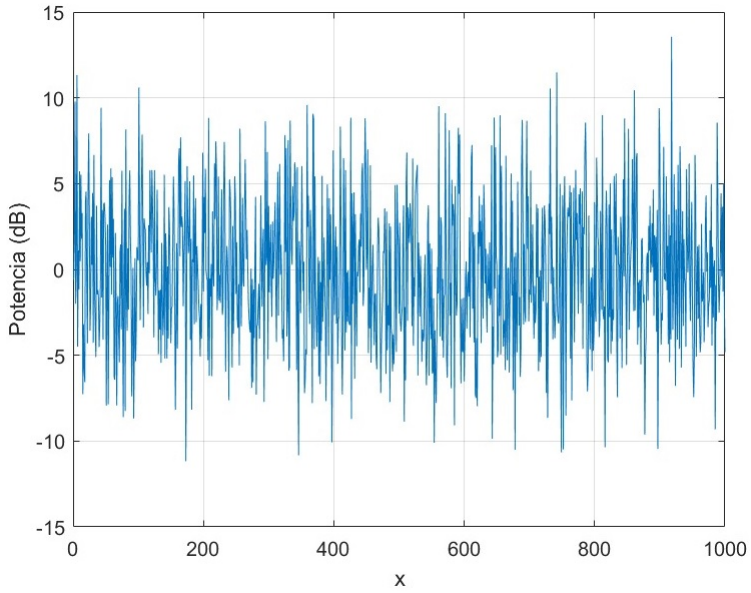


Figura 78: Canal Log-normal.

El calculo de los canales con desvanecimientos puede servir para calcular la ganancia del canal, este análisis se utiliza en el Acceso Multiple No Ortogonal (**NOMA**, por sus siglas en inglés) en el dominio de la potencia, que es una de las propuestas paras las redes celulares 5G. Estos resultados sirven para poder mitigar la interferencia de los usuarios que comparten el mismo recurso, esto se logra utilizando la Cancelación de Interferencias Sucesivas (**SIC**, por sus siglas en inglés).

Una vez obtenidos los canales, lo que procede es generar la *función de densidad de probabilidad* (**PDF**) de las variables aleatorias y graficarlas sobrepuestas con las fórmulas teóricas para observar la aproximación.

La **PDF** de la variable aleatoria *Rayleigh* está dada por la **ecuación 45**, **(Papoulis & Saunders, 2002)**.

$$f(r) = \frac{r}{2\sigma^2} e^{-r^2/2\sigma^2}; \quad r \geq 0; \tag{45}$$

La **PDF** de la variable aleatoria *Log-normal* está dada por la **ecuación 46**, **(Papoulis & Saunders, 2002)**.

$$g(x) = \frac{1}{x\sqrt{2\pi\sigma^2}} e^{-\left(\frac{(\ln(x)-\mu)^2}{2\sigma^2}\right)} \tag{46}$$

Aquí, x representa una variable aleatoria normal. El siguiente **código 7.4** es una función *"M"*, la cual utilizaremos para generar y graficar la **PDF** para todo tipo de variable aleatoria.

Código 7.4.

Código

```
function[]= pdfva(variable, bines, total) % parametros
    que recibe la funcion
[rr,u]=hist(variable,bines); % Calculo de histograma
delta=u(2)-u(1); % Calculo de intervalo
r2=rr/(total*delta); % Normalizar valor del eje Y
bar(u,r2); % Graficar la funcion
colormap([1 1 1]); % Colocar en blanco el fondo de cada
    barra
hold on
```

El siguiente **código 7.5** calcula y muestra la gráfica de la **PDF** de los desvanecimientos, tanto de manera teórica como de forma simulada. Este código produce las gráficas de la **Figura 79** y **80**.

Código 7.5.

Código

```
sigma=1;media=0;N=5000;
x1=randn(N,1)*sigma + media; % Creacion de N variables
    aleatorias normales con media cero y desviacion
    estandar 1
x2=randn(N,1)*sigma + media;
x3=randn(N,1)*sigma + media;
Z=sqrt(x1.^2 + x2.^2); % Calculo de V.A. Rayleigh
W=exp(x3); % Calculo de V.A. Log-Normal
y=length(Z); % Tama no de mi variable
bin=30;
pdfva(Z, bin, y) % Llamado de la funcion
u2=max(Z);u1=min(Z);
m=(u2-u1)/y; % Calculo de intervalo
j=u1:m:u2;
f=(j/sigma.^2).*exp(-j.^2/(2*sigma.^2));
plot(j,f, 'r'); % Grafica de funcion teorica
xlabel('x') ylabel('pdf (x)')
legend('EMPIRICO', 'TEORICO');
```

Código

```
%% Cont...
figure(2)
y2=length(W); % Tama no de mi variable
pdfva(W, bin, y2)% Llamado de la funcin
u4=max(W);
u3=min(W);
m2=(u4-u3)/y;
k=u3:m2:u4;
f2=(k/sigma.^2).*exp(-k.^2/(2*sigma.^2));
plot(k,f2,'r'); % Grafica de funcion teorica
xlabel('x')
ylabel('pdf (x)')
legend('EMPIRICO','TEORICO');
```

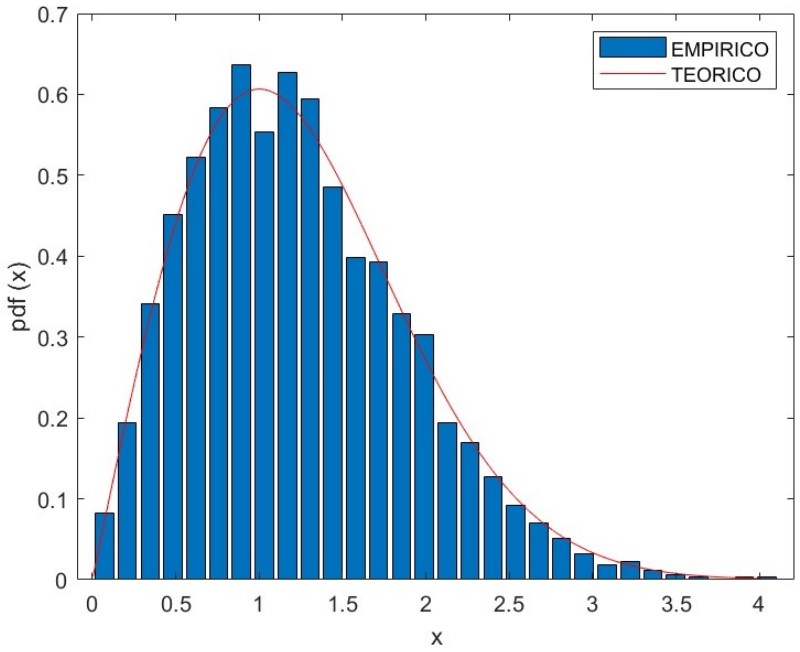


Figura 79: PDF Rayleigh.

5.3. Diseño de radioenlace

Un *radioenlace* se entiende como un enlace de ondas electromagnéticas en la banda de señales de radio. Este enlace se presenta como una conexión ya sea entre terminales fijas o móviles, donde el canal de comunicación es a través del espacio libre.

Cuando se habla de radiocomunicaciones, entendemos que los radioenlaces generalmente se diseñan sobre las bandas de frecuencia del orden de las *microondas*, es decir, a partir de los 300 MHz y hasta los 300 GHz. En esta banda de frecuencia, la longitud de onda (λ) es muy pequeña, y como consecuencia de ello, éstas son muy sensibles a los *fenómenos meteorológicos* como la lluvia y la nieve **(Salema, 2002)**.

Existen condiciones que se deben cumplir para un radioenlace; en primer lugar, es vital que la ruta directa entre las dos antenas (la ruta de la línea de vista) esté libre de obstrucciones. Sin

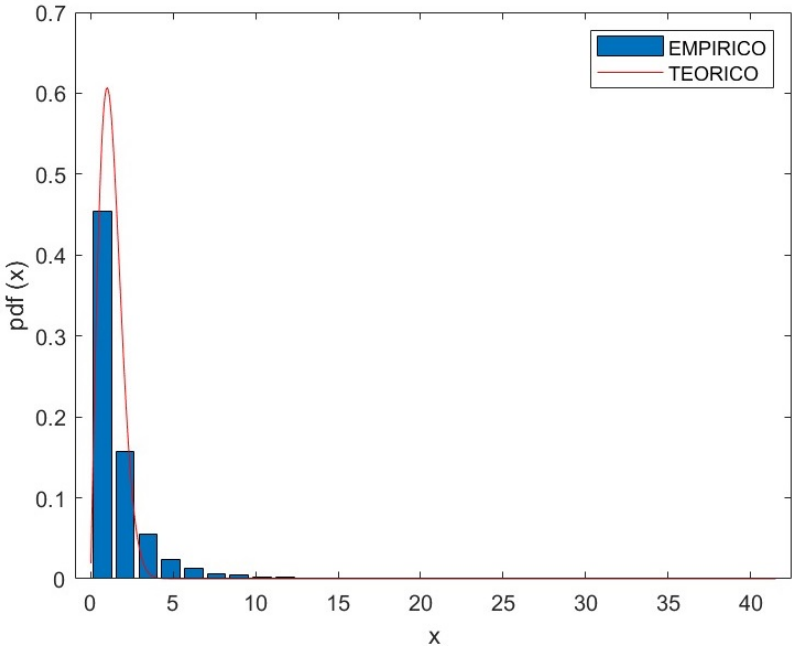


Figura 80: PDF Log-normal.

embargo, esto en sí mismo no es suficiente, sino que es muy conveniente que no haya obstáculos cerca de la ruta de la línea de visión, ya que podrían provocar reflexiones y pérdidas en la recepción. La **Figura 81** muestra un camino simple de un radioenlace, la elipse punteada define una región conocida como la 1ra. *zona de Fresnel* **(Salema, 2002)**.

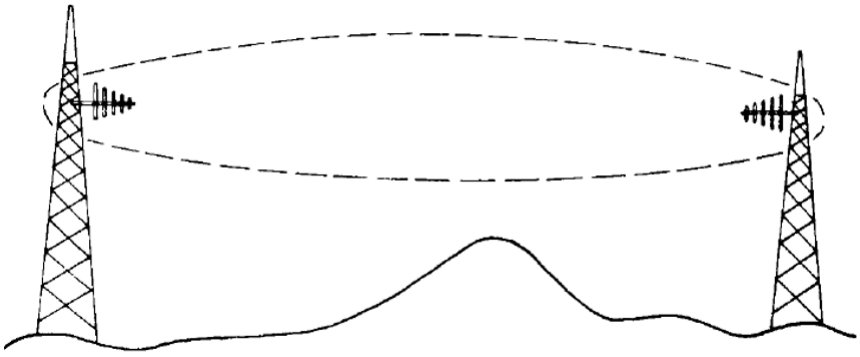


Figura 81: Radioenlace punto a punto.

Cuando se calculan las *pérdidas en la transmisión*, también se deben considerar los efectos como son la *pérdida en el espacio libre*, *atenuación* y *dispersión*. Resultado de este cúmulo de pérdidas, la potencia (P_R) de la señal se ve disminuida en el receptor, y coloquialmente se le conoce como *pérdidas en el espacio libre*. A manera de conclusión, este tipo de pérdidas se ocasionan porque la energía de la señal que se está radiando se expande en función de la distancia desde el transmisor (**Tx**), es decir, entre más alejado esté el **Tx** del receptor (**Rx**) más pequeña es la señal recibida.

En todo el trayecto del radioenlace, la potencia de radiofrecuencia que se le está suministrando al transmisor se dispersa, quiere decir que, una parte de la potencia de **RF** llega directamente a la antena receptora (**Rx**), como se muestra en el trayecto *A* de la **Figura 82**, mientras que otra se refleja en la tierra (trayecto *B* y *C*).

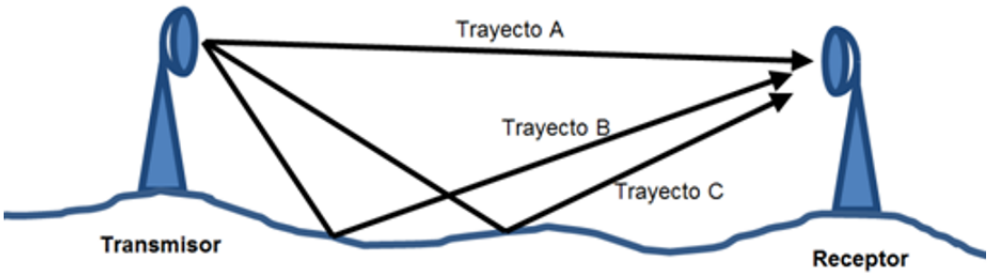


Figura 82: Efecto multitrayectoria.

Cuando en un radioenlace se calculan las pérdida en el *espacio libre*, lo que se analiza también es una estimación aproximada de la viabilidad del enlace. En este sentido, se puede evaluar solamente las *pérdidas en el espacio libre*, pero si se requiere una evaluación exacta se deben considerar también las pérdidas que se generan en la atmósfera.

Al trabajar radioenlaces en la banda de las microondas se vuelve complejo, ya que a estas frecuencias, las ondas electromagnéticas pareciera que se van ensanchando a medida que se van propagando. Las ondas electromagnéticas en esta banda de frecuencia tienen longitudes de onda de pocos centímetros, por lo tanto, sus haces son anchos y necesitan más espacio. Para estos radioenlaces, la línea de vista necesaria para lograr una conexión óptima desde el punto *A* hasta el punto *B* de la **Figura 82**, es más que una simple línea delgada, en sí es un volumen en forma de elipsoide. El ancho de este volumen se puede calcular a través del concepto de *zonas de Fresnel*. La fórmula generalizada se presenta en la **ecuación 47**.

$$R_n = \sqrt{\frac{n\lambda d_1 d_2}{d_1 + d_2}} \tag{47}$$

5.3.1. Pérdida en el espacio libre

Para calcular las pérdidas por trayectoria, se deben tomar en cuenta los efectos ocasionados por la *pérdida en el espacio libre*, las diferentes *atenuaciones* y *dispersiones*. Dependiendo de la longitud de la trayectoria, se deben estudiar los efectos de propagación a través de la atmósfera y las características de elevación del terreno **(Salema, 2002)**.

Como ya se ha mencionado, la potencia de la señal se ve disminuida a lo largo del trayecto por la dispersión del frente de onda ocasionada por la distancia, a este efecto se le conoce comúnmente como *pérdida en el espacio libre*. Se observa que entre más lejanos estén las dos terminales (T_x y R_x), menor será la potencia medida en el receptor. Esta *"pérdida"* (dispersión) ocurre debido a que la onda electromagnética radiada se expande en función de la distancia desde el transmisor (T_x).

La pérdida o atenuación de energía en la trayectoria entre el transmisor (T_x) y el receptor (R_x) es simplemente la relación de la potencia recibida y la potencia transmitida. La fórmula para calcular dicha pérdida se conoce como la *ecuación de Friis*, como se muestra en la **ecuación 48** **(Salema, 2002)**.

$$L_{fs} = \frac{P_R}{P_T} = G_T G_R (\frac{\lambda}{4\pi r})^2 \tag{48}$$

donde

P_R	Es la potencia recibida [W].
P_T	Es la potencia transmitida [W].
G_T	Es la ganancia de la antena Tx.
G_R	Es la ganancia de la antena Rx.
λ	Es la longitud de onda [m].
$(4\pi r)^2$	Es el área de superficie de la esfera en el radio r $[m^2]$.

La **ecuación 48** es comúnmente utilizada en decibeles que está dada por la siguiente **ecuación 49**.

$$L_{fs}(dB) = 32.44 - 10\log G_T - 10\log G_R + 20\log f + 20\log d \tag{49}$$

donde

f	Es la frecuencia de operación [MHz].
d	Es la distancia [km].

