ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DISEÑO, SIMULACIÓN, E IMPLEMENTACIÓN DE CÓDIGOS DE CANAL EN SISTEMAS OFDM"

TESINA DE SEMINARIO

Previo a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Freddy Manuel Orosco Villamagua

Claudia Sofía Pintos Castro

Guayaquil – Ecuador

2013

AGRADECIMIENTO

Agradezco a Dios por su infinita bondad, a mis padres por su amor y apoyo constante, a mis queridas hermanas por su cariño brindado, a Sofía mi compañera incondicional y a esta, mi universidad que junto a sus docentes han sido mentores de mis conocimientos.

Freddy Manuel Orosco Villamagua

Agradezco a Dios por darme fortaleza, a mi mamá por su apoyo constante, al Ingeniero Jorge Gómez por sus valiosos consejos, al Dr. Boris Ramos por ser nuestro guía, a Freddy por ser un buen compañero y a la Espol por brindarme conocimientos durante los últimos cinco años.

Claudia Sofía Pintos Castro

DEDICATORIA

Este trabajo va dedicado a mis padres, quienes con su paciencia y consejos me brindaron siempre apoyo incondicional y un ejemplo a seguir.

Freddy Manuel Orosco Villamagua

Este trabajo va dedicado a los futuros ingenieros en telecomunicaciones del país para que no se rindan y continúen sus estudios en una de las ramas más asombrosas de la ingeniería.

Claudia Sofía Pintos Castro

TRIBUNAL DE SUSTENTACIÓN

PhD. BORIS RAMOS

PROFESOR DEL SEMINARIO

DE GRADUACIÓN

ING. JORGE GÓMEZ

PROFESOR DELEGADO POR

LA UNIDAD ACADÉMICA

DECLARACIÓN EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestas en esta tesina de seminario, nos corresponden exclusivamente; y el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de Graduación de la ESPOL)

Freddy Manuel Orosco Villamagua

Claudia Sofía Pintos Castro

RESUMEN

El presente proyecto se enfoca en el diseño y simulación de un sistema de comunicaciones inalámbrico empleando métodos de corrección de errores con el objetivo de obtener el menor error posible durante el proceso de transmisión y recepción; siempre tomando en cuenta las capacidades de infraestructura así como la complejidad del sistema.

Empezaremos realizando una explicación que englobará conceptos importantes sobre comunicaciones inalámbricas, se presentarán diagramas de bloques que nos permitan entender el funcionamiento del transmisor y del receptor y su implementación. A continuación estudiaremos los factores que pueden afectar nuestra comunicación durante su transmisión por el medio o canal.

Luego se presentará el diseño y desarrollo de los bloques elaborados en LABVIEW explicando en detalle cada uno de los componentes de los mismos. En ésta parte también se indicarán los parámetros a configurar en los respectivos paneles frontales de los instrumentos virtuales (Vis) necesarios para poder establecer nuestro canal de comunicaciones.

Finalmente se mostraran los resultados obtenidos a través de diagramas de constelaciones y de gráficas Tasa de error (BER) vs Relación Señal a Ruido (SNR). Se realizará un análisis del desempeño de dos sistemas de corrección de errores sobre tres diferentes canales utilizando diferentes modulaciones.

ÍNDICE GENERAL

ABREVIATURAS

NI	National Instruments
USRP	Universal Software Radio Peripheral
SDR	Software Defined Radio
BPSK	Binary Phase Shift Keying
QPSK	Quadrature Phase Shift Keying
QAM	Quadrature Amplitude Modulation
OFDM	Orthogonal Frequency Division Mutiplexing
COFDM	Coded Orthogonal Frequency Division Mutiplexing
AWGN	Additive White Gaussian Noise
FEC	Forward Error Correction
ARQ	Automatic Repeat Request
ISI	Inter-Symbol Interference
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
BER	Bit Error Rate
SNR	Signal-to Noise Ratio

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Desde las primeras comunicaciones se ha buscado asiduamente que la información pueda llegar a su destino sin sufrir modificaciones de un extremo a otro. Nuestro proyecto se enfoca en implementar y analizar el comportamiento de ciertos algoritmos de codificación para corrección de errores sobre un sistema de comunicaciones trabajando sobre OFDM frente a posibles errores introducidos por los canales AWGN, ISI y Rayleigh. A éste sistema de comunicaciones se lo conoce como COFDM.

El objetivo de este proyecto es ofrecer al lector mayor información sobre métodos existentes para obtener una comunicación más fiable y robusta, garantizando que en la mayoría de los casos la información viaje segura hasta su destino, obteniéndose menor cantidad de errores e incluso poder corregir muchos de los mismos.

En el capítulo uno se dará una breve explicación de las razones que motivan el estudio de esquemas de codificación para control de errores, así como la importancia que merece estudiar su desempeño en diferentes medios considerando tres diferentes modulaciones.

En el segundo capítulo nos enfocaremos en el marco teórico que sustenta cada una de las partes fundamentales del proyecto; transmisor, receptor y medio inalámbrico (canal). Se estudiará las causas más comunes para la aparición de errores en las secuencias recibidas analizada desde los puntos de vista: tipo de modulación, la potencia del ruido y otras interferencias e imperfecciones que puedan ser introducidas por el medio; finalizando éste capítulo estudiaremos los algoritmos a ser implementados para detección y corrección de errores: Códigos de bloques lineales y Códigos convolucionales.

De manera puntual en el capítulo tres explicaremos el proceso de implementación en el software LabView de los algoritmos de detección y corrección de errores arriba cuyo sustento se estudiará en el capítulo 2, priorizando el rendimiento del sistema y evitando una alta complejidad en su implementación.

Seguido a esto en el capítulo cuatro nos enfocaremos en la simulación de nuestros algoritmos creados en LABVIEW, haciendo variaciones dentro de los parámetros de transmisión y probando a su vez sobre códigos de bloques lineales y códigos convolucionales generando los diagramas de constelaciones recibidos, variaciones de error, y curvas BER vs SNR.

Para cumplir con nuestro objetivo finalmente se mostrarán las conclusiones a las que hemos llegado luego de una investigación y experimentación extensa sobre el tema, así como los inconvenientes tenidos durante el desarrollo de la misma, las acciones tomadas y algunas teorías que en un futuro podrían hacer nuestro sistema aún más robusto.

CAPÍTULO 1

PLANTEAMIENTO DEL PROBLEMA

1.1 DESCRIPCIÓN DEL PROBLEMA

La comunicación en la actualidad se ha convertido en uno de los factores claves de un país y del mundo entero estando muy ligada incluso a su nivel de desarrollo socioeconómico. Cada vez los sistemas, alámbrico e inalámbricos son utilizados con mayor frecuencia en diversas aplicaciones, lo que exige que estos cada vez sean más robustos para evitar ataques mal intencionados y garantizar que la información que nos llegue sea correcta. Existen muchos procesos en donde se requiere una transmisión/recepción casi perfecta y como consecuencia los mensajes de un extremo al otro deben ser recibidos sin errores.

La modulación ortogonal por división de frecuencias (OFDM) es un esquema de modulación presente en la mayoría de los modelos de comunicación inalámbricos de banda ancha. A diferencia de las comunicaciones de única portadora, donde cada símbolo se transmite individualmente ocupando todo el ancho de banda disponible, la modulación multiportadora envían los símbolos paralelamente en sub-portadoras adyacentes, utilizando un método de multiplexación por división de frecuencias.

La detección y corrección de errores, junto a la encriptación de información, son bloques fundamentales en los sistemas de comunicación modernos. Lo que se busca al implementar éstos procesos es reducir la tasa de errores en el receptor, de modo que el mensaje recibido después de haber sido procesado en el receptor sea lo más cercano al transmitido. Si esto lo implementamos en nuestra modulación OFDM tenemos algo conocido como COFDM (Multiplexación por División de Frecuencia Ortogonal Codificada).

Para llevar a cabo nuestro objetivo primero implementaremos algoritmos de codificación y decodificación de canal en el software LABVIEW, implementados en módulos de comunicación inalámbricos NI-USRP desarrollados por National Instruments.

Realizaremos especial énfasis en analizar cómo influye el algoritmo de codificación de canal en la tasa de errores considerando variaciones en el medio y variaciones a la modulación.

1.2 OBJETIVOS GENERALES

En el presente proyecto nos enfocaremos en implementar dos algoritmos de detección y corrección de errores: códigos de bloques lineales y códigos convolucionales. Utilizaremos herramientas matemáticas como la operación de matrices a través de polinomios generadores, así como también el uso de diagramas de estados.

Identificar las ventajas que ofrecen los sistemas que usan codificación de canal frente a uno que no lo hace. Diferenciar los tipos de codificación y realizar un análisis en realización a su eficiencia tomando como base la tasa de error de bit BER.

Someter nuestro sistema a diferentes pruebas para observar el funcionamiento del mismo en lo que sería diferentes aplicaciones reales de la vida cotidiana. Esto incluye, variar el tipo de modulación utilizada, simular diferentes canales e incluso aumentar la potencia del ruido frente

a la señal transmitida, de modo que se entienda el efecto de esto sobre nuestro mensaje y cómo los algoritmos trabajan para poder sobrellevarlo.

1.3 JUSTIFICACIÓN

La importancia que nuestros mensajes lleguen al receptor seguro y con la menor cantidad de errores es la motivación de este proyecto. En toda transmisión se busca abolir el efecto del canal sobre la señal, esto es el efecto multipaso y la interferencia intersimbólica; las técnicas de detección y corrección de errores nos permiten hacer frente a estos problemas.

Una reducción de la tasa de error de bit conlleva a un mensaje en el receptor muy próximo al mensaje original. Poder diferenciar entre los diferentes tipos de codificación de canal que existen a fin de obtener el máximo beneficio para cada aplicación que se realice, así como analizar sus ventajas y desventajas.

CAPÍTULO 2

FUNDAMENTO TEÓRICO

En éste capítulo se explicarán los principios de funcionamiento de un sistema OFDM codificado. Inicialmente se realizará una breve exposición acerca de cada uno de los bloques que componen un sistema de comunicaciones digital haciendo especial énfasis en sistemas OFDM. A continuación expondremos las características más importantes de los canales AWGN, ISI y Rayleigh, canales sobre los cuales se observó el comportamiento del sistema codificado. Los fundamentos y características matemáticas que nos permitan comprender el funcionamiento de los códigos de bloques lineales y códigos convolucionales serán detallados al final del presente capítulo.

2.1 EL SISTEMA DE COMUNICACIONES DIGITAL

Un sistema de comunicaciones digital básicamente está conformado por los bloques descritos en la figura 2.1.

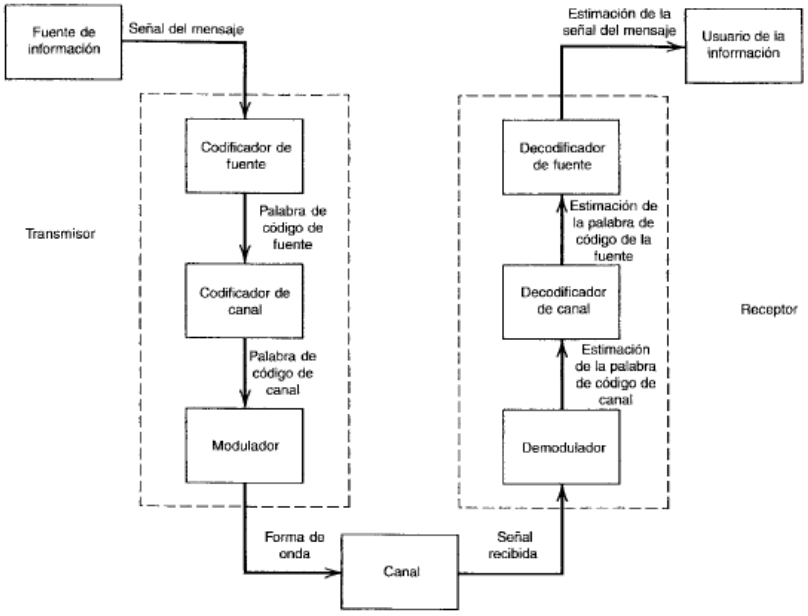


Fig. 2.1 Esquema de un sistema de comunicaciones estándar [1]

El número de bloques a utilizar en un sistema de comunicaciones real puede ser más extenso, dependiendo de las características de propagación del medio y de los problemas que podrían afectar a las señales durante el proceso de transmisión y recepción. En el caso del sistema de comunicaciones OFDM a estudiarse, está compuesto por bloques adicionales como ecualizadores de canal, sincronizadores de trama, entre otros; sin embargo nuestro estudio se centra en los bloques codificador/decodificador de canal.

2.1.1 TRANSMISOR

En el transmisor básicamente tenemos los siguientes bloques: Codificador de Fuente, Codificador de Canal y Modulador.

El codificador de fuente se encarga de comprimir la información que lleva el mensaje eliminando partes del dicho mensaje que sean redundantes comparando las demás partes del mismo, de ésta manera el resultado es una señal cuya transmisión por el canal será más eficiente que si no existiera éste bloque. Al pasar por codificador de fuente se obtiene una secuencia denominada palabra de código de fuente.

El codificador de canal se encarga de agregar redundancia de manera controlada a la palabra de código de fuente previendo errores que puedan afectar la decodificación del mensaje original, lo cual produce una nueva secuencia de símbolos denominada la palabra de código del canal.

Finalmente el modulador se encarga de representar la palabra de código de canal en símbolos analógicos, la secuencia de estos

símbolos analógicos se denomina forma de onda, la cual es adecuada para la transmisión por el canal [1].

2.1.2 CANAL

Podemos ver un canal de comunicación como el enlace entre dos puntos a lo largo de una trayectoria de comunicación. Los medios de transmisión pueden ser en forma general alámbricos e inalámbricos. Los medios alámbricos comúnmente utilizados son el cable de cobre, cables coaxiales y fibra óptica, siendo esta última la más utilizada en la actualidad debido a que proporciona características de menor atenuación y vulnerabilidades a fenómenos eléctricos. Sin embargo, la principal desventaja que presentan los medios alámbricos son los costos asociados a su implementación y mantenimiento, así como en la falta de movilidad que proporciona a los dispositivos finales. Es por esto que el uso de tecnologías inalámbricas de comunicaciones es cada vez más frecuente para muchas aplicaciones. Por otro lado las señales que viajan por el medio aire son más susceptibles a distorsiones por fenómenos como desvanecimientos, ruido, falta de línea de vista y otros efectos multitrayectoria. Es por ésto que el diseñador de un sistema de comunicaciones inalámbrico debe utilizar algoritmos

que aseguren que el mensaje llegará de un extremo a otro con la mínima distorsión posible.

2.1.3 RECEPTOR

En el receptor tenemos básicamente los siguientes bloques: Demodulador, Decodificador de Canal y Decodificador de Fuente.

El demodulador obtiene las señales analógicas del medio, esto es, recibe señales atenuadas y desfasadas y se encarga de realizar una aproximación de la posible secuencia de símbolos binarios conociendo el alfabeto de señales y las características de modulación del transmisor (tipo de modulación, frecuencia, ancho de banda, intervalos de guarda).

El decodificador de canal, con ayuda de los bits de redundancia agregados en el transmisor, se encarga de detectar si hubo errores en la trama del mensaje recibido, inclusive podría, dependiendo del tipo de codificación, corregir uno o más de estos errores.

El decodificador de fuente recibe los bits decodificados y entrega al usuario la información original prediciendo la redundancia que fue eliminada en el transmisor, de modo que pueda ser utilizada por la aplicación que lo haya requerido.

2.2 MODULACIÓN EN AMPLITUD Y FASE

La información cuando requiere ser transmitida de un lugar a otro sobre un medio compartido por otros usuarios debe ser modulada con el fin de poder ser detectada por el receptor sin problemas. Algunas técnicas de modulación incluyen cambios en su amplitud, frecuencia, fase, tiempo, entre otras. En la modulación en amplitud y fase el flujo de bits de información está contenido en la amplitud y/o fase de la señal transmitida s(t), 0≤t<Ts, cuya representación en banda base compleja es:

$$S(t) = \mathcal{R}\{x(t)e^{j(2\pi f_c t)}\} \quad \textbf{(2.1)}$$

Donde x(t) = sI(t) + jsQ(t) = (si1 + jsi2) g(t), siendo (si1,si2) el símbolo a ser transmitido y perteneciente a la constelación de señales y Ts es el periodo de símbolo. La tasa de bits para esta modulación es K bits por símbolo o R=$\log_2$M/Ts símbolos por segundo.

2.2.1 PHASE SHIFT KEYING (PSK)

Para MPSK toda la información está codificada en la fase de la señal transmitida, entonces durante un periodo de símbolo tiene la siguiente forma:

$$s_i(t) = si_1 \cos(2\pi fct) - si_2 \operatorname{sen}(2\pi fct) \quad \textbf{(2.2)}$$

Los símbolos $(si1, si2)$ están dados por $si1 = Acos\left[\frac{2\pi(i-1)}{M}\right]$ y $si2 = Asen\left[\frac{2\pi(i-1)}{M}\right]$ para i=1,…,M, siendo A la amplitud de la señal y M el número de símbolos. La constelación de señales puede tener un desfase dado por $\theta i = \frac{2\pi(i-1)}{M}\ siendo\ i = 1,2,\dots,M$. La distancia mínima entre los puntos de la constelación es dmin=2Asen(π/M) donde A es función de la energía de la señal. La modulación 2PSK es más conocida como BPSK así como 4PSK es más conocida como QPSK y es la misma que 4QAM [2].

Todas las posibles señales transmitidas si(t) tienen la misma energía:

$$Esi = A^2 \quad \textbf{(2.3)}$$

Las figuras 2.2 y 2.3 ilustran las constelaciones de símbolos para BPSK y QPSK respectivamente.

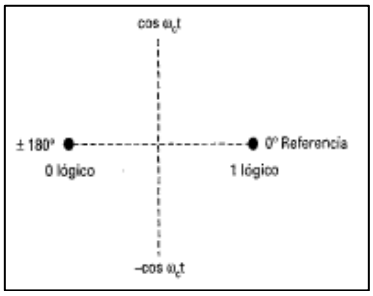


Fig. 2.2 Diagrama de constelación BPSK [3]

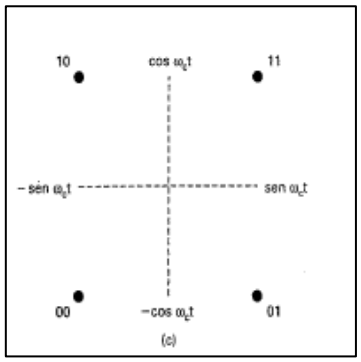


Fig. 2.3 Diagrama de constelación BPSK [3]

La regla de decisión de máxima verosimilitud para BPSK será:

$$Zi = \begin{cases} 0; \Re(si) \geq 0 \\ 1; \Re(si) < 0 \end{cases} \quad (2.4)$$

La regla de decisión de máxima verosimilitud para QPSK será:

$$Zi = \begin{cases} 00; \Re(si) \geq 0 \; y \; Im(si) \geq 0 \\ 01; \Re(si) < 0 \; y \; Im(si) \geq 0 \\ 11; \Re(si) < 0 \; y \; Im(si) < 0 \\ 10; \Re(si) \geq 0 \; y \; Im(si) < 0 \end{cases} \quad (2.5)$$

2.2.2 MODULACIÓN DE AMPLITUD EN CUADRATURA (QAM)

Para MQAM la información está codificada en la amplitud y en la fase de la señal transmitida. Esta característica hace que QAM tenga una mejor eficiencia espectral que MPAM y MPSK ya que puede codificar una mayor cantidad de bits por símbolo dada una cantidad de energía [2].La señal transmitida en QAM durante un periodo de símbolo Ts tiene la siguiente forma:

$$si(t) = Aicos(\theta i)\cos(2\pi fct) - Aisen(\theta i)\,\mathrm{sen}(2\pi fct) \quad (2.6)$$

La energía de la señal si(t) es :

$$Esi = A^2 \quad \textbf{(2.7)}$$

La distancia mínima entre cualquier par de símbolos i y j se la puede encontrar como la distancia entre dos puntos, esto es:

$$dij = \sqrt{(s_{i1} - s_{j1})^2 + (s_{i2} - s_{j2})^2} \quad \textbf{(2.8)}$$

Algunas constelaciones QAM cuadradas son 4QAM y 16QAM que se muestran en la figura 2.4.

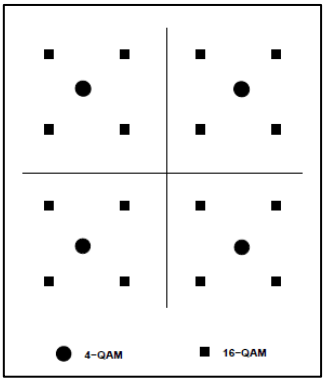


Fig. 2.4 Diagrama de constelación 4QAM y 16QAM[2]

2.3 ¿QUÉ ES OFDM?

OFDM (Orthogonal Frecuency-Division Multiplexing) es una técnica de modulación digital cuya función principal es dividir un ancho de banda asignado en múltiples sub-bandas o sub-canales, cada uno con su frecuencia central las cuales son ortogonales entre sí, multiplexando la información a ser enviada entre los sub-canales obtenidos.

Gracias a que frecuencias centrales de los sub-canales son ortogonales se dice que el cross-talk o interferencia con el sub-canal adyacente se elimina, por lo tanto no es necesario utilizar filtros para cada sub-canal simplificando de esta manera el diseño tanto del transmisor como del receptor.

Cada uno de los sub-canales tiene su propio sub-ancho de banda, el cual es relativamente pequeño debido a que normalmente el número de sub-portadoras es muy grande; con esto es posible modelar la atenuación de cada sub-canal prácticamente constante, lo cual simplifica el bloque de ecualización respecto a los utilizados en sistemas de portadora única. Por lo expuesto, se puede concluir que la técnica OFDM permite una mayor eficiencia del uso del espectro permitiendo acercarse a la capacidad máxima del canal.

De manera conceptual OFDM ha existido durante décadas; pero su implementación real y a costos aceptables no fue posible sino con el advenimiento y propagación de tecnologías como los microprocesadores de alta velocidad de procesamiento y los dispositivos de lógica programable.

Uno de los descubrimientos que hizo posible la implementación real de OFDM es la utilización de la Transformada Rápida de Fourier o FFT para lograr la transmisión por medio de sub-portadoras paralelas de manera que se elimina en lo posible la interferencia o traslape entre ellas. Por ésta razón se puede decir que el número de sub-portadoras está ligado al número de muestras que usa la FFT identificada por N_{FFT}.

2.3.1 ESQUEMA DE UN SISTEMA OFDM

El diagrama de bloques mostrado en la figura 2.5 nos ilustra el esquema de un sistema OFDM:

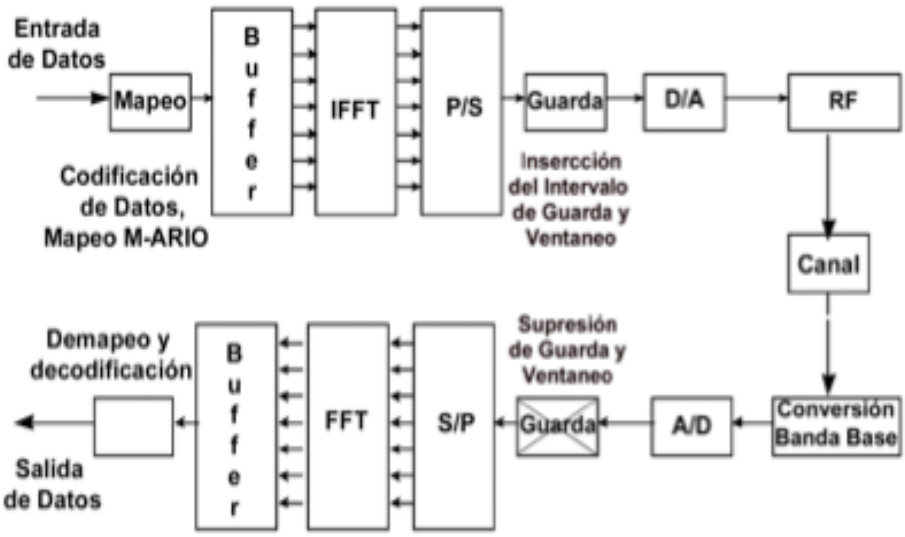


Fig. 2.5 Esquema de un sistema que modula en OFDM[4]

Para explicar el funcionamiento de un sistema OFDM consideraremos una secuencia de bits de longitud igual a 500. A la entrada del sistema OFDM ingresa una trama que contiene bits en serie los cuales pueden o no estar codificados. Estos bits primero se convierten en símbolos, de acuerdo a una regla de correspondencia determinada por el tipo de modulación escogida tales como M-PSK o M-QAM; a éste proceso se le conoce como mapeo [4]. Podríamos suponer que la cantidad de símbolos generados son 200, entonces, tenemos hasta el momento una matriz de 1 x 200.

Los símbolos obtenidos son temporalmente almacenados en un Buffer cuya función es almacenar todos los símbolos generados por el bloque anterior y tomar grupos de N símbolos, asumiendo que N=10, entonces tomaría el grupo total de 200 símbolos y los organiza en sub-grupos de 10 símbolos, es decir, tendríamos una matriz de 20 x 10.

Ésta matriz entra a continuación por la Transformada Inversa Rápida de Fourier IFFT cuya función es encontrar la función en el dominio del tiempo que representan cada grupo de símbolos. Si recordamos nuestra matriz de 20 x 10, la IFFT toma la primera fila de dicha matriz, el cual es un arreglo de 1 x 10 símbolos. Como se asocian los símbolos con el dominio de la frecuencia, internamente se asocia cada uno de los diez símbolos con una portadora ortogonal diferente, es decir, en este caso se tendrían 10 portadoras ortogonales. Con ésta información, se encarga de estimar la señal en el dominio del tiempo representada por dicho espectro de frecuencias y lo que envía en la correspondiente salida son N coeficientes de dicha señal, en nuestro ejemplo anterior serían 10 puntos.

El bloque IFFT se encarga de generar internamente múltiples portadoras ortogonales entre sí; esto lo hace dividiendo el ancho de banda asignado para un número N determinado. El proceso descrito anteriormente es conocido como la modulación OFDM.

Éste proceso se realiza con cada una de las 20 filas que inicialmente se envía del Buffer al mismo tiempo, de modo que la IFFT envía como salida una matriz de 20x10. La unidad de datos obtenida hasta ésta parte del diagrama de bloques recibe el nombre de Símbolo OFDM.

La siguiente parada de nuestro grupo de datos será un convertidor de paralelo a serial, éste convierte nuestra matriz en un arreglo de una dimensión, colocando al inicio la primera fila, seguido de la segunda fila, después la tercera y así sucesivamente.

Previo a la conversión de digital a analógico, se adiciona el intervalo de guarda al final de cada símbolo OFDM, importante para prevenir la interferencia entre símbolos OFDM causada por la multitrayectoria de la señal sobre el canal inalámbrico.

Finalmente, la señal analógica se pasa por un bloque de RF, de modo que la señal obtenida es adecuada para ser enviada por el medio.

La señal se transmite a través de un canal que agrega desvanecimiento multitrayectoria y ruido aditivo blanco gaussiano (AWGN) [4].

Del lado del receptor se esperaría realizar el proceso contrario, esto es, primero debe realizarse la conversión de RF a una señal banda base y en seguida convertirse de señal analógica a señal digital.

A continuación se pasa por un bloque de Sincronización, para conocer exactamente desde dónde empiezan los datos. Después de haber sido sincronizado, se elimina el prefijo cíclico del símbolo OFDM recibido.

Previo a pasar por el bloque FFT, debe convertirse los datos a formato paralelo para que puedan ser procesados adecuadamente. Es decir, en el ejemplo anterior en el transmisor generamos una

matriz de 1x200, de modo que deben nuevamente ser convertidos a una matriz de dos dimensiones de 20x10.

El siguiente bloque es la Transforma Rápida de Fourier que convierte nuestras “muestras” que representan la señal en el tiempo a los correspondientes coeficientes en el dominio de la frecuencia, siendo la salida de éste bloque una matriz de 20x10 para el ejemplo tratado, sabiendo que cada una de las columnas es asociada con cada una de las frecuencias ortogonales.

Los bits de la FFT de la señal en el dominio del tiempo corresponden al peso de cada una de las portadoras, las cuales son subsecuentemente desintercaladas, decodificadas con el algoritmos apropiado y mapeadas inversamente para dar un estimativo de los originales.

2.3.2 ORTOGONALIDAD

La clave para la eficiencia espectral de OFDM sobre su antecesor FDM radica en la ortogonalidad de sus portadoras, por ello es posible que las bandas laterales en el dominio de la frecuencia de las portadoras se solapen sin que exista interferencia co-canal.

Dos frecuencias son ortogonales cuando la separación entre ellas es exactamente igual al recíproco del periodo del símbolo útil, es decir, que la separación en frecuencia sea f=1/T, lográndose que las portadoras no se interfieran entre sí, como se muestra en la siguiente figura:

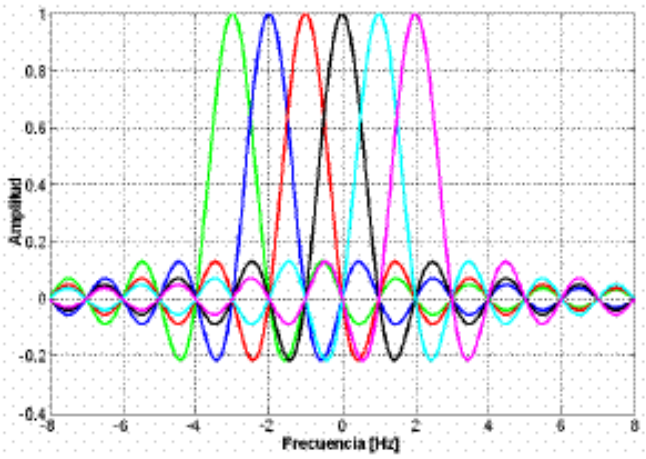


Fig. 2.6 Espectro de una señal OFDM con 6 sub-portadoras[5]

Las portadoras forman lo que se conoce como un conjunto ortogonal, debido que la portadora k-ésima (en banda base) puede escribirse como:

$$\varphi_k(t) = e^{jkw_u t} \qquad (2.9)$$

Donde $w_u = 2\pi/T$, por lo tanto cada una de las portadoras deben cumplir la siguiente condición:

$$\int_{\tau}^{\tau+T_u} \varphi_k(t)\varphi_1^*(t)dt = 0; k \neq 1 \qquad (2.10)$$

Todo esto con la finalidad de poder traslapar varias portadoras en el mismo espacio que ocuparían pocas portadoras empleando FDM, reduciendo de ésta manera el ancho de banda utilizado en comparación con FDM. En un sistema OFDM la ortogonalidad en el dominio de la frecuencia se logra a través de la conversión a símbolos que realiza el mapeador.

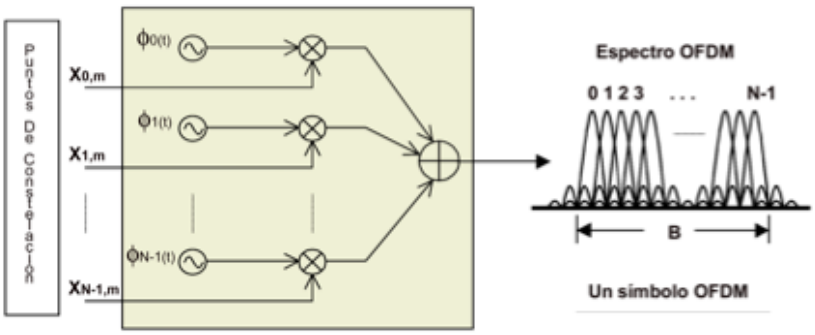


Fig. 2.7 Espectro OFDM[4]

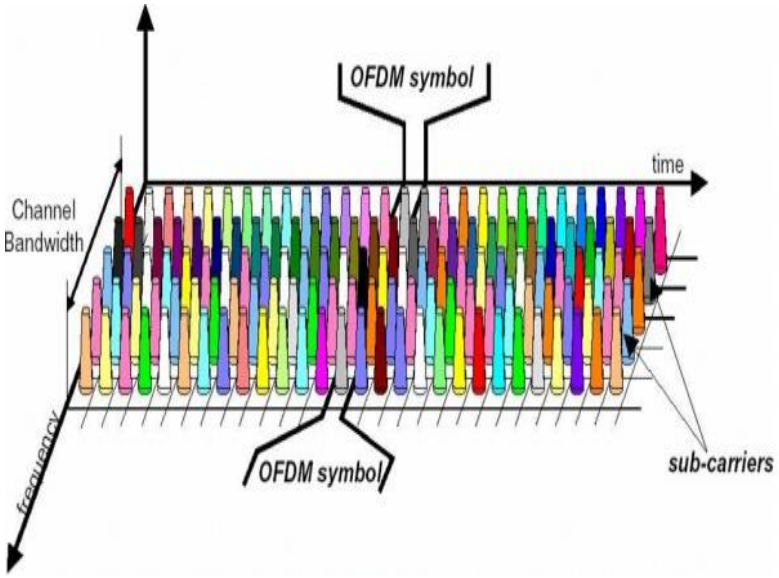


Fig. 2.8 Distribución de símbolos OFDM[6]

2.3.3 TRANSFORMADA RÁPIDA DE FOURIER FFT/IFFT

Es evidente que la implementación en hardware de un sistema FDM como el mostrado en la figura 2.9 para un número alto de portadoras, no sería posible ya que se requeriría de miles de osciladores, filtros, multiplicadores e integradores, es decir, sería un sistema muy costoso por el número de equipos requeridos sino también por su alto consumo de potencia.

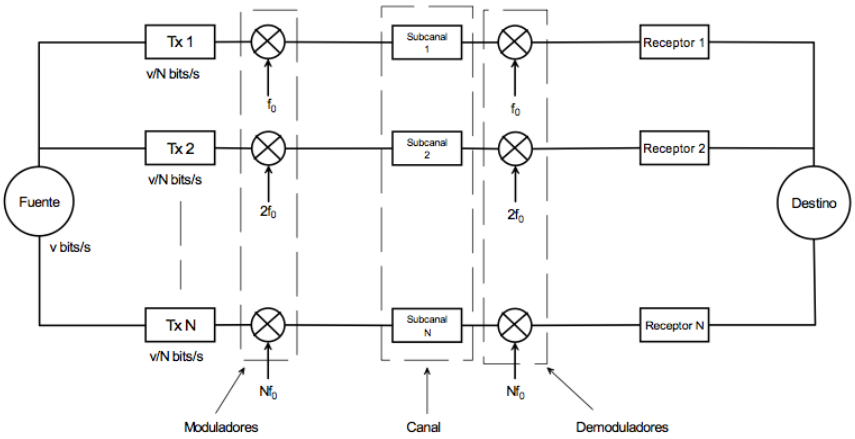


Fig. 2.9 Multiplexado por división de frecuencia (FDM)[7]

Lo que hizo que la modulación OFDM pueda ser implementada en sistemas de comunicación comerciales es la utilización de la transformada inversa de Fourier. Debido a que las señales están compuestas por componentes de frecuencias ortogonales, se evita el empleo de filtros y permite trabajar con la señal muestreada, como es de esperarse, a una frecuencia superior a la frecuencia de Nyquist.

En estas condiciones, el proceso de integración se convierte en uno de suma y todo el proceso de demodulación es idéntico a una transformada discreta de Fourier. En la actualidad existe la disponibilidad de numerosos circuitos integrados que permiten realizar estas operaciones, con lo que la implementación práctica del modulador y demodulador OFDM resulta relativamente sencilla.