5.3.2. Cálculo de zonas de Fresnel

La trayectoria de un radioenlace puede que no sea una línea recta, sino que presente cambios en su ruta debido a la refracción bajo la presencia de obstáculos y de las condiciones atmosféricas del terreno, generándose multitrayectorias con interferencias *constructivas* y *destructivas*, es por ello que para evaluar la viabilidad de un radioenlace es necesario recurrir al concepto de las *zonas de Fresnel* **(Salema, 2002)**.

La *zona de Fresnel* es aquella región del espacio en forma de elipsoide por la que viajan las ondas entre dos puntos, desde un transmisor hasta un receptor. La teoría de zona de Fresnel evalúa la ruta de propagación del enlace y al espacio (volumen) alrededor del eje directo del mismo, los puntos que en teoría no se encuentran en la línea directa de propagación también radian potencia y contribuyen a que la señal llegue hasta el receptor.

En la **Figura 83** se muestra que existe más de una zona de Fresnel alrededor de la línea directa entre un transmisor y un receptor. En la zona de Fresnel existen puntos donde se satisface la **ecuación 50**.

$$d_1 + d_2 + \frac{n\lambda}{2} = a + b \tag{50}$$

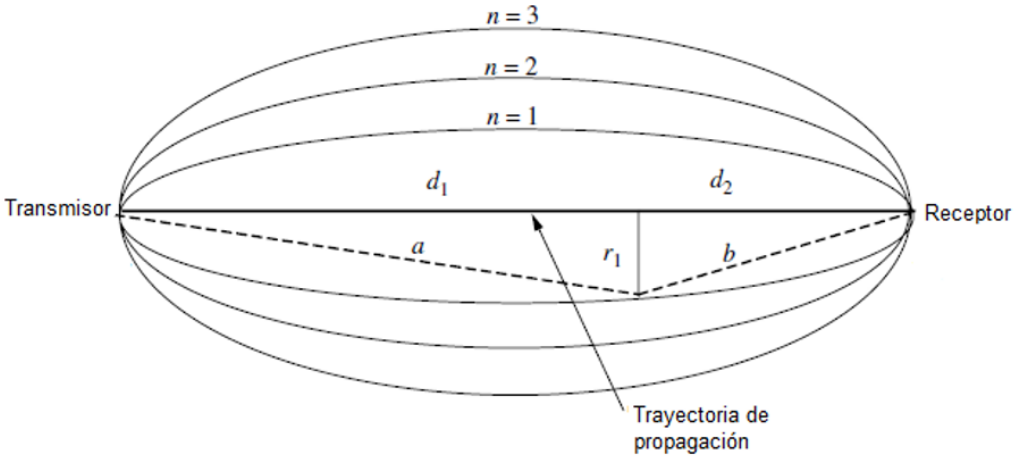


Figura 83: Zonas de Fresnel.

donde

n Es el número de la zona de Fresnel a calcular (número entero).

Si la longitud de la trayectoria es mayor que el radio del volumen de la zona de Fresnel r_n, éste puede aproximarse con la **ecuación 51 (Salema, 2002)**.

$$r_n = \sqrt{\frac{n\lambda d_1 d_2}{d_1 + d_2}} \tag{51}$$

Nota Los valores de r_n, d_1 y d_2 están dados en metros $[m]$.

En los sistemas inalámbricos de banda ancha que operan a frecuencias de microondas, el radio del volumen de la zona de Fresnel se calcula mediante la **ecuación 52**, **(Salema, 2002)**.

$$r_n = \sqrt{\frac{n d_1 d_2}{f * (d_1 + d_2)}} \tag{52}$$

donde

r_n Es el radio del volumen de la zona de Fresnel [m].
n Es la zona de Fresnel bajo cálculo.
f Es la frecuencia de operación [GHz].
d_1, d_2 Son las distancias entre los puntos del enlace [m].

La primera zona de Fresnel se conceptualiza como la región en la cual se transmite la potencia significativa, de esta manera si la primera zona de Fresnel se ve obstruida o bloqueada, la potencia que llegará al receptor estará disminuida.

5.3.3. Cálculo de la altura de las antenas en un radioenlace

En el diseño de sistemas de radioenlaces, mantener la zona de Fresnel despejada es un punto importante, puesto que para tener un trayecto libre de obstáculos, se debe considerar la ruta en todas sus direcciones, la mayoría de los casos, si no es que siempre, las obstrucciones se encuentran por debajo de la ruta del enlace, ocasionadas por objetos fijos propios del relieve, copas de los árboles, edificios, etc. La **Figura 84** muestra un enlace cuya zona de Fresnel está bloqueada parcialmente por la cima de un monte, aunque la línea de vista no se encuentra obstruida.

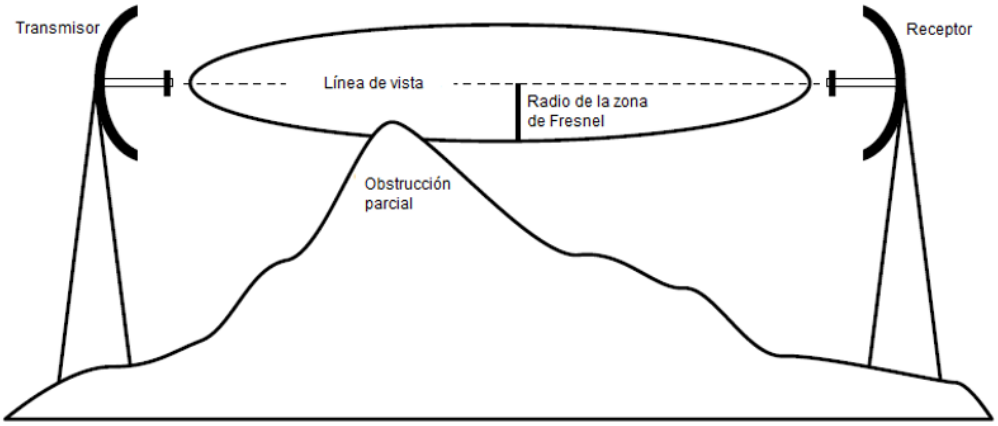


Figura 84: Zona de Fresnel parcialmente bloqueada.

Un criterio general para establecer radioenlaces efectivos es que la primera zona de Fresnel debe tener un radio del volumen de al menos el $60\,\%$ libre de su magnitud total en el punto del obstáculo de mayor altura, para lograr esto, las torres de las antenas deben tener la altura necesaria que satisfaga esta consideración.

Para el cálculo de las alturas de las torres se pueden tomar varios puntos de partida, uno de ellos, es el siguiente:

- Se calcula con $n = 1$ la 1ra. zona de Fresnel entre los puntos donde se colocarán las torres mediante la **ecuación 52**.
- Se identifica la obstrucción más significativa del perfil del radioenlace, el radio del volumen de la zona de Fresnel en este punto corresponde a un valor r_n.
- Se determina la altura entre el punto más elevado de la obstrucción y el punto de la 1ra. zona de Fresnel. Este valor es la altura de las torres y obviamente es la elevación a la cual tienen que estar colocadas las antenas en las mismas para librar el $100\,\%$ de la primera zona de Fresnel, sin embargo, como se mencionó anteriormente, basta que esta zona esté despejada en un $60\,\%$ en relación a su radio $(0.6r_n)$ por lo que la altura de las torres se puede estimar desde el punto $0.6r_n$ hasta el punto más alto del obstáculo.

La altura mínima de las torres de las antenas es de $15m$ y una máxima de $120m$, si el valor calculado del tamaño de las torres excede este intervalo, se procede a la implementación de

antenas repetidoras.

5.3.4. Uso de antenas repetidoras

Las *antenas repetidoras* tienen la función de ser nodos configurados para transmitir el tráfico de las ondas destinadas a otro nodo, ya sea otra repetidora o al receptor de destino, y son utilizadas para evitar obstáculos a lo largo de la trayectoria de un enlace. La **Figura 85** muestra una antena repetidora que reenvía el tráfico de datos por el espacio libre a otras antenas, aquí se observa que las antenas donde se inicia la transmisión no tienen línea de vista con las antenas terminales.

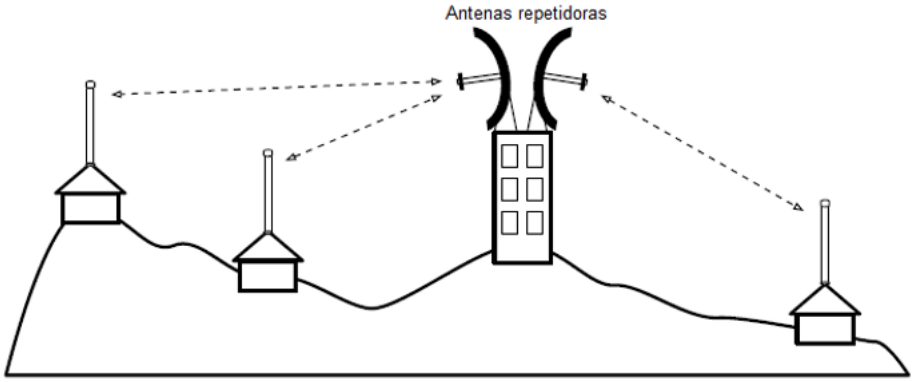


Figura 85: Antena repetidora.

Las antenas repetidoras se utilizan siempre y cuando la altura calculada de las torres sea mayor a $120m$, colocadas en el punto de mayor obstrucción garantizan un enlace de radio viable, así mismo, se recurre a ellas cuando la distancia total del enlace de radio supera los $64km$.

5.3.5. Método de diseño

Cuando se desea diseñar un enlace de radiocomunicaciones se requiere conocer la altura de la superficie terrestre (altitud), desde el transmisor, que sería el punto inicial ($Km\ 0$) hasta el receptor, que sería el punto final ($Km\ n$). Como se debe conocer las altitudes en cada cierta distancia, lo recomendable es utilizar un mapa topográfico.

Una vez que sabemos las ubicaciones del transmisor y receptor, trazamos el perfil, el cual se obtiene marcando sobre el mapa una línea recta entre punto y punto. La línea recta funge como línea de vista entre nuestro **Tx** y **Rx**, desde el punto inicial hasta el punto final del enlace. Sobre esta línea recta en el mapa se marcan todos los puntos discretos, los cuales son tomados a $\leq 250m$ entre punto y punto. En la **Figura 86** se representa una línea de vista trazada sobre un mapa topográfico **(Salema, 2002)**.

Una vez que se hayan marcado ya todos los puntos sobre la línea de vista en el mapa, se continúa con obtener la altura superficial de cada uno de estas marcas. Conociendo ubicación y altura de cada marca, podemos ahora generar una gráfica que represente el relieve de toda la línea de vista. Como se observa en la **Figura 87**, los mapas topográficos cuentan con curvas de nivel las cuales están acotadas en metros y, además, se puede observar que las líneas A y B representan las curvas de nivel ordinarias y están acotadas cada $20m$.

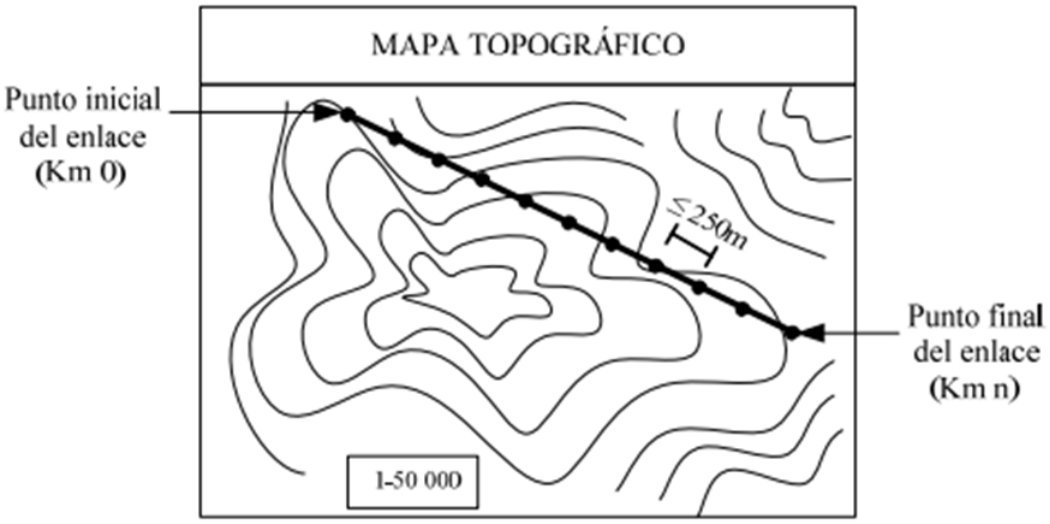


Figura 86: Perfil topográfico.

Cada una de estas curvas representan la altura del relieve sobre el nivel del mar, para este ejemplo, denotaremos como C_i a estas alturas . Entonces si el punto 1 cae sobre la curva de nivel A se le asignará una altura de $5000m$ ($C_i = 5000m$), si caen en la línea X_1 se le asignará una altura de $5020m$ ($C_i = 5020m$).

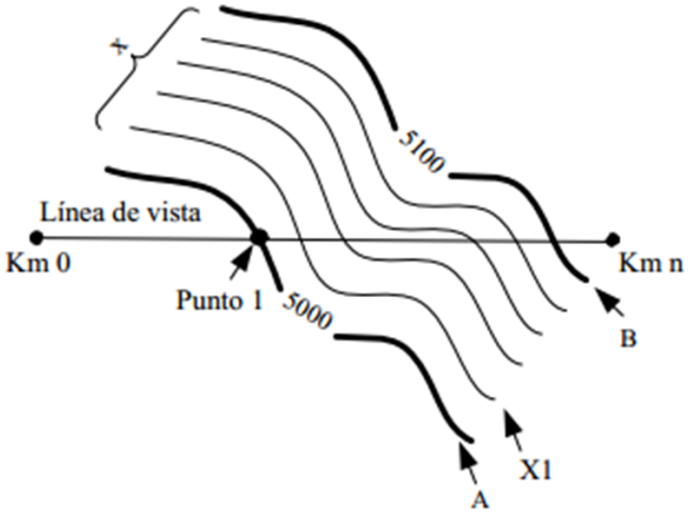


Figura 87: Ejemplo de curvas de nivel en un mapa topográfico.

Generando la base de datos con todas las alturas de los puntos marcados podemos realizar la gráfica C_i con respecto a la distancia X_1, donde X_1 son los puntos de referencia, o bien la distancia que hay desde el $Km\ 0$ y el punto de referencia que se requiere graficar como se muestra en la **Figura 88**.

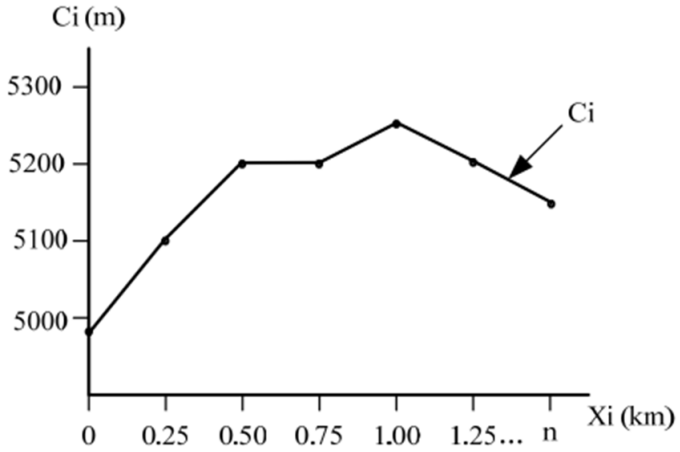


Figura 88: Gráfica de las alturas del relieve (Ci).