La etapa correspondiente a IFFT/FFT en nuestro esquema mencionado forma una de las partes esenciales en un esquema de modulación OFDM.

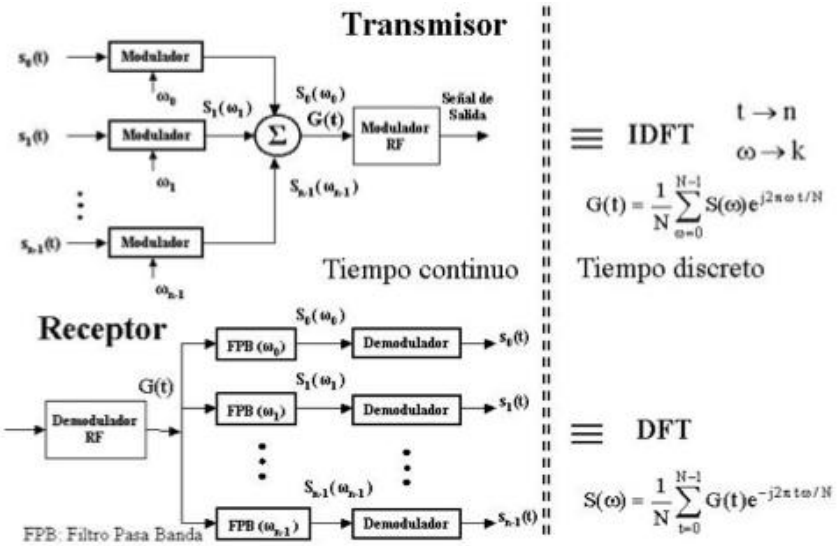


Fig. 2.10 Equivalencia de FDM y DFT[8]

Para concluir con este apartado decimos que la Transformada de Fourier F{x(t)} constituye la relación entre la señal x(t) y su representación en el dominio de la frecuencia X(w) definida de la siguiente manera:

$$X(w) = \int_{-\infty}^{+\infty} x(t)e^{-jwt}dt = F\{x(t)\} \qquad (2.11)$$

De la misma forma, la transformada inversa de Fourier $F^{-1}\{X(w)\}$ constituye la relación entre la señal X(w) y su representación en el dominio del tiempo x(t) definida de la siguiente manera [7]:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(w)e^{jwt}\,dw = F^{-1}\{X(w)\} \qquad (2.12)$$

2.3.4 INTERVALO DE GUARDA (PREFIJO CÍCLICO)

Cada una de las sub-portadoras está modulada por los símbolos representados por números complejos. Previendo que no todas las componentes de las frecuencias portadoras llegarán al mismo tiempo al receptor, es decir, llegarán vestigios de la señal original retrasada que pudieran causar interferencia con los demás símbolos y por consiguiente en sus frecuencias portadoras [4]. Para evitar esta situación, se agrega un intervalo de guarda, como se muestra en la figura 2.11.

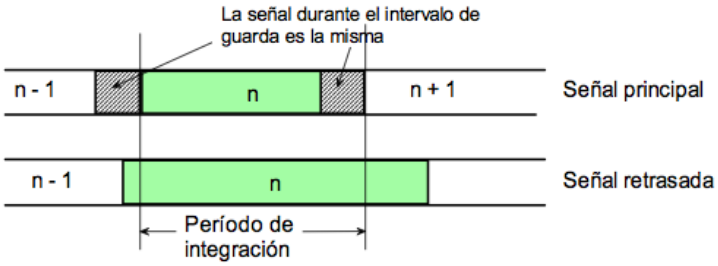


Fig. 2.11 Adición del intervalo de guarda[8]

El procedimiento a seguir es el siguiente, se agrega una porción de la señal tanto al inicio como al final de la misma. Ésta porción corresponde a la última parte de la señal a transmitirse, por lo que nuestro transmisor se ve obligado a extender la duración de los símbolos OFDM de modo se exceda el periodo de integración Tu. Todas las subportadoras son cíclicas durante *Tu*, de modo que también lo es la señal modulada completa [4].

Siempre que el retardo que sufra la señal sea menor que el intervalo de guarda, todas las componentes de la señal durante el período de integración proceden del mismo símbolo. La interferencia entre símbolos o entre portadoras ocurrirá solamente cuando el retardo relativo exceda la duración del intervalo de guarda.

Mientras mayor sea el intervalo de guarda asignado a cada símbolo OFDM, menor será la interferencia causada por los efectos multipaso. Por otro lado, la adición de éste intervalo de guarda ocasiona una pérdida de la eficiencia espectral y un mayor requerimiento de potencia en el transmisor.

La pérdida de eficiencia espectral es causada al transmitir muestras duplicadas que no aportan nueva información. El número de muestras que contiene el Intervalo de Guarda para algunos casos se ilustra en la figura 2.12 donde el ancho del Intervalo de Guarda (Δ) es A × Tu siendo A=1/2n; n = 2, 3, 4, 5, 6.

Número de puntos de la FFT	**Ancho del Intervalo de Guarda**				
	1/4	1/8	1/16	1/32	1/64
64	16	8	4	2	1
128	32	16	8	4	2
256	64	32	16	8	4
512	128	64	32	16	8
1024	256	128	64	32	16
2048	512	256	128	64	32
4096	1024	512	256	128	64
8192	2048	1024	512	256	128

Fig. 2.12 Número de muestras contenidos según el Intervalo de Guarda [4]

2.3.5 SINCRONIZACIÓN DEL CANAL

Para que la demodulación pueda llevarse a cabo sin problemas, en el receptor se deben tomar las muestras exactamente mientras dure el periodo útil del símbolo OFDM, esto es, sin contar el intervalo de guarda agregado anteriormente.

Podríamos hacer la analogía de los sistemas de comunicación analógicos, en los que a éste proceso se le llamaba demodulación coherente o síncrona en el receptor, para lo cual es indispensable que estén sincronizadas la portadora del transmisor y del receptor, en la misma frecuencia y en la misma fase.

Para nuestro sistema OFDM, se sincronizan ubicando sub-portadoras “pilotos”, como se muestra en la figura 2.13, las cuales están distribuidas en el canal de transmisión y que actúan como “marcas de sincronismo”.

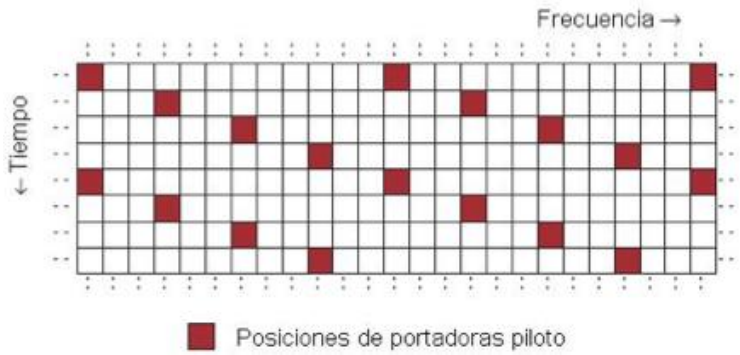


Fig. 2.13 Distribución de portadoras piloto[4]

La información de las señales piloto es conocida, por lo tanto, el receptor utiliza ésta condición para realizar una estimación de la respuesta en frecuencia del canal. La estimación así obtenida para una portadora piloto puede interpolarse para llenar los huecos que separan a los pilotos y emplearse para ecualizar todas las constelaciones que transportan datos [4].

Las portadoras pilotos se utilizan para realizar la sincronización en tiempo y frecuencia, así también son útiles en el proceso de ecualización y la corrección de errores de fase común en el receptor. Son transmitidas con un nivel de potencia reforzado E=16/9, son las encargadas de llevar información de referencia, la cual es generada por un aleatorizador de energía, su esquema de modulación es BPSK o DPBSK.

Las portadoras pilotos se dividen en 2 tipos:

- ✓ Portadoras Piloto Fijas: Son aquellas que siempre están en las mismas posiciones dentro de un símbolo OFDM. Usadas para la sincronización en tiempo y frecuencia, la ecualización y la corrección de error de fase común en el receptor.

- ✓ Portadoras Piloto Dispersas: Son aquellas cuya posición varía dentro de un símbolo OFDM siguiendo un patrón conocido por ejemplo cada 4 símbolos OFDM. Son utilizadas para la sincronización en tiempo y frecuencia y para la ecualización en el receptor.

2.4 MULTIPLEXACIÓN POR DIVISIÓN DE FRECUENCIA ORTOGONAL CODIFICADA (COFDM)

Hemos llegado a éste punto, al cual le dedicaremos mucho tiempo pues será sobre lo que se enfocará toda nuestra investigación. La codificación es uno de los aspectos más importantes en un sistema de comunicaciones COFDM, pues dependiendo del algoritmo a utilizarse será posible la detección y corrección de errores, así como también la

utilización de mecanismos de encriptación que garanticen que nuestra información viaje de un lugar a otro de manera segura.

2.4.1 ¿QUÉ ES COFDM?

Es una técnica compleja de modulación de banda de banda ancha utilizada para transmitir información digital a altas velocidades, a través de un canal de comunicaciones, cuya base es el sistema de comunicaciones OFDM. A parte de combinar las bondades de OFDM expuestas en la sección 2.3, agrega bloques de codificación y decodificación de canal y usualmente se agrega también un bloque de entrelazamiento [9].

De ésta manera, se obtiene una modulación específicamente diseñada para combatir los efectos del multitrayecto y otros tipos de interferencias que afectan a receptores fijos y móviles. Comparando el sistema COFDM con otros sistemas de comunicación digital se obtiene que COFDM comparte los bloques de modulación y codificación del canal. Tal como ya lo hemos analizado COFDM tiene la misma estructura que OFDM con la única diferencia que ésta añade codificación antes que la señal sea modulada y transmitida y el receptor realiza la decodificación.

2.4.2 COMPARACIONES Y DIFERENCIAS ENTRE COFDM Y OFDM

Para canales muy selectivos o variantes, COFDM presenta mejores características que OFDM ya que puede soportar multi-trayecto severo, la presencia de interferencia de banda estrecha co-canal, la cancelación de la señal, el ruido de impulsos y la reducción rápida de la amplitud de la señal [9].

A pesar que la codificación en sí proporciona mejores bondades al sistema OFDM, éste puede ser utilizado junto con el entrelazamiento de portadoras.

Algunas características comunes entre COFDM y OFDM son:

- ✓ La presencia de ortogonalidad de las frecuencias subportadoras.
- ✓ La modulación de las subportadoras.
- ✓ La presencia de Intervalos de Guarda entre cada símbolo.

- ✓ La sincronización por medio de portadoras piloto.

- ✓ Requiere un bloque de ecualización.

Algunas mejoras de COFDM frente a OFDM son:

- ✓ Presencia de codificación de control de errores.

- ✓ El entrelazamiento de las portadoras de datos en frecuencia o en tiempo y frecuencia [8].

2.5 CANAL DE COMUNICACIONES INALÁMBRICO

Un canal de comunicaciones inalámbrico hace referencia al uso del aire como medio de propagación entre dos extremos, emisor y receptor. En general, la tecnología inalámbrica utiliza ondas de radiofrecuencia de baja potencia y una frecuencia de portadora específica, que puede ser de uso libre o privado, para poder comunicarse entre dispositivos. La búsqueda de la movilidad y la ubicuidad ha conllevado a que las comunicaciones inalámbricas avancen a un ritmo acelerado; sin embargo la capacidad de estos sistemas está limitada por el canal de propagación. La propagación por multitrayectoria, presente en los canales móviles de radio, ocasionada

por la difracción y dispersión debido a las características del terreno y edificios es una de estas limitaciones.

La propagación por multitrayectoria ocasiona distorsión en los sistemas analógicos y afecta severamente el desempeño de los sistemas digitales reduciendo las relaciones señal a ruido y la señal a interferencia. Conocer el modelo matemático que rige al canal es de suma importancia pues le permite al diseñador adoptar mecanismos que ayuden a eliminar los efectos negativos introducidos por el canal.

2.5.1 PARÁMETROS DEL CANAL DE COMUNICACIONES MÓVILES MULTIPASO

Con el fin de caracterizar el canal de comunicaciones móviles se utilizan los siguientes parámetros: retardo de propagación rms (*rms delay spread*), exceso de retardo promedio (*mean excess delay*) y exceso de retardo de propagación (excess delay spread) los cuales pueden ser determinados a partir del *power delay profile*. Las propiedades de dispersión en el tiempo para canales multipaso son usualmente cuantificadas por el exceso de retardo promedio y el retardo de propagación rms. El exceso de retardo promedio es el primer momento del *power delay profile* y se define como:

$$\bar{\tau} = \frac{\sum_k P(\tau_k)\tau_k}{\sum_k P(\tau_k)} \quad (2.13)$$

El retardo de propagación rms es el segundo momento del *power delay profile* y se define como:

$$\sigma_\tau = \sqrt{\overline{\tau^2} - (\bar{\tau})^2} \quad (2.14)$$

$$\overline{\tau^2} = \frac{\sum_k P(\tau_k)\tau_k^2}{\sum_k P(\tau_k)} \quad (2.15)$$

Los valores típicos para el exceso de retardo promedio están en el orden de los microsegundos en canales de radio *outdoor* y en el orden de los nanosegundos en canales de radio *indoor*.

2.5.2 ANCHO DE BANDA COHERENTE

Así como se puede caracterizar el canal en el dominio del tiempo por medio de sus parámetros en el *power delay profile*, de manera análoga se puede caracterizar en el dominio de la frecuencia por medio del ancho de banda coherente. El ancho de banda coherente es una medición del rango de frecuencias sobre las

cuales el canal puede ser considerado plano (todas las frecuencias pasan por el canal con igual amplitud y fase).

Considerando que la función de correlación de la frecuencia con el ancho de banda sea de 0.9 el ancho de banda coherente es aproximadamente igual a:

$$B_c \approx \frac{1}{50\sigma_\tau} \quad \textbf{(2.16)}$$

Mientras que si la correlación es igual a 0.5 se aproxima a la siguiente expresión:

$$B_c \approx \frac{1}{5\sigma_\tau} \quad \textbf{(2.17)}$$

Siendo σ_τ = exceso de retardo promedio.

2.5.3 RESPUESTA AL IMPULSO DE UN CANAL MULTITRAYECTORIA

La respuesta al impulso es una caracterización de banda ancha que contiene toda la información necesaria para simular y analizar la transmisión de radio a través de dicho canal. La respuesta al impulso puede ser utilizada para predecir y comparar el funcionamiento de varios sistemas de comunicaciones móviles y diferentes anchos de banda de transmisión para determinadas condiciones del canal [10].

Si consideramos la señal transmitida como:

$$s(t) = \Re\{u(t)e^{j2\pi f_c t}\} \quad \textbf{(2.18)}$$

La correspondiente señal recibida es producto de la suma entre la componente de la trayectoria con línea de vista y todas las componentes debidas al multipaso.

La n-ésima componente debida al multipaso puede corresponder a una reflexión simple o a múltiples reflexiones. Si el retardo entre

dos componentes es mayor al tiempo de símbolo se dice que tienen resolución, caso contrario se dice que no tienen resolución. En general los canales de banda ancha tienen componentes con resolución y los canales de banda de banda estrecha tienden a tener componentes sin resolución.

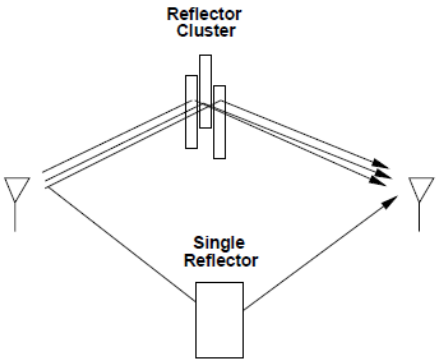


Fig. 2.14Reflexión simple y Reflexión múltiple

La respuesta impulso de un canal variante en el tiempo está definida como sigue:

$$c(\tau,t) = \sum_{n=0}^{N(t)} \alpha_n(t) e^{-j\varphi_n(t)} \delta(\tau - \tau_n(t)) \qquad \textbf{(2.19)}$$

Donde $c(\tau, t)$ representa la respuesta de paso bajo de un canal al tiempo t frente a un impulso en el tiempo $t - \tau$ [2].

2.5.4 MODELOS CON DESVANECIMIENTO PLANO Y SELECTIVO EN FRECUENCIA

El retardo de propagación debido al multipaso puede causar dispersión en el tiempo así como desvanecimiento selectivo en frecuencia, como se muestra en la siguiente figura.

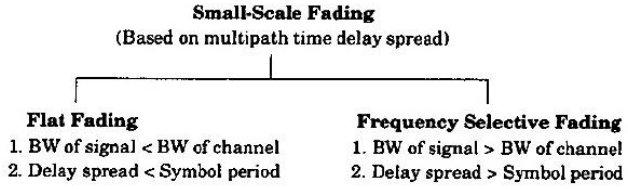


Fig. 2.15 Tipos de canal según el retardo de propagación

2.5.4.1 DESVANECIMIENTO PLANO

Si el canal de radio tiene una respuesta de ganancia constante en amplitud y fase sobre un ancho de banda que es mayor que el ancho de banda de la señal, la señal recibida tendrá desvanecimiento plano [10].Las características de espectro de la señal transmitida se

conservan en la señal que llega al receptor afectando únicamente la amplitud de la misma.

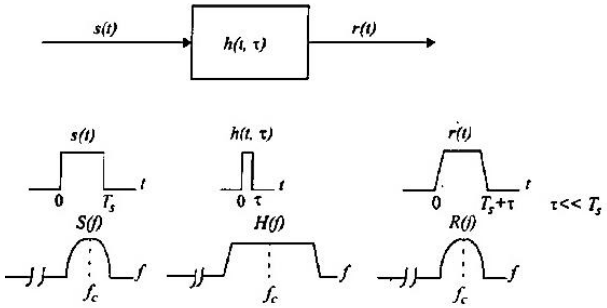


Fig. 2.16 Respuesta de un canal con desvanecimiento plano

En canales con desvanecimiento plano el reciproco del ancho de banda es mucho mayor que el *multipath time delay spread* del canal.

$$B_s \ll B_c \quad y \quad \mathrm{Ts} \gg \sigma_\tau \qquad \textbf{(2.20)}$$

La distribución de la ganancia instantánea más utilizada para modelar canales con desvanecimiento plano es la distribución Rayleigh que será revisado en la sección 2.5.6.

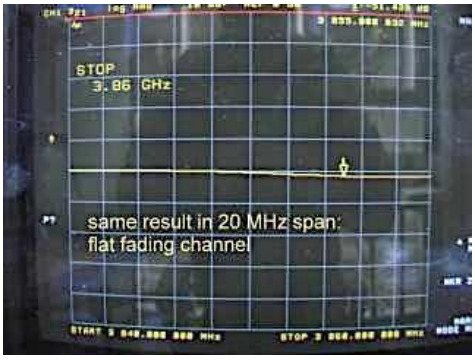


Fig. 2.17 Respuesta real de un canal con desvanecimiento plano [11]

2.5.4.2 CANAL SELECTIVO EN FRECUENCIA

Si el canal tiene una respuesta en amplitud y fase sobre un ancho de banda que es menor que el ancho de banda de la señal, el canal crea desvanecimientos selectivos en frecuencia en la señal recibida [10].

En este tipo de canales el retardo de propagación rms σ_τ debido al multipaso es mayor que el tiempo del símbolo de la señal, en el dominio de la frecuencia el ancho de banda de la señal transmitida es mayor que el ancho de banda coherente $B_c$del canal.

$$B_s > B_c \quad y\, Ts < \sigma_\tau \tag{2.21}$$

La señal es recibida con distorsiones debido a que es producto de múltiples versiones de la señal transmitida con sus respectivas atenuaciones y retardos. En el dominio del tiempo la selectividad en frecuencia se observa como interferencia intersimbólica ISI [12].

La figura muestra como en el dominio de la frecuencia este tipo de canales tienen ciertas componentes con ganancias mayores que otras.

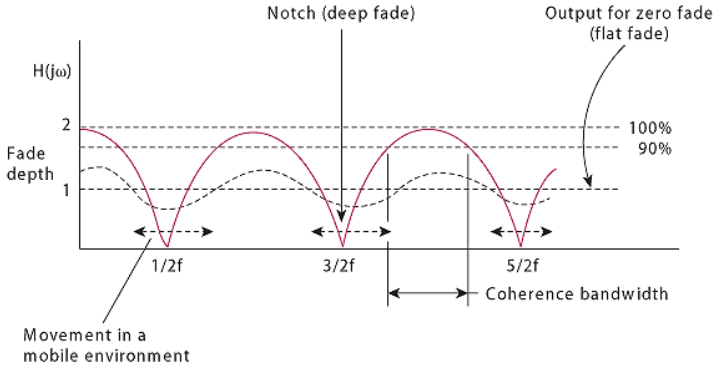


Fig. 2.18 Respuesta en amplitud de un canal selectivo en frecuencia[12]

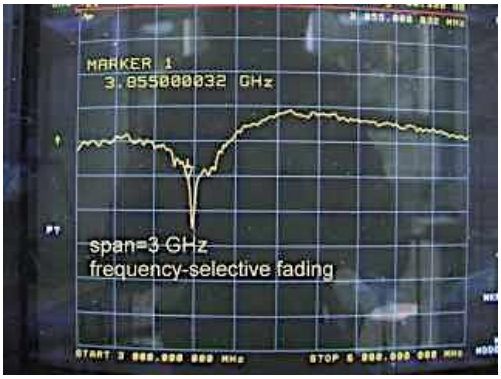


Fig. 2.19 Respuesta real de un canal con desvanecimiento selectivo en frecuencia [11]

2.5.5 CANAL AWGN

El canal más simple a modelar es el canal Gaussiano, llamado también AWGN. Este canal es representado como un canal ideal, esto es, que no existe multitrayectoria y cuya fuente de ruido se asume que tiene una densidad espectral de potencia constante sobre el ancho de banda del canal.

La salida de un canal AWGN, el cual es un canal aleatorio, está modelada por la siguiente expresión:

$$r_k = s_k(t) + n(t);\ kT < t < (k+t)T \qquad (2.22)$$

Donde sk(t) es la señal de entrada al canal en el intervalo de tiempo [kTb, kTb+Tb] y n(t) es un proceso aleatorio que representa un ruido ideal denominado blanco gaussiano, con media cero y densidad espectral de potencia N0/2 constante para todas las frecuencias. Adicionalmente se conoce que ruido blanco gaussiano es independiente de la señal de entrada, la cual tiene ancho de banda W y potencia finita [13].

El modelo del canal AWGN no es físicamente realizable ya que el ruido no afecta de la misma manera a todo el espectro de frecuencias. En este caso, es usual definir un ruido limitado en banda a partir del ruido blanco gaussiano. El modelo del canal AWGN con ruido limitado en banda está caracterizado por la siguiente expresión:

$$r_k(t) = s_k(t) + nd(t)\cos(2\pi f_0 t) - nq(t)\sin(2\pi f_0 t) \quad \textbf{(2.23)}$$

Siendo f0 la frecuencia central de operación y las funciones nd(t) y nq(t) procesos estocásticos banda base con ancho de banda W, mutuamente independientes, con media cero y varianza $N_o W/2$ [13].

Sin embargo un canal gaussiano es importante ya que provee una idea de cuál sería el mejor rendimiento del sistema sin la presencia de multitrayectoria, así para un esquema de modulación dado, podríamos calcular la tasa de bit erróneos en presencia del mencionado canal.

2.5.6 RAYLEIGH

La distribución de Rayleigh es utilizado en comunicaciones inalámbricas con desvanecimiento a pequeña escala (small-scale fading) con lo que se entiende que no existe línea de vista (LOS) entre el receptor y el transmisor. Una característica de este tipo de distribución es que no existe una componente dominante de las señales recibidas debidas a multipaso [11].

La distribución de Rayleigh describe en forma estadística la variación de tiempo de llegada de la señal a un receptor que ha atravesado un canal con desvanecimiento plano, es decir con desvanecimiento rápido por multitrayectoria.

Se conoce que la envolvente de la suma de dos señales de ruido gaussiano ortogonales obedece a una distribución Rayleigh. La

figura 2.20 muestra una señal envolvente con distribución Rayleigh como una función del tiempo.

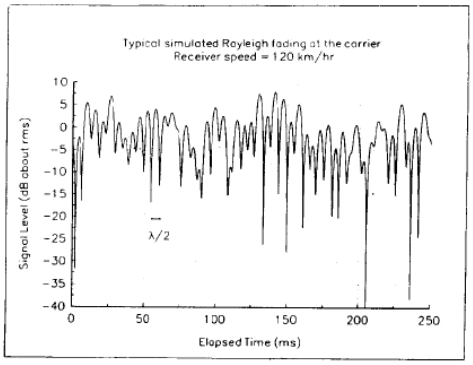


Fig. 2.20 Desvanecimiento Rayleigh típico simulado [10]

La distribución de Rayleigh tiene una función de densidad de probabilidad igual a:

$$p(r) = \begin{cases} \dfrac{r}{\sigma^2}\exp\left(-\dfrac{r^2}{2\sigma^2}\right); (0 \le r \le \infty) \\ 0\,; (r < 0) \end{cases} \qquad (2.24)$$

Donde σ representa el valor rms de la señal de voltaje recibida y σ^2 es la potencia promedio, ambas antes de la detección de envolvente. A continuación se muestra una gráfica.

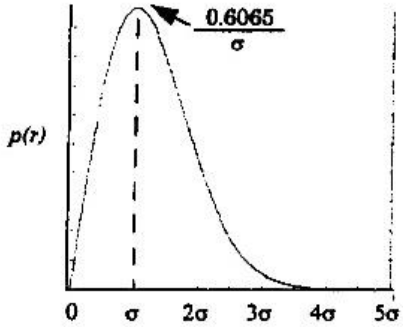


Fig. 2.21 Función de densidad de probabilidad Rayleigh [10]

La probabilidad que la envolvente de la señal recibida no exceda un valor específico de R está dado por la correspondiente función de distribución acumulada.

$$P(R) = 1 - exp\left(-\frac{R^2}{2\sigma^2}\right) \qquad (2.25)$$

La figura 2.22 muestra la función de distribución acumulada para un canal Rayleigh.

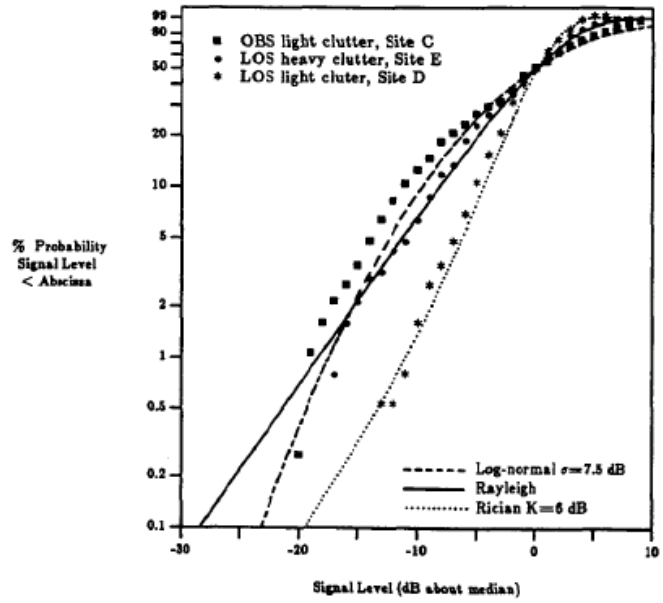


Fig. 2.22 Función de distribución de probabilidad acumulada en canales Log-normal, Rayleigh y Rician [10]

El valor medio rmean de la distribución de Rayleigh está dada por:

$$\text{Rmean} = 1{,}2533\,\sigma \qquad (2.26)$$

La varianza de la distribución de Rayleigh está dada por σ_r^2, que representa la potencia ac de la envolvente de la señal.

$$\sigma_r^2 = 0{,}4292\,\sigma^2 2 \qquad (2.27)$$

La mediana de r se encuentra:

$$\text{Rmedian} = 1{,}177\ \sigma \qquad (2.28)$$

Por lo tanto la media y la mediana difieren sólo en 0.55 dB en una señal con desvanecimiento Rayleigh. Usualmente en la práctica se utiliza la mediana en vez de la media, esto es debido a que es más sencillo realizar comparaciones entre distribuciones que pueden tener medias muy diferentes [10].

La aplicación de este modelo la encontramos en las zonas urbanas ya que de las señales que llegan al receptor ninguna es dominante debido a la gran cantidad de obstáculos y a que no existe un camino con línea de vista.

2.6 TEORÍA DE LA INFORMACIÓN

Cuando se diseña un sistema de comunicaciones el objetivo es siempre que la información llegue al receptor de manera confiable aun después de haber pasado por un canal ruidoso tomando en cuenta las limitaciones propias del sistema como potencia máxima de transmisión, ancho de banda, complejidad y por ende costos del sistema.

2.6.1 CAPACIDAD DEL CANAL

La velocidad a la que se puede transmitir los datos a través de un canal o ruta de comunicación se conoce como capacidad del canal y se la mide en función de la cantidad de bits por símbolos [bits/s].

Por tanto si la tasa de transmisión de un sistema permanece por debajo de C (Capacidad del Canal) entonces la probabilidad de error en la información que se transmite será baja empleando señales de transmisión codificadas suficientemente largas, caso contrario si la tasa de transmisión pasó por sobre este valor no es será posible una transmisión confiable por ningún medio [14].

2.6.2 TEOREMA DE CODIFICACIÓN DEL CANAL

La presencia de ruido y los diferentes factores que existen en el medio o canal afectan significativamente los bits que llegan al receptor. Lo que se busca con la codificación de canal es aumentar precisamente la resistencia de nuestras secuencias transmitidas con el fin de poder recuperarlas lo más próximo a lo que fue transmitido. Para lograr esto, el transmisor debe contar con un codificador de canal que aplicará bits redundantes o secuencias estructuradas, mientras que el receptor contará con un

decodificador que se encargara de realizar el proceso contrario. Logrando de esta forma que los efectos del ruido sobre los bits se disminuyan, de manera que se pueda reconstruir la secuencia original lo más exactamente posible [15].

2.6.3 TEOREMA DE CAPACIDAD DE INFORMACIÓN

Cuando se diseña un sistema de comunicaciones el objetivo es siempre que la información llegue al receptor de manera confiable aun después de haber pasado por un canal ruidoso tomando en cuenta las limitaciones propias del sistema como potencia máxima de transmisión, ancho de banda, complejidad y por ende costos del sistema.

Cuando el sistema de comunicación es digital una forma de medir la confiabilidad de la información recibida es a través de la probabilidad de error de bit (BER); mientras más cercana a cero el sistema es más confiable.

En teoría, si deseamos obtener bajas tasas de error, existe una velocidad máxima de transmisión de datos que depende del ancho de banda del canal y de la relación señal a ruido, relacionadas por

el teorema de capacidad del canal de Shannon de la siguiente manera:

$$C = B \log 2 (1 + SNR) \, b/s \qquad (2.29)$$

La capacidad de información se define como la velocidad máxima a la cual puede transmitirse sin error la información a lo largo del canal y se mide en bits por segundo [1].

Es decir, siempre que la velocidad a la cual se transmiten los datos por el canal R sea menor que la capacidad del canal C, el mensaje puede llegar al receptor sin errores aún después de pasar por un canal ruidoso.

La comunicación eficiente desde una fuente hasta el destino del usuario se consigue a través de la codificación de la fuente. La comunicación confiable por un canal ruidoso se alcanza por medio de la codificación de control de errores.

2.7 CODIFICACIÓN DE CONTROL DE ERRORES

Como ya lo analizamos en la sección 2.5 del documento en curso, la naturaleza ruidosa del canal unido a los diversos factores que existen sobre el mismo (atenuación, distorsión, multitrayecto), producen alteraciones en la trama que enviamos y por tanto aparece un BER que dependiendo de las condiciones del medio puede ser significativo o no.

El ruido ha sido siempre el factor que mayor ha golpeado a las comunicaciones, pero fue hasta 1948 en donde Claude Shannon publicó tal vez uno de los artículos más importantes para las comunicaciones, en donde demostró que para todo canal de transmisión digital expuesto a ruido existe un esquema de codificación con probabilidad de error tan pequeño como se desee, evitando así disminuir la eficiencia del canal y haciendo la transmisión-recepción lo más confiable posible. A partir de aquí nace la teoría de la información, y un sin número de técnicas para detectar y corregir errores.

Es así, que la codificación de canal tiene por objetivo transformar tramas de bits en secuencias mejoradas aplicándoles redundancia estructurada con el único objetivo de hacerlas robustas y que el receptor pueda detectar y corregir los errores a fin de obtener una secuencia tan precisa

como la original. Existen 2 técnicas o esquemas de corrección de datos [16]:

2.7.1 ARQ

El ARQ (Automatic Repeat Request) es una técnica de control de errores muy usada que se basa en conseguir la información mediante retransmisiones del mensaje enviado, pues los esquemas con los que cuenta pueden detectar errores más no corregirlos. Para controlar la correcta recepción de los bits(secuencias, tramas, paquetes) se utilizan bits para indicar si el paquetes ha sido o no recibido correctamente, sin embargo puede darse el caso que estos bits se pierdan en el medio, por lo cual tienen implementado un tiempo límite o timeout que le indica al transmisor que envíe nuevamente.

Como se puede deducir, es una técnica que utiliza de muchos recursos cada vez que existen errores dando lugar a series de retransmisiones que se pueden repetir por varias ocasiones. Es usada principalmente en sistemas que no actúan en tiempo real. Cabe mencionar que esta técnica requiere principal de una conexión en 2 vías, Half Duplex o Full Duplex [1].

2.7.2 FEC

FEC (Forward Error Correction) es un tipo de mecanismo de corrección de errores que no necesita una retransmisión de los datos. Muy utilizado en sistemas sin retorno (Simplex), su funcionamiento radica en el uso de unos bits de redundancia que se colocan justo a la salida de la fuente digital (generadora de bits) obteniendo palabras de códigos para ser transmitidas. Por su lado el receptor cuenta con el decodificador que al aplicar los algoritmos de corrección obtendrá la secuencia original.

En la actualidad existen varios tipos de codificación que se usan. Las diferencias radican en la cantidad de bits erróneos que pueden corregir y en la complejidad que resultado su implementación principalmente el decodificador. Es precisamente sobre este aspecto fundamental de codificación/decodificación sobre el cual enfocaremos nuestro tema.

2.7.3 EFECTIVIDAD DE UN CÓDIGO

La efectividad de un código está caracterizado fundamentalmente por 3 factores: (1) la distancia mínima entre las palabras de código, (2) la capacidad de detección de ráfagas, y (3) la

probabilidad de que una secuencia aleatoria de datos sea aceptada como libre de errores. La distancia mínima entre palabras de código hace referencia al mínimo número de bits para diferenciar una palabra de otra, mientras que una ráfaga de errores en un bloque está determinada por una serie de bits erróneos consecutivos cuya longitud puede variar. La capacidad de detección de ráfagas de un código se define como la longitud máxima de una ráfaga de errores que puede ser detectada.

2.8 CÓDIGOS DE BLOQUES LINEALES

Un código de bloque consiste de un grupo de vectores de longitud fija llamado palabras de código. Cuando los elementos de las palabras de códigos son ceros y unos se les llama bits y se dice que el código es binario, lo que implica que sigue las reglas de la aritmética de módulo 2 [17].

Si dos palabras de código al sumarse en aritmética módulo 2 dan como resultado una tercera palabra de código se dice que un código es lineal [1].

Se puede considerar que la longitud de las palabras de código *n* es mayor que la longitud de las secuencias de información a codificarse k, considerando que una secuencia codificada normalmente está compuesta de los bits de información además de bits de redundancia. De éste modo, el código de bloques lineal (n,k) utiliza una regla de correspondencia definida para asignar a cada secuencia de información de longitud k una secuencia codificada de longitud n. La relación Rc= k/n se define como la tasa del código.

2.8.1 MATEMÁTICA DEL CÓDIGO DE BLOQUE LINEAL (N,K)

En éste trabajo, se ha implementado un codificador y decodificador de bloques lineal sistemático, esto es, los bits de mensaje forman parte de la palabra de código, como se ilustra en la siguiente figura:

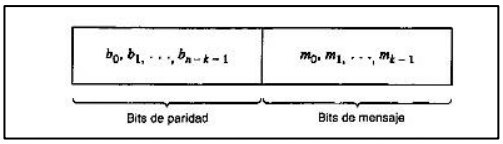


Fig. 2.23 Estructura de un código sistemático [1]

La estructura matemática para la palabra de código se puede escribir de la siguiente manera:

$$c = \begin{Bmatrix} b_i, & i = 0,1,\dots,n-k-1 \\ m_{i+k-n} & i = n-k, n-k+1,\dots,n-1 \end{Bmatrix} \quad (2.30)$$

Los (n-k) bits de paridad son sumas lineales de los k bits del mensaje, como se indica a continuación[1]:

$$b_i = p_{0i}m_0 + p_{1i}m_1 + \cdots + p_{k-1}m_{k-1} \quad (2.31)$$

$$p = \begin{Bmatrix} 1\ si\ b_i\ depende\ de\ m_j \\ 0\ en\ otro\ caso \end{Bmatrix}$$

Donde los coeficientes pi se eligen de modo que los renglones de la matriz generadora sean linealmente independientes. Estas expresiones matemáticas podrían ser reformuladas en forma matricial, definiendo como vectores renglón ‘m’ para el mensaje, ‘b’ para los bits de paridad y ‘c’ para la palabra de código:

$$m = [m_0, m_1, \dots, m_{k-1}] \quad (2.32)$$

$$b = [b_0, b_1, \dots, b_{k-1}] \quad (2.33)$$

$$c = [c_0, c_1, \dots, c_{k-1}] \quad (2.34)$$

Del mismo modo podemos definir la matriz de coeficientes P de k por (n-k) como sigue, siendo cada uno de los coeficientes valores binarios.

$$P = \begin{bmatrix} p_{00} & p_{01\dots\dots} & p_{0,n-k-1} \\ p_{10} & p_{11\dots\dots} & p_{1,n-k-1} \\ p_{k-1,0} & p_{k-1,1} \cdots & p_{k-1,n-k-1} \end{bmatrix} \quad (2.35)$$

Podemos definir una matriz generadora de palabras de código que cumpla con el siguiente orden:

$$G = [P : I_k] \quad (2.36)$$

De modo que las palabras de código pueden ser obtenidas utilizando la siguiente ecuación:

$$c = mG \quad (2.37)$$

Otra forma de expresar la relación entre los k bits de mensaje y los (n-k) bits de paridad de un código de bloques lineal, seria considerando una matriz H de (n-k) por n definida como:

$$H = [I_{n-k} : P^T] \quad (2.38)$$

Donde P^T es la transpuesta de la matriz de coeficientes P e I_{n-k} es una matriz identidad de (n-k) por (n-k). La matriz H cumple la propiedad que:

$$cH^T = 0 \quad (2.39)$$

La matriz H se conoce como matriz de verificación de paridad del código y $cH^T = 0$se le conoce como ecuación de verificación de paridad. Las relaciones entre el vector mensaje, el vector de código, la matriz generadora y la matriz de verificación de paridad se ilustran en la siguiente figura:

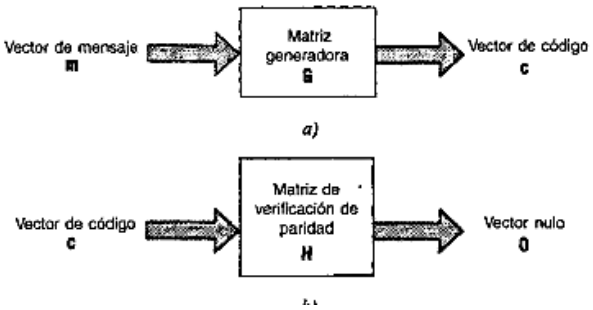


Fig. 2.24 Relación del vector de código con la Matriz Generadora y de Verificación de Paridad.[1]

2.8.2 MATRIZ DE SÍNDROME

En el transmisor se recibe una secuencia de bits que puede contener errores introducidos por el canal, los que podríamos representar de la siguiente manera:

$$r = c + e \qquad (2.40)$$

El receptor debe decodificar el vector de código c a partir del vector recibido r. Inicialmente, se debe encontrar el vector de síndrome, el cual nos indica la cantidad de errores que tiene la secuencia recibida y en algunas ocasiones la posición exacta de los mismos. Podríamos encontrar el síndrome de la siguiente manera:

$$s = rH^T \qquad (2.41)$$

2.8.3 LIMITACIONES DE LA CODIFICACIÓN DE SÍNDROME

La principal limitación del uso de síndromes en la codificación de canal, es que no se tiene información exacta acerca del vector de error afectando las secuencias recibidas, sin embargo, reduce las posibles opciones de patrones de error entre las cuales escoger de 2^n a 2^{n-k}. En la práctica, esto puede ser observado sabiendo que se asigna un síndrome a cada error sin embargo, algunos patrones de error son asignados dentro de otro síndrome, cometiendo de ésta forma errores al decidirse por la secuencia equivocada.

2.8.4 DISTANCIA MÍNIMA EN LAS PALABRAS DE CÓDIGO

La distancia mínima de un código de bloques lineal se define como la distancia de Hamming más pequeña entre cualquier par de vectores de código, esta distancia es un parámetro importante del código ya que determina la capacidad que tiene para corregir errores. A su vez, el peso de Hamming de un vector de código está definido como el número de bits que son diferentes de cero en c.

Si se quiere diseñar un código de bloques lineal (n.k) que sea capaz de corregir patrones de error cuyo peso de Hamming (número de bits que son diferentes de cero) de la palabra de código, sea menor o igual que un valor t, el decodificador tendría que decidirse a favor de aquel vector de código cuya distancia de Hamming sea la más cercana al vector recibido.

2.8.5 DECODIFICACIÓN DEL SÍNDROME

Se puede definir un procedimiento para la decodificación de un código de bloques lineales. Primero se debe determinar una tabla de síndrome y patrón de errores, partiendo con el vector cero como patrón de errores y continuando con los errores de un bit. Para cada patrón de errores se asocia un síndrome siendo obtenido de la multiplicación $s=eH^T$.

Para proceder a la decodificación se encuentra el síndrome del vector recibido, multiplicando por la matriz de verificación de paridad H transpuesta. Al encontrar alguna coincidencia con alguno de los síndromes de la tabla de síndrome y patrón de errores definida, es posible encontrar el patrón de errores con el cual decodificar el vector recibido.

Si existieran elementos de síndrome que no hayan sido asignados a un patrón de errores, se escoge los patrones de error con mayor probabilidad de ocurrencia que generen aquellos síndromes.

2.8.6 LA IMPORTANCIA DE ELEGIR UNA MATRIZ GENERADORA

La presente sección corresponde a un aporte de los autores de ésta documento. La matriz generadora es una parte esencial en la detección y corrección de errores, pues como ya se ha mencionado de ella dependerá la cantidad de bits que se puedan corregir. Es importante analizar el funcionamiento de la misma, para ello vamos a elegir 3 matrices generadoras, recordando que cada una de ellas debe ser ortogonal G1, G2 y G3:

$$G1 = \begin{vmatrix} 0 & 010 \\ 1 & 101 \end{vmatrix} \quad (2.42)$$

$$G2 = \begin{vmatrix} 0 & 110 \\ 1 & 101 \end{vmatrix} \quad (2.43)$$

$$G3 = \begin{vmatrix} 1 & 1110 \\ 0 & 1101 \end{vmatrix} \quad (2.44)$$

Como se ha anticipado, las matrices generadoras definirán la cantidad de bits redundantes que se incluirán en la secuencia a transmitir. En el caso de las generadoras que se muestran arriba la matriz identidad a utilizar será I2, mientras que las columnas restantes determinarán si serán 2 o 3 los bits que se añadirán.

Con nuestra primera matriz podemos corregir un error de cada 4 bits transmitidos al igual que la segunda, sin embargo si comparamos una misma secuencia utilizando las 2 matrices, la segunda es más eficiente que la primera; esto se debe a que en el primer caso existen síndromes duplicados que produce que el receptor decida a favor de la secuencia equivocada. Pero nos podemos dar cuenta que la cantidad de errores aún es significativa en los 2 casos, cosa que no ocurre si utilizamos nuestra tercera matriz, aquí tenemos la facultad de corregir hasta 2 errores de cada 5 bits, incluso si ocurre una ráfaga de errores en los 2 últimos bits. Esto se debe principalmente a las secuencias que se generan luego de la codificación de canal, pues en los primeros 2 casos nos podemos dar cuenta que la distancia que existe entre las diferentes combinaciones es de uno o 2 bits mientras que en nuestra última selección la distancia mínima existente es de 3 pudiendo llegar a ser hasta 4.

La distancia entre las secuencia es sumamente importante. Como lo indica claramente la sección 2.7.3 en la que se analizó su definición. Fijémonos en la figura 2.25 para poder entender el rol de la misma.

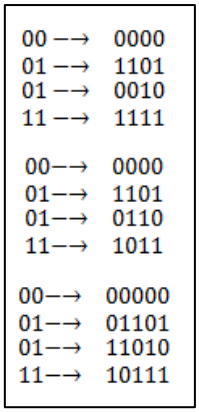


Fig. 2.25 Combinación de di-bits con su respectiva palabra de código

En el primer ejemplo si tomamos la secuencia 0000 y la 0010 la distancia entre ellas es de uno, por lo tanto si el canal llegase a distorsionar el receptor no identificaría el error puesto que la secuencia recibida se encuentra dentro del espacio de las posibles secuencia a recibir, causando desde ya que el BER se incremente. De aquí la importancia de tener la mayor distancia posible entre las secuencias.

Otro parámetro muy importante en el proceso de la codificación/decodificación de canal es la matriz de verificación o también conocida como H, la cual depende de la matriz Generadora ver ecuación 2.37.

Por lo tanto tendremos que escoger una matriz generadora de tal forma que genere palabras de código con una distancia entre ellas lo más grande posible y además que al obtener una matriz de verificación esta no cree síndromes repetidos al menos para los errores de bits únicos, puesto es imposible crear un síndromes diferente para todas las posibles combinaciones.

2.9 CÓDIGOS CONVOLUCIONALES

El sistema de comunicaciones considerando codificación y decodificación Convolucional se puede ilustrar en la siguiente figura:

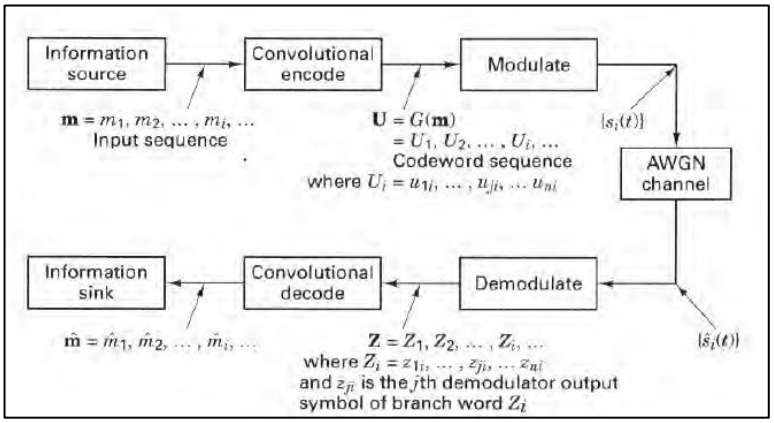


Fig. 2.26 Esquema de comunicación que incluye codificación convolucional [18]

Un mensaje de entrada puede ser definido como m=m1,m2,…,mi donde el índice i es utilizado para indicar tiempo o ubicación en la respectiva secuencia. Se puede asumir que cada uno de los elementos de m son independientes, es decir, cada mi tiene igual probabilidad de ser cero o de ser uno. Esta secuencia entra a un codificador Convolucional, generándose a la salida una secuencia G=f(m). Una de las características de este tipo de codificadores es que los bits de la secuencia codificada no serán independientes entre ellos ya que dependen tanto de la secuencia entrante actual como de las secuencias anteriores, como será explicado más adelante.

La figura 2.27 muestra la forma general de un codificador Convolucional, el cual puede ser implementado con un registro de desplazamiento y sumadores de módulo 2 de M=kK-etapas y n sumadores de módulo 2,

donde el valor de K es llamada “Longitud de restricción” y está definida como el número de desplazamientos de k bits que un bit puede influir en la salida del codificador.

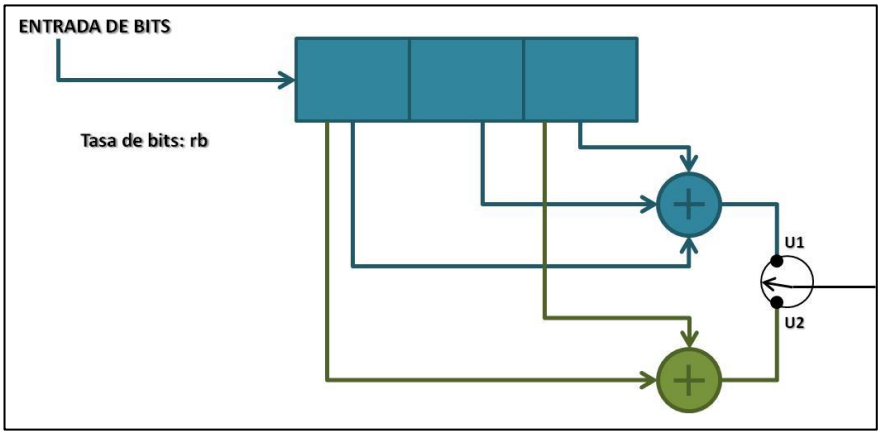


Fig. 2.27 Esquema de un registro de desplazamiento de un codificador convolucional.

[19]

Se puede considerar a un codificador Convolucional como una máquina de estado finito compuesto de registros de desplazamientos de M etapas (o capacidad M) que se conectan a n sumadores de módulo 2 y al final un multiplexor envía a la salida los generados de los sumadores, los cuales conforman los bits de redundancia. Siendo una secuencia de bits de entrada de longitud L, el número de bits generados a la salida del codificador es n (L+M), por lo que la tasa de código es:

$$\mathbf{r} = \mathrm{L} / \mathrm{n}(\mathrm{L} + \mathrm{M}) \qquad (2.45)$$

Normalmente la longitud del mensaje L es mucho mayor al número de etapas de los registros de desplazamiento, por lo que es posible simplificar la expresión anterior:

$$\mathbf{r} = 1/\mathrm{n}\ [\mathrm{bits/símbolo}] \qquad (2.46)$$

Siendo n el número de sumadores del codificador. Se puede observar en la figura 2.19 que si el registro de desplazamiento tiene M etapas, cuando un bit entra al codificador deberá desplazarse M+1 veces para salir del sistema y dejar de influir sobre la salida actual, por tal motivo se dice que la longitud de restricción de un codificador convolucional es M+1.

Considere que los registros de desplazamiento para ser inicializados en cero, deben agregarse una secuencia entrante que consiste en K-1 ceros a la cola del mensaje.

2.9.1 FORMAS DE REPRESENTACIÓN DE UN CODIFICADOR CONVOLUCIONAL

2.9.1.1 REPRESENTACIÓN DE CONEXIONES

Como se puede observar en la Figura 2.28, se tiene un registro de desplazamiento con capacidad para tres bits, por lo que su longitud de restricción sería K=4. Los bits se van desplazando hacia la derecha uno a la vez, de modo que k=1. En este codificador se tienen 2 sumadores de módulo 2, el superior que realiza la suma de los tres bits y el inferior que realiza la suma en módulo 2 del primer y el último bit. La tasa de código de este codificador Convolucional es ½.

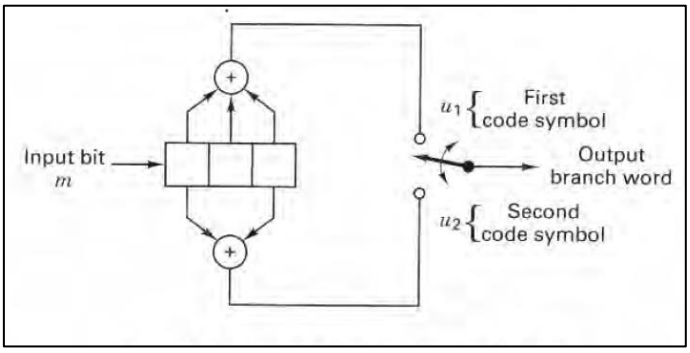


Fig. 2.28 Proceso de codificación. Codificador Convolucional ½. [18]

La desventaja de encerar los registros de desplazamiento del codificador convolucional es que la tasa efectiva de datos en una palabra de código disminuye debajo del valor de k/n, sin embargo cuando las secuencias de bits son de longitud grande, ésta tasa tiende nuevamente al valor de k/n.

2.9.1.2 RESPUESTA AL IMPULSO DEL CODIFICADOR

Se define la respuesta al impulso como la respuesta del codificador cuando se aplica a la entrada un alto “1” y éste atraviesa al codificador. De modo que la secuencia codificada puede obtenerse como la suma desplazada en el tiempo del producto de cada uno de los bits de mensaje por la respuesta al impulso, es decir, un codificador convolucional cumple con la propiedad matemática de linealidad. Precisamente a éste desplazamiento en el tiempo o también llamado convolución debe su nombre.

2.9.1.3 DIAGRAMA DE ÁRBOL

Es posible construir un árbol de código de un codificador convolucional recordando la estructura de desplazamiento

que ocasiona que los bits anteriores influyan sobre los bits a codificarse actualmente. Se construye desde un estado cero donde asumimos que el registro de desplazamiento está encerado y se colocan las dos posibles entradas, un cero y un uno. Se dibuja una rama superior para identificar la salida del codificador cuando a la entrada hay un cero y una rama inferior cuando a la entrada hay un uno. En la rama se coloca la secuencia codificada resultante para la correspondiente entrada. El siguiente gráfico muestra cómo nos quedaría el árbol de código cumpliendo estas características.

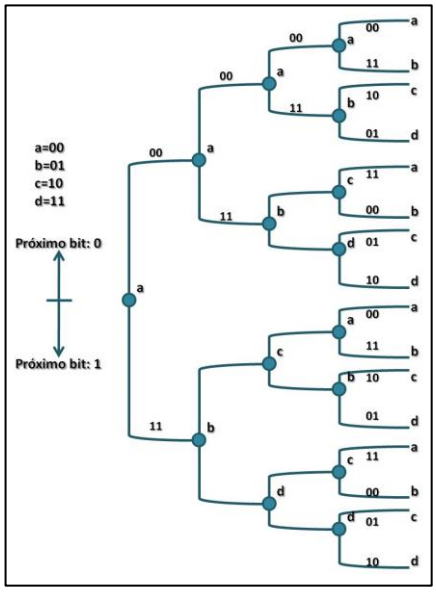


Fig. 2.29 Representación de un código convolucional. Diagrama de Árbol [19]

2.9.1.4 DIAGRAMA DE TRELLIS

Se puede observar que en el árbol de la figura 2.30 algunos estados se repiten y el árbol se vuelve continuo a partir de las tres primeras ramas. Considerando esto, se puede reducir el árbol a uno más compacto, que es que se puede observar en la siguiente figura y recibe el nombre de enramado.

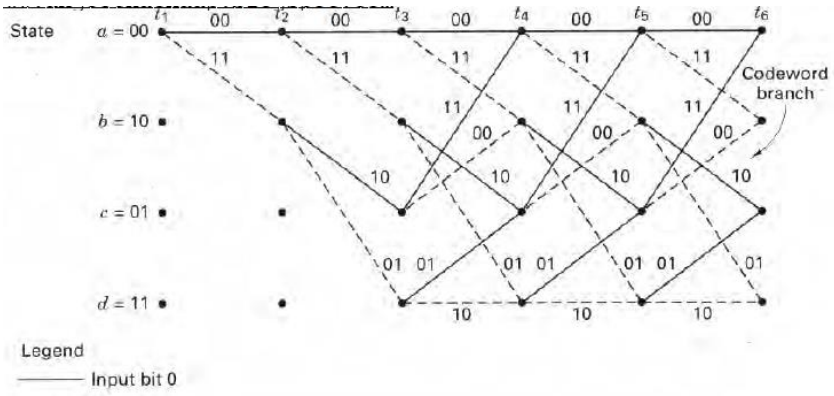


Fig. 2.30 Diagrama de Trellis. Mecanismo detección de errores en receptor [18]

Al diagrama de enramado también se lo conoce como Diagrama de Trellis. La simbología utilizada para este tipo de diagramas es que una línea continua representa el instante en que ingresa un cero al codificador y una línea segmentada representa el instante que ingresa un uno al

codificador. De igual manera, la secuencia de codificación para un grupo de bits entrantes se sigue de izquierda a derecha. De ésta forma se puede observar la naturaleza de máquina de estado finito del codificador ya que se asemeja a la forma de un diagrama de estados.

Un estado de un codificador Convolucional se define como los bits que están almacenados en su registro de desplazamiento, si consideramos un codificador con registros de desplazamiento con capacidad para 2 bits, entonces los posibles estados son cuatro, los cuales se detallan a continuación:

Estado	*Descripción binaria*
a	00
b	10
c	01
d	11

Fig. 2.31 Estados de un codificador convolucional [1]

2.9.1.5 DIAGRAMA DE ESTADOS

Como se mencionó en los párrafos anteriores, un codificador convolucional pertenece a los dispositivos

conocidos como máquinas de estado finito, que es el nombre general que se les da a las máquinas que tienen memoria acerca de las señales pasadas. Se conoce como estado de una máquina de estado finito, aquel que describe el valor que toman las señales de salida para una determinada señal de entrada siguiendo una secuencia lógica, de modo que el estado futuro depende del estado actual y de las señales de entrada [18].

Observando el diagrama de Trellis podríamos identificar la forma en que se construye la secuencia codificada por medio de niveles *j*, se puede observar en la figura que a partir del nivel j=3 el diagrama de enramado sigue el mismo patrón de estados y es el que observamos en la figura 2.32:

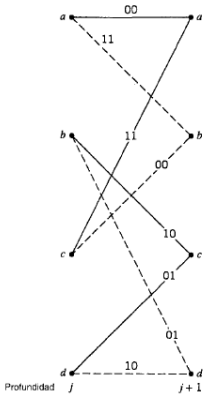


Fig. 2.32 Estados iniciales de un diagrama de trellis [1]

Donde del lado izquierdo se han colocado los posibles estados presentes y del lado derecho los posibles estados siguientes para las correspondientes di-bits generados por el codificador identificados por las ramas. Ésta misma porción se puede ilustrar construyendo un diagrama de estados, siendo los estados a, b, c y d y las transiciones los posibles di-bits salientes de él, que son únicamente dos productos de que se desplaza un bit a la vez. De modo que nos quedaría un diagrama de estados como se ilustra en la figura 2.33.

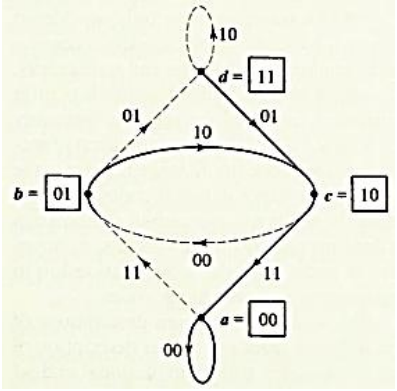


Fig. 2.33 Equivalencia del diagrama de trellis expresado en diagrama de flujo [20]

De la misma manera, la nomenclatura utilizada en la figura anterior es que una línea segmentada representa una

transición cuando se ingresa un bit 1 y una línea continua para un bit 0.

2.9.2 DECODIFICACIÓN DE CÓDIGOS CONVOLUCIONALES MEDIANTE EL ALGORITMO DE VITERBI

Se puede observar en la diagrama de enramado que a partir del nivel j=3 existen siempre dos ramas entrantes a cada uno de los estados: a, b, c o d. Podría un decodificador de distancia mínima tomar una decisión respecto a cuál trayectoria retener en éste punto sin tener pérdidas en el desempeño. Podría tomar la siguiente decisión en el nivel j=4. Estas decisiones sucesivas nos dan una idea del funcionamiento del algoritmo de Viterbi, utilizado para la decodificación Convolucional. El algoritmo de Viterbi opera de la siguiente manera:

Primero se divide la secuencia en grupos de longitud igual a la capacidad de los registros de desplazamiento. El primer grupo formado se debe comparar con las dos ramas salientes del nivel j=0. La comparación a realizarse se refiere a encontrar la distancia de Hamming entre los bits del primer grupo y los bits posibles a la

salida, que identifican a cada rama, en éste caso son 00 y 11. A éste número se le llama etiqueta.

Al llegar al siguiente nivel j=1, existen dos posibles estados, a los cuales se les debe asignar un valor llamado métrica. La métrica es el mínimo valor acumulado de las etiquetas de las ramas que llegan a ése estado. Debido a que en el nivel j, cada estado tiene una única rama de entrada, se asigna a ésos dos estados le métrica que corresponde la etiqueta encontrada en el paso anterior.

Antes de llegar al siguiente nivel, debemos tomar el grupo dos y encontrar las etiquetas que identifican a cada una de las cuatro posibles ramas.

En el nivel j=2, existen cuatro posibles estados, sin embargo, cada estado tiene un única rama, por lo que la métrica asignada corresponde a la suma de la etiqueta de la rama que llega a ese estado más la métrica del estado desde la que partió.

Nuevamente, se toma el siguiente grupo y se encuentran las correspondientes etiquetas a cada una de las ocho posibles ramas.

En el nivel j=3, existen cuatro posibles estados siguientes y cada estado tiene dos ramas de entrada. Entre las dos se debe escoger aquella cuya métrica sea menor, es decir, se decide por aquella rama cuya suma de su etiqueta más la métrica del estado presente sea el mínimo posible.

Se repiten los pasos anteriores hasta terminar de evaluar todos los grupos formados por la secuencia recibida.

"*De esta forma, se asegura que las trayectorias contendrán la elección de máxima verosimilitud [1].*"

Al llegar al último nodo, se puede obtener finalmente la decisión de la trayectoria de distancia mínima y es liberada hacia el destino como versión decodificada de la secuencia recibida.

2.10 GANANCIA DE CODIFICACIÓN

La ganancia de codificación es un parámetro bien establecido para medir el rendimiento de los códigos de canal. Considerando $P_{e,c}(\gamma)$ la probabilidad de error de un sistema codificada y $P_{e,uc}(\gamma)$ la probabilidad de error de un sistema sin codificación [21]. Se podría obtener la

relación Señal a ruido para un determinado Pe a partir de las curvas BER vs SNR del sistema. La ganancia de codificación está definida como:

$$C_{dB} = \lim_{\mu \to \infty} \left[10 \log(\mu) - 10 \log(P_{e,c}^{-1}\left(P_{e,uc}(\mu)\right)) \right] \quad \textbf{(2.47)}$$

Siendo μ un determinado nivel señal a ruido. La ganancia de codificación puede ser interpretada como la cantidad de potencia de transmisión adicional necesaria para obtener una probabilidad de error Pe en un sistema sin codificación con respecto a uno con codificación de canal.

CAPÍTULO 3

DISEÑO DE CODIFICADORES Y DECODIFICADORES

En este capítulo se explicará el diseño realizado en el software LabView de cada uno de los algoritmos de codificación para control de errores tanto del lado del transmisor, al cual se le conoce como codificador, así como del lado del receptor al cual se le conoce decodificador. La mayoría de las funciones utilizadas para la implementación de los mismos se tomaron de las librerías propias de LabView, así como funciones de la librería Modulation Toolkit de propiedad de National Instruments.

Para facilitar la implementación de los códigos de canal se ha utilizado sub-VI agrupados en librerías, con el fin de obtener una mejor estructuración y escalabilidad de los algoritmos.

3.1 MODULACIÓN EN AMPLITUD Y FASE

3.1.1 MODULADOR BPSK

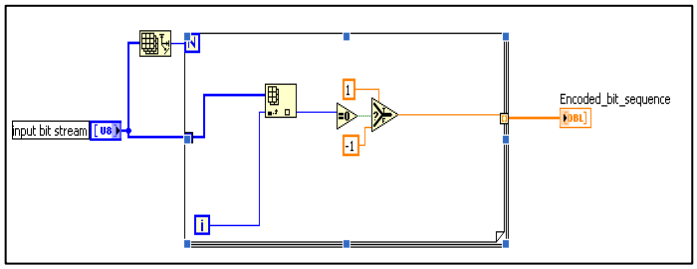


Fig. 3.1Diagrama de bloques del modulador BPSK

Inicialmente el algoritmo selecciona cada uno de los bits que forman parte de la secuencia de entrada, esto con ayuda de la función *Index Array*  *(Functions >> Programming >> Array >> IndexArray)*. El programa verifica si el bit actual es igual a cero con lo cual se modula por un uno, caso contrario se modula por un cero, tal como fue explicado en el capítulo 2. Las funciones *Equalto0?*  y *Select*  de la librería de funciones de comparación *(Functions >> Programming >> Comparison)* hacen posible esta conversión.

Este procedimiento debe ser realizado para cada uno de los bits de mensaje a ser modulados, lo cual lo obtenemos con ayuda de

la función *Array Size*  *(Functions >> Programming >> Array >> ArraySize)*.

3.1.2 DEMODULADOR BPSK

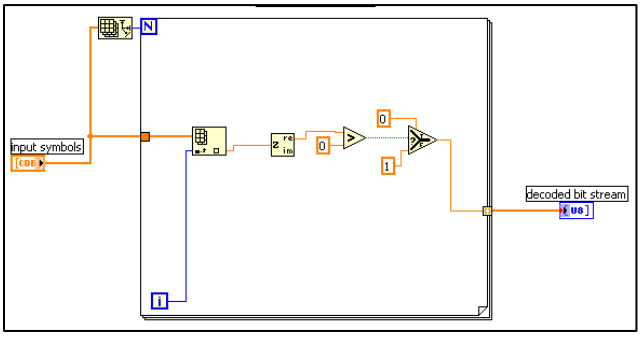


Fig. 3.2Diagrama de bloques de un demodulador BPSK

Inicialmente el algoritmo evalúa el primer símbolo del cual extraerá la parte real. Utilizamos las funciones *Index Array*  *(Functions >> Programming >> Array >> IndexArray)* para seleccionar un símbolo dentro de un arreglo, la función Complex to Polar  (Functions >> Programming >> Numeric >> Complex >> Complex to Polar). La parte real del símbolo se evalúa si tiene un valor mayor a cero, en caso que ésta proposición sea verdadera el bit demodulado será un cero, caso contrario el bit será un uno. Son parte de éste procedimiento la función *Greather than 0?*  *(Functions >> Programming >> Comparison >> Greather than 0?)*,

así como la función *Select*  *(Functions >> Programming >> Comparison>> Select).*

Este procedimiento hasta ser evaluados todos los símbolos enviados para lo cual utilizamos la función *Array Size*  *(Functions >> Programming >> Array >> ArraySize)*.

3.1.3 MODULADOR QPSK

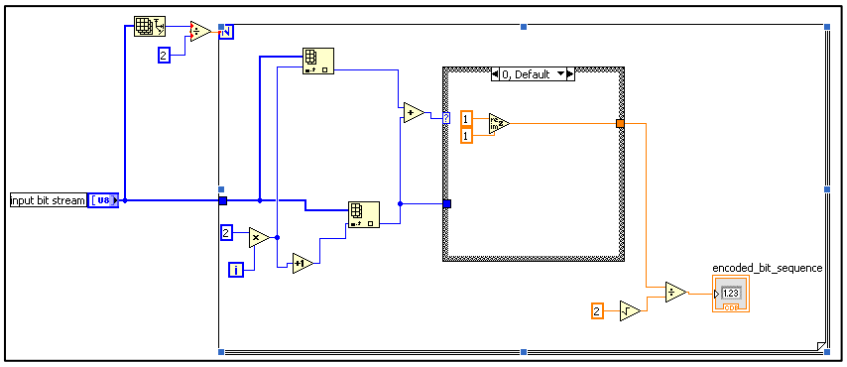


Fig. 3.3Diagrama de bloques del modulador QPSK Case=0

Inicialmente el algoritmo evalúa los dos primeros bits de la secuencia de bits utilizando las funciones *Index Array*  *(Functions >> Programming >> Array >> IndexArray)* y la función *Increment*  *(Functions >> Programming >> Numeric >> Increment*). Se suman los dos elementos con el objetivo de

evaluar el resultado y seleccionar los símbolos correspondientes. El caso que se muestra en la figura 3.3 corresponde al resultado igual a cero. Se observa que, de cumplirse ésta condición el símbolo a ser enviado es 1 + j1. Las funciones utilizadas hasta éste punto son: *Add*  *(Functions >> Programming >> Numeric >> Add)*, la estructura *case (Functions >> Programming >> Structures >> CaseStructure)* y *Re/Im to Complex*  *(Functions >> Programming >> Numeric >> Complex >> Re/Im to Complex)*.

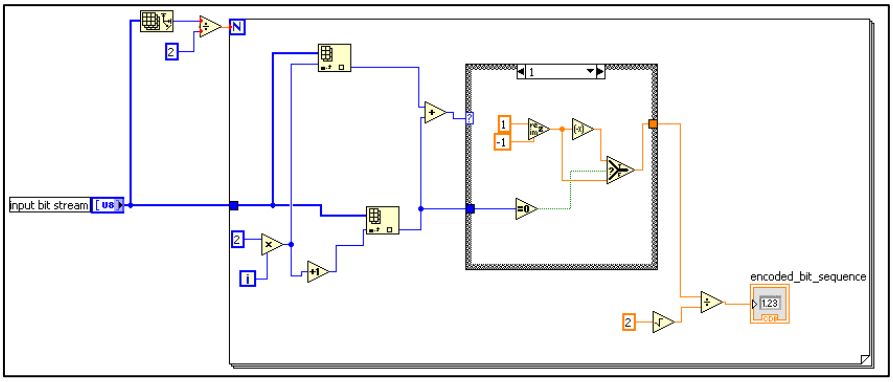


Fig. 3.4 Diagrama de bloques del modulador QPSK Case=1

El caso que se muestra en la figura 3.4 corresponde al resultado igual a uno. Para que ésta proposición sea verdadera pueden darse dos casos: que el segundo bit sea cero o que el segundo bit sea uno. De cumplirse la primera condición, se observa que el símbolo a ser enviado es 1-j1, caso contrario el símbolo a ser

enviado es -1+j1. Las funciones utilizadas en éste caso son: Negate  (Functions >> Programming >> Numeric >> Negate), Equalto0?  (Functions >> Programming >> Comparison >> Equalto0?), Select  (Functions >> Programming >> Comparison >> Select) y Re/Im to Complex  (Functions >> Programming >> Numeric >> Complex >> Re/Im to Complex).