Se procede a calcular la curvatura de la tierra utilizando la **ecuación 53**.

$$f_i = \frac{X_i * (X_n - X_i)}{2KR_0} * 1000 \tag{53}$$

donde

f_i	Representa la curvatura de la Tierra en metros [m].
Xi	Es la distancia del punto inicial al punto final, es decir distancia desde el transmisor T_x hasta el receptor (R_x) en kilómetros [Km].
Xn	Es la distancia total del enlace.
K	Representa el factor de corrección de la Tierra.
R_0	Es el radio de la Tierra dado en kilómetros [Km]. $R_0 = 6370Km$.

La curvatura de la Tierra representada por f_i se calcula para obtener la altura real de la superficie terrestre. Ya que se realiza este cálculo, asignaremos el nombre de Z_i, que en resumen sería la suma de C_i y f_i, resultando en $Z_i = C_i + f_i$.

Lo que prosigue es el cálculo de las zonas de Fresnel, y aquí, una vez que se ha graficado Z_i, identificamos el punto en dónde existe mayor obstrucción. Aquí podemmos distinguir la altura que existe entre el punto de la obstrucción (Z_i) hasta la zona inferior del radio del volumen de la zona de Fresnel. Conociendo el valor de esta diferencia, entre la altura del relieve y el límite de la elipsoide de la zona de Fresnel, podemos identificar la altura que deberán cumplir las antenas, es decir, se deberá elevar a las antenas esta altura para que el limite inferior de la elipsoide quede completamente libre de la obstrucción.

Debemos considerar que la altura mínima que debe cumplir una torre de un radioenlce es de $15\ m$, por lo que si la obstrucción que se distingue entre el relieve y la elipsoide inferior de Fresnel es menor a esta altura, la torre siempre será de $15\ m$. El siguiente **código 7.6** calcula y genera las gráficas para un ejemplo de un radio enlace (ver **Figura 89-93**).

Código 7.6.

Código

```
% Distancia del enlace
Xi=0:0.250:8.75;
% Altura del terreno
Ci=
    [2942,2940,2930,2930,2935,2945,2940,2920,2910...

2905,2890,2880,2880,2870,2865,2865,2865,2870,2880,2898...

2910,2920,2930,2930,2930,2922,2895,2865,2865,2865,2870...

2890,2900,2910,2900,2920];
x1=0;
Xn=8.75;
K=4/3;
Ro=6370;
% Curvatura de la tierra
fi=((Xi.*(Xn-Xi))/(2*K*Ro))*1000;
Zi=Ci+fi;
% Gráficas
plot (Xi,fi)
xlabel('Distancia X_i [Km]')
ylabel('Curvatura de la Tierra [m]')
grid on
figure (2)
plot(Xi,Ci,'-ro')
hold on
plot(Xi,Zi,'-k*')
grid on
xlabel('Distancia X_i [Km]') ylabel('Alturas [
    m]') legend ('Ci', 'Zi')
figure(3)
y1=2942; % Altura en el x1
y2=2920; % Altura en el Xn
y=(((y2-y1)/(Xn-x1)).*(Xi))+y1;
plot(Xi,Zi,'-k*')
hold on
plot(Xi,y)
title('Línea de vista entre Tx y Rx') xlabel('
    X_i [Km]') ylabel('Z_i [m]')
grid on;
f=1.8*10^9; c=3*10^8; lamda=c/f;
R1=sqrt((lamda*(Xi*1000).*((Xn-Xi)*1000))/(Xn
    *1000)); % Rn de la elipsoide, para este
    caso especial n=1
elip_inf=y-R1;
elip_sup=y+R1;
figure (4)
plot(Xi,Zi,'-k*')
hold on
plot(Xi,y)
% Grafica de la elipsoide inferior
plot(Xi,elip_inf,'--*')
% Grafica de la elipsoide superior
plot(Xi,elip_sup,'--*')
title('Elipsoide de Fresnel') xlabel('X_i [Km]
    ') ylabel('Z_i [m]')
grid on
```

Código

```
%% cont...
figure (5)
for k=1:length(y)
 if (Zi(k)>y(k))
 dif(k)=abs(Zi(k)-elip_inf(k));
 else
 dif(k)=0;
 end
end
plot(Xi,Zi,'-k*')
hold on
plot(Xi,y+max(dif))
plot(Xi,elip_inf+max(dif),'--*')
plot(Xi,elip_sup+max(dif),'--*')
title('Elevación de la línea de vista')
xlabel('X_i [Km]')
ylabel('Z_i [m]')
grid on;
```

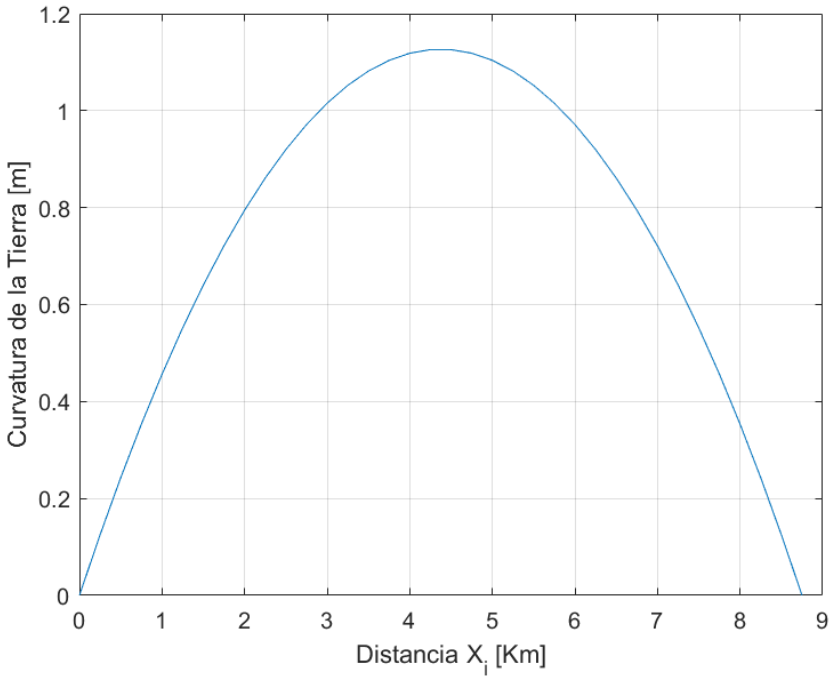


Figura 89: Curvatura de la Tierra.

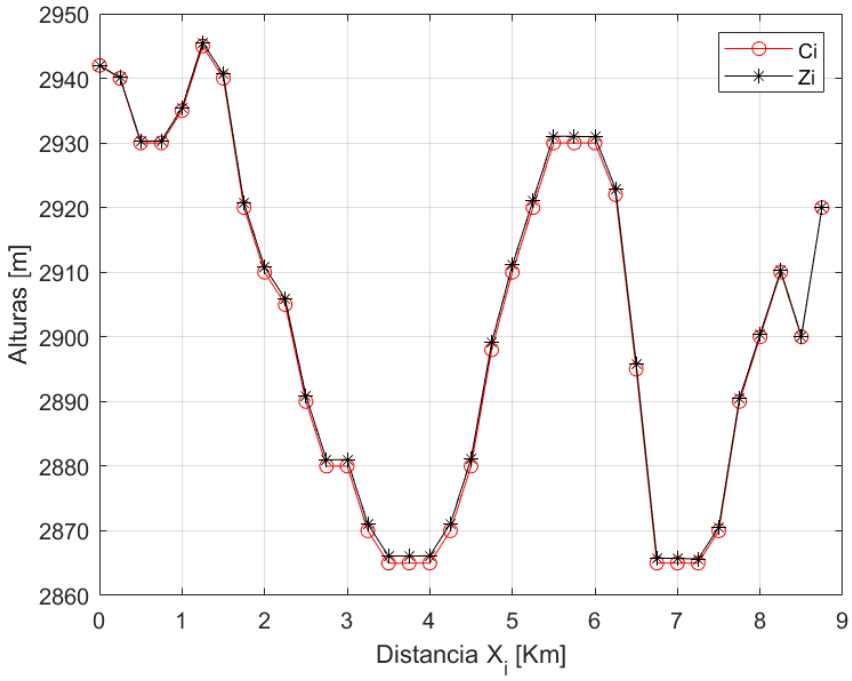


Figura 90: Gráfica de Ci y Zi.

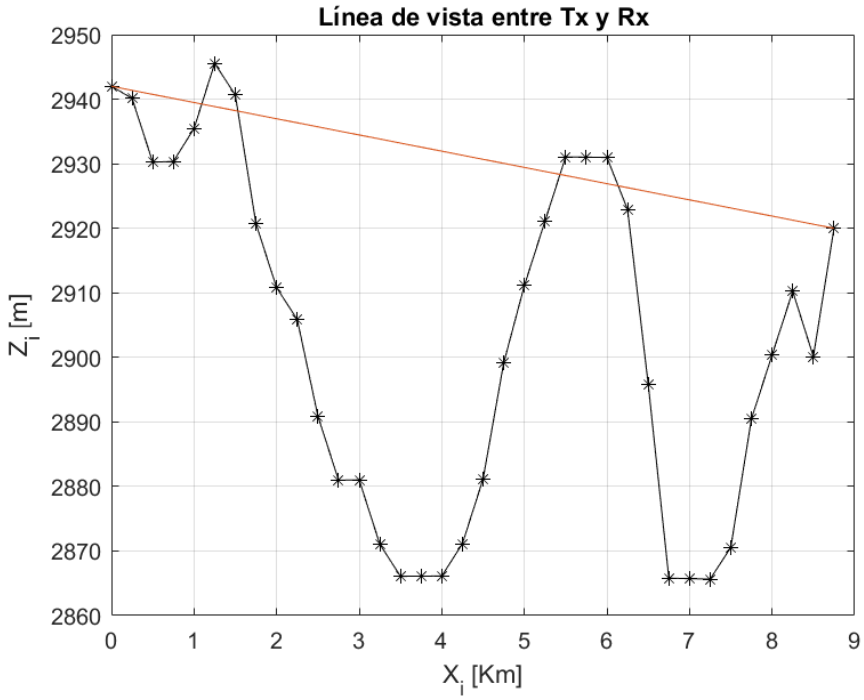


Figura 91: Línea de vista.

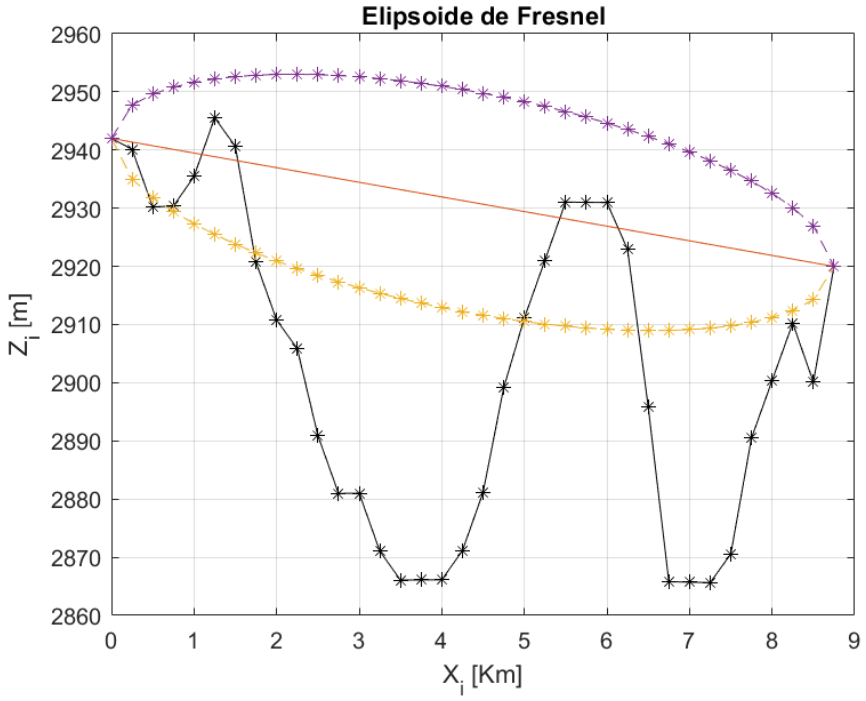


Figura 92: Zona de Fresnel.

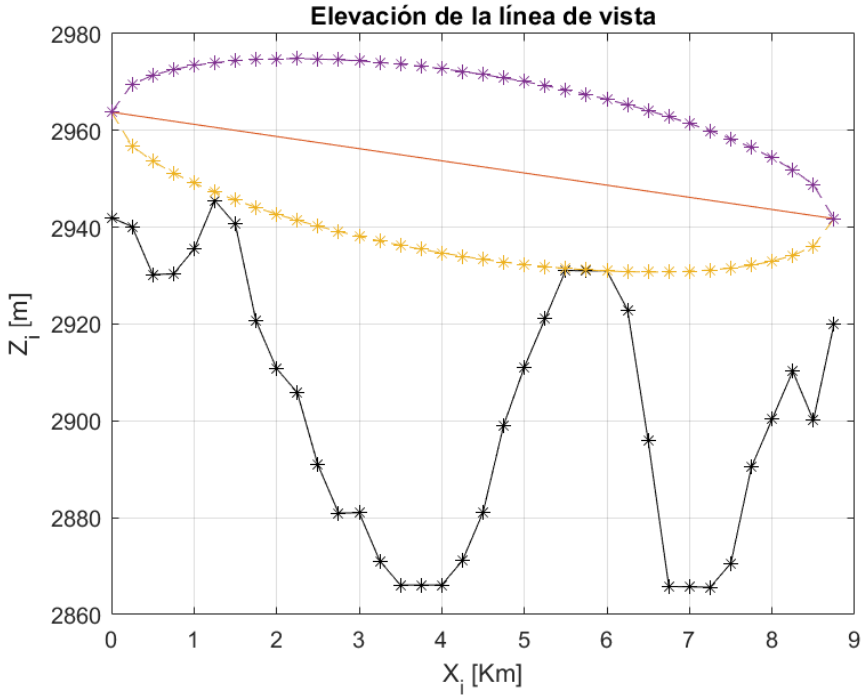


Figura 93: Radioenlace.

6. Aplicaciones en Modulación y Telefonía

En este capítulo se tratarán ejemplos sobre modulación *QAM* en donde se mostrará la constelación del modulador y se calculará la probabilidad de bit erróneo. También se realizará un ejemplo de redes jerárquicas, que son utilizadas en la telefonía, en donde se diseñará la red y se calculará la probabilidad de bloqueo en la red.

6.1. 4QAM

La *modulación* consiste en transportar la señal de información en una portadora de mayor frecuencia. Una de las modulaciones más utilizadas es la *modulación de amplitud en cuadratura* (QAM, por sus siglas en inglés), y esta puede realizar la modulación de la portadora tanto en amplitud como en fase. Por ejemplo, tenemos que la **Figura 94** muestra una constelación de una modulación *4QAM* **(Haykin, 2002.)**.