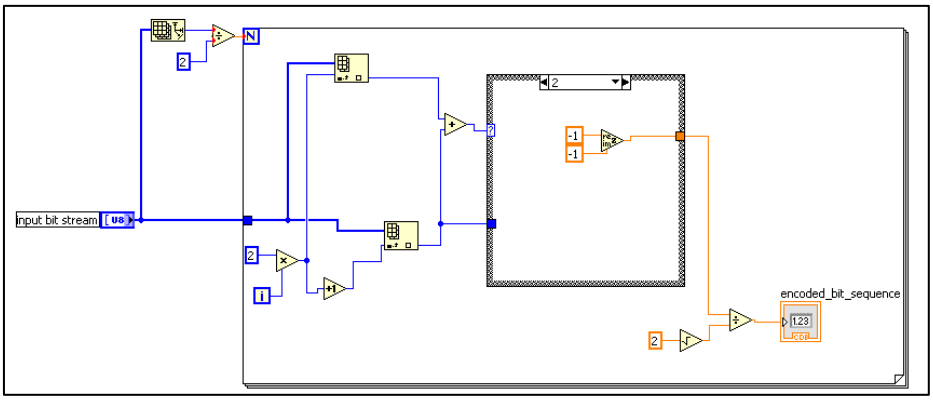


Fig. 3.5 Diagrama de bloques del modulador QPSK Case=2

El caso que se muestra en la figura 3.5 corresponde al resultado igual a dos. Se observa que, de cumplirse ésta condición el símbolo a ser enviado es -1 - j1. Se utiliza una vez más la función *Re/Im to Complex*  *(Functions >> Programming >> Numeric >> Complex >> Re/Im to Complex)*.

3.1.4 DEMODULADOR QPSK

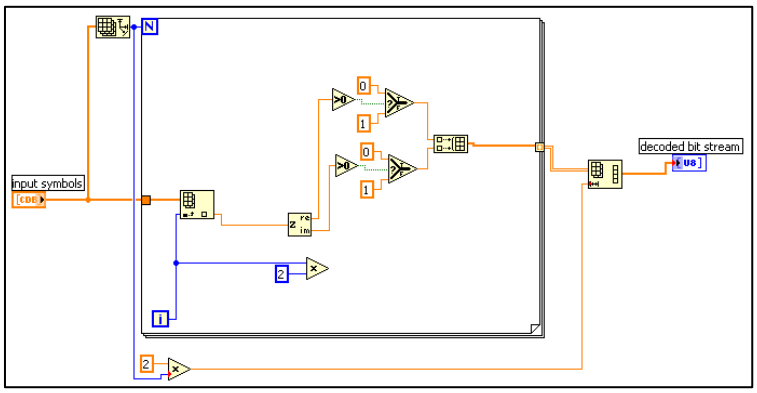


Fig. 3.6Diagrama de bloques del demodulador QPSK

Inicialmente el algoritmo toma el primer símbolo del cual extrae la parte real y la parte imaginaria, esto con la función *Complex to Polar* *(Functions >> Programming >> Numeric >> Complex >> Complex to Polar)*. Si la parte real es mayor a cero el primer bit obtenido es un cero, caso contrario se obtiene un uno. De la misma forma, si la parte imaginaria es mayor a cero el segundo bit obtenido es un cero, caso contrario se obtiene un uno. Este procedimiento con la función *Select* *(Functions >> Programming >> Comparison >> Select),* en su orden. Una vez obtenidos estos dos bits se forma un arreglo de dimensión 1x2, esto es posible con la función *Build Array* *(Functions >> Programming >> Array >> BuildArray)*. Este procedimiento es realizado para todos los símbolos recibidos, realizando una conversión de una matriz de

Nx2 a un arreglo de una dimensión con la función *Reshape Array*  *(Functions >> Programming >> Array >> Reshape Array).*

3.1.5 MODULADOR 16QAM

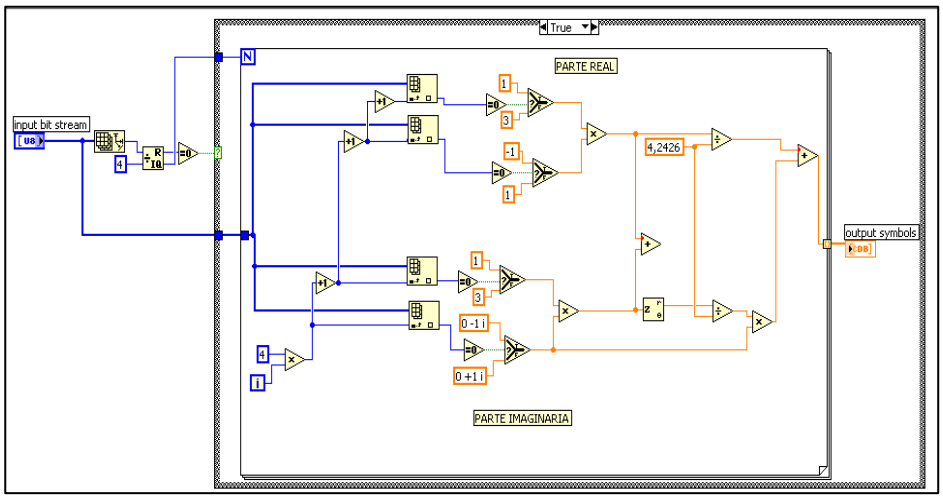


Fig. 3.7 Diagrama de bloques del modulador 16QAM

Inicialmente el sistema debe comprobar que la longitud de bits entrantes sea múltiplo de cuatro, de modo que sea posible su modulación y demodulación, caso contrario, ignora la secuencia y envía un mensaje de error. Para ésta comprobación utilizamos las funciones *Array Size*  *(Functions >> Programming >> Array >> ArraySize),Quotient&Remainder*  *(Functions >> Programming >> Numeric >> Quotient&Remainder)* y *Equal to 0?*  *(Functions >> Programming >> Comparison >> Equal to 0?).*

A continuación se evalúan los cuatro primeros bits cuyo valor determinará una característica del símbolo. Si el primer bit es igual a cero, la parte imaginaria del símbolo tendrá signo negativo, caso contrario tendrá signo positivo. Si el segundo bit es igual a cero, la magnitud de la parte imaginaria será igual a uno, caso contrario será igual a tres. Si el tercer bits es igual a cero, la parte real del símbolo tendrá signo negativo caso contrario tendrá signo positivo. Si el cuarto bit es igual a cero, la magnitud de la parte real será igual a uno, caso contrario será igual a tres. Éste procedimiento de evaluación y selección es realizado con la función *Index Array*  *(Functions >> Programming >> Array >> Index Array*), *Equal to 0?*  *(Functions >> Programming >> Comparison >> Equal to 0?), Select*  *(Functions >> Programming >> Comparison >> Select) y Re/Im to Complex*  *(Functions >> Programming >> Numeric >> Complex >> Re/Im to Complex).*

3.1.6 DEMODULADOR 16QAM

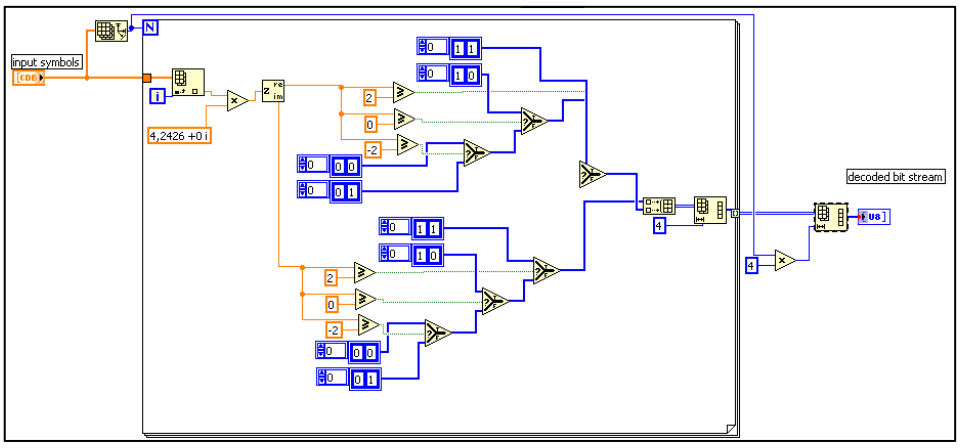


Fig. 3.8Diagrama de bloques del demodulador 16QAM

Para el demodulador 16QAM los límites de las regiones de decisión a favor de un símbolo están situados en dos positivo, cero y dos negativo, tanto en el eje real como en el eje imaginario. Por lo que inicialmente el sistema de evaluar tanto la parte real como la parte imaginaria del símbolo actual. Si la parte imaginaria es mayor a +2 se decide a favor del dibit “11”; caso contrario evalúa si la misma es mayor a cero pero menor a +2 en cuyo caso se decide a favor del dibit “10”; si ésta condición es falsa se evalúa si ésta parte imaginaria es mayor a -2 pero menor a cero para decidirse a favor del dibit “00”; si no se cumplen ninguna de las condiciones nombradas se entiende que la parte imaginaria del símbolo es menor a -2 y por lo tanto debe decidirse a favor de un “01”. Se utilizaron las funciones *Select (Functions >> Programming*

>> Comparison >> Select) y *Greather or Equal?*  *(Functions >> Programming >> Comparison >> Greather or Equal?)*

Éste procedimiento se evalúa para la parte real del símbolo, quedándonos dos dibits, esto es, una matriz de 2x2, lo cual debemos cambiar a un arreglo de 1x4; esto es posible utilizando la función *Reshape Array*  *(Functions >> Programming >> Array >> Reshape Array).*

3.2 CÓDIGOS DE BLOQUE LINEALES

3.2.1 CODIFICADOR

Como se estudió en el capítulo 2, el codificador de canal es el encargado de construir secuencias de bits con redundancia estructurada. La idea principal del algoritmo codificador de bloques lineales diseñado es de una secuencia de bits de mensaje, agruparlos en di-bits de información y pasarlos por una matriz generadora cuyo resultado permitirá que el receptor identifique y corrija errores. Se ha diseñado un codificador sistemático, de modo que los bits de información forman parte de la palabra de código, la cual puede estar al inicio o al final de cada palabra. Para el algoritmo codificador diseñado, cada palabra de código

estará compuesta por 3 bits de redundancia y 2 de información ubicados en ese orden. Se puede demostrar que a la salida de nuestro codificador tendremos 4 posibles secuencias, cada una de ellas de 5 bits. La matriz generadora ha sido diseñada de modo que la distancia de Hamming de las palabras de código obtenidas en el codificador sea la más grande posible, como consecuencia de esto, la probabilidad de error del sistema debido a la cercanía de las palabras de código es la menor posible.

El procedimiento seguido por nuestro algoritmo se ilustra mediante un diagrama de flujo en la figura 3.9 y el diagrama de bloques en la figura 3.10.

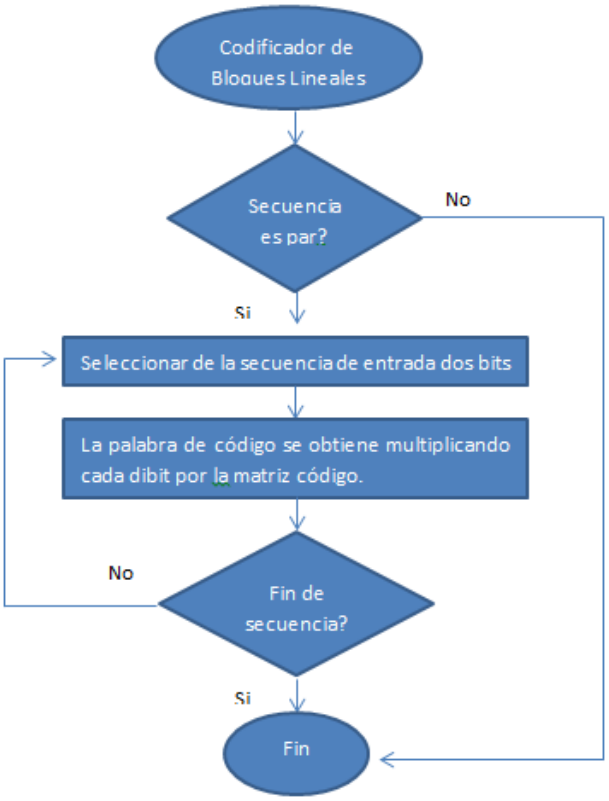


Fig. 3.9 Diagrama de flujo del codificador convolucional

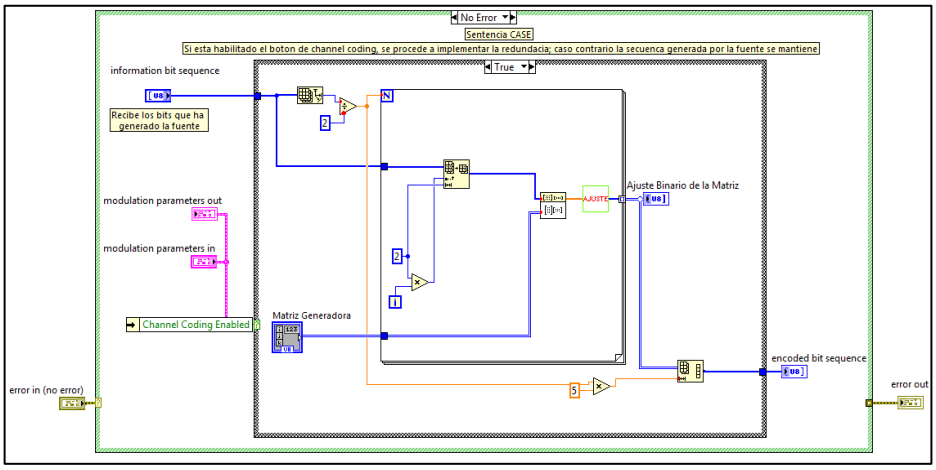


Fig. 3.10 Diagrama de Bloques del Codificador de Bloques Lineal

Inicialmente contamos con el cluster “modulation parameters in” el cual tiene varios parámetros importantes para el proceso de modulación en el sistema OFDM.

De éste clusterse extrae el control“Channel Coding Enabled” 

que es una variable de tipo booleano que sirve de referencia a nuestro algoritmo para aplicar la codificación de canal o no. Esto se realiza con la función *Unbundle by Name*  *(Functions>>Programming>> Cluster,Class&Variant>>UnbundleBy Name)*que extrae los parámetros de un cluster.

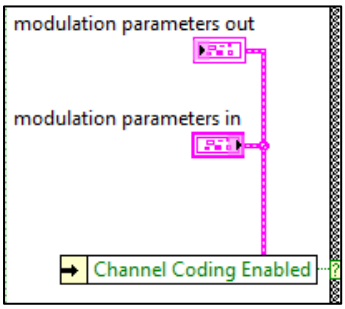


Fig. 3.11 Cluster que contiene parámetro importante de la transmisión

El procedimiento a seguir es diferente para un sistema que utilice codificación y uno sin codificación, por lo que para diferenciar esto utilizamos la estructura CASE (Functions >> Programming >>

Structures >> Case). Si se encuentra habilitada la codificación de canal procedemos a procedimiento que se detalla a continuación:

El algoritmo obtendrá el tamaño de la secuencia de bits que ha sido generada por la fuente. Para ello utilizamos la función*Array Size*  *(Functions>> Programming>>Array>>Array Size),* la cual extrae el tamaño de un arreglo que en nuestro caso representa la longitud de la secuencia de bits.

A esta longitud la dividiremos para 2, puesto que así será como tomaremos nuestros datos, el mismo que será utilizado para indicarle a nuestro programa el número de veces que deberá realizar el procedimiento. Esto es posible con la función *Divide*  *(Functions>>Programming>>Numeric>>Divide),* la cual divide dos números.

Aplicamos redundancia estructurada a los bits, para ello tomamos los 2 primeros bits y los multiplicamos por una matriz generadora G de dimensión (2x5), con el objetivo de obtener una secuencia de 5 bits como palabra de código. Para lo cual se extrae una parte del arreglo con la función *Array Subset* 

(Functions>>Programming>>Array>>ArraySubset) y del sub-VI AJUSTE.vi explicando más adelante.

Se repite este procedimiento con todos los bits restantes de la secuencia con la ayuda de un lazo FOR, como se ilustra en la figura 3.12.

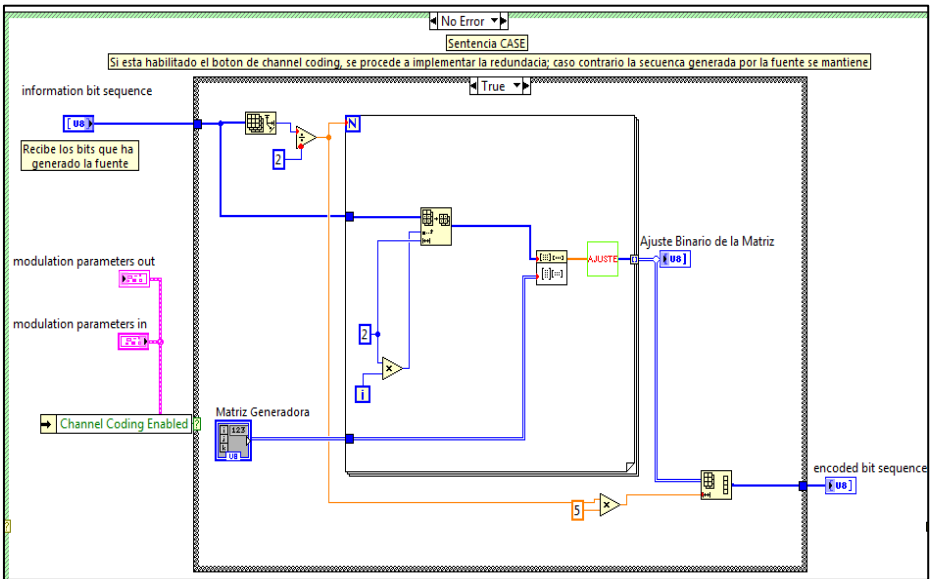


Fig. 3.12Diagrama de Bloques del Codificador de Bloques Lineal para Case True

La Matriz generadora seleccionada tiene la propiedad que sus filas son ortogonales entre sí y que las palabras de código obtenidas de ella tienen una distancia de hamming mayor o igual a uno, como fue explicado en el capítulo 2.

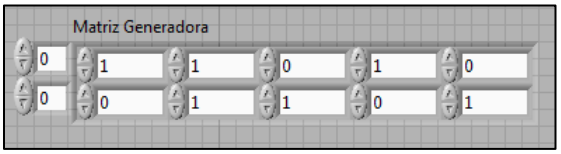


Fig. 3.13 Matriz Generadora implementada

De la multiplicación del paso anterior sabemos que cada secuencia generada puede contener cualquier número entero positivo, pero para efecto de nuestra implementación necesitamos que las secuencias contengan únicamente unos y ceros. AjusteBinario.vi se encarga de realizar ésta conversión mediante el siguiente procedimiento:

Toma uno a uno los 3 bits de la secuencia generada con ayuda de la función *IndexArray* (Functions >> Programming >> Array >> IndexArray) y los divide para dos con la función *Quotient&Remainder (Functions >> Programming >> Numeric >> Quotient&Remainder).*

Se compara el residuo de la división anterior con el cero con la función *EqualTo0? (Functions >> Programming >> Comparison >> EqualTo0?).*

La condición de verdad es utilizada por el selector *Select*  *(Functions >> Programming >> Comparison >> Select)* para decidir que el número en binario equivale a un 0 caso contrario indicará que es un 1, obteniendo finalmente una secuencia exclusivamente binaria. El resultado de cada una de las secuencias con su respectivo ajuste es llamado síndrome, y será utilizado para detectar y corregir los errores. El funcionamiento del algoritmo de ajuste binario se ilustra en las figuras 3.14 y 3.15:

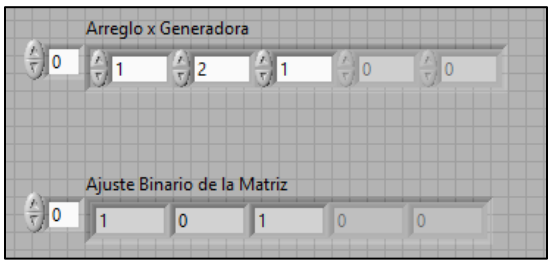


Fig. 3.14 Panel Frontal de "ajusteBinario.vi"

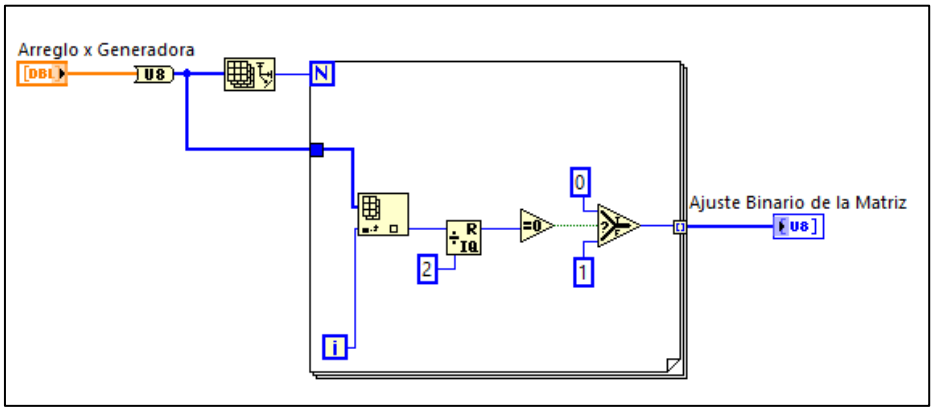


Fig. 3.15 Diagrama de Bloques de "ajusteBinario.vi"

Continuando con el instrumento virtual del codificador de canal, como último paso se realiza un ajuste de dimensiones de nuestro arreglo de forma que tengamos en un arreglo de una sola fila todas las nuevas secuencias de bits, a fin de obtener nuestro arreglo final al cual llamaremos encoded_bit_sequence, procedimiento que se ilustra en las figuras 3.16 y 3.17. En el caso que la codificación de canal estuviera deshabilitada, el algoritmo procede simplemente a dejar la secuencia generada por la fuente sin cambios. La función utilizada es *ReshapeArray*  *(Functions >> Programming >> Array >> Reshape Array).*

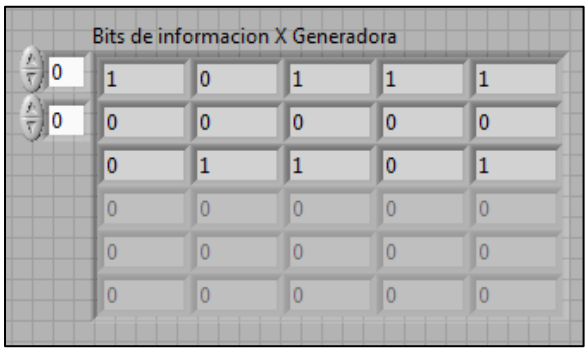


Fig. 3.16 Matriz de sub-secuencias de bits codificada

Fig. 3.17 Secuencia de bits codificada en forma de 1 dimensión

3.2.2 DECODIFICADOR

El siguiente algoritmo realizará el proceso inverso al algoritmo codificador. El objetivo es detectar y corregir errores que podrían ser introducidos por el canal inalámbrico utilizando los bits redundantes agregados por el codificador. En primer lugar tomaremos secuencias de 5 bits y los multiplicaremos por su respectiva matriz de comprobación previamente establecida y obtenida a partir de la definición de la matriz generadora. Utilizaremos el teorema del vector nulo para detectar si existen errores y de ser el caso buscaremos dentro de nuestros 8 síndromes posibles con el objetivo de identificar el bit erróneo y proceder a corregirlo. Finalmente descartaremos los bits redundantes y extraemos los bits de información de nuestras secuencias ya corregidas que como lo mencionamos en la sección del codificador serán siempre las 2 últimos bits de cada palabra de código.

El procedimiento realizado por nuestro algoritmo se ilustra mediante un diagrama de flujo en la figura 3.18 y el diagrama de bloques en la figura 3.19.

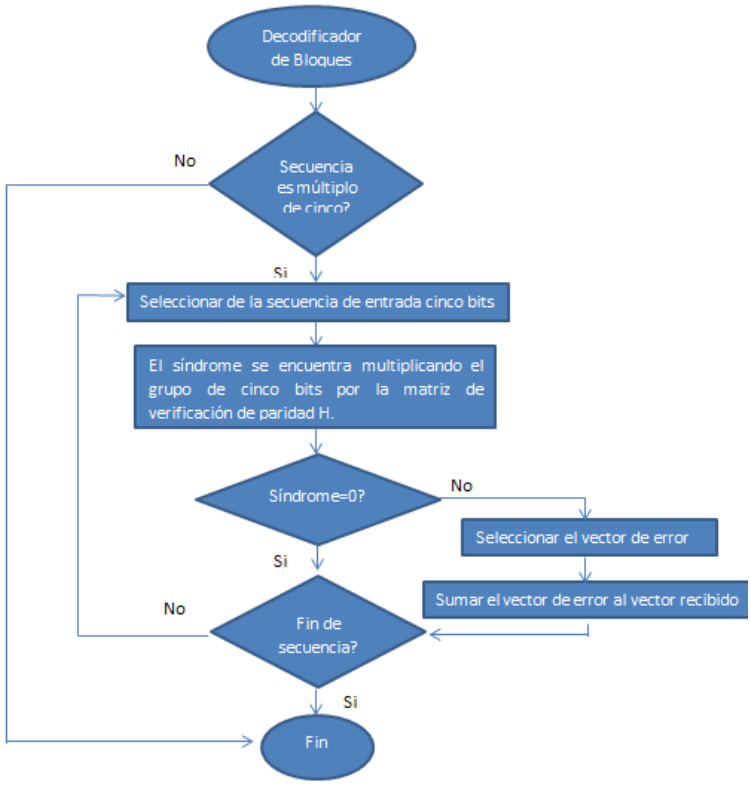


Fig. 3.18 Diagrama de flujo del decodificador de bloques lineales

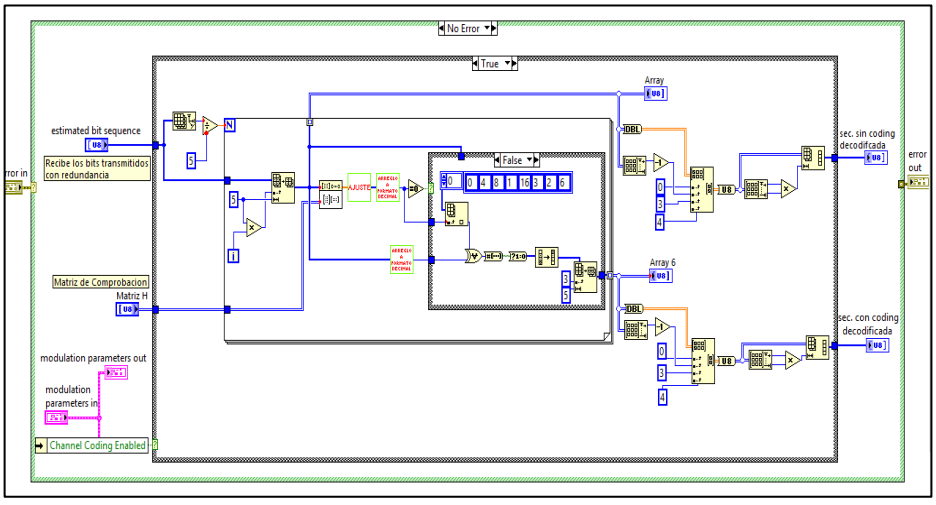


Fig. 3.19 Diagrama de Bloques del Decodificador de Bloques Lineal

Del cluster “modulation parameters out” extraemos el valor booleano que nos indicará si el sistema está utilizando channel coding o no. Utilizamos un Unbundle by name *(Functions >> Programming >> Cluster,Class & Variant >> Unbundle by Name)*; este nos permite extraer cualquier parámetro de la modulación usando únicamente el nombre de la variable asociado a ella.

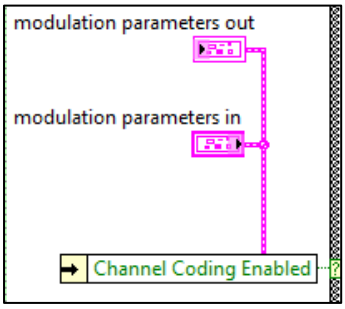


Fig. 3.20 Cluster *Modulation Parameters in* en el Decodificador

Extraemos el número de bits que se han receptado en nuestra estimate_bit_secuence con ayuda de la función *Array Size (Functions >> Programming >> Array >> Array Size)* y lo dividimos para 5, esto determinara el número de secuencias que debemos analizar. De aquí que utilizamos la función *Array Subset (Functions >> Programming >> Array >> Array Subset)* para crear secuencias de 5 bits.

Fig. 3.21 Secuencia de bits recibida

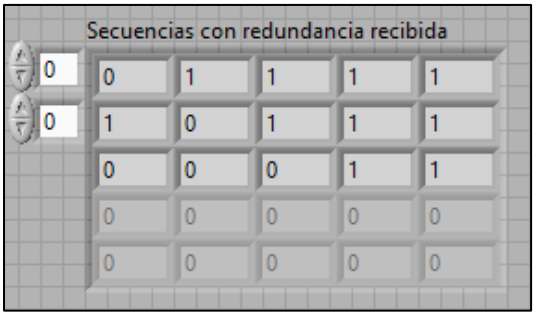


Fig. 3.22 Sub-secuencias de bits en el decodificador

Estas secuencias obtenidas las multiplicaremos una a una por la matriz de verificación de paridad H que es un parámetro del decodificador haciendo uso de la función *AxB*  *(Functions >> Programming >> Array >> Matrix >> AxB)*.

Nuestra matriz de comprobación de paridad H ingresada es de tamaño 5x3 tal como se muestra en la figura 3.23, la cual fue obtenida a partir dela matriz generadora G según lo explicado en el capítulo 2.

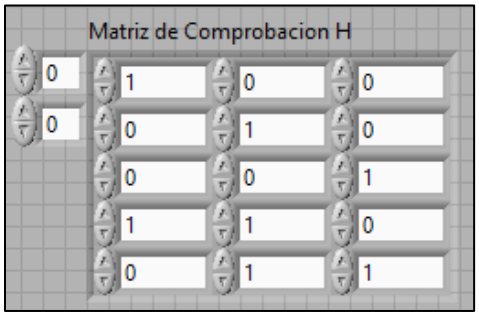


Fig. 3.23 Matriz de comprobación H

En este momento como podemos interpretar, el resultado de la multiplicación de la secuencia de cinco bits recibidos con la matriz de verificación de paridad H será un arreglo de tamaño 1x3 que puede contener cualquier número entero positivo. Por lo tanto debemos realizar el ajuste necesario de tal forma que las secuencias sean únicamente binarias.

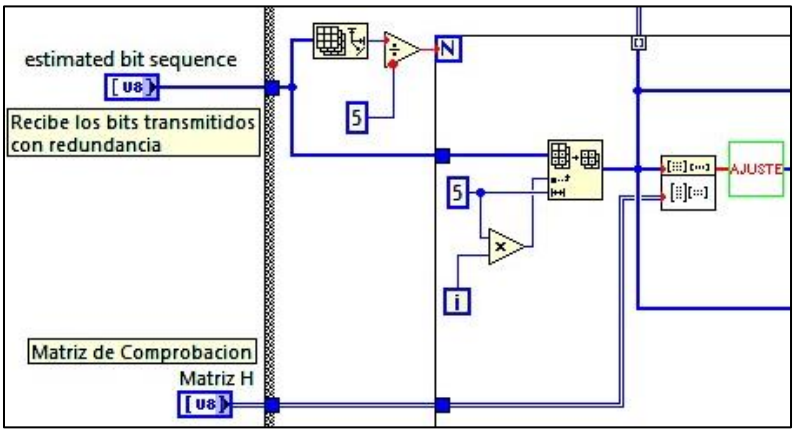


Fig. 3.24 Procedimiento inicial en el decodificador

Se puede observar que la función ajusteBinario.vi se utiliza también del lado del decodificador. Las figuras 3.14y 3.15 ilustraron su funcionamiento. La secuencia de bits obtenida hasta éste momento es conocida como síndrome.

Como se estudió en el capítulo dos, conocemos que si el producto entre la secuencia recibida y la matriz verificación no es un vector nulo, esto indicará la presencia de errores en la secuencia recibida caso contrario la secuencia es igual a la transmitida. Por lo tanto verificamos el síndrome. Si el vector no es nulo (los 3 bits no son ceros) procedemos a obtener cual fue el valor obtenido en representación en base 10 para lo cual hemos creado la función BinaDec.vi cuyo procedimiento a seguir es el siguiente:

Obtenemos uno a uno los bits y los multiplicamos por la base 2 elevado a su respectiva potencia y se suman todos los números obtenidos. Se utilizó la función *Array Size*  *(Functions >> Programming >> Array >> Array Size)* para tomar la longitud de la secuencia recibida como límite de un lazo for para realizar un procedimiento algunas veces. La matriz de datos binarios se convierte en un arreglo de una dimensión con ayuda de la función

ReshapeArray  (Functions >> Programming >> Array >> Reshape Array). A continuación debemos obtener cada uno de los elementos del arreglo de datos binarios y obtendremos su equivalente decimal utilizando las potencias de dos con la función *Scale by Power of 2*  *(Functions >> Programming >> Numeric >> Scale By Power of 2).* Finalmente se realiza una suma acumulada de cada uno de los valores obtenidos de las potencias de dos que representan el número decimal. El proceso mencionado del sub-VI BinaDec.vi se ilustra en las figuras 3.25 y 3.26.

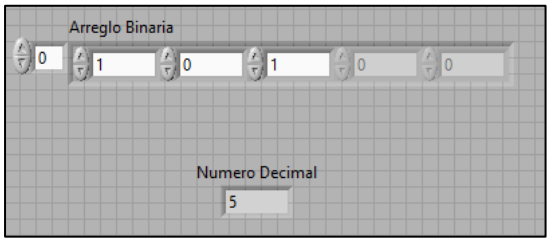


Fig. 3.25 Panel Frontal del VI "BinaDec.vi"

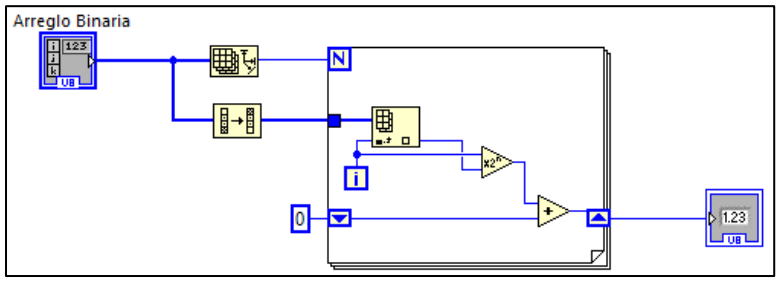


Fig. 3.26 Diagrama de bloques del VI "BinaDec.vi"

Si el resultado obtenido a la salida de BinaDec.vi es igual a cero, el decodificador no realizará ningún tipo de corrección y pasará la misma secuencia a la salida tal como fue recibida, caso contrario procederemos a corregir los errores. Para realizar ésta decisión se utiliza una estructura de selección *Case (Functions >> Programming >> Structures >> Case Structure).*

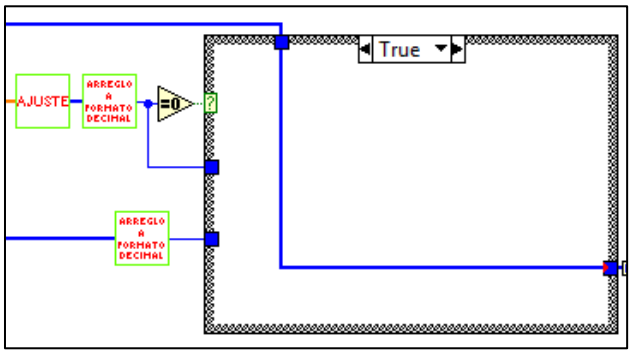


Fig. 3.27 Secuencia sin errores se envía directamente con secuencia decodificada

Para corregir los errores, el algoritmo toma el resultado en base decimal y procede a verificar en la tabla de síndromes y vectores de error. Se extrae la secuencia de error que correspondiente al síndrome obtenido teniendo un arreglo constante donde ingresamos en formato decimal los vectores de error que forman parte de nuestro decodificador de bloques lineales. Esto es

realizado con la función *Array Constant*  *(Functions >> Programming >> Array >> ArrayConstant).*

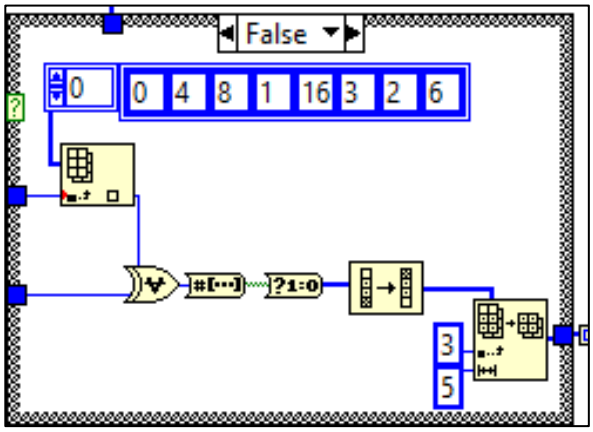


Fig. 3.28 Arreglo con los posibles síndromes

El proceso resta en módulo 2 se realiza con ayuda de la función *Array Size*  *(Functions >> Programming >> Array >> Array Size)*, en donde el índice del arreglo corresponde al valor en base decimal obtenido en el paso anterior al cual se aplicamos la función *ExclusiveOr*  *(Functions >> Programming >> Boolean >> ExclusiveOr)* con el fin de encontrar la resta en módulo 2 de los dígitos.

Se cambia el formato de la secuencia de entero sin signo de 8 bits a un arreglo booleano primero colocando en bloque *Number to*

Boolean Array  *(Functions >> Programming >> Boolean >> Number to Boolean Array)* cuyo resultado será un arreglo de valores T o F cuya representación estará dada por un tamaño de 8 bits y en orden de significancia cambiada, esto es el bit menos significativo a la izquierda; seguimos con la función *Boolean to (0,1)*  *(Functions >> Programming >> Boolean >> Boolean to (0,1))*. Como el orden de los bits está invertido es necesario utilizar la función *Reverse 1D Array*  *(Functions >> Programming >> Array >> Reverse 1D Array).*

Descartamos los primeros 3 bits ya que estos no contendrán información alguna, pues corresponden a la parte más alta del ajuste binario que contendrá siempre ceros; obteniendo de esta manera los 5 bits que nos llegaron siendo corregidos sus errores. Se utiliza la función *Array Subset*  *(Functions >> Programming >> Array >> ArraySubset).*

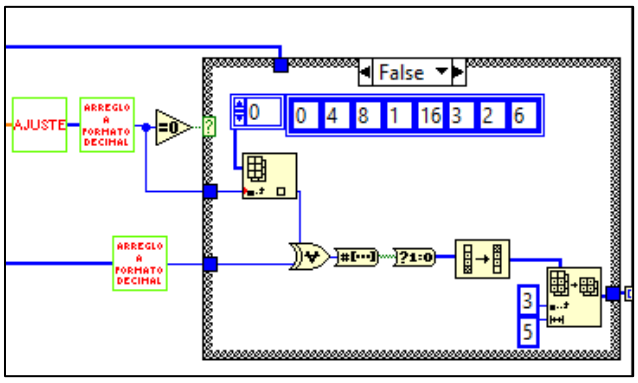


Fig. 3.29 Procedimiento para corrección de errores en el decodificador

A continuación, las figuras 3.30 y 3.31 nos ilustran los resultados del proceso de detección y corrección de errores utilizando codificación de bloques lineales.

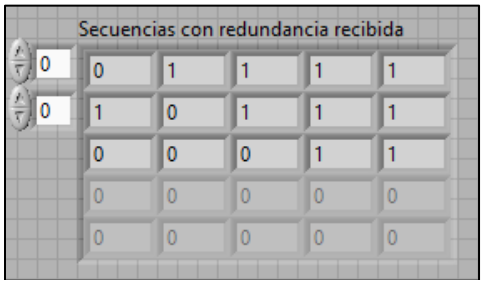


Fig. 3.30 Secuencia de bits recibida con errores

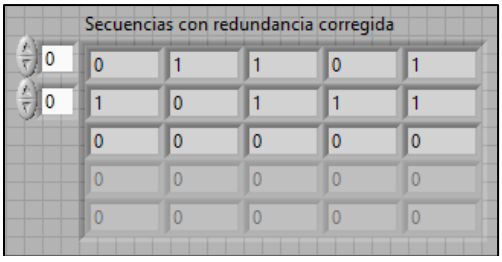


Fig. 3.31 Secuencia de bits recibida después de corregir los errores

Se puede observar que gracias a la redundancia estructurada agregada en las secuencias generadas en el transmisor, de los cinco bits que forman la trama sólo 2 son bits de información y los 3 bits restantes son bits de redundancia.

Empleamos la función *Get Submatrix*  *(Functions >> Programming>> Array >> Matrix >> Get Submatrix),* cuyos índices están determinamos como se muestra en la figura 3.32.

Las siguientes funciones también hacen posible la implementación de la última parte del decodificador mostrado en la figura 3.19:

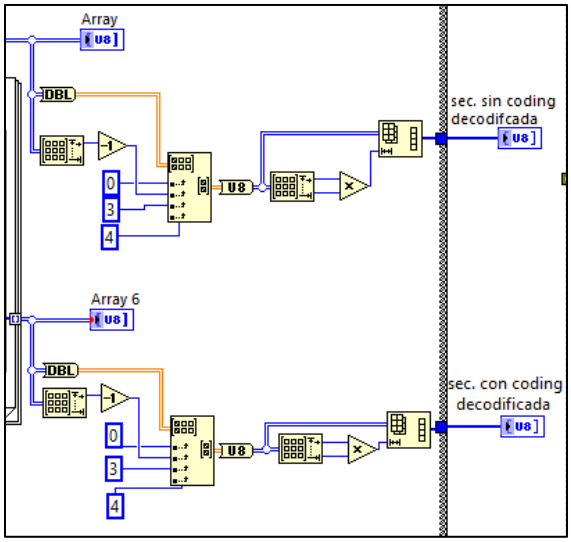


Fig. 3.32 Procedimiento para obtener la secuencia de bits de información decodificada

La figura 3.33 ilustra los resultados obtenidos enviando una secuencia “011100”, se puede observar que cuando no se aplica ningún tipo de codificación de canal la secuencia llega con errores que son corregidos por el algoritmo codificador y decodificador de bloques lineales.

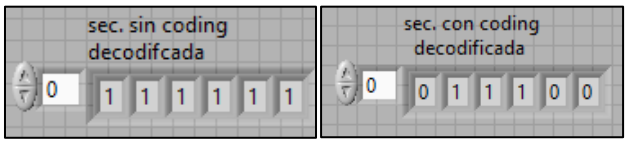


Fig. 3.33 Comparación entre una secuencia de bits de información obtenida sin utilizar codificación de canal y una que utiliza codificación de bloques lineal

3.3 CÓDIGOS CONVOLUCIONALES

Como es evidente, los códigos de bloques lineales tienen limitaciones a la hora de corregir errores, ya sea porque el código decide a favor de otra secuencia debido a la cantidad de posibilidades que existen o bien porque la cantidad de síndromes es limitada frente a todos los posibles errores que puede ocurrir. Los códigos convolucionales, como fue estudiado en el capítulo2, intentarán superar varios de estos problemas a fin de mejorar la Tasa de Errores por Bit (BER). Debido a que los bits de las secuencias que llegan dependen de sus estados anteriores, esto nos permite tener un mecanismo adicional de control sobre las secuencias. Para la implementación partimos tomando 1 bit de la secuencia generada

por la fuente, y lo ingresamos a través de un registro de desplazamiento, que para nuestro ejemplo es de tamaño 3 y que esta encerado inicialmente. Procedemos a generar una secuencia de 3 bits; en la cual el primer bit será el bit de información extraído desde la fuente, el segundo bit será la suma en módulo 2 del bit del primer y el tercer lugar del registro de desplazamiento y el tercer bit será la suma de los 3 bits del registro de desplazamiento, obteniendo de esta forma la secuencia final que será transmitida. Se realiza el mismo procedimiento con cada uno de los bits de la fuente considerando que a partir de éste momento los registros de desplazamiento no estarán encerados sino que tendrán los primeros 2 bits del estado anterior. Esta característica es precisamente lo que genera un estado de dependencia en todas las secuencias transmitidas mejorando, como ya se explicó en la sección 2.8, la tasa de errores, ya que existirá un menor número de opciones sobre las cuales escoger el decodificador.

El procedimiento seguido por el algoritmo implementado se ilustra mediante un diagrama de flujo en la figura 3.34.

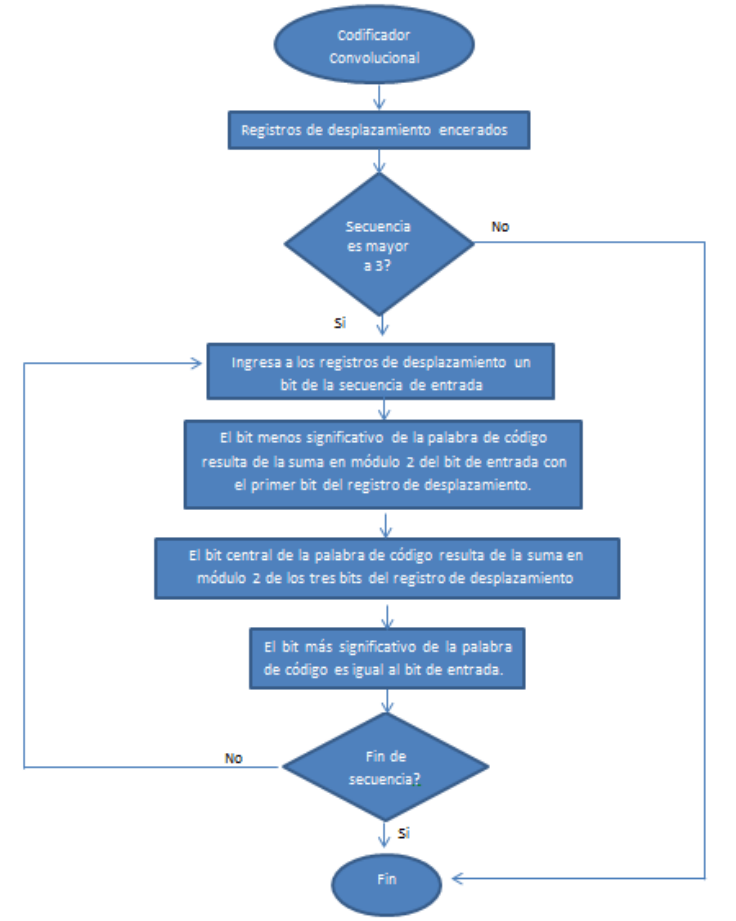


Fig. 3.34 Diagrama de flujo del codificador convolucional

3.3.1 CODIFICADOR CONVOLUCIONAL

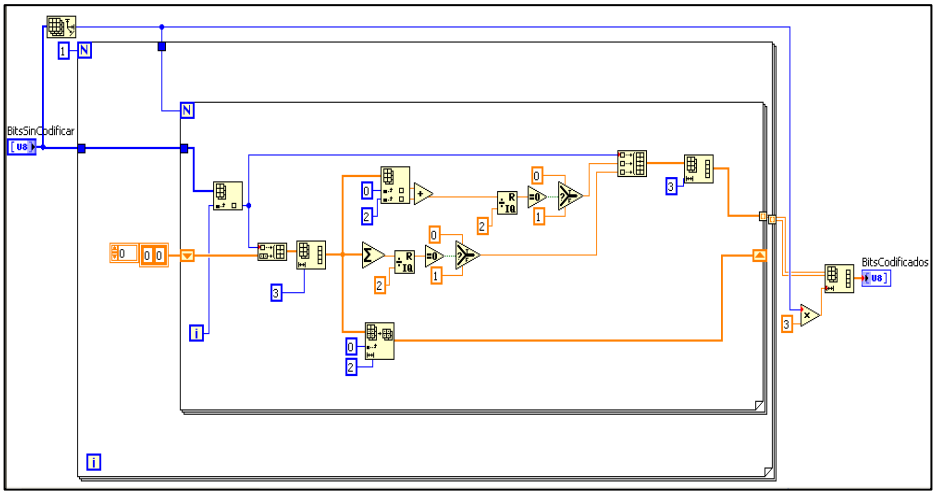


Fig. 3.35 Diagrama de Bloques del Codificador Convolucional

Tomamos el primer bit con la función *Index Array* *(Functions >> Programming >> Array >> Index Array)* y lo concatenamos junto con el arreglo 00 para formar nuestro registro de desplazamiento para lo cual utilizamos la función*Build Array* *(Functions >> Programming >> Array >> Build Array).*Como de la función anterior obtenemos una matriz debemos cambiar su formato a un arreglo de 3 elementos con la función*Reshape* Array *(Functions >> Programming >> Array >> Reshape Array).*A continuación, utilizamos el bloque *Index Array* *(Functions>> Programming >> Array >> Index Array)*para obtener el primer y el ultimo bit, los cuales me servirán para realizar la suma en módulo dos que

corresponde al primer bit de la codificación. Esto es posible sumando normalmente los dos bits y después para convertir a formato binario, éste resultado se divide para dos con la función *Quotient&Remainder*  *(Functions >> Programming >> Numeric >> Quotient&Remainder)*. y se evalúa si el residuo es cero, en cuyo caso se muestra el bit es un cero caso contrario será un uno, lo cual es posible gracias a la función *Select*  *(Functions >> Programming >> Comparison >> Select).*

Al mismo tiempo se realiza la suma en módulo 2 de los tres elementos del registro de desplazamiento que tiene capacidad 3. Esto es posible usando la función *Add Array Elements*  *(Functions >> Programming >> Numeric >> Add Array Elements)*y el mismo procedimiento para conversión a formato binario.

Finalmente para concatenar el bit de información, con el resultado de la suma en módulo 2 entre el primer y tercer bit y con el resultado de la suma en módulo 2 entre los tres bits, se utiliza la función *Build Array* *(Functions >> Programming >> Array >> Build Array).*Como la salida que se obtiene hasta este punto es una matriz, debe ser convertida a arreglo de 1x3 utilizando la

función *Reshape* Array  *(Functions >> Programming >> Array >> Reshape Array).*

Este procedimiento se realiza para cada uno de los bits de la secuencia entrante, con la diferencia que los registros se van desplazando hacia la derecha por cada vez que ingresa un bit, procedimiento que se realiza dentro del lazo de repetición for colocando un *shift register* el cual tendrá por valor inicial 00, que será el arreglo para el primer bit y como valor final o de realimentación tendrá los 2 primeros bits de cada registro de desplazamiento del estado anterior.

3.3.2 DECODIFICADOR CONVOLUCIONAL

El decodificador convolucional diseñado ha sido basado en el algoritmo de Viterbi, cuyo funcionamiento ha sido explicado en el capítulo 2. Debido a la complejidad de su implementación, se diseñaron funciones o sub-VI con el fin de distribuir cada uno de los procedimientos a ser realizados. A su vez, éstas funciones se agruparon en una librería llamada ConvolutionalLibrary.llb. A continuación, explicaremos el programa principal seguido de cada una de las funciones implementadas.

El procedimiento seguido por el algoritmo implementado se ilustra mediante un diagrama de flujo en la figura 3.36 y su diagrama de bloques en la figura 3.37.

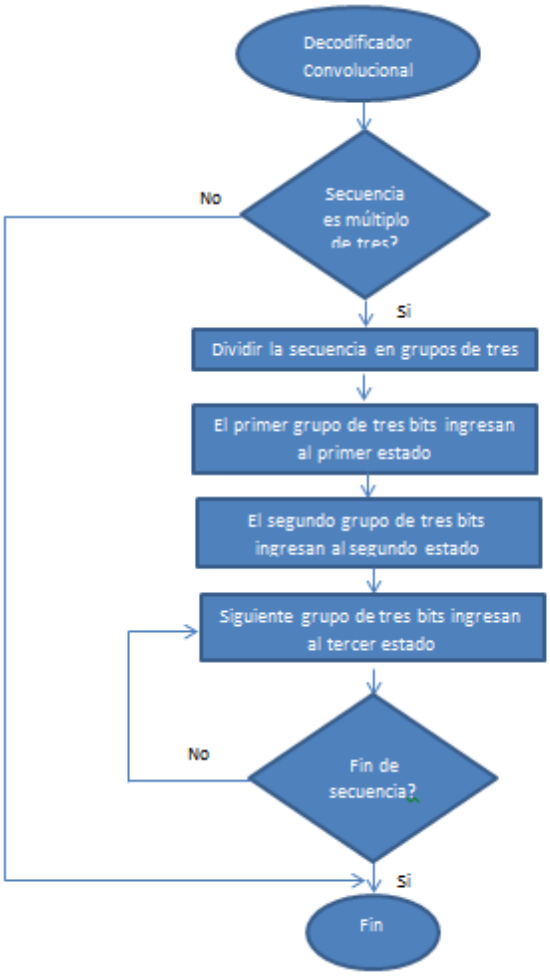


Fig. 3.36 Diagrama de flujo del decodificador convolucional

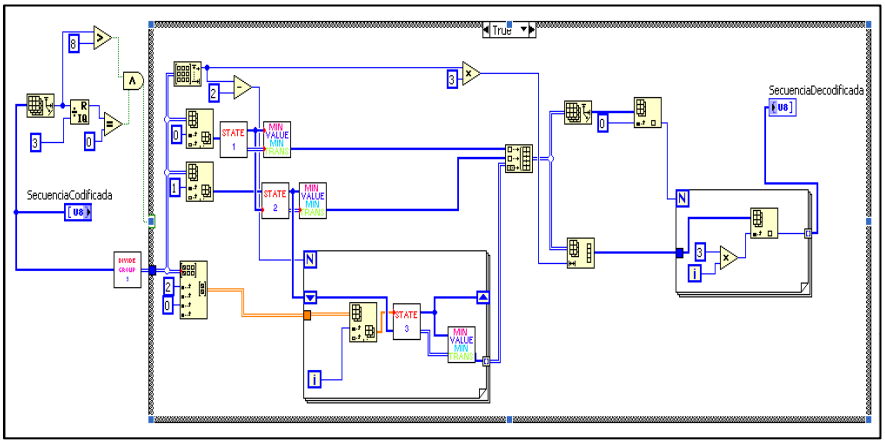


Fig. 3.37 Diagrama de bloques del decodificador convolucional

Inicialmente el programa verifica que el tamaño de la secuencia recibida sea múltiplo de tres, lo cual es de esperarse debido a que el codificador convolucional implementado en éste proyecto, por cada bit de información genera un grupo de tres bits. Esto lo hace con ayuda de la función *Array Size*  *(Functions >> Programming >> Array >> Array Size),* así como de la función *Quotient and Remainder*  *(Functions >> Programming >> Numeric >> Quotient and Remainder),* siendo Remainder la salida correspondiente al residuo de la división que se utiliza para verificar que la secuencia sea de longitud mayor o igual a nueve bits, esto es con el objetivo de realizar la decodificación en al menos tres estados, uno para cada terna de bits. En esta parte también se utiliza la función *Greater or Equal?*  *(Functions >>*

Programming >> Comparison >> Greater or Equal?), siendo la constante de comparación el número 9.

Con esto se pretende verificar que la secuencia recibida sea válida para la decodificación del codificador convolucional. A continuación, la secuencia grande de bits es divida en grupos de tres bits, esto es posible con ayuda de la función *DivideGroup3.vi* 

(ConvolutionalLibrary >> DivideGroup3.vi) detallada en las funciones de la librería Convolutional.llb.

El primer grupo de bits se pasa al bloque *State1.vi* 

(ConvolutionalLibrary >> State1.vi) que implementa el primer estado correspondiente a la decodificación convolucional, es decir, el estado en que los registros estaban encerados e ingresó al sistema el primer bit. Ésta función tiene dos salidas: LabelMatrix y TransitionMatrix, la primera identifica la matriz de distancias entre la terna de bits actual y cada una de las ramas del mismo estado y la segunda representa las ternas de bits que identifican cada una de las ramas de salida del estado. El algoritmo implementado por ésta función será detallado más adelante.

El segundo grupo de bits se pasa al bloque *State2.vi*  *(ConvolutionalLibrary >> State2.vi)* que implementa el segundo estado correspondiente a la decodificación convolucional, es decir, el estado en que el tercer bit es aún un cero, el bit central es el primer bit entrante y el bit más significativo es el bit que recién entra al sistema. Para el correcto funcionamiento de éste VI necesita la terna actual de bits a evaluarse y la matriz de distancias del estado anterior. Ésta función tiene dos salidas: LabelMatrix y TransitionMatrix, las cuales representan lo mismo explicado para la función State1. El algoritmo implementado por ésta función será detallado más adelante.

Como se estudió en el capítulo 2, a partir del estado 3, los demás estados se vuelven repetitivos por lo que es posible utilizar la función *State3.vi*  *(ConvolutionalLibrary >> State3.vi)*dentro de un lazo for que se repita para el número de ternas de bits faltantes por analizar. El valor que debe ser conectado a N es el número de ternas menos dos, para lo cual se utilizaron las funciones *Array Size*  *(Functions >> Programming >> Array >> Array Size) y Substract*  *(Functions >> Programming >> Numeric >> Substract)*.

State3.vi implementa el tercer estado del decodificador, recibe como entradas la terna actual de bits y la matriz de distancias del estado anterior; siendo sus salidas la matriz de transiciones de éste estado como su matriz de distancias de la terna de bits actual con cada uno de los elementos de la misma matriz de transiciones. Es ésta matriz de transiciones del estado actual que necesita el siguiente estado para su funcionamiento, es decir, al encontrarse dentro de un lazo for se retroalimenta a sí mismo para la próxima ejecución. Para esto se añade un *Shift Register*, siendo el valor inicial la matriz de distancias del estado 2.

A la salida de cada estado, se ubica un bloque *MinValueMinTransition.vi* 

(ConvolutionalLibrary >> SendMinValueMinTransition), que toma la decisión de escoger la transición con menor distancia conociendo la matriz de transiciones y la matriz de distancias.

En éste punto de programa, tenemos cada una de las ternas que más se aproxima a la secuencia original de mensaje enviada por el receptor, formando un arreglo de dos dimensiones Nx3. Para decodificar los bits de cada una de las ternas de mayor

aproximación, se debe primero realizar la conversión a un arreglo de una dimensión; para esto se utiliza la función *ReshapeArray*  *(Functions >> Programming >> Array >> ReshapeArray),* siendo la dimensión a ajustarse igual a la secuencia recibida.

Para realizar la decodificación, se toma el primer bit de cada terna, que corresponde al bit de información. Esto debe colocarse dentro de un lazo FOR que se repita un número de veces igual al número de ternas de bits formados a partir de la secuencia recibida; esto es posible utilizando la función *Array Size*  *(Functions >> Programming >> Array >> Array Size)* y la estructura *For (Functions >> Programming >> Structures >> For Loop)*.

Para llegar a cabo la implementación de este algoritmo hemos creado una librería llamada Convolutional Library, la cual pretende implementar los instrumentos virtuales necesarios para el correcto funcionamiento de un decodificador convolucional haciendo uso del diagrama de enramado estudiado en el capítulo 2.

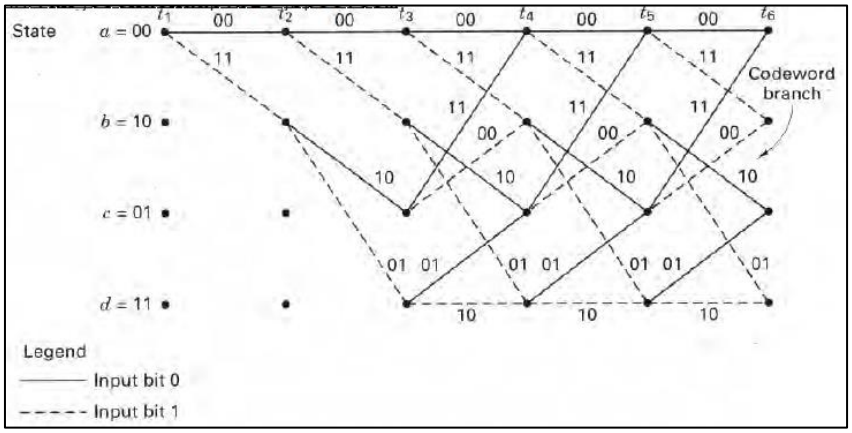


Fig. 3.38 Diagrama de enramado de un codificador convolucional

Cabe indicar que la forma como se ha tomado los estados, es en forma vertical, como se ilustra en la siguiente figura.

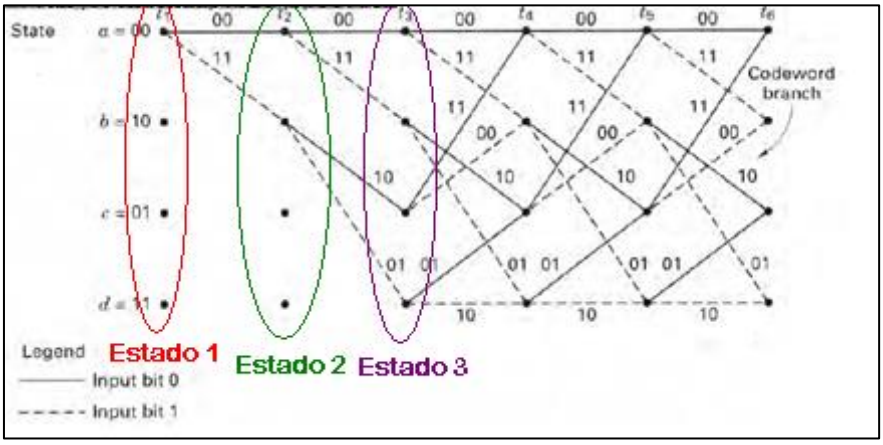


Fig. 3.39 Estados 1, 2 y 3 del diagrama de enramado de un codificador convolucional

El estado 1 está compuesto como lo indica la figura 3.40, la cual busca implementar el primer ramal que constituye un diagrama de enramado.

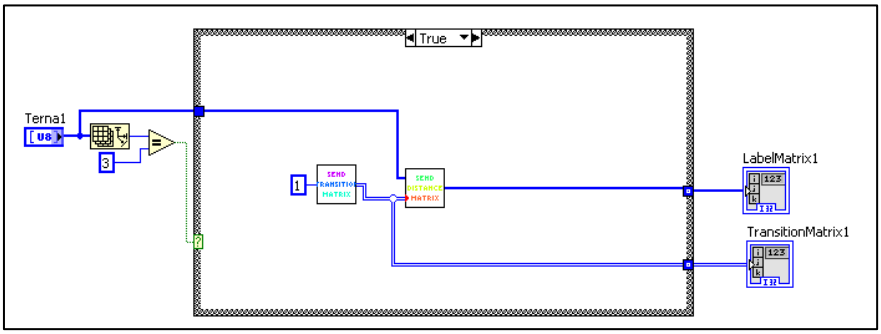


Fig. 3.40 Diagrama de bloques del sub-VI "State1.vi"

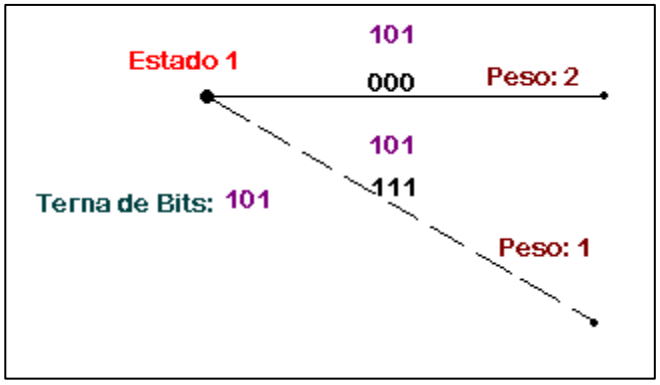


Fig. 3.41 Estado 1 del diagrama de enramado

Este VI está diseñado para verificar que la secuencia que recibe sea de tres bits, para poderla procesar, caso contrario, será ignorada.

Primero debe solicitar la matriz de transiciones de éste primer estado, esto lo hace con ayuda de la función *SendTransitionMatrix.vi*  *(ConvolutionalLibrary >> SendTransitionMatrix),*la cual necesita el número del estado actual, es decir, 1.

Con ésta matriz se debe obtener la matriz de distancias entre la terna de bits actual y las posibles ramas de salida del mismo estado. Esto es posible con ayuda de la función *SendDistanceMatrix.vi*  *(ConvolutionalLibrary >> SendDistanceMatrix)*, que devuelve una matriz que cuyos elementos son las distancias de Hamming entre la terna actual de bits y cada uno de los elementos de la matriz de transiciones de éste estado. Las salidas de éste VI son tanto la matriz de transiciones TransitionMatrix1 de éste estado como la matriz de distancias LabelMatrix1.

La figura 3.42 muestra la implementación del Estado 2 del diagrama de enramado de un codificador convolucional, que se representa en la figura 3.43 a través de un ejemplo.

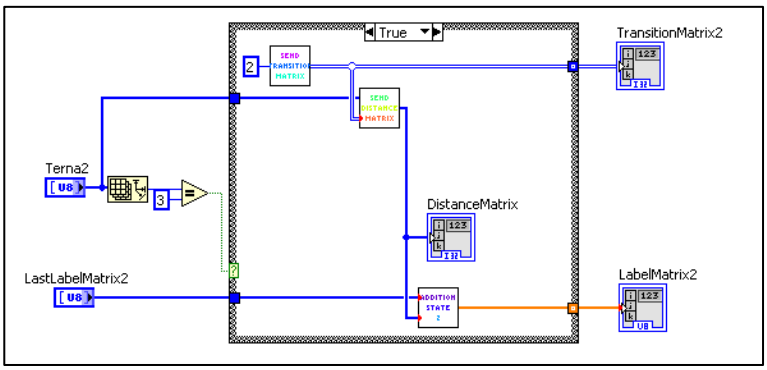


Fig. 3.42 Diagrama de bloques del sub-VI "State2.vi"

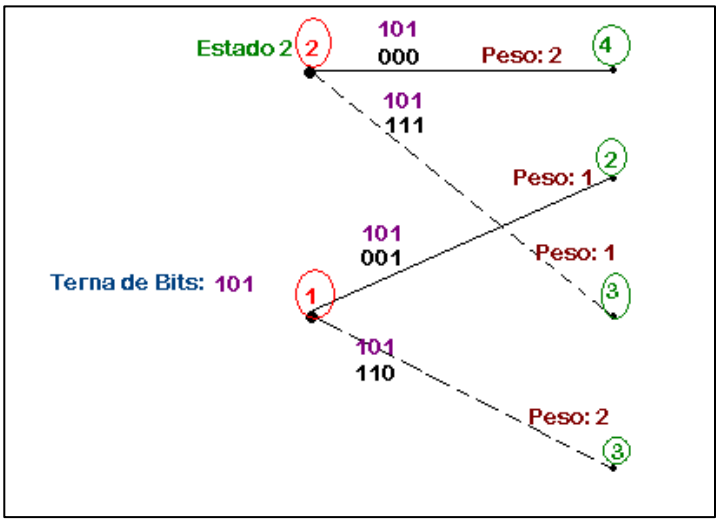


Fig. 3.43 Estado 2 del diagrama de enramado

Este VI, al igual que para State1.vi, está diseñado para verificar que la secuencia que recibe sea de tres bits, para poderla procesar, caso contrario, será ignorada.

Primero debe solicitar la matriz de transiciones de éste estado, esto lo hace con ayuda de la función SendTransitionMatrix.vi que está incluida dentro de ésta librería, la cual necesita el número del estado actual, es decir, 2.

Con ésta matriz se debe obtener la matriz de distancias entre la terna de bits actual y las posibles ramas de salida del estado 2. Esto es posible con ayuda de la función SendDistanceMatrix.vi que se incluye dentro de ésta librería, que devuelve una matriz de distancias de Hamming entre la terna actual de bits y cada uno de los elementos de la matriz de transiciones del estado actual.

Para calcular la matriz de distancias que recibirá el siguiente estado, ha sido implementado el VI *AditionState2.vi* 

(ConvolutionalLibrary >> AditionState2.vi) cuyo funcionamiento será detallado más adelante.

Las salidas de éste VI son tanto la matriz de transiciones de éste estado como la matriz de distancias.

La figura 3.44 muestra la implementación del Estado 3 del diagrama de enramado de un codificador convolucional, que se representa en la figura 3.45 a través de un ejemplo.

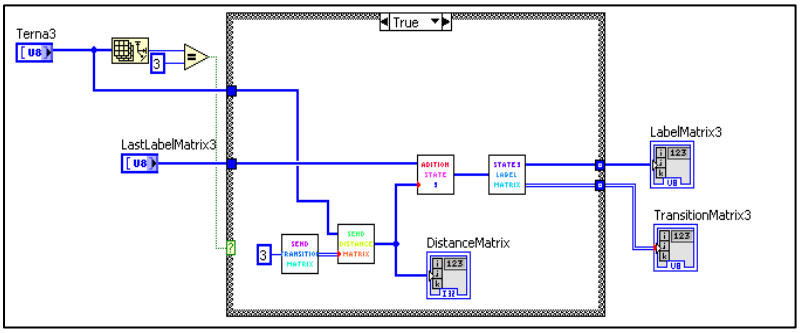


Fig. 3.44 Diagrama de bloques del sub-VI "State3.vi"

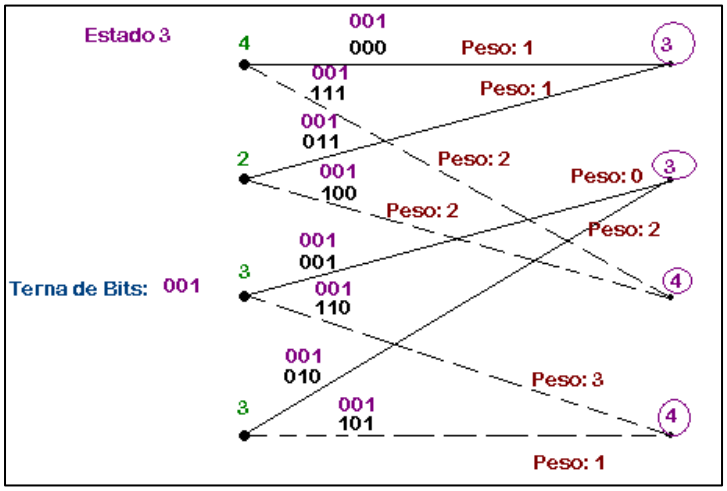


Fig. 3.45 Estado 3 del diagrama de enramado

Su implementación es muy semejante a los estados uno y dos descritos anteriormente. Está diseñado para verificar que la secuencia que recibe sea de tres bits, para poderla procesar, caso contrario, será ignorada.

Primero debe solicitar la matriz de transiciones de éste estado, esto lo hace con ayuda de la función SendTransitionMatrix.vi que está incluida dentro de ésta librería, la cual necesita el número del estado actual, es decir, 3.

Con ésta matriz se debe obtener la matriz de distancias entre la terna de bits actual y cada una de las posibles ramas de salida del estado 3. Esto es posible con ayuda de la función SendDistanceMatrix.vi que se incluye dentro de ésta librería, que devuelve una matriz de distancias de Hamming entre la terna actual de bits y cada uno de los elementos de la matriz de transiciones del estado tres.