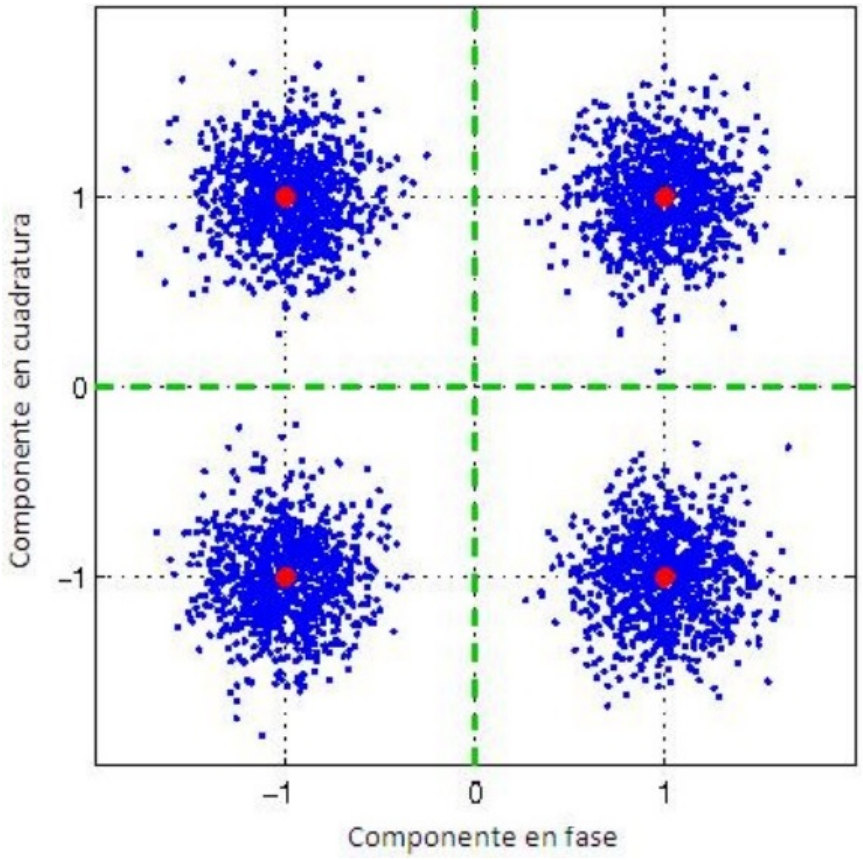


Figura 94: Constelación 4QAM.

El **código 8.1** produce gráfica de la constelación *4QAM* para un canal con ruido Gaussiano (ver **Figura 95**).

Código 8.1.

Código

```
% Generación de componentes principales
N=1000;
x1=3+3*j;
x2=-3+3*j;
x3=-3-3*j;
x4=3-3*j;
% Grafica de componentes
plot(real(x1),imag(x1), 'kp')
hold on
plot(real(x2),imag(x2), 'kp')
plot(real(x3),imag(x3), 'kp')
plot(real(x4),imag(x4), 'kp')
axis([-6 6 -6 6]);
% Suma de ruido Gaussiano a las componentes
r1=x1+randn(1,N)+j.*randn(1,N);
r2=x2+randn(1,N)+j.*randn(1,N);
r3=x3+randn(1,N)+j.*randn(1,N);
r4=x4+randn(1,N)+j.*randn(1,N);
% Grafica de componente con canal Gaussiano
plot(real(r1),imag(r1), 'ro')
plot(real(r2),imag(r2), 'bo')
plot(real(r3),imag(r3), 'go')
plot(real(r4),imag(r4), 'yo')
xlabel('Componente en fase')
ylabel('Componente en cuadratura')
grid on
```

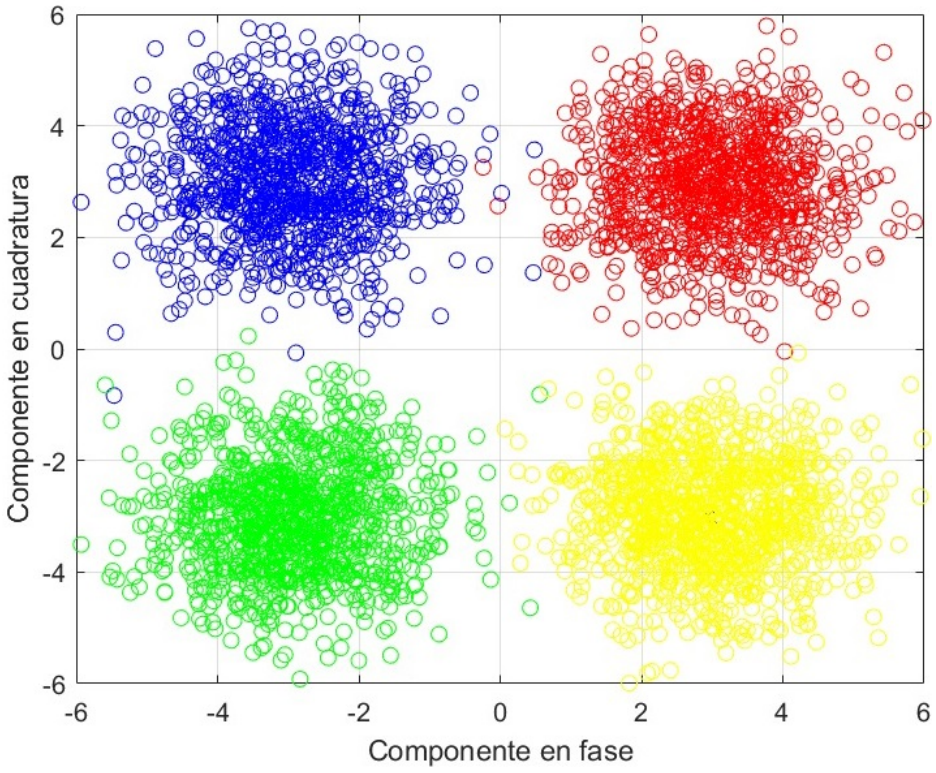


Figura 95: Constelación 4QAM.

Uno de los parámetros utilizados para la medición de la calidad de servicio (QoS, por sus siglas en inglés) es la probabilidad de bit erróneo (BER, por sus siglas en inglés), y el **código 8.2** calcula y muestra la gráfica de esta probabilidad (ver **Figura 96**).

Código 8.2.

Código

```
clc
clear
b=1; % Bandera para contar elementos
N=5000; % Numero de V.A. Gaussianas
for sir=0.1:0.1:2
% Generacion de componentes principales
x1=sir+(sir)*j;
x2=-sir+(sir)*j;
x3=-sir-(sir)*j;
x4=sir-(sir)*j;
% Suma de ruido Gaussiano a las componentes
r1=x1+randn(1,N)+j.*randn(1,N);
r2=x2+randn(1,N)+j.*randn(1,N);
r3=x3+randn(1,N)+j.*randn(1,N);
r4=x4+randn(1,N)+j.*randn(1,N);
conta=0; % Contador de bits erroneos
% Busqueda de bit erroneos
for k=1:N
    % Primer Cuadrante
    d1=sqrt((real(r1(1,k))-real(x1)).^2+((imag(r1(1,k))
       -imag(x1))).^2);
    d2=sqrt((real(r2(1,k))-real(x1)).^2+((imag(r2(1,k))
       -imag(x1))).^2);
    d3=sqrt((real(r3(1,k))-real(x1)).^2+((imag(r3(1,k))
       -imag(x1))).^2);
    d4=sqrt((real(r4(1,k))-real(x1)).^2+((imag(r4(1,k))
       -imag(x1))).^2);
    if (d2<d1 || d3<d1 || d4<d1)
        conta=conta+1;
    end
    % Segundo Cuadrante
    d1=sqrt((real(r1(1,k))-real(x2)).^2+((imag(r1(1,k))
       -imag(x2))).^2);
    d2=sqrt((real(r2(1,k))-real(x2)).^2+((imag(r2(1,k))
       -imag(x2))).^2);
    d3=sqrt((real(r3(1,k))-real(x2)).^2+((imag(r3(1,k))
       -imag(x2))).^2);
    d4=sqrt((real(r4(1,k))-real(x2)).^2+((imag(r4(1,k))
       -imag(x2))).^2);
    if (d1<d2 || d3<d2 || d4<d2)
        conta=conta+1;
    end
```

Código

```
%% Cont...
    % Tercer Cuadrante
    d1=sqrt((real(r1(1,k))-real(x3)).^2+((imag(r1(1,k))
       -imag(x3))).^2);
    d2=sqrt((real(r2(1,k))-real(x3)).^2+((imag(r2(1,k))
       -imag(x3))).^2);
    d3=sqrt((real(r3(1,k))-real(x3)).^2+((imag(r3(1,k))
       -imag(x3))).^2);
    d4=sqrt((real(r4(1,k))-real(x3)).^2+((imag(r4(1,k))
       -imag(x3))).^2);
    if (d1<d3 || d2<d3 || d4<d3)
        conta=conta+1;
    end
    % Cuarto Cuadrante
    d1=sqrt((real(r1(1,k))-real(x4)).^2+((imag(r1(1,k))
       -imag(x4))).^2);
    d2=sqrt((real(r2(1,k))-real(x4)).^2+((imag(r2(1,k))
       -imag(x4))).^2);
    d3=sqrt((real(r3(1,k))-real(x4)).^2+((imag(r3(1,k))
       -imag(x4))).^2);
    d4=sqrt((real(r4(1,k))-real(x4)).^2+((imag(r4(1,k))
       -imag(x4))).^2);
    if (d1<d4 || d2<d4 || d3<d4)
        conta=conta+1;
    end
end
Pbe(b)=conta/(4*N); % Calculo de probabilidad de bit
   erroneo
b=b+1;
end
% Generar grafica
sir=0.1:0.1:2;
semilogy(sir,Pbe)
xlabel('Eb/N0')
ylabel('Probabilidad de bit erroneo')
grid on
```

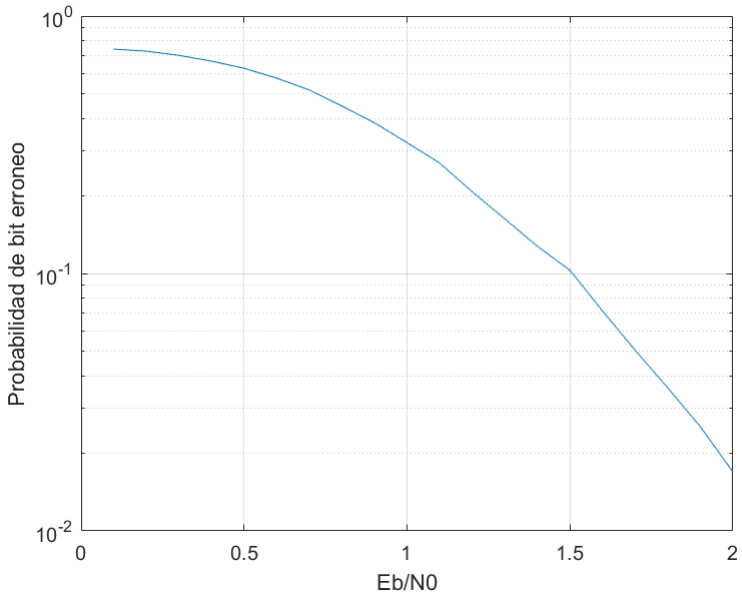


Figura 96: Probabilidad de error.

6.2. MPSK

Otra forma de calcular la probabilidad de bit erróneo es utilizando los comandos de MATLAB. El **código 8.3** calcula la razón de símbolo erróneo para un sistema *MPSK* con ruido Gaussiano (ver **Figura 97**).

Código 8.3.

Código

```
%% Cont...
    % Tercer Cuadrante
    d1=sqrt((real(r1(1,k))-real(x3)).^2+((imag(r1(1,k))
       -imag(x3))).^2);
    d2=sqrt((real(r2(1,k))-real(x3)).^2+((imag(r2(1,k))
       -imag(x3))).^2);
    d3=sqrt((real(r3(1,k))-real(x3)).^2+((imag(r3(1,k))
       -imag(x3))).^2);
    d4=sqrt((real(r4(1,k))-real(x3)).^2+((imag(r4(1,k))
       -imag(x3))).^2);
    if (d1<d3 || d2<d3 || d4<d3)
        conta=conta+1;
    end
    % Cuarto Cuadrante
    d1=sqrt((real(r1(1,k))-real(x4)).^2+((imag(r1(1,k))
       -imag(x4))).^2);
    d2=sqrt((real(r2(1,k))-real(x4)).^2+((imag(r2(1,k))
       -imag(x4))).^2);
    d3=sqrt((real(r3(1,k))-real(x4)).^2+((imag(r3(1,k))
       -imag(x4))).^2);
    d4=sqrt((real(r4(1,k))-real(x4)).^2+((imag(r4(1,k))
       -imag(x4))).^2);
    if (d1<d4 || d2<d4 || d3<d4)
        conta=conta+1;
    end
end
Pbe(b)=conta/(4*N); % Calculo de probabilidad de bit
   erroneo
b=b+1;
end
% Generar grafica
sir=0.1:0.1:2;
semilogy(sir,Pbe)
xlabel('Eb/N0')
ylabel('Probabilidad de bit erroneo')
grid on
```

6.3. Telefonía

Una parte medular de las redes telefónicas es el enrutamiento, el cual consiste en asignar o buscar trayectorias entre el transmisor y el receptor. El Sector de Normalización de las Telecomunicaciones de la Unión Internacional de Telecomunicaciones (ITU-T, por sus siglas en inglés) define varios tipos de enrutamientos, de los cuales uno de los más utilizados es el enrutamiento jerárquico, en el cual a partir de reglas específicas permite escoger rutas alternativas de manera fija, es decir, hay niveles (jerarquías). El tráfico siempre se enruta hacia los niveles más bajos en la red **(Bellamy, 2000)**.

En la **Figura 98** se muestra un ejemplo de una red, en donde P_1 y P_2 representan la probabilidad de ocupación de la ruta y $(1P_k)$ es la probabilidad de que la ruta esté libre. La probabilidad de bloqueo en la red está dada por la **ecuación 54**.

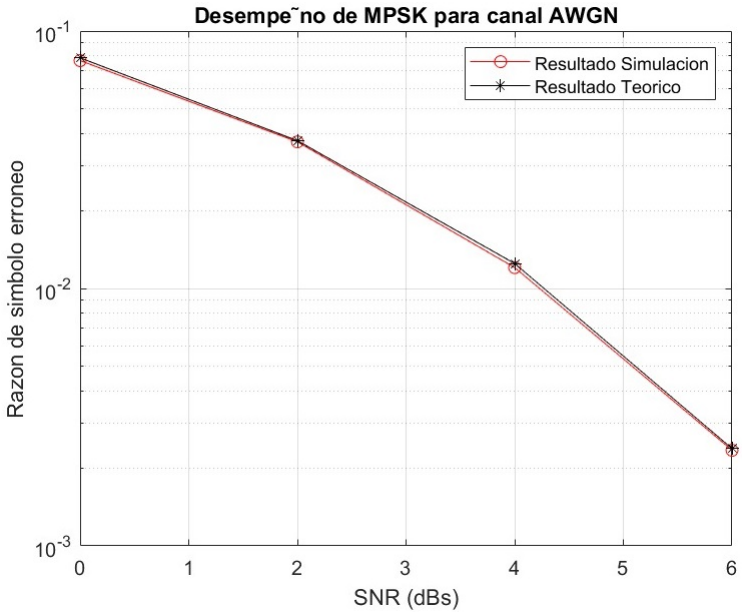


Figura 97: Sistema MPSK.

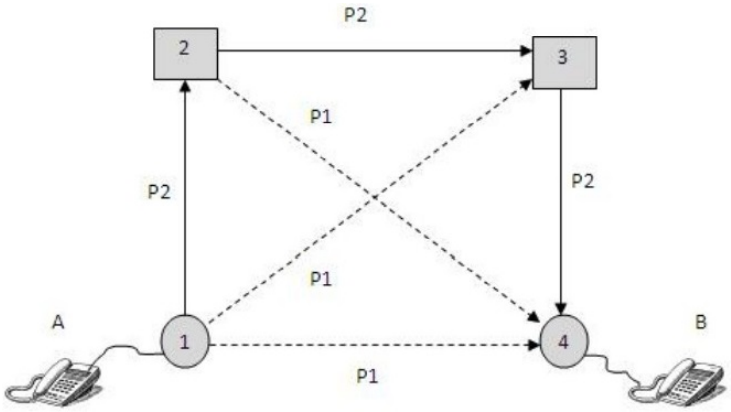


Figura 98: Redes jerárquicas.

$$PB = P_1^2\,P_2 + P_1\,P_2(1-P_1) + P_1^3(1-P_2)P_2 + P_1^3(1-P_2)^2P_2 \tag{54}$$

Se observa en la **Figura 98** que no se pueden dar rutas hacia atrás, es decir siempre son de los niveles bajos hacia los niveles altos. El **código 8.4** calcula la probabilidad de bloqueo utilizando enrutamiento jerárquico, para el ejemplo de la **Figura 98**, cuando el *abonado A* desea comunicarse con el *abonado B*, para valores de $P_1 = 0.3$ y $P_2 = 0.005$, el programa solicita al usuario la probabilidad de ocupación de las rutas, si la ruta no existe la probabilidad es 1, también solicita como dato el nodo alterno, por ejemplo de la **Figura 98** se observa que si la ruta que une el nodo 1 con el nodo 4 está bloqueado, entonces el nodo alterno es el 3. El siguiente **código 8.4** genera un archivo de función *"M"*, para el cálculo de la probabilidad de bloqueo.

Código 8.4.

Código

```
% FUNCION QUE CALCULA LA PROBABILIDAD DE BLOQUEO, RECIBE COMO PARAMETROS LOS NODOS DE ORIGEN, DESTINO
% INDICES, LA MATRIX DE PROBABILIDAD Y LA MATRIX DE NODOS ALTERNOS
function[prob_bloqueo]=pb(i,d,a,b,prob,alter);
if(a==d) % CONDICION DE PARO
prob_bloqueo=0;
elseif(a~=d && b==0) % CONDICION DE PARO
prob_bloqueo=1;
else
% CALCULO DE PROBABILIDAD DE BLOQUEO CON USO RECURSIVO DE LA FUNCION
prob_bloqueo=(1-prob(a,b)).*pb(i,d,b,d,prob,alter)+ (prob(a,b).*pb(i,d,a,alter(a,b),prob,alter));
end
```

El **código 8.5** produce los siguientes resultados:

Código 8.5.

Código

```
% CONDICIONES DEL CODIGO
disp('PROGRAMA QUE CALCULA LA PROBABILIDAD DE BLOQUEO DE EXTREMO A EXTREMO DE UNA RED DADA');
disp('SI NO EXISTE ENLACE LA PROBABILIDAD DEBE SER 1');
disp('SI NO EXISTE ENLACE ALTERNO, ENTONCES EL VALOR DE NODO ALTERNO DEBE SER 0');
% CONDICIONES INICIALES
x=4;
mat=zeros(x,x); % SE CREA MATRIX DE CEROS DE 4X4
traf=zeros(x,x); % SE CREA MATRIX DE CEROS DE 4x4
t=1; % VARIABLE PARA RECORER LA MATRIX
for j=1:x-1
for i=x-1:-1:t
strcat('INGRESE PROBABILIDAD DE BLOQUEO ENTRE NODO ', num2str(j), 'Y',num2str(i+1))
% PEDIR PROBABILIDAD DE BLOQUEO PARA EL ENLACE
prob=input('PROBABILIDAD DE BLOQUEO= ');
alter=input('INGRESE NODO ALTERNO: ');
mat(j,i+1)=prob; % MATRIZ DE PROBABILIDADES DE BLOQUEO
traf(j,i+1)=alter; % MATRIZ DE NODOS ALTERNOS
end
t=t+1;
end
m=pb(1,x,1,x,mat,traf); % LLAMADO DE LA FUNCION PARA EL CALCULO DE LA PROBABILIDAD DE BLOQUEO
disp('LA PROBABILIDAD DE BLOQUEO PARA ESTA RED ES: ')
disp(m)
```

```
PROGRAMA QUE CALCULA LA PROBABILIDAD DE BLOQUEO DE EXTREMO A EXTREMO DE UNA RED DADA
SI NO EXISTE ENLACE LA PROBABILIDAD DEBE SER 1
SI NO EXISTE ENLACE ALTERNO, ENTONCES EL VALOR DE NODO ALTERNO DEBE SER 0

ans =

    'INGRESE PROBABILIDAD DE BLOQUEO ENTRE NODO1Y4'

PROBABILIDAD DE BLOQUEO= 0.3
INGRESE NODO ALTERNO: 3

ans =

    'INGRESE PROBABILIDAD DE BLOQUEO ENTRE NODO1Y3'

PROBABILIDAD DE BLOQUEO= 0.3
INGRESE NODO ALTERNO: 2

ans =

    'INGRESE PROBABILIDAD DE BLOQUEO ENTRE NODO1Y2'

PROBABILIDAD DE BLOQUEO= 0.005
INGRESE NODO ALTERNO: 0

ans =

    'INGRESE PROBABILIDAD DE BLOQUEO ENTRE NODO2Y4'

PROBABILIDAD DE BLOQUEO= 0.3
INGRESE NODO ALTERNO: 3

ans =

    'INGRESE PROBABILIDAD DE BLOQUEO ENTRE NODO2Y3'

PROBABILIDAD DE BLOQUEO= 0.005
INGRESE NODO ALTERNO: 0

ans =

    'INGRESE PROBABILIDAD DE BLOQUEO ENTRE NODO3Y4'

PROBABILIDAD DE BLOQUEO= 0.005
INGRESE NODO ALTERNO: 0
LA PROBABILIDAD DE BLOQUEO PARA ESTA RED ES:
    0.0018
```

Figura 99: Resultados.

7. Aplicaciones en Sistemas Celulares

En esta unidad nos enfocaremos en mostrar el uso importante que tiene **MATLAB** en los sistemas celulares. Se mostrarán celdas celulares, usuarios en las celdas y se les dará movimiento a los usuarios, esto para mostrar una idea de lo que sucede en los sistemas reales de comunicación inalámbrica. Los códigos que se desarrollan en esta unidad se pueden utilizar para realizar un

análisis del comportamiento de los usuarios en una red celular 5G. Podemos simular la distribución de los usuarios en la celdas, calcular la interferencia que se presenta en cada banda y la velocidad de transmisión, esto se realiza para un esquema NOMA.

7.1. Celdas celulares

El área de cobertura de las celdas celulares en un principio se representaba mediante circunferencias, pero existen zonas de traslape y zonas que quedan sin cobertura es por esto que las celdas celulares típicamente se representan por hexágonos, para facilitar el análisis matemático.

El siguiente **código 9.1** genera la gráfica de una celda celular hexagonal, en este ejemplo la distancia desde el centro de la celda al vértice es de 10 como se puede observar en la **Figura 100**.

Código 9.1.

```
1 i=-pi:pi/3:pi;
2 x=10*cos(i);
3 y=10*sin(i);
4 plot(x,y)
```

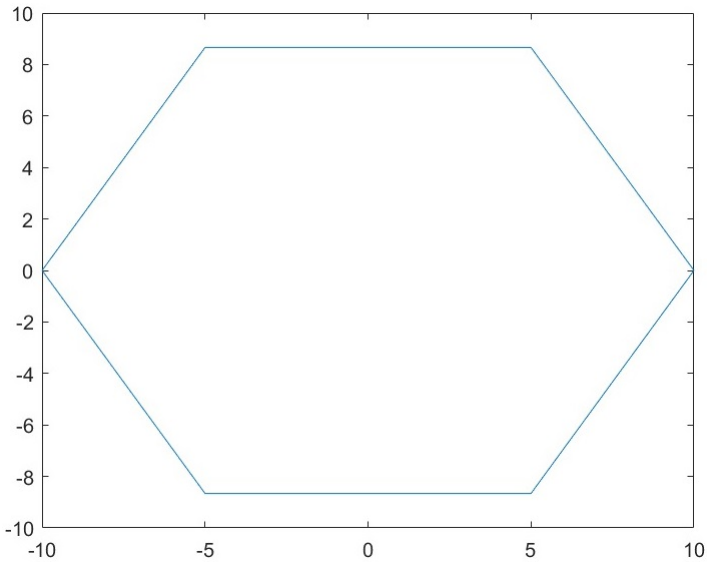


Figura 100: Celda celular.

Al analizar un sistema celular, se toma en cuenta una red de celdas, es decir, una interconexión de celdas adyacentes, esto para considera la transferencia de llamadas, es decir el cambio de celdas. El **código 9.2** genera la gráfica de una red de 19 celdas conjuntas. Esto se reduce a un análisis geométrico (ver **Figura 101**).

Código 9.2.

```
i=-pi:pi/3:pi;
x0=10*cos(i); y0=10*sin(i);
x1=x0+15; y1=y0+8.6603;
x2=x0; y2=y1+8.6603;
x3=x1; y3=y0-8.6603;
x4=x0; y4=y3-8.6603;
x5=x0-15; y5=y1;
x6=x5; y6=y0-8.6603;
x7=x0; y7=y2+8.6603+8.6603;
x8=x1; y8=y2+8.6603;
x9=x8+15; y9=y2;
x10=x9; y10=y1-8.6603;
x11=x10; y11=y10-8.6603-8.6603;
x12=x1; y12=y11-8.6603;
x13=x0; y13=y12-8.6603;
x14=x5; y14=y12;
x15=x14-15; y15=y11;
x16=x15; y16=y10;
x17=x16; y17=y9;
x18=x5; y18=y8;
plot(x0,y0,'k') hold on;
plot(x1,y1,'k')
plot(x2,y2,'k')
plot(x3,y3,'k')
plot(x4,y4,'k')
plot(x5,y5,'k')
plot(x6,y6,'k')
plot(x7,y7,'k')
plot(x8,y8,'k')
plot(x9,y9,'k')
plot(x10,y10,'k')
plot(x11,y11,'k')
plot(x12,y12,'k')
plot(x13,y13,'k')
plot(x14,y14,'k')
plot(x15,y15,'k')
plot(x16,y16,'k')
plot(x17,y17,'k')
plot(x18,y18,'k')
```

7.2. Usuarios en las celdas celulares

Una vez creadas las celdas celulares lo que resta para visualizar el sistemas celular completo, es decir, generar usuarios dentro de la celda. El **código 9.3** genera usuarios aleatorios dentro de una celda, los grafica y genera el efecto de movimiento de los usuarios, los cuales tienen 4 grados de libertad, es decir se mueven hacia el norte, sur, este y oeste. Cada movimiento se realiza cada 0.7 segundos, esto para visualizarse.

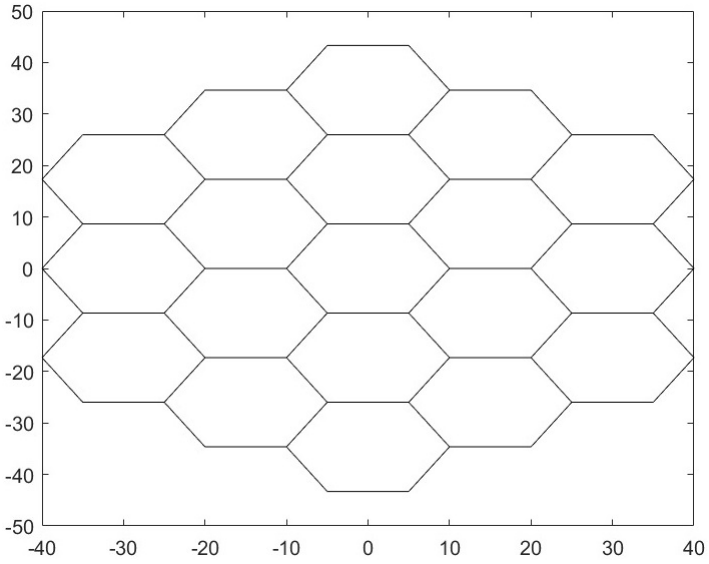


Figura 101: Red celular.

Código 9.3.

Código

```
% Generar Celdas
i=-pi:pi/3:pi;
x=10*cos(i);
y=10*sin(i);
% Generar N usuarios
N=1000;
xv=10*randn(N,1);
yv=10*randn(N,1);
% Colocar los usuarios dentro de la celda
in=inpolygon(xv,yv,x,y);
m1=xv(in);
m2=yv(in);
% Generar movimientos de los usuarios cada 0.7 seg
for u=1:30
pause (0.7)
drawnow
clf
nuevo=rand(length(m1),1);
if (sum(nuevo)>length(m1)/2)
    mnu=m1+nuevo;
    mnu2=m2+nuevo;
else
    mnu=m1-nuevo;
    mnu2=m2-nuevo;
end
```

Código

```
%% Cont...
% Limites de la celda, para asegurar que los usuarios queden dentro
for p=1:length(m1)
if (mnu(p)<=-10)
mnu(p)=mnu(p)+1;
elseif(mnu(p)>=10)
mnu(p)=mnu(p)-1
elseif(mnu2(p)<=-8.6603)
mnu2(p)=mnu2(p)+1;
elseif(mnu2(p)>=8.6603)
mnu2(p)=mnu2(p)-1;
end
if (atan(mnu(p)/mnu2(p))>=0.52 || atan(mnu(p)/mnu2(p))<=-0.52)
if (mnu(p)>=0 && mnu2(p)>=0)
mnu(p)=mnu(p)-1;
mnu2(p)=mnu2(p)-1;
elseif(mnu(p)>=0 && mnu2(p)<0)
mnu(p)=mnu(p)-1;
mnu2(p)=mnu2(p)+1;
elseif(mnu(p)<0 && mnu2(p)>=0)
mnu(p)=mnu(p)+1;
mnu2(p)=mnu2(p)-1;
elseif(mnu(p)<0 && mnu2(p)<0)
mnu(p)=mnu(p)+1;
mnu2(p)=mnu2(p)+1;
end
end
plot(x,y,mnu(p),mnu2(p),'.b')
hold on
end
end
```

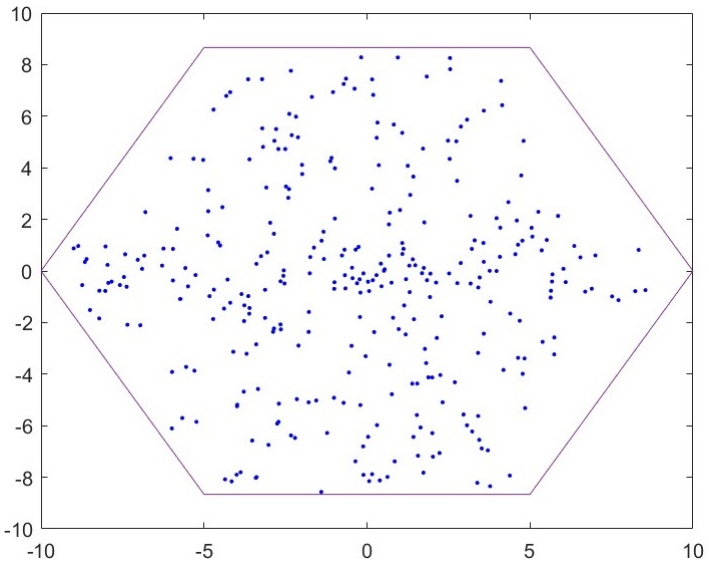


Figura 102: Usuarios en la celda.

Por último debemos distinguir a los usuarios, es decir, los usuarios cercanos a la estación base los consideramos con buena condición de canal (esto sin considerar los desvanecimientos) y los colocaremos en color ocre, los usuarios que se encuentren en condiciones aceptables lo colocaremos con color azul y los usuarios que se ubican en el borde de la celda, en malas condiciones de canal, los colocaremos en color rojo, esto se muestra en el **código 9.4**.

Código 9.4.