Para decidir la matriz de distancias a utilizarse en el siguiente estado, es necesario decidir entre dos caminos que llegan al mismo elemento de la matriz de distancias, la cual se calcula

sumando la distancia acumulada con la nueva distancia de Hamming producto de evaluar la terna de bits actual. Esta parte ha sido implementada a través del VI *AditionState3.vi*  *(ConvolutionalLibrary >> AditionState3.vi).*

La siguiente parte corresponde a tomar la decisión de la transición con menor distancia, en caso de haber dos caminos con la misma distancia acumulada total debe decidir aleatoriamente por cualquiera de los dos. Esto ha sido implementado en la función *State3LabelMatrix.vi*  *(ConvolutionalLibrary >> State3LabelMatrix.vi)*que se incluye en ésta librería y cuyo funcionamiento será detallado más adelante. Las salidas de éste VI son tanto la matriz de transiciones de éste estado como la matriz de distancias.

La función Send Transition Matrix envía la matriz de transiciones de un estado, recibiendo como entrada el número del estado y devuelve como salida la matriz de transiciones del estado solicitado.

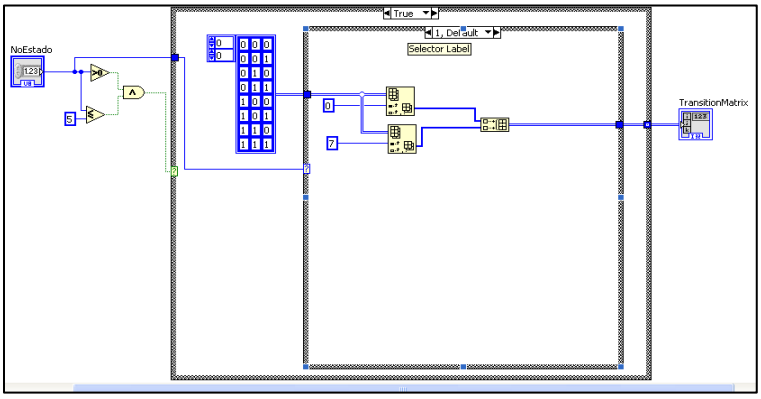


Fig. 3.46 Diagrama de bloques del sub-VI "SemdTransitionMatrix.vi" Case=1

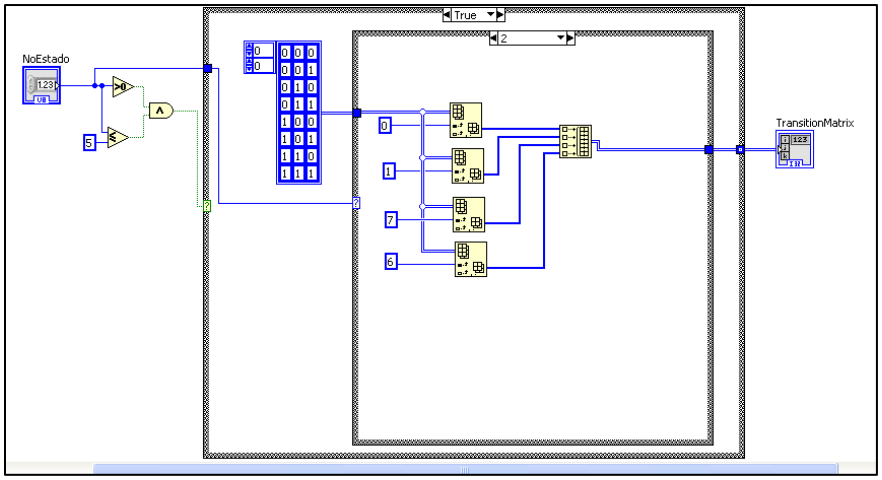


Fig. 3.47 Diagrama de bloques del sub-VI "SendTransitionMatrix.vi" Case=2

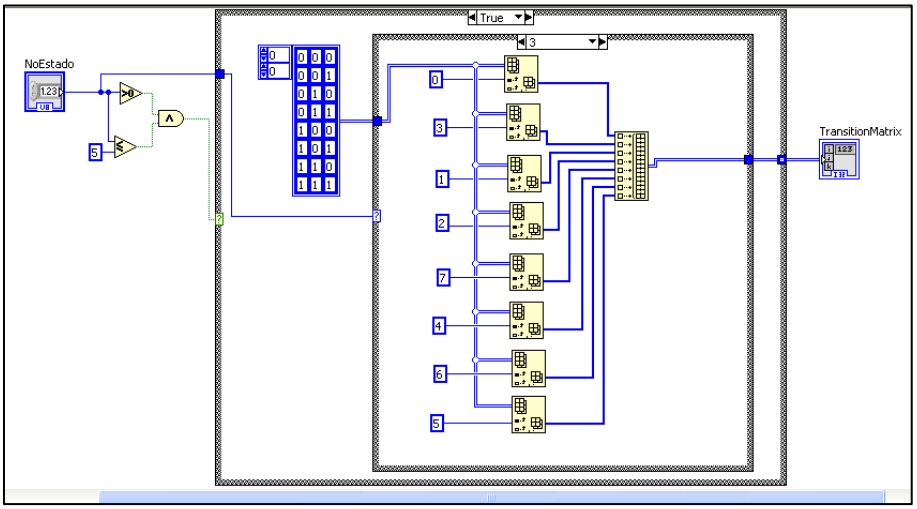


Fig. 3.48 Diagrama de bloques del sub-VI "SendTransitionMatrix.vi" Case=3

Primero verifica que el número del estado se encuentre en un rango válido, esto es, que sea mayor a cero y menor o igual a 3; esto se puede realizar con ayuda de las funciones *Greather than 0?* *(Functions >> Programming >> Comparison >> Greather than 0?),* *Less or Equal? (Functions >> Programming >> Comparison >> Less or Equal?)* y *Función AND* *(Functions >> Programming >> Boolean >> AND)*.

El valor de No.Estado será la decisión que debe tomarse dentro de un CASE, siendo los posibles casos 1, 2 y 3; para esto se ha definido una matriz contante que contiene los números del cero al

siete en binario, que representan las ocho posibles combinaciones a obtenerse de tres bits.

Para calcular las distancias de Hamming entre 1 terna y una matriz de ternas se ha implementado la función Send Distance Matrix. Éste sub-VI se encarga básicamente de tomar de la matriz de transiciones de Nx3 y tomar cada una de sus filas.

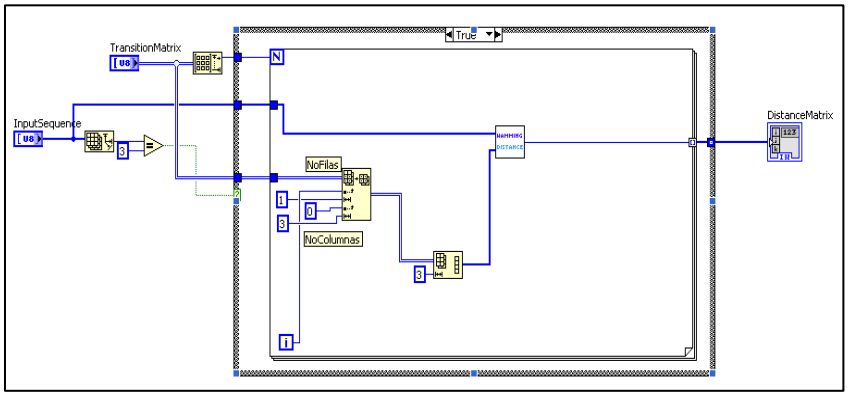


Fig. 3.49 Diagrama de bloques sub-VI "SendDistanceMatrix.vi"

Esto es necesario para poder calcular la distancia de Hamming, lo cual es realizado por el sub-VI HammingDistance.vi y la que recibe dos ternas de bits; la misma que se encuentra implementada en la presente librería. Ésta función se encarga de enviar la terna actual de bits y cada una de las ternas que forman parte de la Matriz de Transiciones. Esto se programó utilizando las funciones

IndexArray  *(Functions >> Programming >> Array >> IndexArray)* y *ReshapeArray*  *(Functions >> Programming >> Array >> ReshapeArray).*

SendHammingDistance complementa nuestro vi anterior ya que se encarga de calcular la distancia de Hamming entre dos Ternas de bits.

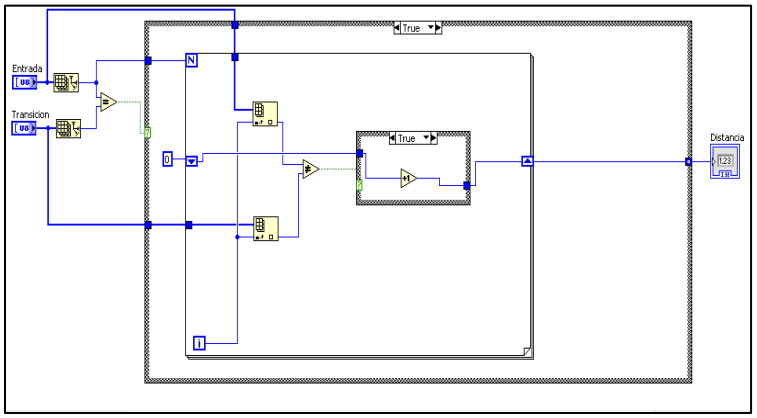


Fig. 3.50 Diagrama de bloques del sub-VI "SendHammingDistance.vi" Case=True

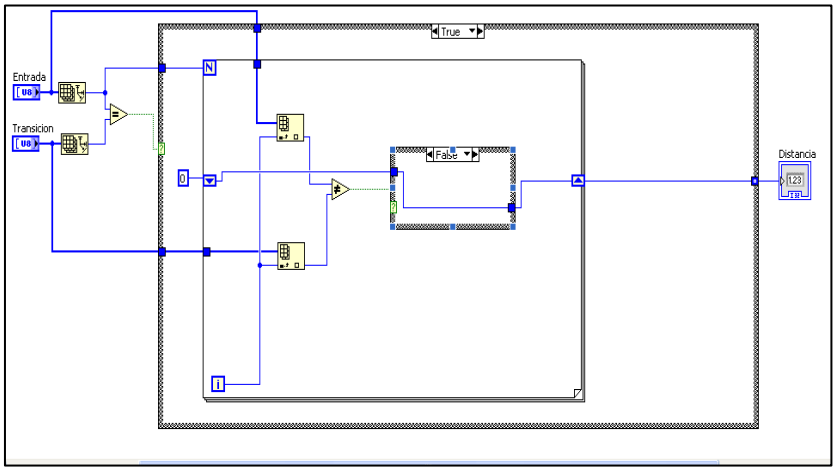


Fig. 3.51 Diagrama de bloques del sub-vi "SendHammingDistance.vi"

Case=False

Primero debe verifica que las dos secuencias sean de la misma longitud, esto se ha programado utilizando la función *ArraySize* *(Functions >> Programming >> Array >> ArraySize)* y *Equal?* *(Functions >> Programming >> Comparison >> Equal?).* Dentro de un FOR cuya condición de parada es la longitud de la secuencia de Entrada, se extraen el primer elemento de ambas Ternas de Bits y se pregunta si los dos elementos son diferentes; siempre que sean diferentes se debe sumar una unidad a la distancia acumulada. Esto se lo realiza dentro de un CASE donde la condición de verdad es la desigualdad entre los dos bits (función *NotEqual?* *(Functions >> Programming >> Comparison >>NotEqual?).* El caso TRUE agrega en una unidad la distancia

que ha sido acumulada. El caso FALSE no altera la distancia acumulada. Se utiliza un *shift register* dentro del lazo FOR inicializado en cero y que es utilizado para acumular la distancia de Hamming total.

La función Adition State 2 pretende realizar la función de sumar las métricas acumuladas hasta llegar al estado 2. Inicialmente el sub-VI verifica que la longitud de la matriz de distancias anterior sea igual a dos, lo cual se ha programado con las funciones *ArraySize*  *(Functions >> Programming >> Array >> ArraySize)* y *Quotient and Remainder*  *(Functions >> Programming >> Numeric >> Quotient and Remainder)*, de la librería de funciones Numéricas.

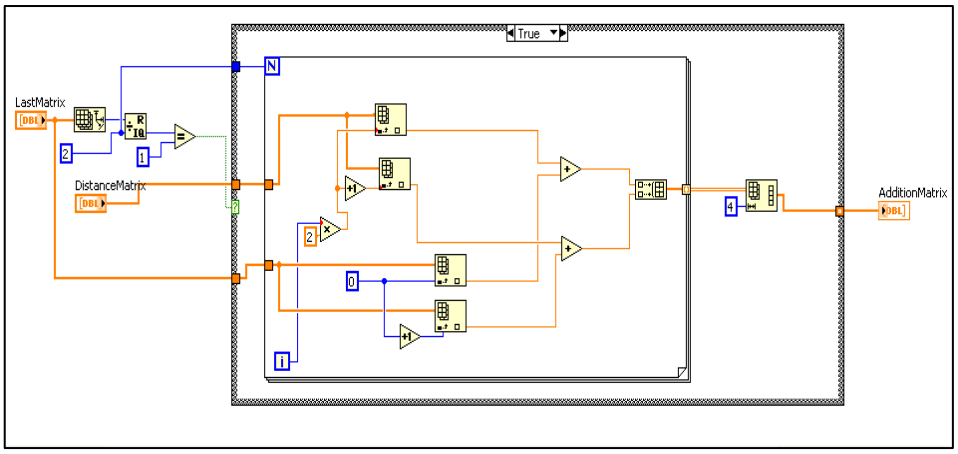


Fig. 3.52 Diagrama de bloques del sub-VI "AditionState2.vi"

A continuación el programa debe sumar las métricas provenientes del estado anterior con cada una de las etiquetas generadas del cálculo de la distancia de Hamming. Para realizar esto, se utiliza un lazo *FOR* que se repetirá dos veces. Con la función *IndexArray* se toman los elementos 0 y 1 tanto del arreglo DistanceMatrix como de LastMatrix y se suman los elementos DistanceMatrix[0] y LastMatrix[0], así también DistanceMatrix[1] y LastMatrix[1] . La siguiente vez tomará los elementos 2 y 3 del arreglo DistanceMatrix y los elementos 0 y 1 del arreglo LastMatrix y sumará los DistanceMatrix[2] y LastMatrix[0]; y DistanceMatrix[3] y LastMatrix[1]. Esto se programa con las funciones *IndexArray* y *Build Array* de la librería de arreglos *(Functions >> Programming >> Array)*y la función *Add* de la librería de funciones numéricas*(Functions >> Programming >> Numeric)*.

Estos cuatro elementos obtenidos en forma de un arreglo de dos dimensiones, se convierte a un arreglo de una dimensión, siendo ésta la salida AdditionMatrix.

La función Adition State 3 pretende realizar la función de sumar las métricas acumuladas hasta llegar al estado 3. Inicialmente el sub-VI verifica que la longitud de la matriz de distancias anterior sea

igual a cuatro, lo cual se ha programado con las funciones *ArraySize* de la librería de funciones de arreglos*(Functions >> Programming >> Array)*, *Quotient and Remainder* de la librería de funciones numéricas *(Functions >> Programming >> Numeric)* y *Equal To?* de la librería de funciones de comparación *(Functions >> Programming >> Comparison).*

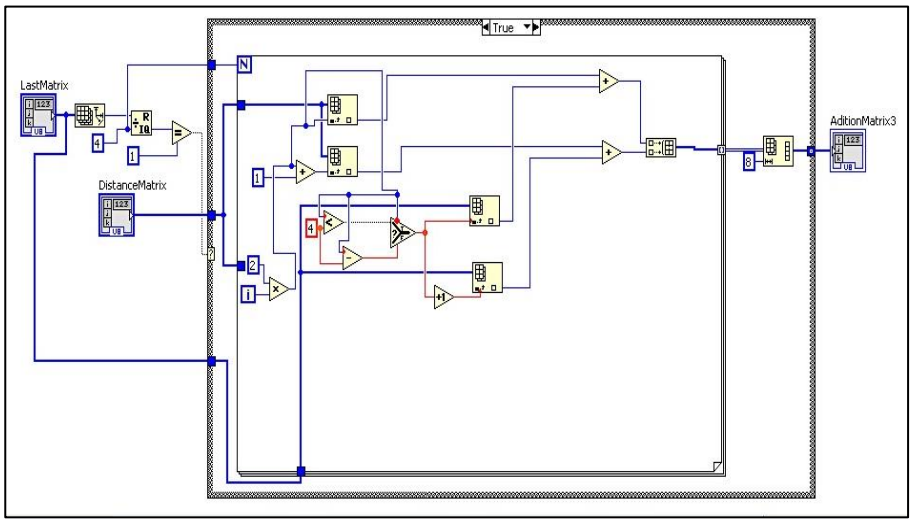


Fig. 3.53 Diagrama de bloques del sub-VI "AditionState3.vi"

A continuación el programa debe sumar las métricas provenientes del estado anterior con cada una de las etiquetas generadas del cálculo de la distancia de Hamming. Para realizar esto, se utiliza un lazo FOR que se repetirá cuatro veces.

Con la función IndexArray se toman los elementos 0 y 1 tanto del arreglo DistanceMatrix como de LastMatrix y se suman los elementos DistanceMatrix[0] y LastMatrix[0], así también DistanceMatrix[1] y LastMatrix[1] .

La siguiente vez tomará los elementos 2 y 3 tanto del arreglo DistanceMatrix como del arreglo LastMatrix y se suman los elementos DistanceMatrix[2] y LastMatrix[2], así también DistanceMatrix[3] y LastMatrix[3] .

A continuación tomará los elementos 4 y 5 del arreglo DistanceMatrix y los elementos 0 y 1 del arreglo LastMatrix y se suman los elementos DistanceMatrix[4] y LastMatrix[0], así también DistanceMatrix[5] y LastMatrix[1] .

Finalmente tomará los elementos 6 y 7 del arreglo DistanceMatrix y los elementos 2 y 3 del arreglo LastMatrix y se suman los elementos DistanceMatrix[6] y LastMatrix[2], así también DistanceMatrix[7] y LastMatrix[3] .

Esto se programa con las funciones *IndexArray*  *y Build Array* de la librería de Arreglos *(Functions >> Programming >> Array)*, las funciones *Add*, *Substract* e *Increment* de la librería de Funciones numéricas y las funciones *Less*  *(Functions >> Programming >> Comparison >> Less?)* y *Select*  *(Functions >> Programming >> Comparison >> Select)* de la librería de Comparaciones.

Estos cuatro elementos obtenidos en forma de un arreglo de dos dimensiones, se convierte a un arreglo de una dimensión, siendo ésta la salida AdditionMatrix.

La función encargada de obtener la matriz de etiquetas de cada una de las ramas salientes del estado 3 es la función State3LabelMatrix.

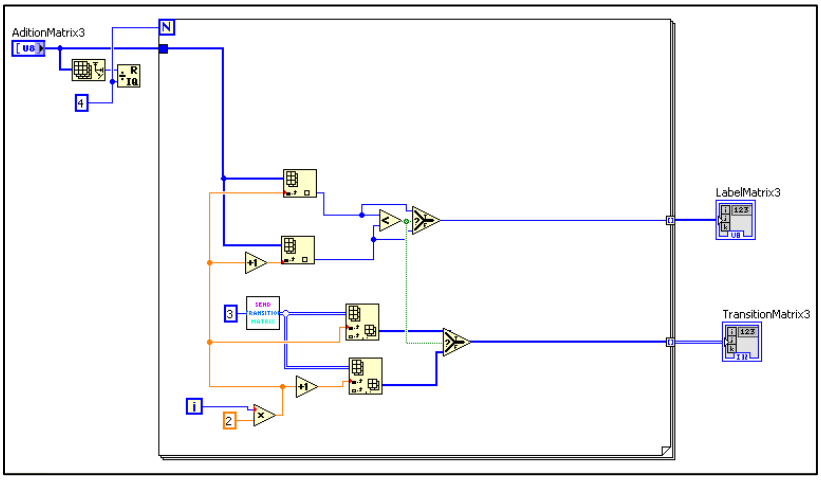


Fig. 3.54 Diagrama de bloques del sub-vi "State3LabelMatrix.vi"

Este sub-VI se encarga de escoger la distancia mínima o Métrica que llega a un nodo dentro de un determinado estado. El programa se encarga de escoger la entero menor entre cada par de elementos, es decir, compara los dos primeros elementos y extrae el valor mínimo entre los dos; de la misma forma sigue comparando tomando de dos en dos. Esto es posible gracias a las funciones *Array Size*  *e Index Array*  de la librería de funciones de Arreglos *(Functions >> Programming >> Array)*, así también con la función *Increment*  de la librería de funciones Numéricas *(Functions >> Programming >> Numeric)* así como las funciones*Less?*  *y Select*  de la librería de funciones de comparación *(Functions >> Programming >> Comparison)*. Con

cada uno de estos elementos escogidos se va formando una matriz llamada LabelMatrix3 la métrica de cada nodo.

A la vez que se escoge la Métrica se escoge su correspondiente Terna de Bits asociada y se va formando la matriz llamada TransitionMatrix3. En ésta parte además de utilizar las funciones detalladas en el párrafo anterior, se utiliza el sub-VI SendTransitionMatrix.vi implementada en ésta librería y explicada anteriormente.

Existe una función encargada de obtener la mínima métrica acumulada con su correspondiente terna asociada de cada estado la cual hemos llamado *SendMinValueMinTransition*.

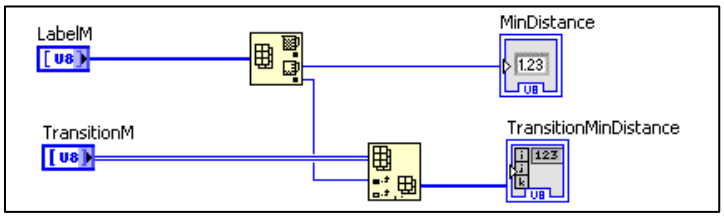


Fig. 3.55 Diagrama de bloques del sub-VI "SendMinValueMinTransition.vi"

Éste sub-VI utiliza las funciones *Array Max&Min*  *y Array Subset*  de la librería de funciones para Arreglos (*Functions* >>

Programming >> Array). La distancia mínima corresponde al valor mínimo encontrado en el arreglo LabelM y el índice es utilizado como valor de fila del arreglo TransitionM, escogiéndose como sub-arreglo su terna de bits.

Finalmente tenemos una función auxiliar encargada de dividir una secuencia de longitud N en grupos de 3 bits (siempre que N sea múltiplo de 3), DivideGroup3 se encargó de esto.

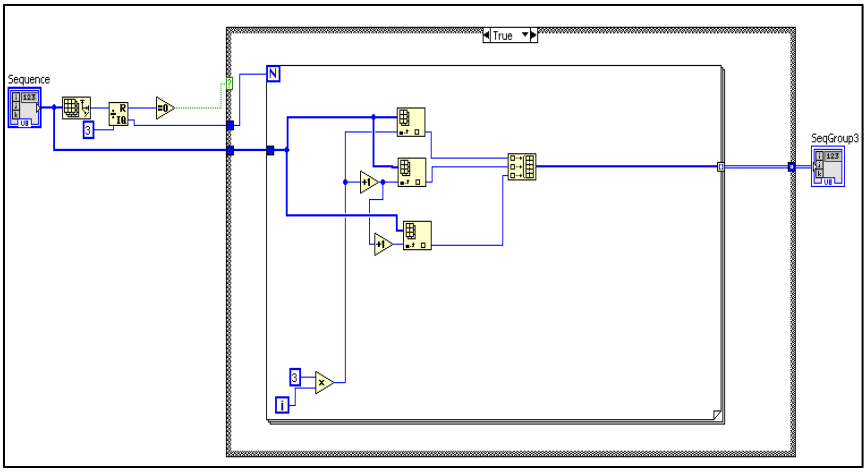


Fig. 3.56 Diagrama de Bloques del sub-VI "DivideGroup3.vi"

Este sub-VI comienza verificando que la secuencia de bits sea múltiplo de 3 caso contrario, no procesa la secuencia de entrada. A continuación se crea un lazo FOR de modo que se repitan un conjunto de instrucciones el número de veces que se pueda

formar ternas de bits. Dentro del lazo de toman inicialmente los bits 0, 1 y 2 de la secuencia grande y se forma el primer arreglo de una dimensión, la siguiente vez toma los bits 3, 4 y 5 de la misma secuencia grande y forma el segundo arreglo de una dimensión y así sucesivamente hasta completar los N arreglos que se puedan formar. En esta se utilizaron las funciones *Index Array*  *, Build Array*  *y Array Size*  de la librería de funciones para Arreglos*(Functions >> Programming >> Array)*; así también las funciones *Quotient&Remainder*  *, Increment*  *, Multiply*  de la librería de funciones numéricas *(Functions >> Programming >> Numeric)*y la función *Equalto0?*  *(Functions >> Programmaing > Comparison >> Equalto0?)*. Éste arreglo de Nx3 corresponde a la salida SeqGrupo3.

CAPÍTULO 4

PRUEBAS Y ANÁLISIS DE RESULTADOS

En ésta sección se describirán los resultados obtenidos después de haber sido implementados los programas elaborados en el capítulo 3. Nos enfocaremos en analizar la tasa de error que existe entre un sistema que utiliza corrección de error y uno que no lo hace, así como también de comparar la eficiencia que existe entre nuestros diferentes métodos de codificación y decodificación de canal.

4.1 CONFIGURACIÓN PREVIA DE LOS PANELES FRONTALES

Las imágenes, resultados y análisis que se detallan en esta sección se han hecho usando LABVIEW y equipos USRP. Para poder realizar las pruebas debemos tener en cuenta ciertos parámetros a configurar. Nuestro sistema está compuesto del transmisor (top_ofdm_tx) y receptor

(top_ofdm_rx). Se tomaran secuencias de bits todas de una misma longitud (1000 bits) siendo la potencia del ruido un parámetro a variar.

El panel frontal de top_ofdm_tx.vi indica parámetros importantes que se deben configurar, así como también graficas que permiten corroborar la información que se transmite. La figura 4.1 muestra la pestaña de configuración inicial; una de las opciones más importantes es la ip que debe asignarle a los equipos para su correcto funcionamiento. Otras características como Frecuencia de portadora, así como la ganancia en dB son factores que se puede elegir como usuario. El modo continuo permite una ejecución continua de los bits que se transmiten, teniendo en claro que con cada ejecución únicamente cambiará las secuencias.

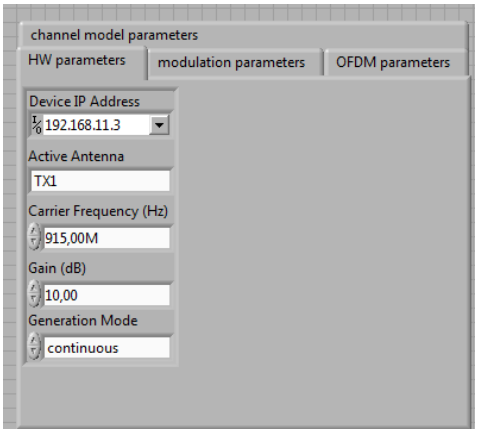


Fig. 4.1 Pestaña de configuración inicial en el transmisor.

La segunda pestaña se muestra en figura 4.2, la cual hace referencia a parámetro como tipo de modulación, factor de muestreo, longitud de la secuencia a transmitir, así como también la secuencia de entrenamiento a usar y algo muy importante que es el selector que habilita la codificación de canal. Para concluir con los parámetros de configuración del transmisor seleccionamos el tipo de canal que usaremos para simular los efectos multicamino y de ruido del medio inalámbrico, permitiéndonos manipular la potencia del ruido sobre nuestra señal así como la respuesta del canal, tal como lo indica la figura 4.3.

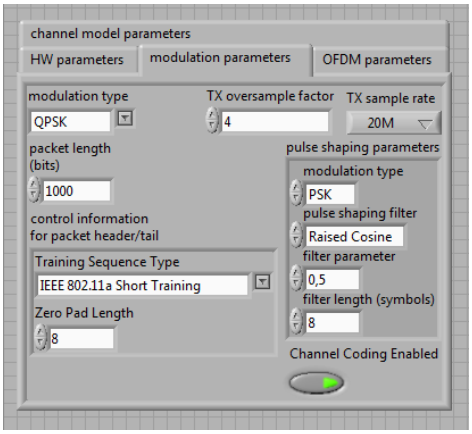


Fig. 4.2 Parámetros de modulación configurables en el transmisor.

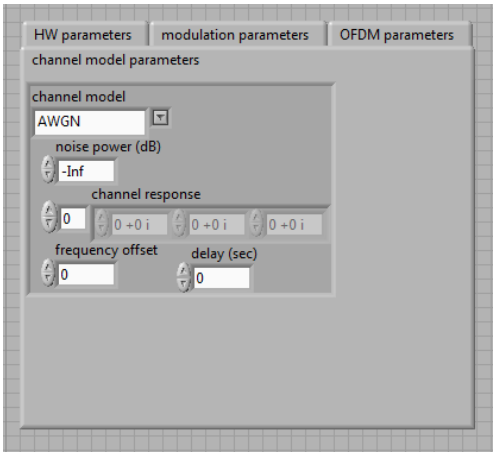


Fig. 4.3 Parámetros de los modelos de canal a usar en la transmisión.

En cuanto respecta al receptor, Top_ofdm_rx.vi, será el encargado de la obtención de los datos que viajarán por el medio así como también de proveer información importante sobre la modulación, la secuencia de entrenamiento que usa, cantidad de símbolos que arriban, diagrama de constelaciones, respuesta del canal y la medición del BER ya sea que estemos usando corrección de canal o no. En este vi los parámetros a configurar son muy pocos, dado que utiliza la recepción de los datos para obtener la información necesaria. Sin embargo un parámetro fundamental a configurar al igual que el transmisor es asignarle una ip valida y un tiempo de captura no menor al tiempo que se demoró el transmisor en generar los símbolos, la figura 4.4 contiene esta descripción.

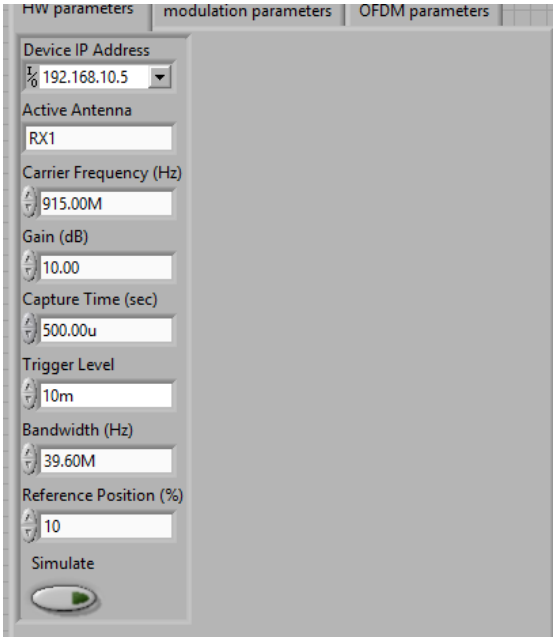


Fig. 4.4 Parámetros de configuración en el receptor.

4.2 RESPUESTA DEL CANAL ISI Y RAYLEIGH EN EL DOMINIO DEL TIEMPO Y FRECUENCIA.

Como lo indica la figura 4.2 usaremos 3 tipos de canales en nuestra experimentación, entre ellos el canal ISI y RAYLEIGH. Resulta importante mostrar las respuestas de estos canales en el dominio del tiempo y de la frecuencia. La figura 4.5 muestra las componentes del canal ISI; se ha seleccionado 3 pasos de características complejas, los cuales determinan la amplitud del pulso con una separación constante entre sí, como se observa en la figura 4.6.

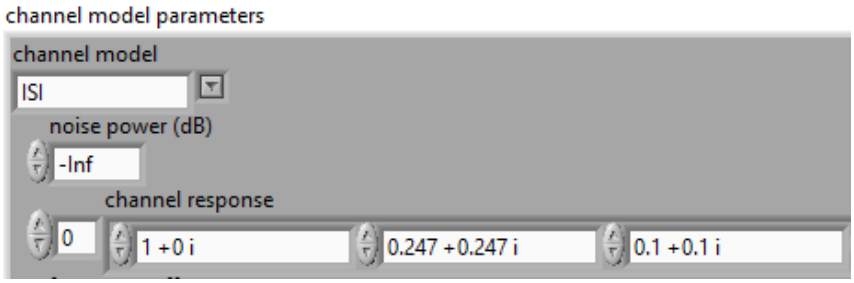


Fig. 4.5 Componentes complejas para un canal ISI

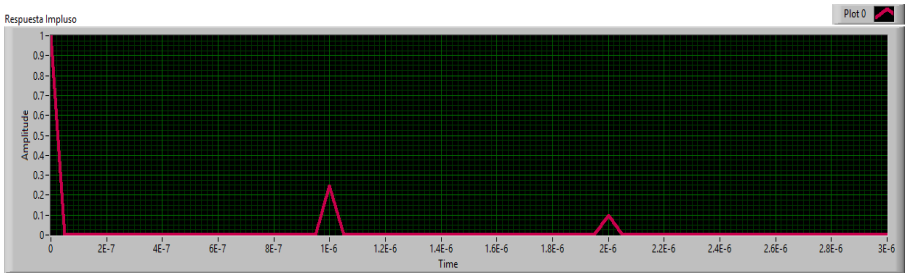


Fig. 4.6 Respuesta en el dominio del tiempo para un canal ISI

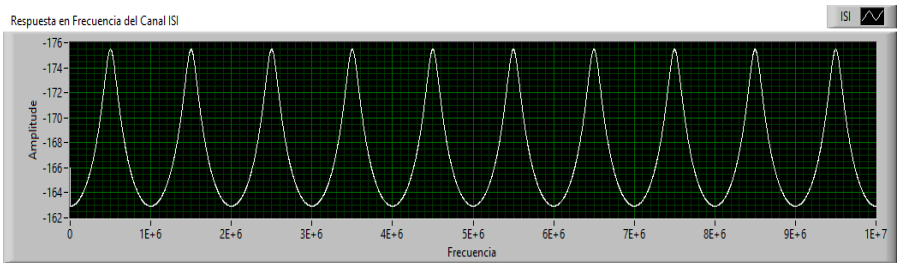


Fig. 4.7 Respuesta en el dominio de la frecuencia para un canal ISI

Se puede observar que, a pesar que las amplitudes de los retardos varía de acuerdo con la respuesta del canal configurado, la separación en el dominio del tiempo es constante y en éste caso igual a 1 ms. La respuesta en el dominio de la frecuencia muestra que las frecuencias

son afectadas siguiendo un patrón, el cual produce atenuaciones cada 1 ms y amplificaciones la frecuencia intermedia.

La figura 4.8 muestra la respuesta en el dominio del tiempo de un canal RAYLEIGH; se puede observar los 4 pulsos cuya amplitud y distancia siguen el estándar ITU-P1225. La respuesta en el dominio de la frecuencia se ilustra en las figuras 4.9 y 4.10 para diferentes ventanas de observación.