Código

```
% Generar Celdas
i=-pi:pi/3:pi;
x=10*cos(i);
y=10*sin(i);
% Generar N usuarios
N=1000;
xv = 10*randn(N,1);
yv = 10*randn(N,1);
% Colocar los usuarios dentro de la celda
in = inpolygon(xv,yv,x,y);
m1=xv(in);
m2=yv(in);
% Generar movimientos de los usuarios cada 0.7 seg
for u=1:30
pause (0.7)
drawnow
clf
nuevo=rand(length(m1),1);
if sum(nuevo)>length(m1)/2
mnu=m1+nuevo;
mnu2=m2+nuevo;
else
mnu=m1-nuevo;
mnu2=m2-nuevo;
end
% Limites de la celda, para asegurar que los usuarios
   queden dentro
for p=1:length(m1)
if (mnu(p)<=-10)
mnu(p)=mnu(p)+1;
elseif(mnu(p)>=10)
mnu(p)=mnu(p)-1;
elseif(mnu2(p)<=-8.6603)
mnu2(p)=mnu2(p)+1;
elseif(mnu2(p)>=8.6603)
mnu2(p)=mnu2(p)-1;
end
if (atan(mnu(p)/mnu2(p))>=0.52 || atan(mnu(p)/mnu2(p))
   <=-0.52)
if (mnu(p)>=0 && mnu2(p)>=0)
mnu(p)=mnu(p)-1;
mnu2(p)=mnu2(p)-1;
elseif(mnu(p)>=0 && mnu2(p)<0)
mnu(p)=mnu(p)-1;
mnu2(p)=mnu2(p)+1;
elseif(mnu(p)<0 && mnu2(p)>=0)
mnu(p)=mnu(p)+1;
mnu2(p)=mnu2(p)-1;
elseif(mnu(p)<0 && mnu2(p)<0)
mnu(p)=mnu(p)+1;
mnu2(p)=mnu2(p)+1;
end
end
end
```

Código

```
%% Cont...
% Asignar Colores
for o=1:length(m1)
if (sqrt(mnu(o)^2 + mnu2(o)^2) <= 5.9 && sqrt(mnu(o)^2+
    mnu2(o)^2) > 1.589)
col= 'bo';
elseif (sqrt(mnu(o)^2+mnu2(o)^2) <= 1.589 && sqrt(mnu(o
    )^2+mnu2(o)^2) > 0.9)
col= 'g+';
elseif (sqrt(mnu(o)^2+ mnu2(o)^2) <= 0.9)
col= 'g+';
else
col= 'r*';
end
hold on
plot(x,y,mnu(o),mnu2(o),col)
end
end
```

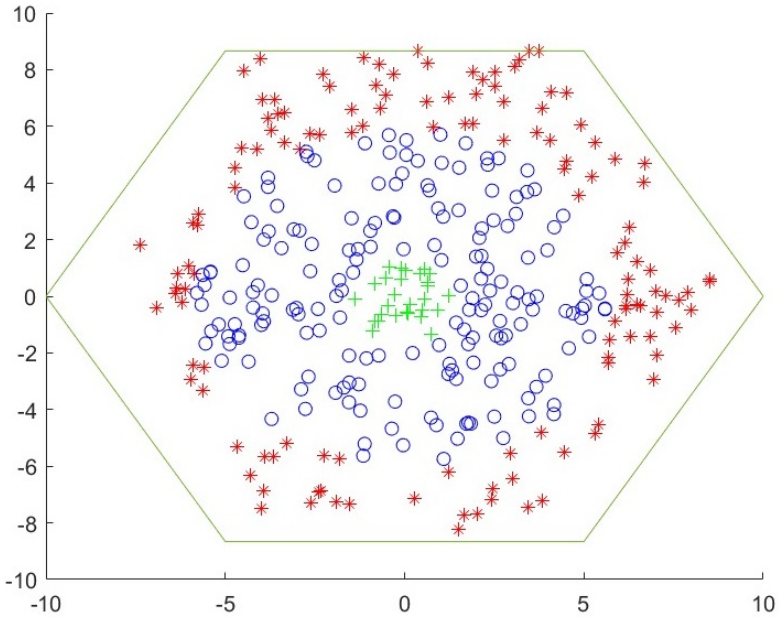


Figura 103: Usuarios en la celda.

7.3. Esquema NOMA para 5G

Hoy en día, las tecnologías de telecomunicaciones en general están tendiendo hacia la integración de todo tipo de clase de servicios, básicamente a través de una red global de comunicaciones. Adicional, se busca que dicha red tenga accesibilidad total, en otras palabras, que todo tipo de usuario pueda acceder a ésta desde cualquier parte, sin tener que estar en una red fija, simplemente mediante la utilización de un terminal móvil **(Butt et al., 2017)**.

Un desafío importante para 5G será la mejora de la eficiencia espectral, ya que con esta se busca ofrecer altas velocidades de transferencia de datos a mayor número de teléfonos inteligentes. La eficiencia espectral se puede incrementar al aumentar el orden de modulación (se refiere a la cantidad de bits de información que se transmiten por símbolo en una señal modulada) y, además, se pueden emplear nuevos esquemas de acceso al medio para mejorar ésta. De los esquemas de acceso al medio podemos destacar el acceso múltiple no ortogonal (*NOMA*),

el cual garantiza que múltiples usuarios puedan compartir el mismo recurso **(Butt et al., 2017)**.

Recientemente *NOMA* se ha destacado como una de las técnicas más prometedoras en cuanto al acceso múltiple para la tecnología 5G, lo anterior debido a que soluciona la problemática de acomodar a más usuarios en el mismo recurso haciendo la asignación de éste en dominios de potencia o código. Estos beneficios se observan commparando esta técnica con la del acceso múltiple ortogonal, **(Vasquez-Toledo & Lara-Rodríguez, 2017)**.

En conclusión, *NOMA* radica en el uso sincrónico del mismo espectro de radioeléctrico para múltiples usuarios a expensas de mínimas interferencias entre ellos. Las redes que trabajan bajo el esquema de *NOMA* pueden contemplar a un mayor número de usuarios en las bandas de espectro radioeléctrico que esté disponible, y pueden ofrecer a los usuarios individuales con un mayor ancho de banda, **(Vasquez-Toledo & Lara-Rodríguez, 2017)**.

A continuación, se mencionan las ventajas que se observan al emplear *NOMA* en comparación con el Acceso Múltiple Ortogonal (*OMA*, por sus siglas en inglés), **(Vasquez-Toledo & Lara-Rodríguez, 2017)**:

- Una mejor eficiencia espectral.
- Existe una conectividad masiva.
- Presenta baja latencia de transmisión y el costo de señalización.

Para que el esquema *OMA* presente resultados exitosos, se deben formar grupos de usuarios y permitir que estos transmitan en el mismo recurso radioeléctrico con cierta potencia adecuada, así, posteriormente emplear técnicas de cancelación de interferencia sucesiva, mejor conocida como *SIC* en el/los receptor/es para finalmente decodificar la señal de mensaje de diferentes usuarios. En la **Figura 104** se presenta el esquema *NOMA* de manera básica a través de un multiplexado en el dominio de la potencia con un receptor SIC, **(Vasquez-Toledo & Lara-Rodríguez, 2017)**.

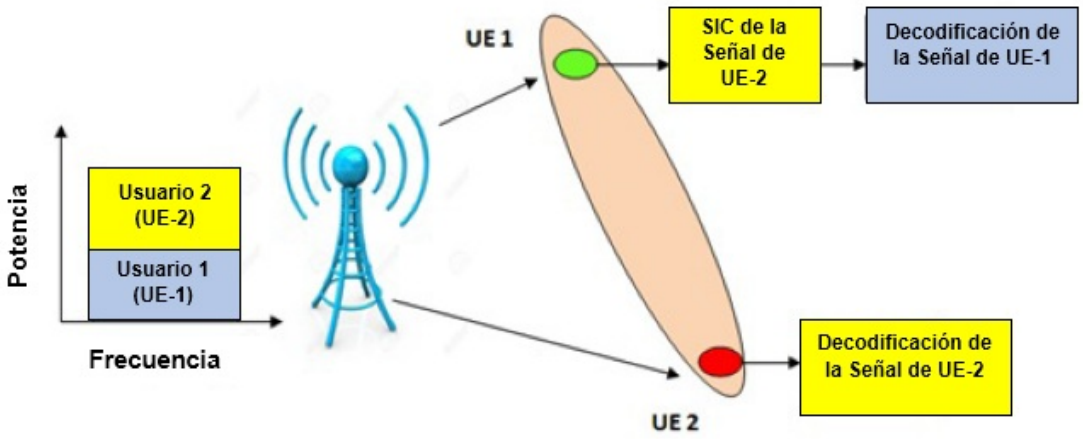


Figura 104: NOMA con SIC en el receptor.

Por sencillez se considera el empleo de una sola antena transmisora y una antena receptora. En el esquema se observa que la estación base transmite una señal (xi) al usuario i (UEi), con potencia Pi, **(Vasquez-Toledo & Lara-Rodríguez, 2017)**.

En $NOMA$, el proceso de cancelar la interferencia sucesiva se realiza en el receptor del UE. El orden óptimo para la decodificación está en el orden decreciente de ganancia del canal normalizado por el ruido y la potencia de interferencia entre celdas (llamado simplemente ganancia del canal). En resumen, se supone que cualquier usuario puede decodificar correctamente las señales de otros usuarios cuyo orden de decodificación viene antes del usuario correspondiente. Por lo tanto, UEi puede eliminar la interferencia del usuario j cuya ganancia de canal $|h_j|^2$ es menor a la del usuario i, $|h_i|^2(j < i)$, es decir, las ganancias de los canales se ordenan de mayor a menor **(Borja-Benítez, Vasquez-Toledo, Tirado-Méndez, Borja-Benítez, & Marcelin-Jiménez, 2020)**.

En la **Figura 105**, se puede observar que el usuario 2 ($UE-2$) no realiza la cancelación de interferencia ya que este es el primero en el orden de decodificación. El $UE-1$ primero decodifica la señal de $UE-2$ y resta de su componente de señal recibida, y a continuación se decodifica la señal del $UE-1$ sin interferencia de la señal del $UE-2$ **(Borja-Benítez et al., 2020)**.

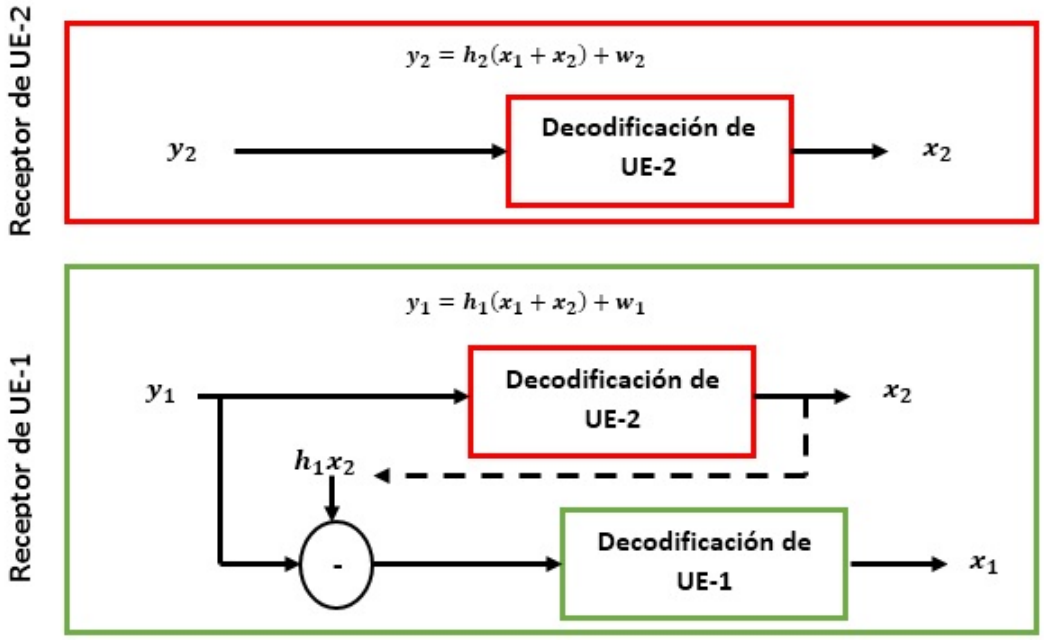


Figura 105: Proceso SIC.

La tasa de transmisión para cada usuario se calcula como sigue:

- Esquema NOMA

$$R_1 = log_2\left(1 + \frac{P_1|h_1|^2}{N_{0,1}}\right) \qquad R_2 = log_2\left(1 + \frac{P_2|h_2|^2}{P_1|h_2|^2N_{0,2}}\right)$$

- Esquema OMA

$$R_1 = \alpha log_2\left(1+\frac{P_1|h_1|^2}{\alpha N_{0,1}}\right) \qquad R_2 = (1-\alpha) log_2\left(1+\frac{P_2|h_2|^2}{(1-\alpha) N_{0,2}}\right)$$

donde

R_1	es la tasa de transmisión normalizado del usuario 1 (bps/Hz).
R_2	es la tasa de transmisión normalizado del usuario 2 (bps/Hz).
P_1	es el porcentaje de potencia asignado al usuario 1.
P_2	es el porcentaje de potencia asignado al usuario 2.
$(\|h_1\|^2)/N_{0,1}$	es la ganancia de canal del usuario 1.
$(\|h_2\|^2)/N_{0,2}$	es la ganancia de canal del usuario 2.
α	es el porcentaje de ancho de banda asignado al usuario 1.

El **código 9.5** simula el canal para dos usuarios que comparten el mismo recurso utilizando la tecnología *NOMA*. Calcula la ganancia del canal, utilizando un canal con desvanecimiento log-normal. Utiliza la función `Ganancia.m` (**código 9.6**) para calcular la ganancia de la antena que depende del ángulo de incidencia. Este código también calcula las tasas de transmisión para cada usuario en ambos esquemas (*OMA* y *NOMA*).

Código 9.5.

Código

```
% Usuario 1
x1=0.2; % Coordenada X del usuario 1
y1=0.66603; % Coordenada Y del usuario 2
d1=sqrt(x1.^2+y1.^2); % Distancia del usuarios 1 a la
    Antena Servidor
ang1=atan(y1./x1)*180/pi; % Angulo
G1=Ganancia(ang1); % Ganancia
% Usuario 2
x2=1; % Coordenada X del usuario 2
y2=-0.86; % Coordenada Y del usuario 2
d2=sqrt(x2.^2+y2.^2); % Distancia del usuarios 2 a la
    Antena Servidor
ang2=atan(y2./x2)*180/pi; % Angulo
G2=Ganancia(ang2); % Ganancia
NGaussianas = 5000; % Numero de variables aleatorias
media = 4; % Valor de la media para ambiente de
    propagación urbano
sigma = 8; % Valor de la varianza para ambiente de
    propagación urbano
% Calculo de Parametros para el usuario 1
media1=10*log10((d1^-4)*G1); % Media
Gaussianas1 = randn(1,NGaussianas);
inter1 = sigma*Gaussianas1 + media1; % Canal Gaussiano
lognormal1= 10.^(inter1/10); % Canal Log-Normal
Gan_canal_1=10*log10(mean(abs(lognormal1))) % Valores
    en dB
```

Código

```
%% cont...
% Calculo de Parametros para el usuario 2
media2=10*log10((d2^-4)*G2); % Media
Gaussianas2 = randn(1,NGaussianas);
inter2 = sigma*Gaussianas2 + media2; % Canal Gaussiano
lognormal2= 10.^(inter2/10); % Canalo Log-Normal
Gan_canal_2=10*log(mean(abs(lognormal2))) % Valores en
    dB
% Asignación de Potencias para cada usuario
P1=0.2;
P2=0.8;
% Tasas para NOMA
R1_NOMA=log2(1+P1*mean(abs(lognormal1)))
R2_NOMA=log2(1+P2*mean(abs(lognormal2))/(P1*mean(abs(
    lognormal2))+1))
% Tasas para OMA
alfa=0.5; % Porcentaje de ancho de banda asignado al
    usuario 1
R1_OMA=alfa*log2(1+(P1*mean(abs(lognormal1))/alfa))
R2_OMA=(1-alfa)*log2(1+(P2*mean(abs(lognormal2))/(1-
    alfa)))
```

Código 9.6.

Código

```
%% Este programa sirve para obtener la ganancia de la
    antena
function [G] = Ganancia(ang)
ar = degtorad(ang);
q=-4; %nivel de ganancia de la antena
p=-15; %nivel promedio de ganancia normalizada para el
     l bulo   lateral.
grado = 0;
raiz=sqrt((1-p)/(1-q))*(pi/3);
%ar=ang;
if(abs(ar)<=raiz)
    grado=1-(((1-q)/((pi/3)^2))*(ar^2));
 elseif (abs(ar)>raiz)
         grado=p;
end
G=grado;
end
```

El **código 9.5** produce:

```
Command Window

Gan_canal_1 =

   21.0926


Gan_canal_2 =

    9.2950
```

Figura 106: Ganancias de canal.

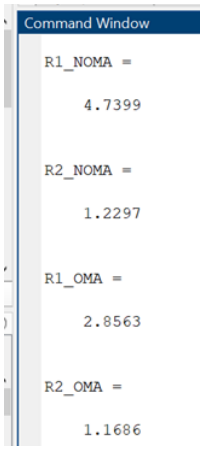


Figura 107: Tasas de transmisión.

Comparando los resultados en las tasas de transmisión para cada esquema se puede observar que *NOMA* tiene ganancia respecto a *OMA* para cada usuario. Para el usuario 1 se tiene una ganancia mayor al 60 %, y para el usuario 2 mayor del 6 %. Con esto se demuestra la ventaja de utilizar el esquema *NOMA*.

8. Introducción a SIMULINK

MATLAB nos ofrece una herramienta llamada **Simulink**, la cual es un entorno muy popular, ya que es un simulador multidominio y su diseño se basa en modelos de diagramas de bloques. Es compatible con:

- El sistema de nivel de diseño.
- Simulación.
- Generación automática de código.
- Prueba continua y verificación de sistemas embebidos.

Además, **Simulink** ofrece el servicio de editor gráfico, contiene bibliotecas de bloques que son personalizadas y, también soluciones para el modelado, así como la simulación de sistemas dinámicos. Este entorno está contenido en **MATLAB**, lo que es muy conveniente ya que podemos añadir códigos de **MATLAB** en los modelos y, además, exportar los resultados obtenidos de las simulaciones para **MATLAB**, lo anterior para realizar análisis posteriores **(Beucher & Weeks, 2008)**.

8.1. Ingresar a SIMULINK

Para ingresar a **Simulink** inicialmente debemos ejecutar la herramienta desde la ventana de comando de **MATLAB**, escribiendo *simulink* en la línea de comandos o la otra opción es dando click en el icono como el mostrado a la derecha.

A continuación, se desplegará una nueva ventana como la mostrada en la **Figura 108**. En ésta, podremos observar todas las librerías de los bloques disponibles en **Simulink**. Del lado izquierdo podemos observar lo correspondiente a todas las librerías disponibles y del lado derecho todo el contenido de cada una de las librerías.

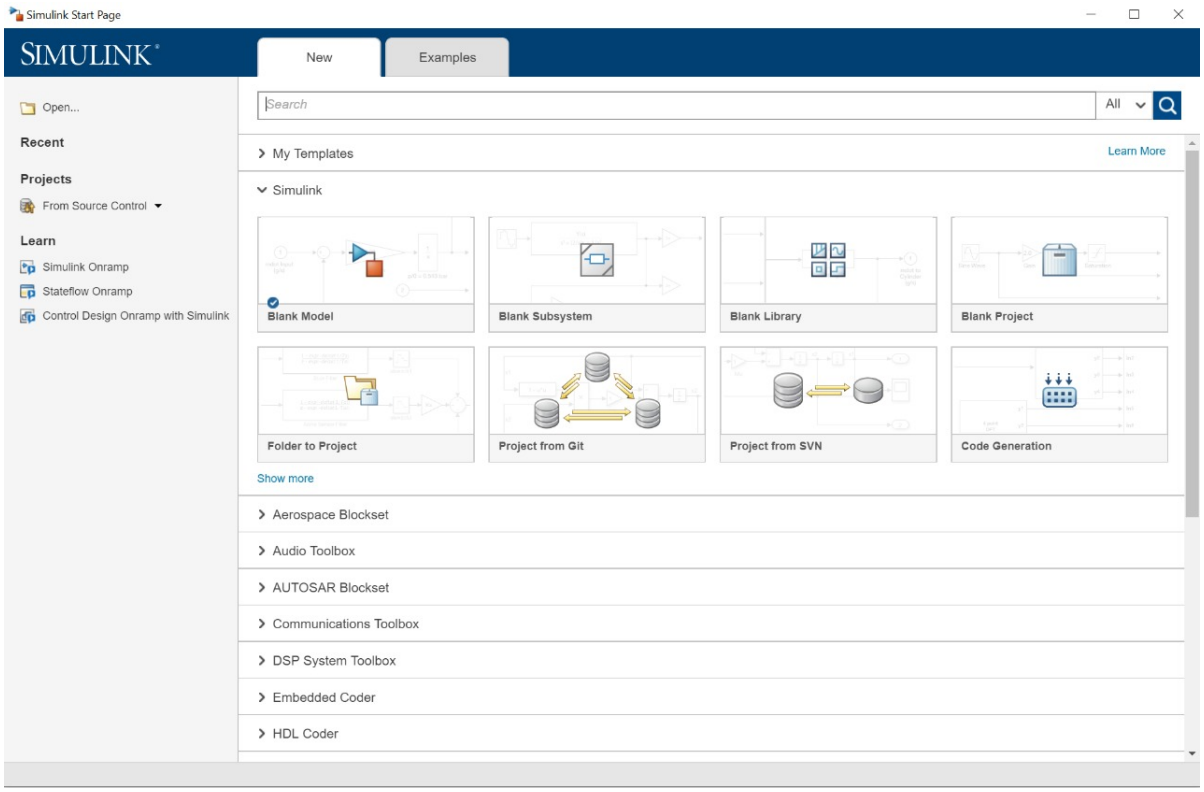


Figura 108: Librerías.

Finalmente, en la ventana de inicio de **Simulink**, en la parte superior, podemos observar diversas herramientas tales como la búsqueda de un bloque determinado a partir de su nombre.

8.2. Entorno de trabajo de SIMULINK

Para poder crear archivos necesitamos abrir un nuevo fichero, para esto, debemos elegir un nuevo fichero en el que se guardará el modelo, damos clic en el menu *NEW*, *Blank Model*, *Create Model*, como se muestra en la **Figura 109 (Beucher & Weeks, 2008)**.

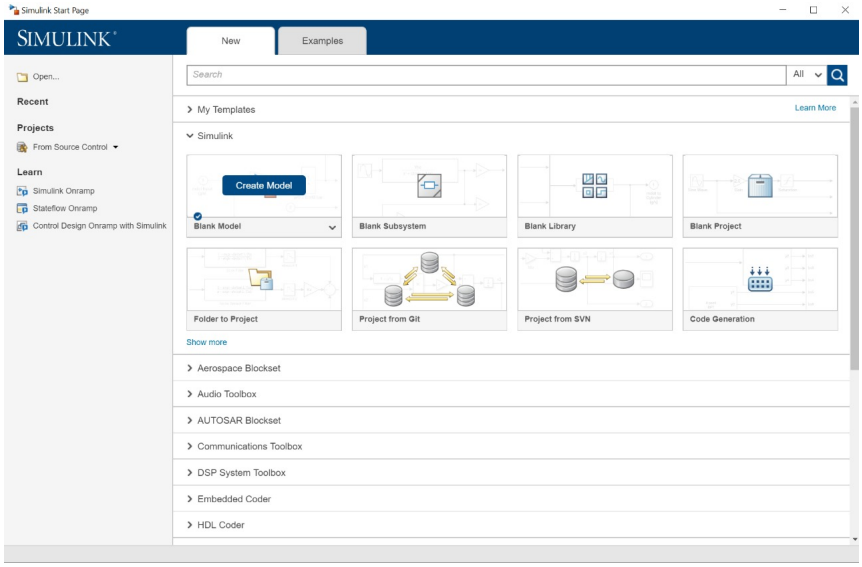


Figura 109: Nuevo fichero.

En esta nueva ventana crearemos el modelo (ver **Figura 110**), observemos que aún no se le ha asignado nombre.

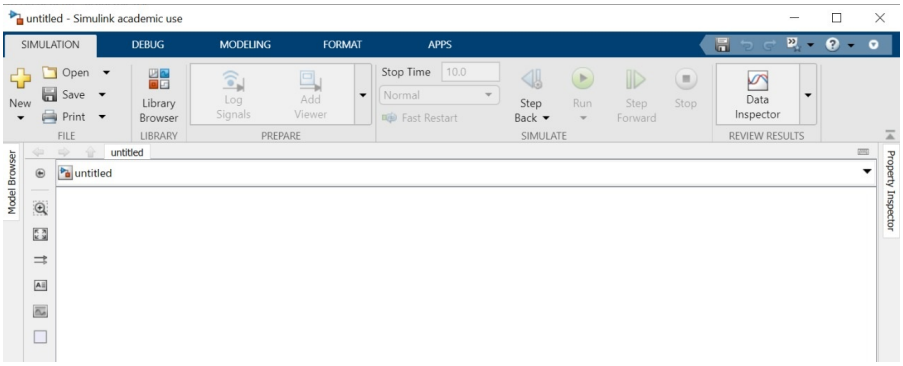


Figura 110: Nuevo fichero.

8.3. Muestreo y Cuantización

Recordando los ejemplos mostrados en el **Capítulo 3**, se trabajó con el teorema de muestreo y se desarrolló el código para el muestreo ideal. En esta sección se desarrollará este ejemplo mediante bloques.

En la ventana de librerías, en la opción **Simulink**, elegir la librería *Sources* y en esta buscar y elegir *Sine Wave*, arrastrar este bloque en la ventana del fichero, como se muestra en la **Figura 111**. En el bloque colocado en el fichero, hacer doble clic, se desplegará una ventana de configuración del bloque en la cual se eligen las características de la señal analógica que queremos generar, esto se observa en la **Figura 112**.

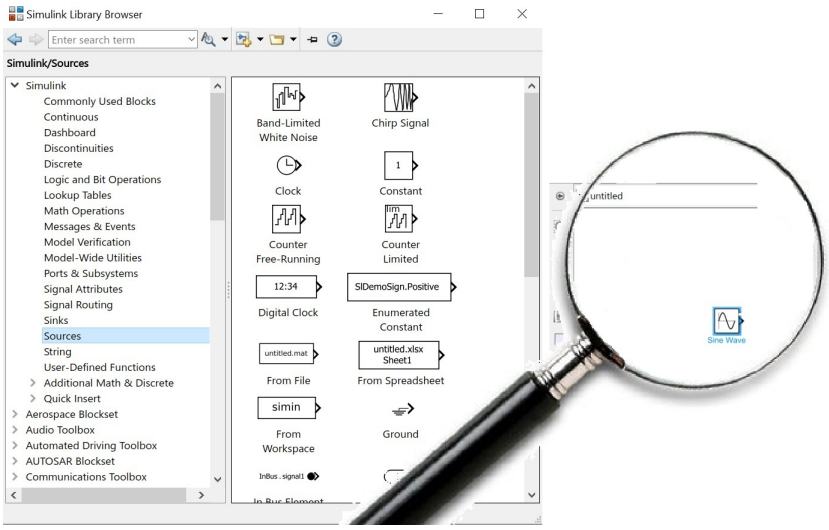


Figura 111: Selección de bloque.

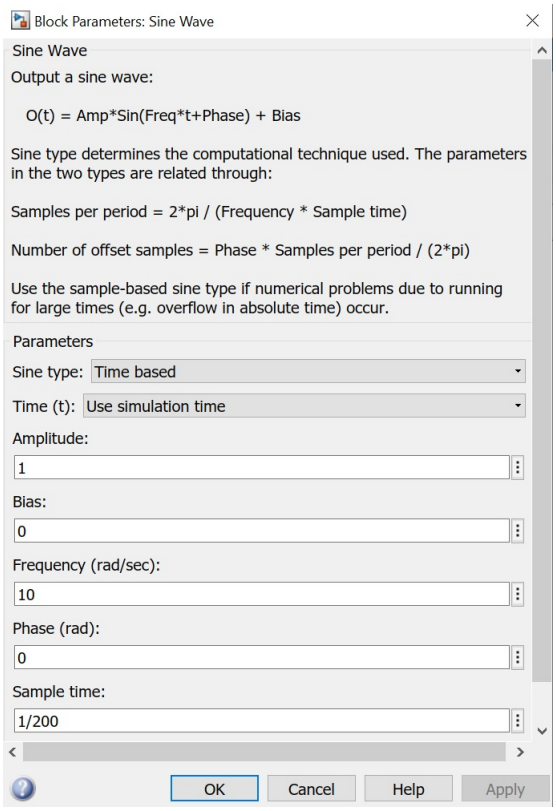


Figura 112: Configuración de características.

En la ventana de librerías, en la opción *SIMULINK*, elegir la librería *Sources* y en esta buscar y elegir *Pulse Generator*, y configurar las características para que la duración del pulso sea muy pequeña, como se muestra en la **Figura 113**.

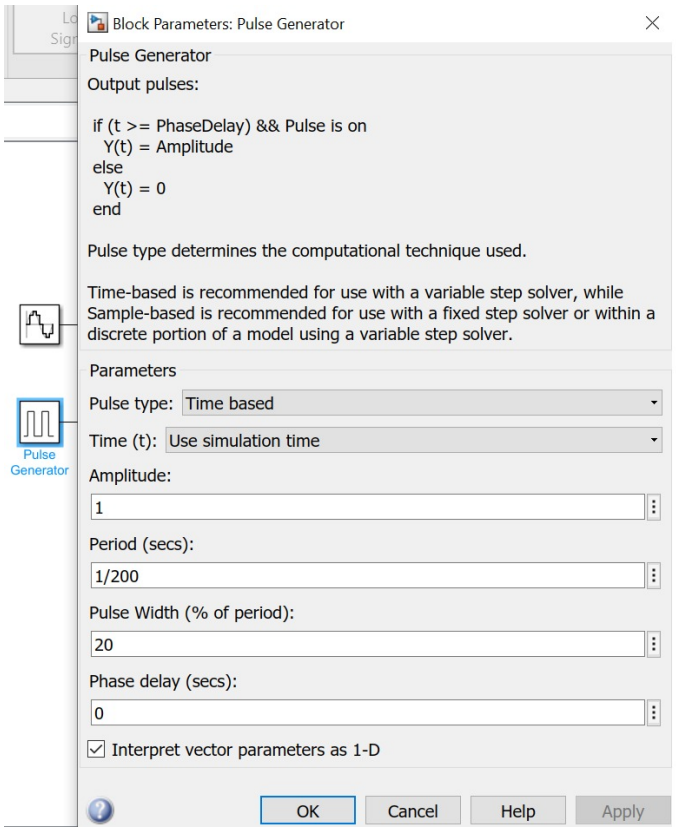


Figura 113: Configuración de características.

En la ventana de librerías, en la opción *SIMULINK*, elegir la librería *Math Operations* y en esta buscar y elegir *Product*, como se muestra en la **Figura 114**.