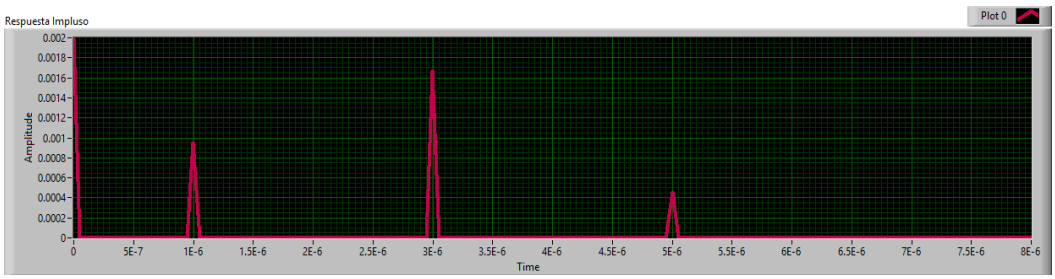


Fig. 4.8 Respuesta en el dominio del tiempo de un canal Rayleigh según el estándar ITU- P1225

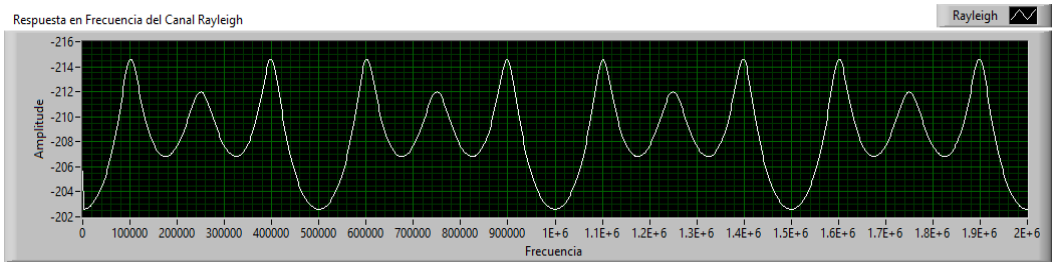


Fig. 4.9 Respuesta en el dominio de la frecuencia de un canal Rayleigh según el estaándar ITU P1225 para una ventana de observación de 2 MHz

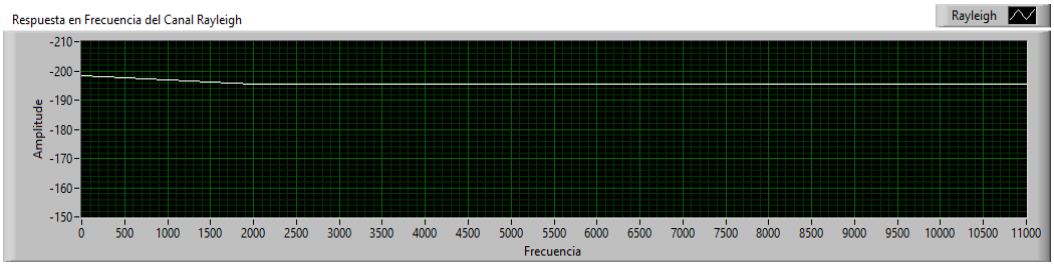


Fig. 4.10 Respuesta en el dominio de la frecuencia de un canal Rayleigh según el estaándar ITU P1225 para una ventana de observación de 11 KHz

Se puede observar en la figura 4.9 que las 64 subportadoras de nuestra señal OFDM, con ancho de banda de 2.5 MHz, está siendo afectada por un canal con desvanecimiento selectivo en frecuencia, ya que las frecuencias no son afectadas en la misma forma. Si tomamos el intervalo entre cero y 500 KHz podemos observar que las frecuencias 150 KHz y 350 KHz se observan desvanecimientos profundos mientras en que las frecuencias 100 KHz, 250 KHz y 400 KHz se observan amplificaciones en la respuesta del canal. Si tomamos el intervalo de frecuencia entre cero y 11 KHz, se observa que las señales cuyo ancho de banda se encuentre en éste rango serán afectadas por un canal con desvanecimiento plano, esto es, que todas las componentes de frecuencia se afectan de manera similar, o bien se atenúan o se amplifican todas a la vez. Se puede demostrar que una subportadora está siendo afectada por la respuesta del canal con desvanecimiento plano.

4.3 COMPORTAMIENTO DEL SISTEMA SIN CODIFICACIÓN DE CANAL

Un sistema de comunicaciones sin codificación de canal normalmente tiene una probabilidad de errores mayor frente a uno sobre el cual han sido implementados esquemas de codificación para control de errores. En esta sección observaremos el comportamiento de un sistema sobre el cual no ha sido implementado un esquema de codificación de canal considerando la tasa de error obtenida en el receptor.

La figura 4.11 nos muestra una constelación de símbolos QPSK recibida simulando una canal AWGN con una potencia de ruido de -10 decibelios. Se puede observar que los símbolos recibidos se encuentran dispersos alrededor de los valores complejos propios de la constelación QPSK, razón por la cual el receptor podría decidirse a favor del símbolo incorrecto incrementando de esta manera la tasa de errores.

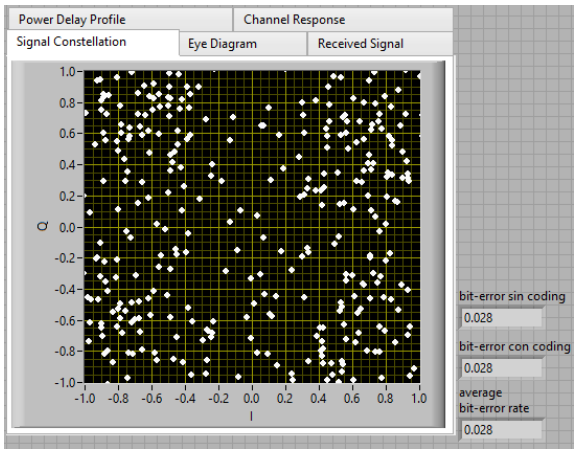


Fig. 4.11 Comportamiento del sistema en un canal AWGN sin codificación de canal

En la figura 4.12 podemos observar la constelación de símbolos QPSK recibidos emulando un canal ISI con una potencia de ruido de -10 decibelios. Se puede observar que la tasa de errores no varía significativamente respecto al canal AWGN.

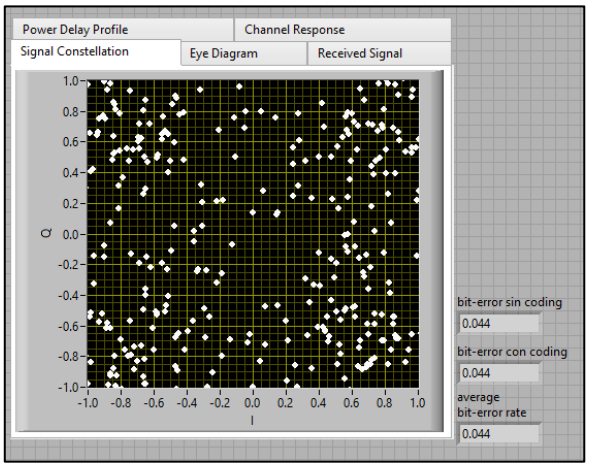


Fig. 4.12 Comportamiento del sistema en un canal ISI sin codificación de canal

La figura 4.13 nos muestra una constelación de símbolos QPSK recibida emulando una canal Rayleigh con una potencia de ruido de -10 decibelios. Se puede observar que la tasa de errores se ha incrementado 10 veces en comparación con el canal AWGN y que la constelación se ha deformado.

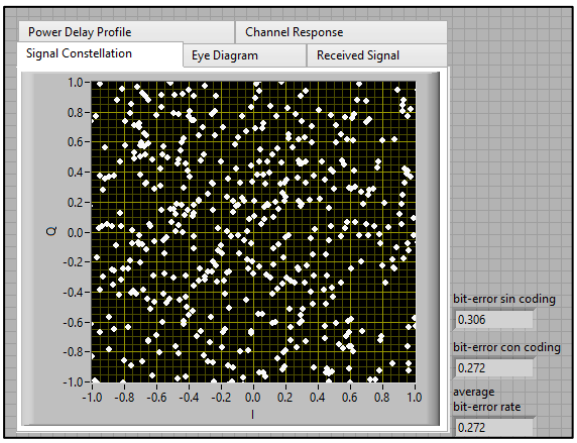


Fig. 4.13 Comportamiento del sistema en un canal Rayleigh sin codificación de canal

Un análisis más profundo se puede observar en las figuras 4.14 y 4.15, las cuales contienen información sobre tasas de errores para diferentes valores de SNR para un canal AWGN y Rayleigh respectivamente.

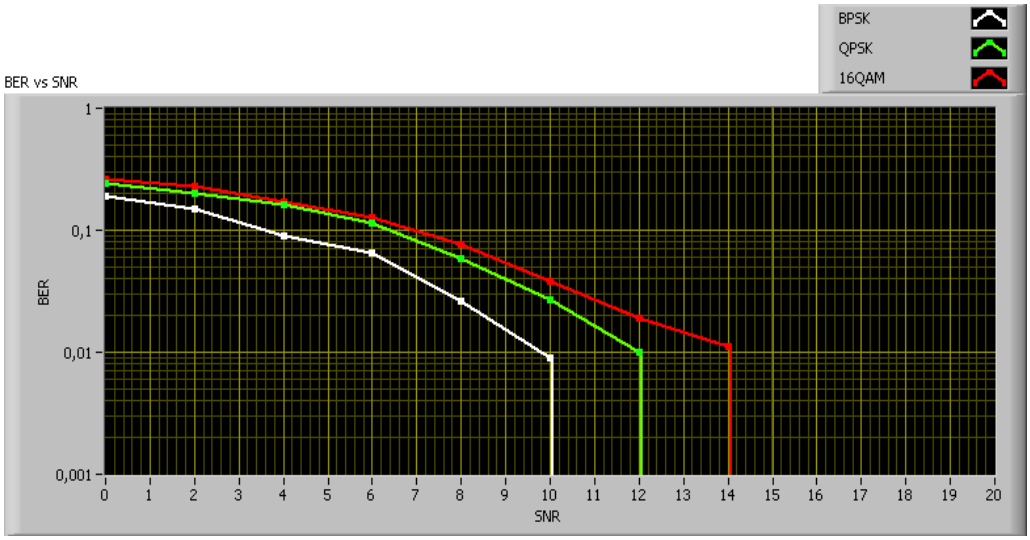


Fig. 4.14 Curvas de BER vs SNR que ilustra el comportamiento del sistema en un canal AWGN sin el uso de codificación de canal.

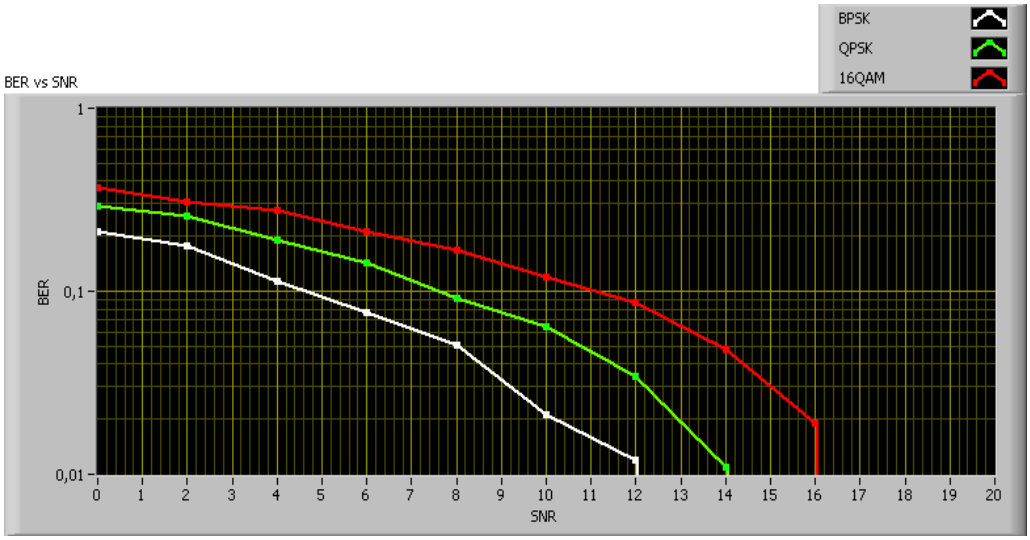


Fig. 4.15 Curvas de BER vs SNR que ilustra el comportamiento del sistema en un canal Rayleigh sin el uso de codificación de canal.

4.4 COMPORTAMIENTO DEL SISTEMA UTILIZANDO CODIFICACIÓN DE BLOQUES LINEALES SOBRE UN CANAL AWGN

Se selecciona la cantidad de bits que se desean transmitir, en nuestro caso 1000 bits, así como el tipo de modulación tal cual la figura 4.16. A la izquierda se observa los parámetros básicos de configuración y a su derecha el diagrama de constelación que nos indica que estamos usando QPSK.

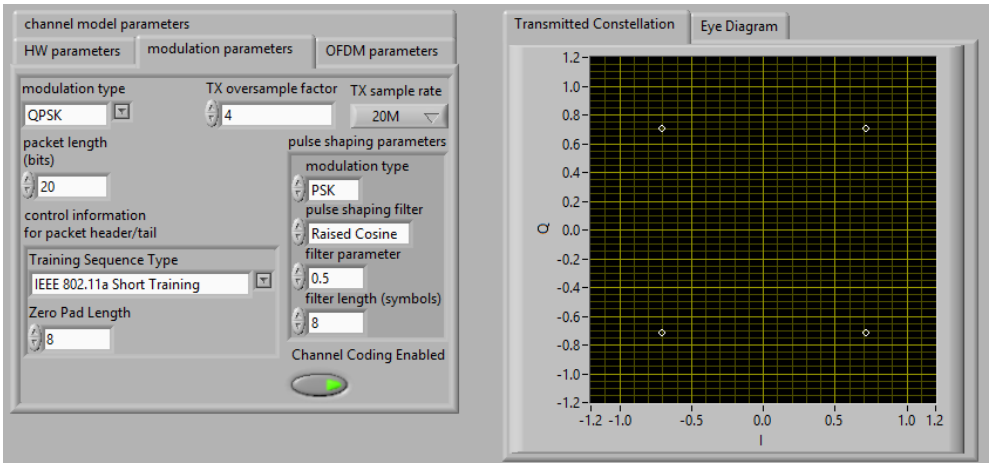


Fig. 4.16 Panel Frontal transmitiendo 1000 bits usando QPSK

En la pestaña del canal simularemos un AWGN con un valor de ruido igual a -Inf, esto será el caso ideal en el que no existen errores durante la transmisión, en la figura 4.17 se ilustra un valor de ruido muy pequeño el cual no provocará un efecto significativo sobre nuestra transmisión.

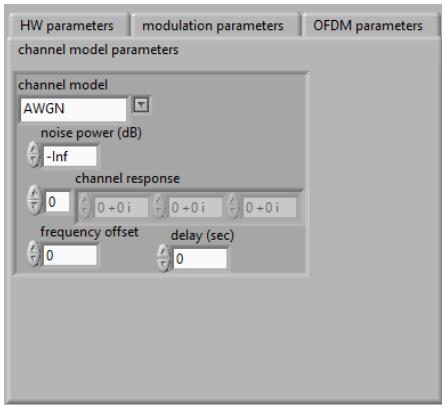


Fig. 4.17 Configuración de los parámetros del canal.

Como se espera, la secuencia fue recibida sin errores; como muestran los valores bit_error_sin_coding y bit_error_con_coding. Al receptor han llegado todos los símbolos y el diagrama de constelación se mantiene bien identificado en su respectivo cuadrante y con una tasa de error de 0%.

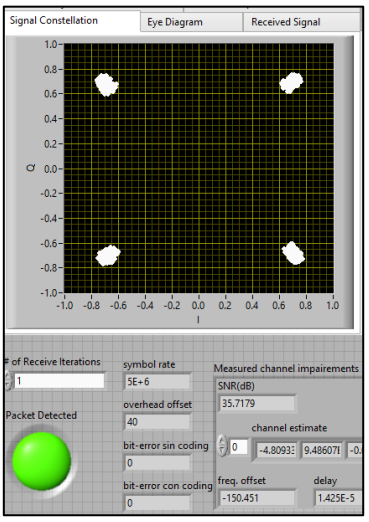


Fig. 4.18 Constelación recibida sin errores usando códigos de bloques lineales sobre AWGN.

Durante la experimentación la potencia del ruido se ha variado desde -20dB hasta 0dB. La figura 4.19 muestra la recepción del mensaje cuando ha pasado por un medio que experimenta un ruido de -20dB; como se puede observar, frente a la presencia de ruido el BER sigue siendo cero; esto se debe fundamentalmente a que la potencia del ruido aún no implica significancia frente a la potencia de la señal.

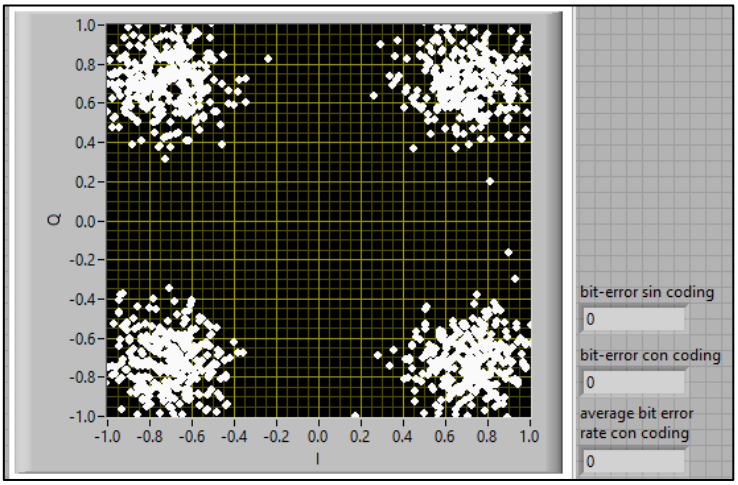


Fig. 4.19 Constelación recibida usando bloques lineales con un valor de ruido pequeño.

Para probar la efectividad de los algoritmos, aumentamos la potencia del ruido a fin de conseguir errores en la transmisión. Probamos con un valor de -8dB como potencia del ruido. Como nos podemos dar cuenta en la figura 4.20 la constelación recibida se ve afectada por este factor y se muestra un poco dispersa, sin embargo aún los símbolos llegan sobre un cuadrante definido. En este punto hemos logrado que en la recepción exista un 5.6% de error bit-error-sin-coding, el cual satisfactoriamente ha sido reducido a 2.3% como lo muestra bit-error-con-coding.

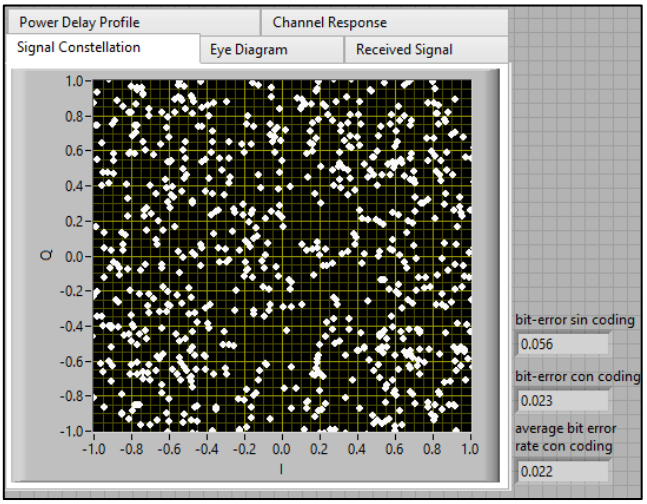


Fig. 4.20 Constelación QPSK recibida en un canal AWGN con -12dB

Como lo hemos advertido, cuando la potencia del ruido alcanza valores de -6dB (para la misma cantidad de bits) el BER se incrementa lo cual se observa en la figura 4.21. Nuestro codificador de bloques lineal aún puede resolver la mitad de los bits erróneos.

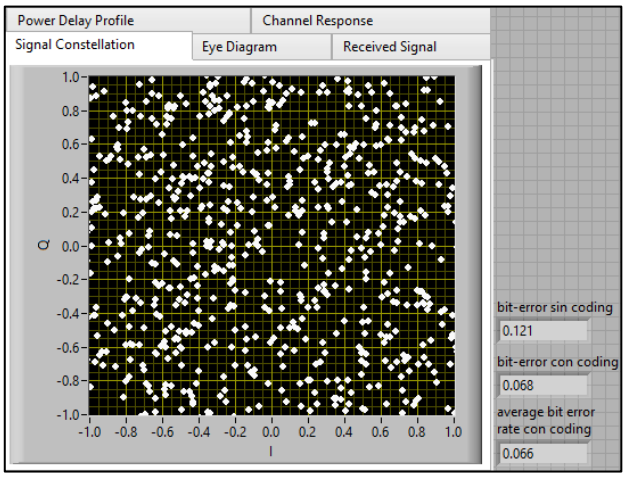


Fig. 4.21 Recepción del mensaje usando -10db de ruido en un canal AWGN.

Para valores en donde la potencia del ruido ha aumentado a -2dB, podemos observar que la constelación ha perdido su forma y que la tasa de errores ha aumentado a un valor del 20.2%, provocando que el algoritmo pierda efectividad y que cada vez sea más difícil corregir los errores, la figura 4.22 nos indica estos resultados.

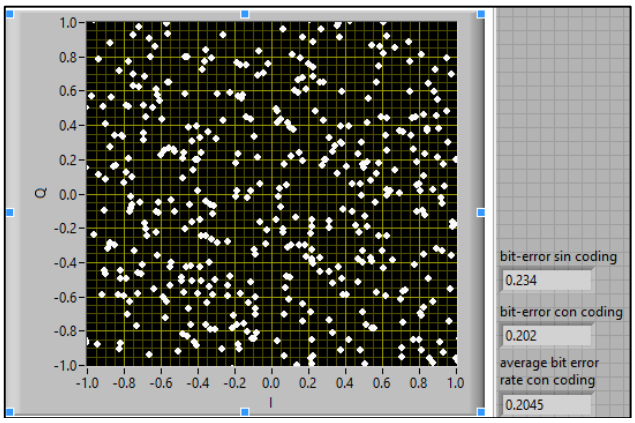


Fig. 4.22 Constelación QPSK recibida sobre AWGN a -6dB

Para valores de ruido en donde la potencia es mayor o igual a la potencia de la señal nos damos cuenta que la constelación se ha perdido por completo en la recepción. Las tasas de errores han ascendido a 32.1%; a estos niveles el efecto del codificador de canal es casi nulo tal como lo ilustra la figura 4.23. Esto se debe a que la potencia del ruido es grande y produce principalmente ráfaga de errores, ante lo cual es sistema es vulnerable.

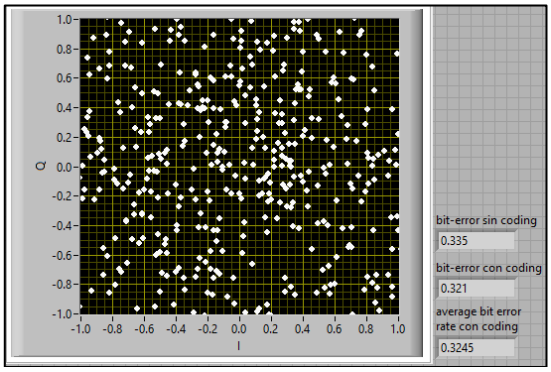


Fig. 4.23 Constelación perturbada por un canal muy ruidoso

Tanto la cantidad de bits transmitidos como el ruido que se introduce a lo largo del canal inciden directamente sobre la tasa de errores obtenida en el receptor; sin embargo también se debe tomar en cuenta el tipo de modulación.

El tipo de modulación es un factor importante al realizar comparaciones acerca del BER, mientras la modulación tenga mayor cantidad de regiones de decisión, la probabilidad de error aumenta y por ende el decodificador es más propenso a equivocarse y cometer errores.

Por tal motivo si utilizamos una modulación que mapee menos símbolos tendremos un mejor rendimiento con respecto al BER. Lo mencionado se ilustra en las figuras 4.20 y 4.24, en donde se utiliza modulación QPSK y BPSK respectivamente. Se puede observar que la tasa de errores a una

potencia del ruido de -8dB se mantiene en 0% para BPSK mientras que utilizando modulación QPSK el BER fue de 2.3%. Esto es debido a que los 2 únicos símbolos tienen una distancia considerable dentro de la constelación haciendo menos probable la interferencia entre sí.

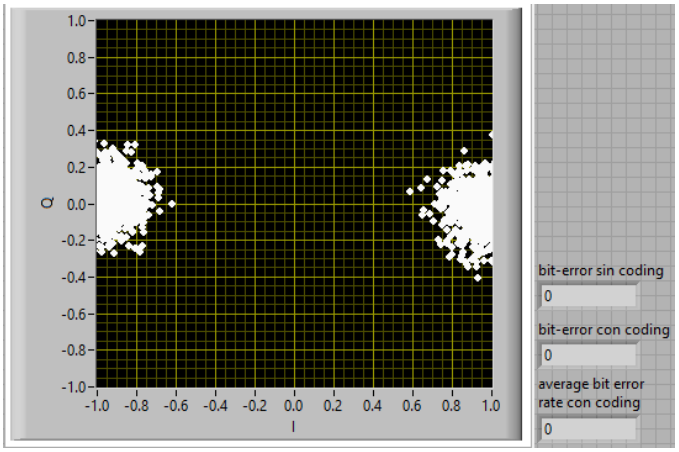


Fig. 4.24 Recepción usando BPSK

En las figuras 4.22 y 4.25 se ilustran los resultados obtenidos con una potencia de ruido de -2db para una constelación QPSK y BPSK, respetivamente. Se observa que al utilizar la modulación QPSK la tasa de errores es 5% mayor que modulando los bits en BPSK.

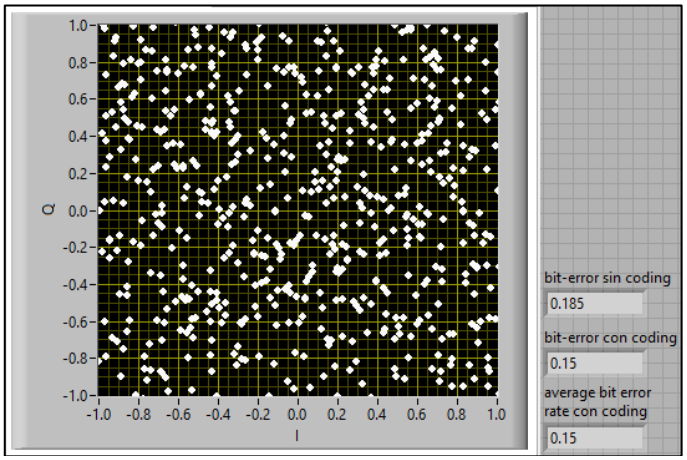


Fig. 4.25 Constelación obtenida con AWGN -6dB QPSK

Ante lo mencionado, esperamos que si se utiliza una modulación que demande un mapeo mayor, los errores aparecerán aun cuando los niveles de ruido sean bajos. En efecto este fenómeno ocurre debido a que mientras mayor número de símbolos existan, la distancia que existe entre ellos será menor, tal como se puede observar en el diagrama de constelación. Una menor distancia implica que existe mayor posibilidad de interferencia inter-simbólica y esto conlleva a tasa de errores más alta. Los diagramas de constelación que se muestran a continuación reflejan claramente éste efecto.

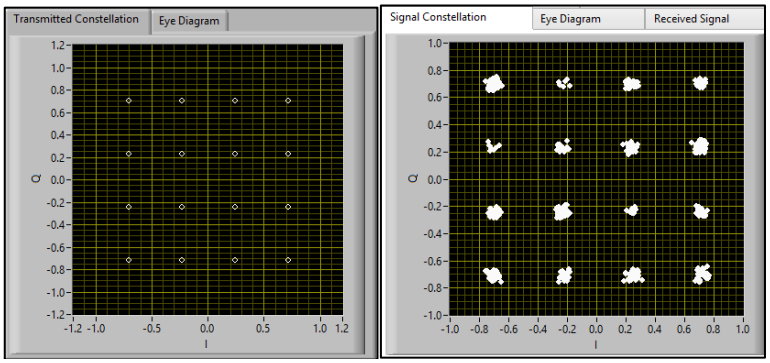


Fig. 4.26Constelación 16QAM sobre un canal AWGN

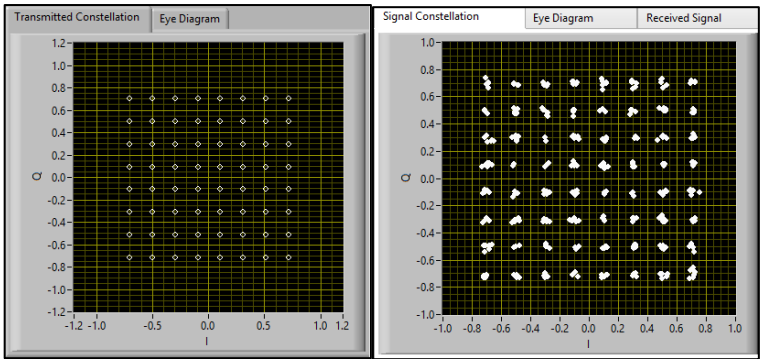


Fig. 4.27 Constelación 64QAM sobre un canal AWGN

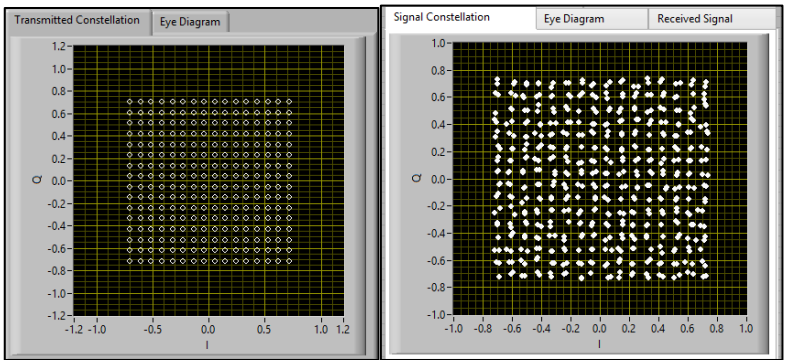


Fig. 4.28 Constelación 256QAM sobre un canal AWGN

El análisis realizado hasta ahora puede ser extendido a una modulación superior. La figura 4.29 y la figura 4.30 nos muestran la constelación usando 16QAM y una potencia de ruido de -8dB y -2dB respectivamente. Como se puede observar la tasa de errores es superior al valor obtenido para las dos modulaciones anteriores.

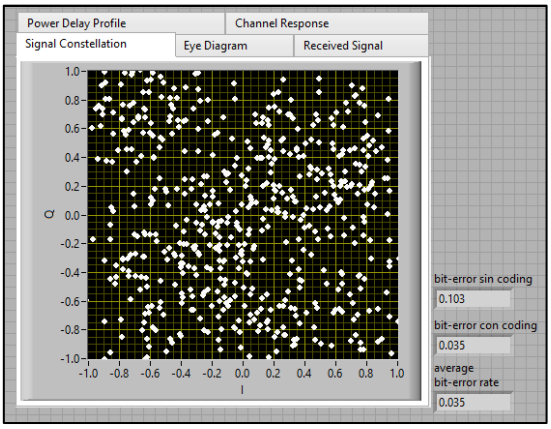


Fig. 4.29 Constelación de la modulación 16QAM obstruida por una potencia de -8db de ruido

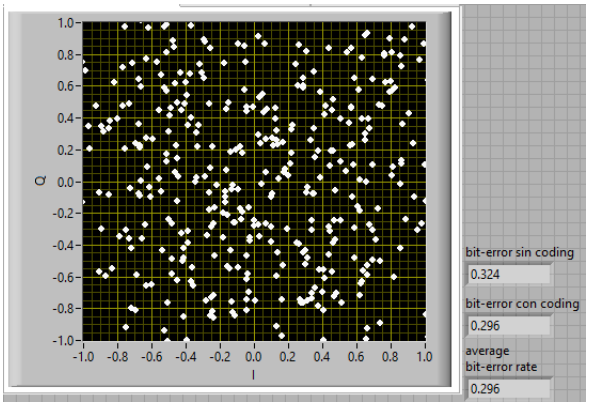


Fig. 4.30 Constelación 16 QAM distorsionada por un canal AWGN ruidoso -2dB de ruido

Luego de obtener las tasas de error para diferentes niveles de ruido en las modulaciones BPSK, QPSK y 16QAM se generaron las curvas BER vs SNR en simulación y considerando datos reales, lo cual se muestra en la figuras 4.31. Las curvas describen al detalle las tasas de errores para diferentes valores de SNR. Como se puede apreciar la modulación BPSK experimenta los niveles de BER más bajos. Se puede observar además como los datos obtenidos de manera simulada se asemejan a los datos reales.

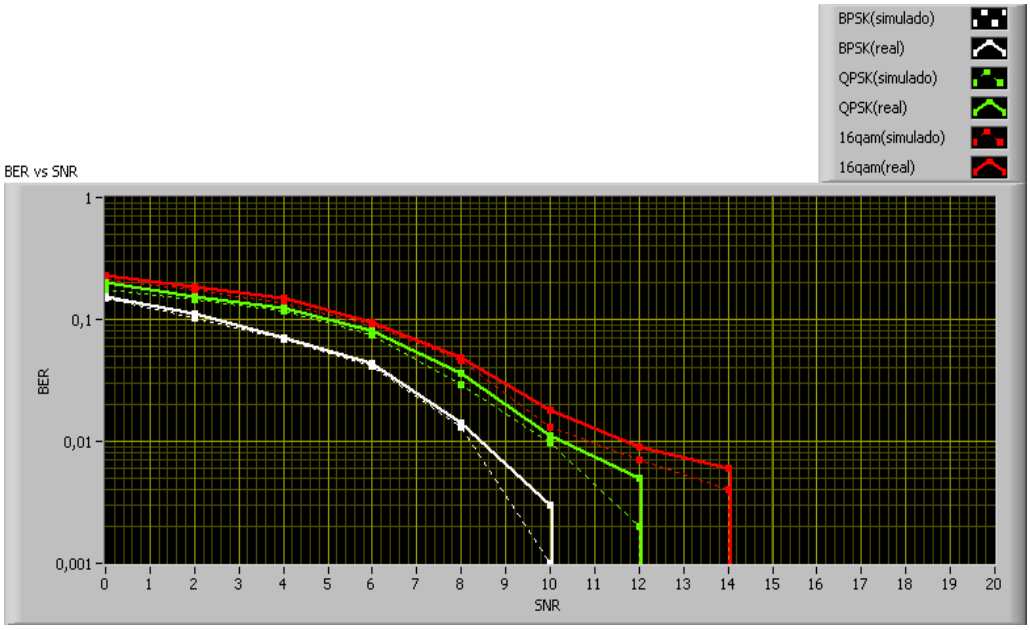


Fig. 4.31 Curva BER vs SNR para canal AWGN, simulado y real, con codificación de bloque lineal.

4.5 COMPORTAMIENTO DEL SISTEMA USANDO CÓDIGOS CONVOLUCIONALES SOBRE UN CANAL AWGN.

Los códigos convolucionales por sus propiedades estudiadas en el capítulo 2, ofrecen mayor robustez frente a la detección y corrección de errores. Para la experimentación tomamos como base las mismas condiciones de los códigos lineales con el objetivo de poder hacer comparaciones más adelante. La figura 4.32 muestra la constelación recibida que se obtuvo de los 1000 bits luego de pasar por un canal ruidoso de potencia -8dB; la tasa error es de apenas 1.4%, haciendo énfasis en la efectividad de la codificación.

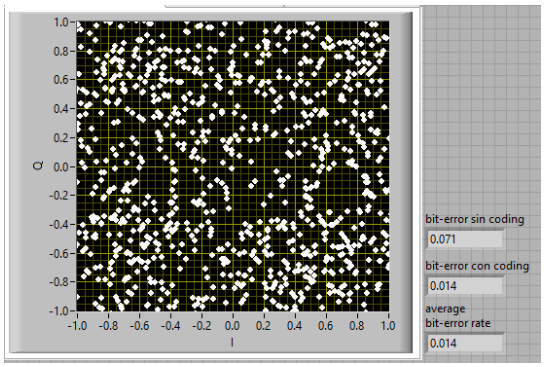


Fig. 4.32 Constelación QPSK que ha pasado por un canal ruidoso utilizando códigos convolucionales.