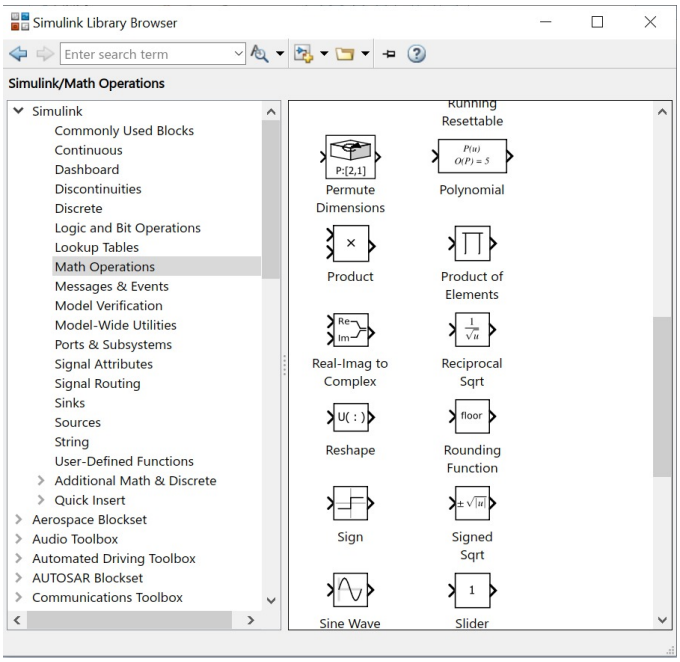


Figura 114: Selección de bloque.

Lo que resta es agregar un bloque para presentar los resultados, para esto, en la ventana de librerías, en la opción *SIMULINK*, elegir la librería *Sinks* y en esta buscar y elegir *Scope*, como se muestra en la **Figura 115**.

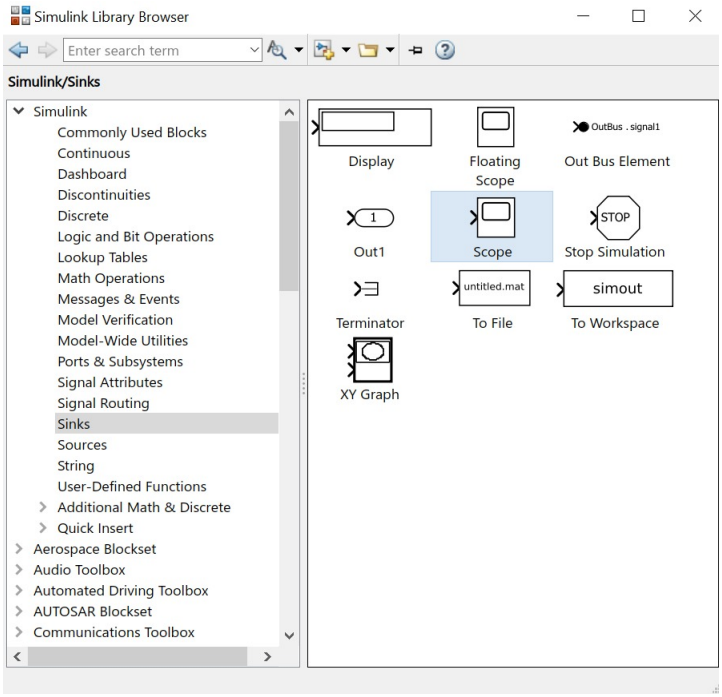


Figura 115: Selección de bloque.

Ya que tenemos estos bloques en el fichero, lo que sigue es la interconexión de estos bloques. Conectar estos bloques como se muestra en la **Figura 116**.

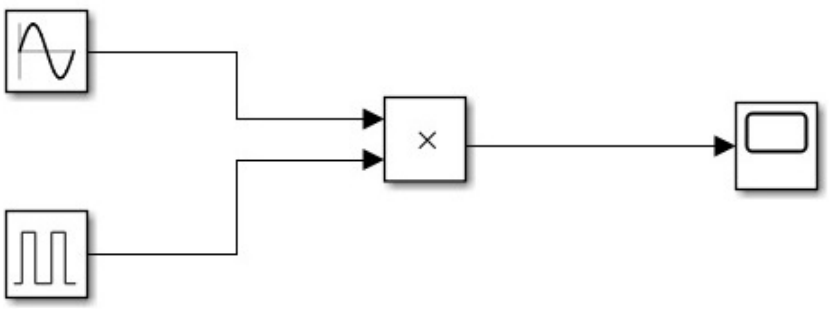


Figura 116: Conexión de bloques.

Teniendo conectado los bloques el siguiente paso es iniciar la simulación como se muestra en la **Figura 117**.

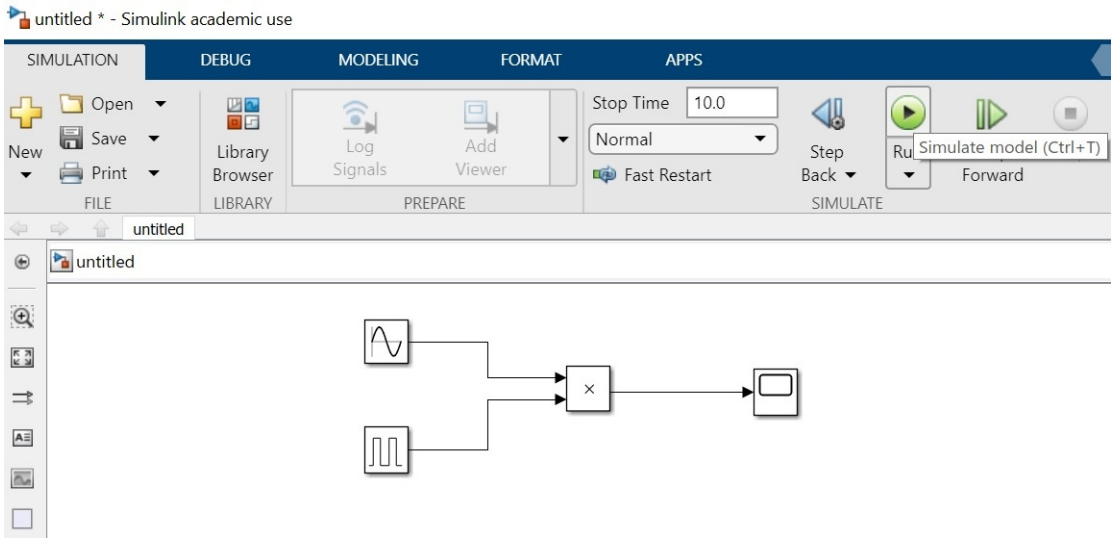


Figura 117: Simulación.

Para observar los resultados de la simulación, debemos hacer un doble clic sobre el bloque *Scope*, el cual arroja el resultado, como se observa en la **Figura 118**.

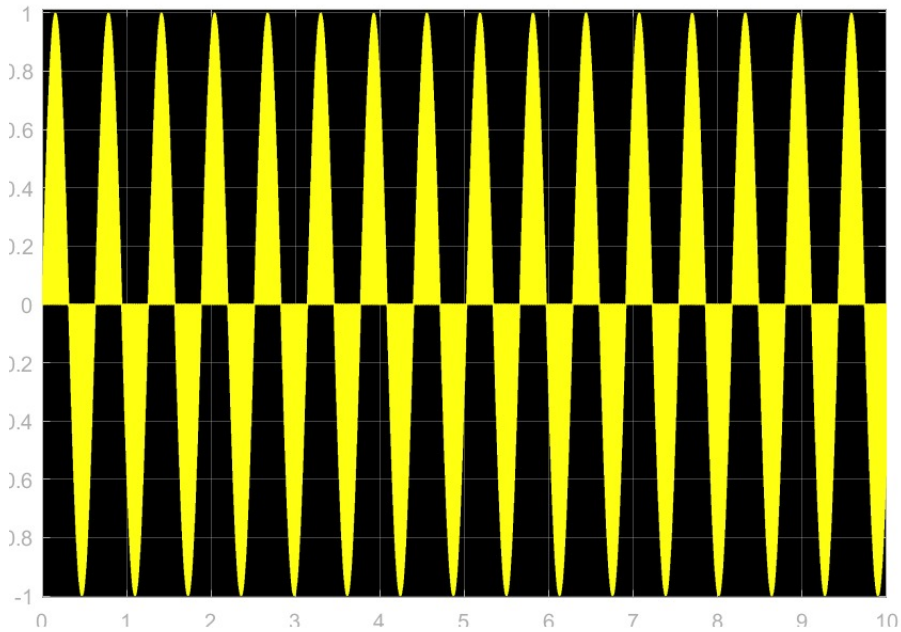


Figura 118: Resultado de simulación.

Se observa en la **Figura 118** el muestreo de la señal analógica. En la **Figura 118**, en donde se despliega el resultado, podemos encontrar *opciones de figura*, en las cuales podemos darle zoom a la figura tanto en el eje x como en el eje y. Utilizando estas opciones obtenemos la **Figura 119**.

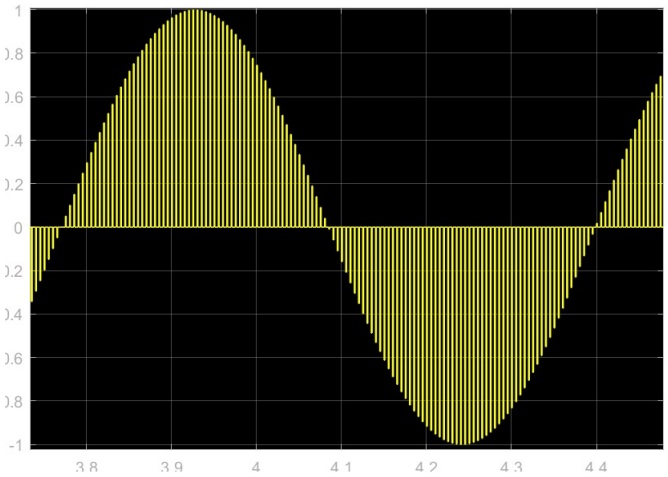


Figura 119: Resultado de simulación.

Para el ejemplo de cuantización, se siguen los mismos pasos, en la ventana de librerías elegir la *simulink*, dentro de la cual se busca *Discontinuities*, y elegir *Quantizers*, en la ventana derecha elegir y arrastra el bloque *Quantizer*, como se muestra en la **Figura 120**. Este bloque tiene como entrada el bloque que genera la señal analógica (*Sine Wave*) y como salida un bloque *Scope*.

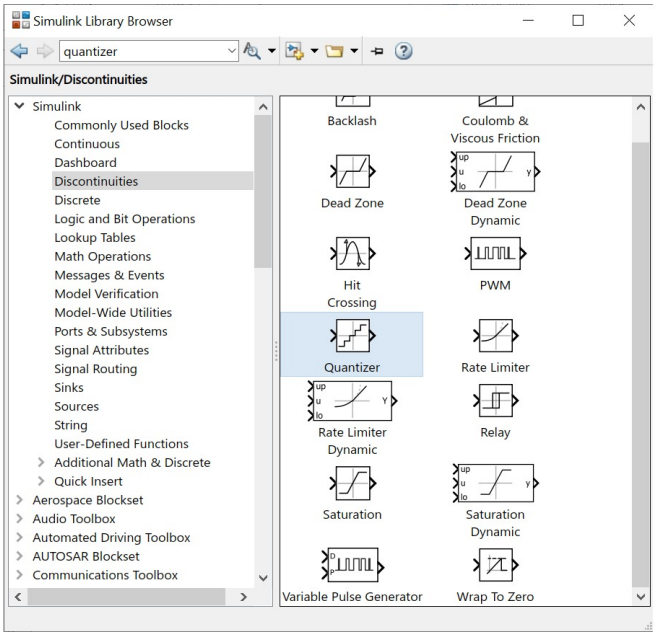


Figura 120: Selección de bloque.

Se siguen los mismos pasos explicado para el ejemplo anterior, para la simulación del fichero, y se obtiene el resultado mostrado en la **Figura 121**.

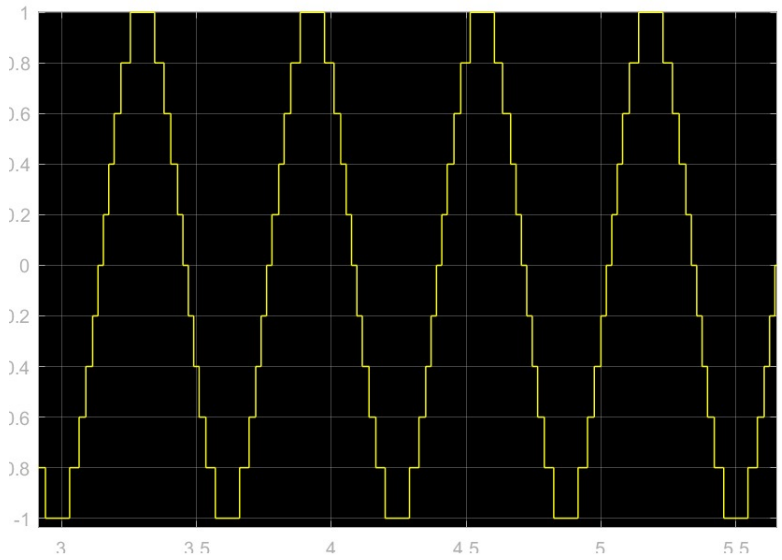


Figura 121: Cuantización.

Por último se guarda el archivo con la extensión *.mdl*, como se muestra en la **Figura 122**.

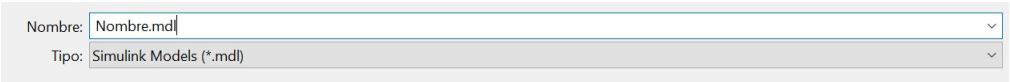


Figura 122: Guardar archivo.

Referencias

Aguilar, H. J., y cols. (2002). *Fundamentos de los sistemas modernos de comunicación* (Vol. 7). Alfaomega.

Bellamy, J. C. (2000). *Digital telephony.*

Beucher, O., & Weeks, M. (2008). *Introduction to matlab & simulink (a project approach).* Laxmi Publications, Ltd.

Borja-Benítez, M. A., Vasquez-Toledo, L. A., Tirado-Méndez, J. A., Borja-Benítez, B., & Marcelin-Jiménez, R. (2020). A novel mathematical teletraffic method to evaluate the interoperation of different standards with noma systems using order statistics. *International Journal of Communication Systems*, *33*(11), 4406.

Butt, M. M., Popovski, P., Shakir, M. Z., López-Pérez, D., Quek, T. Q., & Guizani, M. (2017). Ieee access special section editorial: physical and medium access control layer advances in 5g wireless networks. *IEEE Access*, *5*, 27845–27849.

Gilat, A. (2008). *Matlab: An introduction with applications* (Vol. 3). Wiley New York.

Haykin, S. S. (2002.) *Sistemas de comunicaci.* Mco :: Limusa Wiley,. (Traducido de : Communication systems.)

Kamen, E. W., & Heck, B. S. (2008). *Fundamentos de señales y sistemas usando la web y matlab.* Pearson Educación.

Moya, P. A. (2008). *Señales y sistemas. fundamentos matemáticos.* Pablo Alvarado Moya.

Oppenheim, A. V., Willsky, A. S., & Nawab, S. H. (1998). *Señales y sistemas.* Pearson Educación.

Papoulis, A., & Saunders, H. (2002). Probability, random variables and stochastic processes.

Proakis, J. G., Salehi, M., Zhou, N., & Li, X. (1994). *Communication systems engineering* (Vol. 2). Prentice Hall New Jersey.

Salema, C. (2002). *Microwave radio links: From theory to design.* John Wiley & Sons, Inc.

Stüber, G. L., & Stèuber, G. L. (2012). *Principles of mobile communication* (Vol. 2). Springer.

Vasquez-Toledo, L. A., & Lara-Rodríguez, D. (2017). A coordinated operation access scheme for 5g cellular systems. En *2017 ninth international conference on ubiquitous and future networks (icufn)* (pp. 604–606).

Luis Alberto Vásquez Toledo
Universidad Autónoma Metropolitana,
Unidad Iztapalapa,
División de Ciencias Básicas e Ingeniería,
Departamento de Ingeniería Eléctrica.
Av. Ferrocarril San Rafael Atlixco 186, Col. Leyes de Reforma 1^a Sección,
Alcaldía Iztapalapa, C.P. 09310 CDMX, México.
ORCID: `https://orcid.org/0000-0001-9276-4988`
e-mail: `lvasquezt@xanum.uam.mx`