Como ya se demostró, el tipo de modulación será un factor determinante en el BER. Si comparamos la figura 4.33 y 4.34 se observan que las tasas de error obtenidas en la experimentación son diferentes a pesar

que se ha utilizado la misma cantidad de bits y el mismo nivel de ruido. Es evidente que la constelación que tiene más regiones de decisión tiene más bits erróneos; incluso en la constelación BPSK a este nivel de ruido aún no experimenta errores, lo cual no ocurre en QPSK y 16QAM.

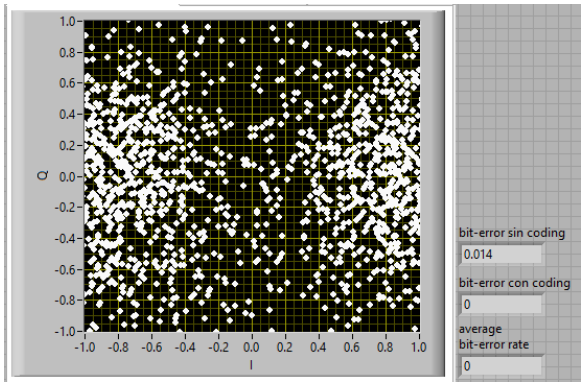


Fig. 4.33 Constelación BPSK usando códigos convolucionales.

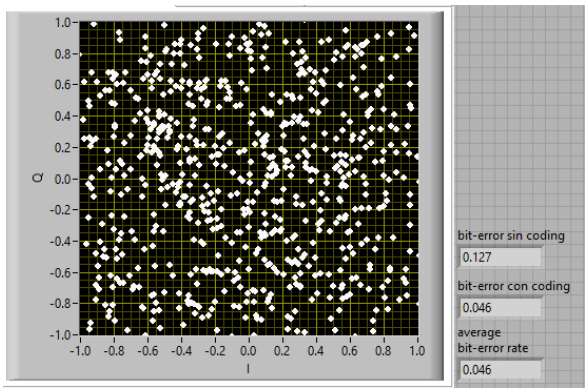


Fig. 4.34 Constelación 16QAM usando códigos convolucionales.

Luego de obtener las tasas de error para diferentes niveles de ruido en las modulaciones BPSK, QPSK y 16QAM se generaron las curvas BER

vs SNR en simulación y considerando datos reales, lo cual se muestra en las figura 4.35. Si comparamos estas curvas con las de la sección 4.3, podemos observar que usando códigos convolucionales las tasas de errores son más bajas que usando los códigos de bloques lineales. Los errores para BPSK tomando datos reales aparecen para un valor de SNR de 6 dB mientras que para codificación de bloques lineal surgieron a partir de 10 dB.

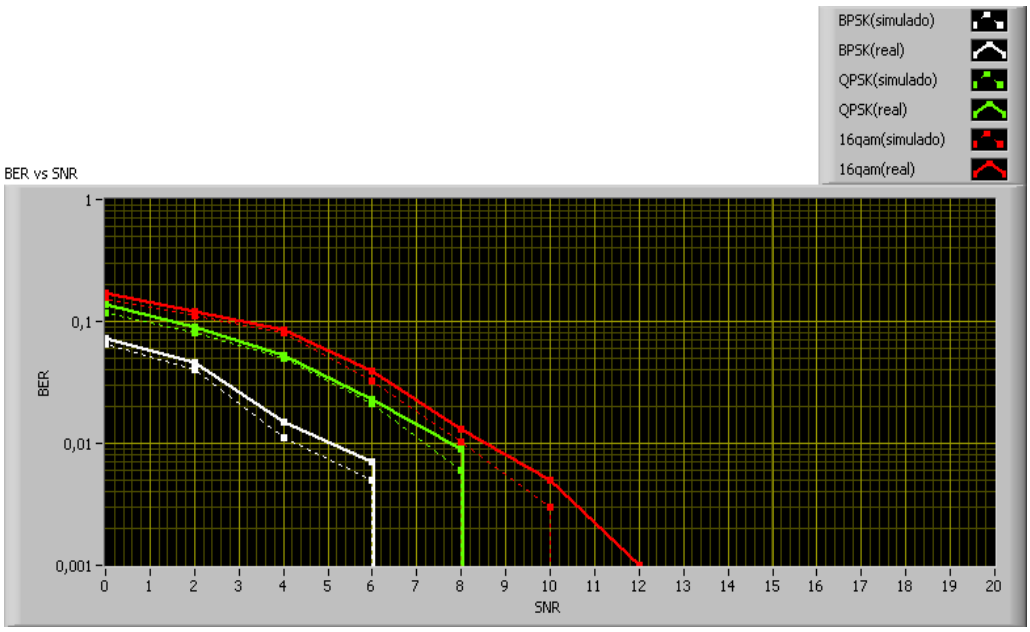


Fig. 4.35 Curva BER vs SNR para canal AWGN, simulado y real, con codificación convolucional.

4.6 COMPORTAMIENTO DEL SISTEMA FRENTE A UN CANAL ISI USANDO BLOQUES LINEALES.

Un sistema OFDM no es afectado de manera significativo por un canal de interferencia intersimbólica ISI debido a sus propiedades con las cuales fue diseñado y que se estudiaron en el capítulo 2. Los retardos y las atenuaciones de potencia que provoca el canal originarían que en el experimentación de un canal con única portadora las tasas de errores se incrementen considerablemente, sin embargo dado la naturaleza y robustez que ofrece OFDM frente a estos problemas nos daremos cuenta que el error introducido por el canal es pequeño. La figura 4.36, 4.37 y 4.38 nos muestran la recepción de las constelaciones BPSK, QPSK y 16QAM respectivamente, tomando una secuencia de 1000 bits expuesta a un ruido de -8dB. En los 3 casos las tasas de errores son bajas y se asemejan a las que obtuvimos con el AWGN, afirmando lo mencionado al inicio de esta sección.

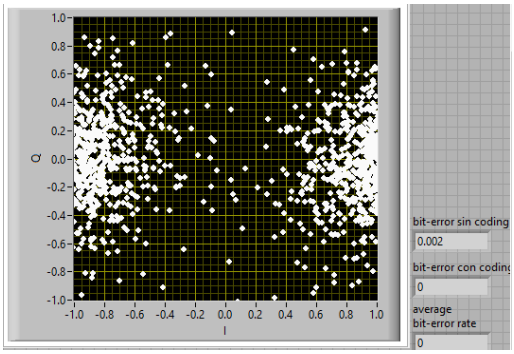


Fig. 4.36 Constelación BPSK al pasar por un canal ISI en un sistema con codificación lineal

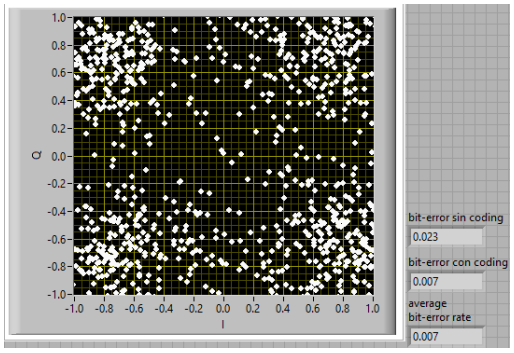


Fig. 4.37 Constelación QPSK al pasar por un canal ISI en un sistema con codificación lineal

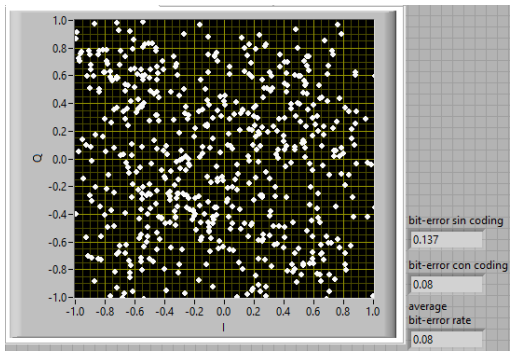


Fig. 4.38 Constelación 16QAM al pasar por un canal ISI en un sistema con codificación lineal

Luego de obtener las tasas de error para diferentes niveles de ruido en las modulaciones BPSK, QPSK y 16QAM se generaron las curvas BER vs SNR en simulación y considerando datos reales, lo cual se muestra en la figura 4.39. A pesar que se está emulando un canal ISI, el porcentaje de bits erróneos que se obtienen en el receptor son muy parecidos a los del canal AWGN, esto se debe a la robustez que ofrece OFDM frente a la interferencia intersimbólica. BPSK mantiene las tasas de errores más bajas.

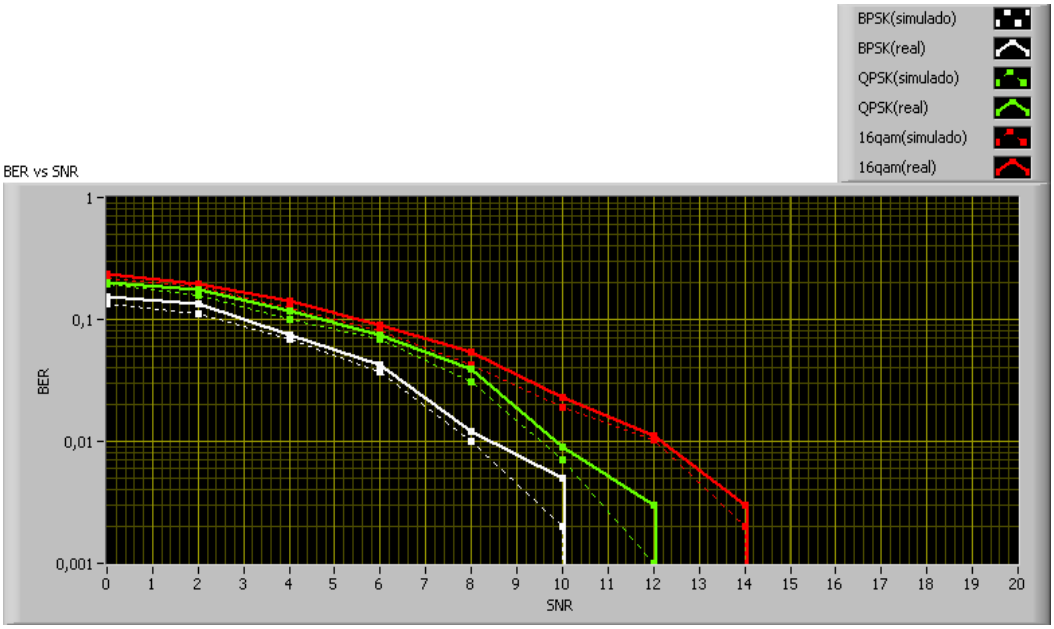


Fig. 4.39 Curva BER vs SNR para canal ISI, simulado y real, con codificación de bloque lineal.

4.7 COMPORTAMIENTO DEL SISTEMA FRENTE A UN CANAL ISI USANDO CÓDIGOS CONVOLUCIONALES.

Al igual que para los códigos de bloque lineal se han realizado experimentaciones para nuestro segundo codificador de canal; los resultados obtenidos se pueden ver en la figura 4.40, 4.41 y 4.42. Como es de esperarse los valores en cuanto al BER difieren un poco si los comparamos con los obtenidos en la sección 4.4 para cada una de las modulaciones. Para BPSK los valores son iguales a 0%, mientras que para QPSK y 16QAM los valores han disminuido.

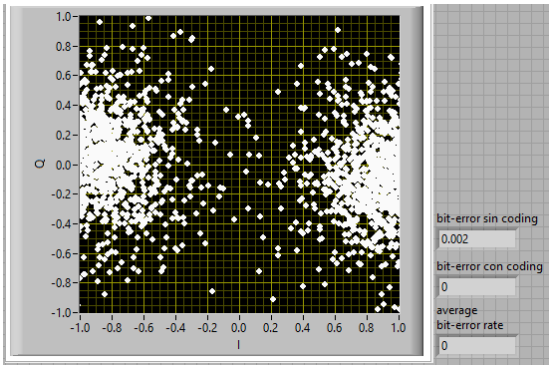


Fig. 4.40 Constelación 16QAM al pasar por un canal ISI en un sistema con codificación convolucional

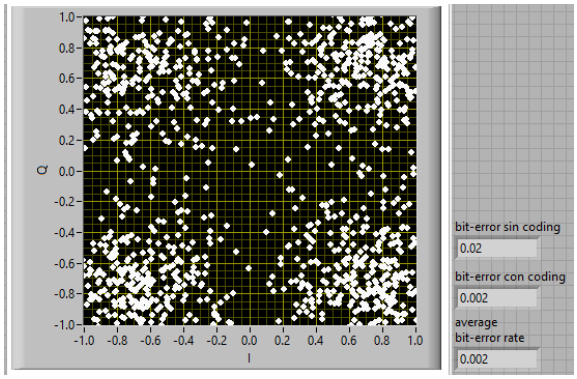


Fig. 4.41 Constelación 16QAM al pasar por un canal ISI en un sistema con codificación convolucional

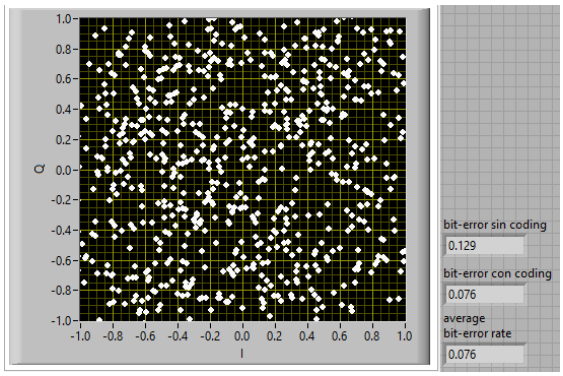


Fig. 4.42 Constelación 16QAM al pasar por un canal ISI en un sistema con codificación convolucional

Luego de obtener las tasas de error para diferentes niveles de ruido en las modulaciones BPSK, QPSK y 16QAM se generaron las curvas BER vs SNR en simulación y considerando datos reales, lo cual se muestra en la figuras 4.43. Si comparamos estas gráficas con las analizadas en la sección 4.5 nos damos cuenta que los resultados en relación al BER vs

SNR son más eficientes cuando se usan codificadores convolucionales tal como sucedió en el canal AWGN.

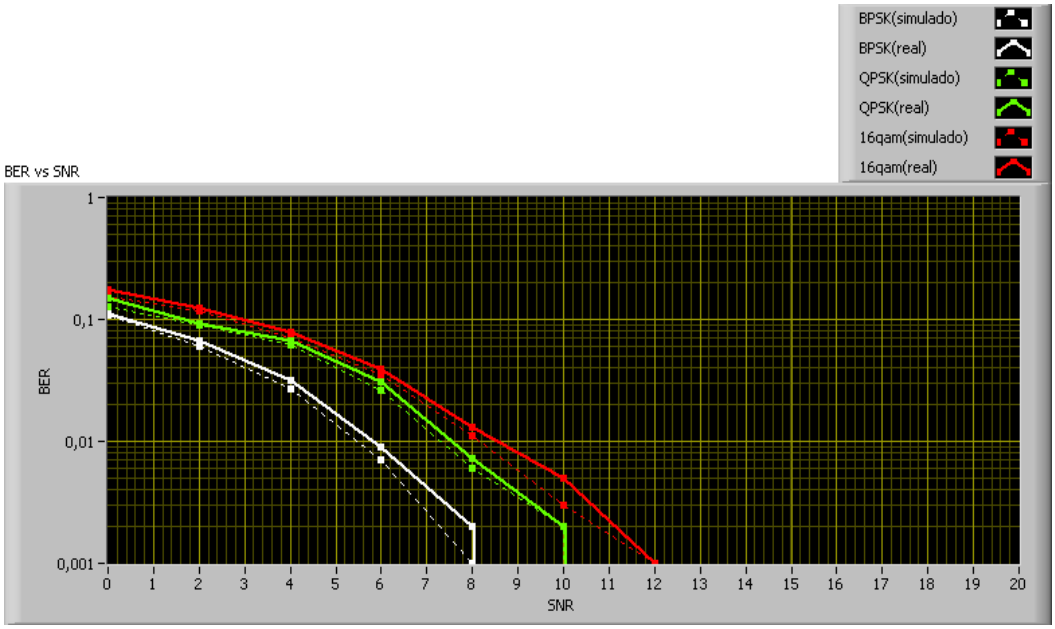


Fig. 4.43 Curvas BER vs SNR para canal ISI, simulado y real, con codificación convolucional.

4.8 COMPORTAMIENTO DEL SISTEMA FRENTE A UN CANAL RAYLEIGH USANDO BLOQUES LINEALES.

En esta sección analizaremos el comportamiento de nuestro sistema al usar un canal destructivo como lo es RAYLEIGH. El multitrayecto y la pérdida de línea de vista son factores que afectan de manera muy significativa a nuestra señal. La Fig. 4.44, Fig. 4.45 y Fig. 4.46 muestran las constelaciones recibidas para cada una de las modulaciones

analizadas. Como podemos observar, éste canal destruye la constelación a niveles de ruido bajos. La experimentación se ha mantenido con una trama de 1000 bits y una potencia de ruido de -8dB sin embargo a este nivel la tasa de error ya se encuentra por sobre el 12% algo que no se observaba en los canales AWGN e ISI. Se confirma una vez más el efecto de la modulación sobre la tasa de error; en 16 QAM el BER es el más alto en todos los casos.

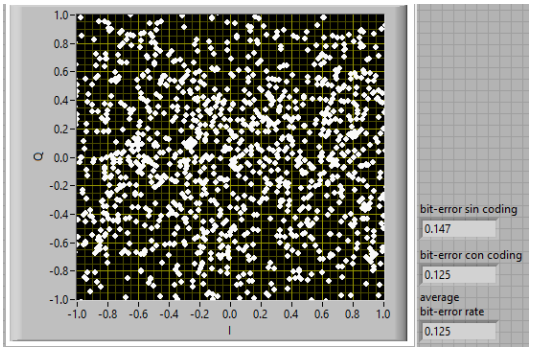


Fig. 4.44 Constelación BPSK al pasar por un canal RAYLEIGH en un sistema con codificación lineal

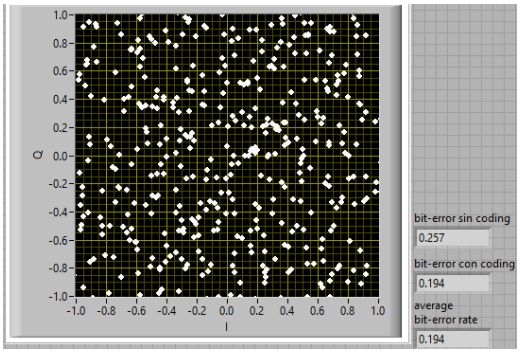


Fig. 4.45 Constelación QPSK al pasar por un canal RAYLEIGH en un sistema con codificación lineal

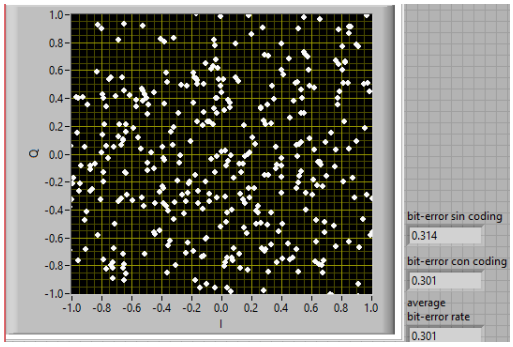


Fig. 4.46 Constelación 16QAM al pasar por un canal RAYLEIGH en un sistema con codificación lineal

Luego de obtener las tasas de error para diferentes niveles de ruido en las modulaciones BPSK, QPSK y 16QAM se generaron las curvas BER vs SNR en simulación y considerando datos reales, lo cual se muestra en las figuras 4.47. En este caso con la emulación de un canal RAYLEIGH se reafirma el efecto destructivo de las componentes de multicamino que tiene sobre el mensaje. Con una modulación 16QAM en Rayleigh las tasas de errores aparecen aun cuando el valor de SNR es de 16 dB; las curvas tienen un crecimiento más rápido del BER a medida que el SNR disminuye, lo que conlleva a tasas de errores más altas en relación a los 2 canales anteriormente analizados.

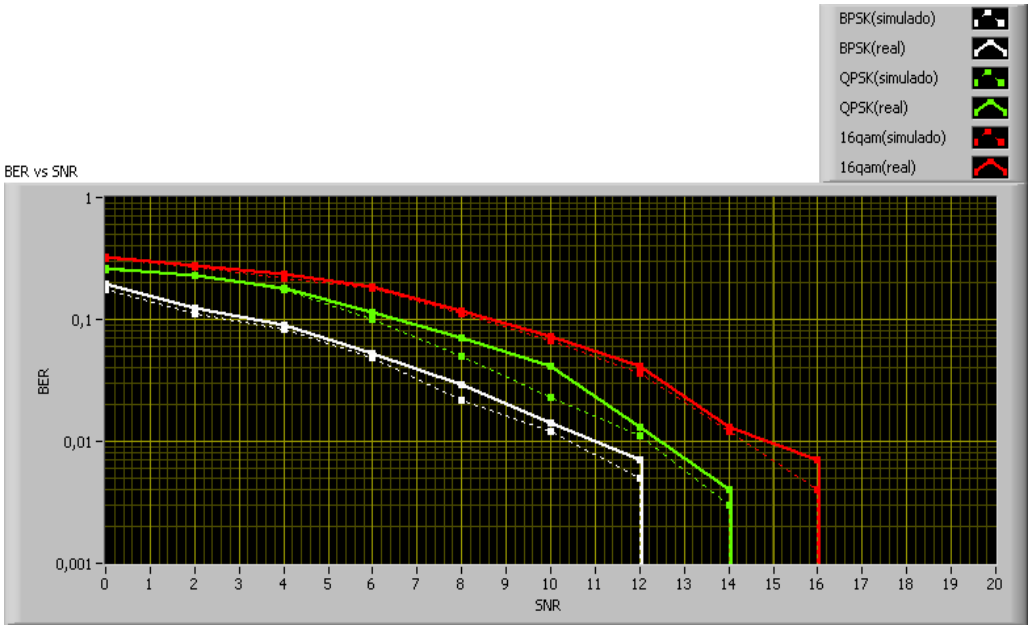


Fig. 4.47 Curvas BER vs SNR para canal RAYLEIGH, simulado y real, con codificación de bloque lineal.

4.9 COMPORTAMIENTO DEL SISTEMA FRENTE A UN CANAL RAYLEIGH USANDO CÓDIGOS CONVOLUCIONALES.

Como en los casos anteriores ahora analizaremos el comportamiento del sistema frente a un canal RAYLEIGH pero usando códigos convolucionales. De la misma forma la figura 4.48, 4.49 y 4.50 nos indican tasas de errores por encima del 11%, que comparándolas con las obtenidas en la sección 4.6 son mejores; sin embargo el canal RAYLEIGH predomina bajo cualquier aspecto y consigue deteriorar la señal de una forma más rápida que los 2 canales anteriores.

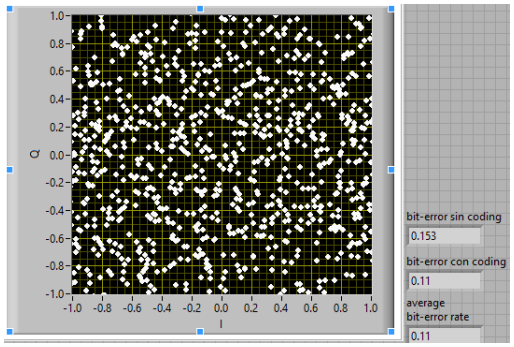


Fig. 4.48 Constelación BPSK al pasar por un canal RAYLEIGH en un sistema con codificación convolucional

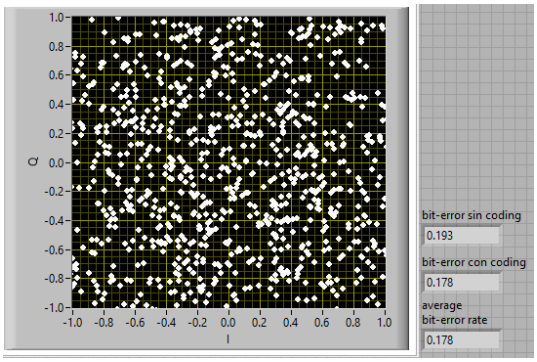


Fig. 4.49 Constelación QPSK al pasar por un canal RAYLEIGH en un sistema con codificación convolucional

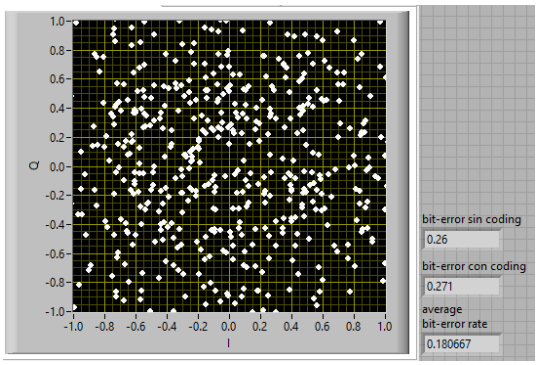


Fig. 4.50 Constelación 16QAM al pasar por un canal RAYLEIGH en un sistema con codificación convolucional

Luego de obtener las tasas de error para diferentes niveles de ruido en las modulaciones BPSK, QPSK y 16QAM se generaron las curvas BER vs SNR en simulación y considerando datos reales, lo cual se muestra en la figura 4.51. Al igual que en la sección 4.7 las tasas de error son mayores que usando los 2 canales iniciales, sin embargo son menores con respecto a los codificadores de bloques lineales.

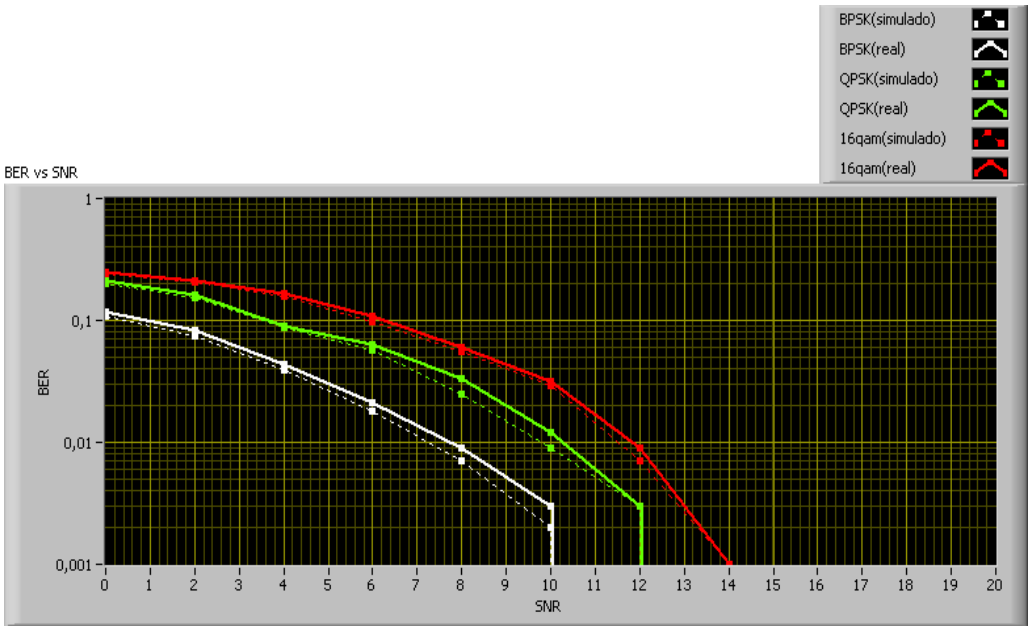


Fig. 4.51 Curvas BER vs SNR para canal RAYLEIGH, simulado y real, con codificación convolucional

4.10 ANÁLISIS COMPARATIVO ENTRE CODIFICACIÓN DE BLOQUES LINEAL Y CONVOLUCIONAL

La figura 4.52, 4.53 y 4.54 muestran curvas de BER vs SNR para un sistema que no utiliza codificación de canal y para uno que si lo hace usando un canal AWGN. Se indican las modulaciones BPSK, QPSK y 16QAM respectivamente; en todos los casos se observa que los códigos convolucionales ofrecen menores tasas de errores y que mientras el esquema de modulación aumenta el BER aparece a niveles de SNR más altos y crece con un patrón más acelerado.

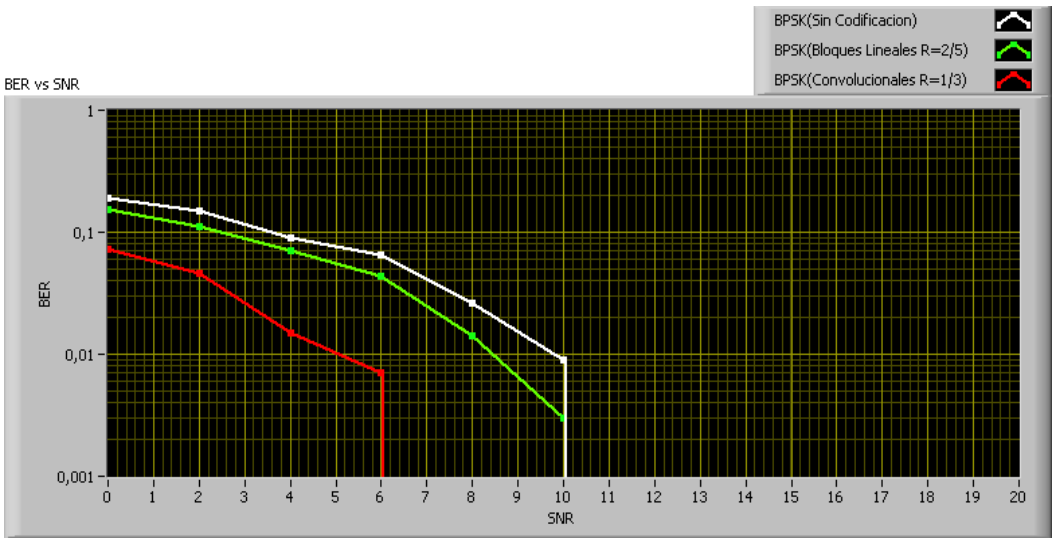


Fig. 4.52 Curvas BER vs SNR para un canal AWGN usando los diferentes algoritmos implementados con modulación BPSK.

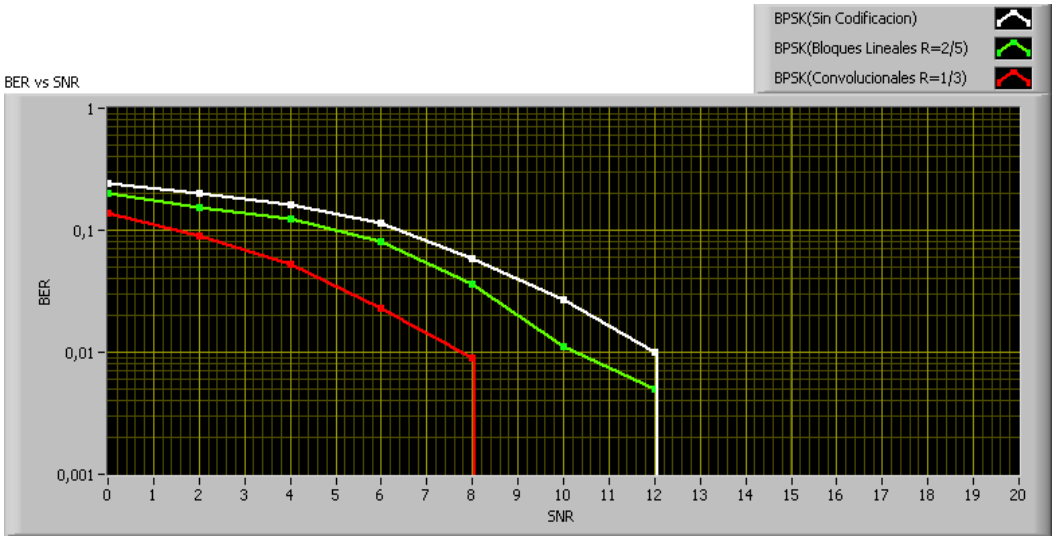


Fig. 4.53 Curvas BER vs SNR para un canal AWGN usando los diferentes algoritmos implementados con modulación QPSK.

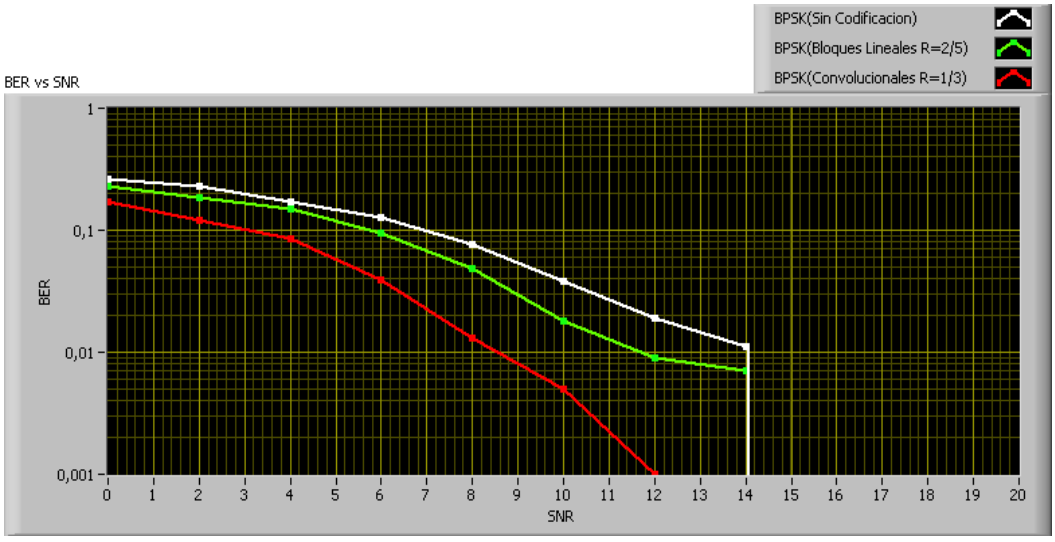


Fig. 4.54 Curvas BER vs SNR para un canal AWGN usando los diferentes algoritmos implementados con modulación 16QAM.

Un aspecto a tener en cuenta cuando hablamos de eficiencia y comparación entre codificadores de canal es la ganancia de codificación. De las gráficas obtenidas de BER vs SNR podemos generar esta ganancia, la cual mide el desempeño que tiene un sistema de comunicación que no usa coding frente a uno que si lo hace. La ganancia del codificador expresa la reducción del SNR para alcanzar una determinada probabilidad de error. Las figuras 4.55, 4.56 y 4.57 han sido generadas usando los resultados obtenidos de las graficas 4.52, 4.53 y 4.54 respectivamente para BPSK, QPSK y 16QAM respectivamente. Se puede observar que sobre un canal AWGN en un sistema de comunicaciones OFDM con codificación de bloques lineales se requieren en promedio 2 dB menos de potencia de transmisión para obtener una tasa de errores entre 0,001 y 0,1, rango sobre el cual el sistema tiene un desempeño optimo. De la misma forma, en un sistema de comunicaciones OFDM con codificación convolucional se requieren en promedio 4 dB menos de potencia de señal para obtener el mismo rango de tasa de errores. Cabe indicar que ésta ganancia de codificación va disminuyendo a media que aumenta la tasa de errores, tanto para la codificación convolucional como para la de bloques lineales

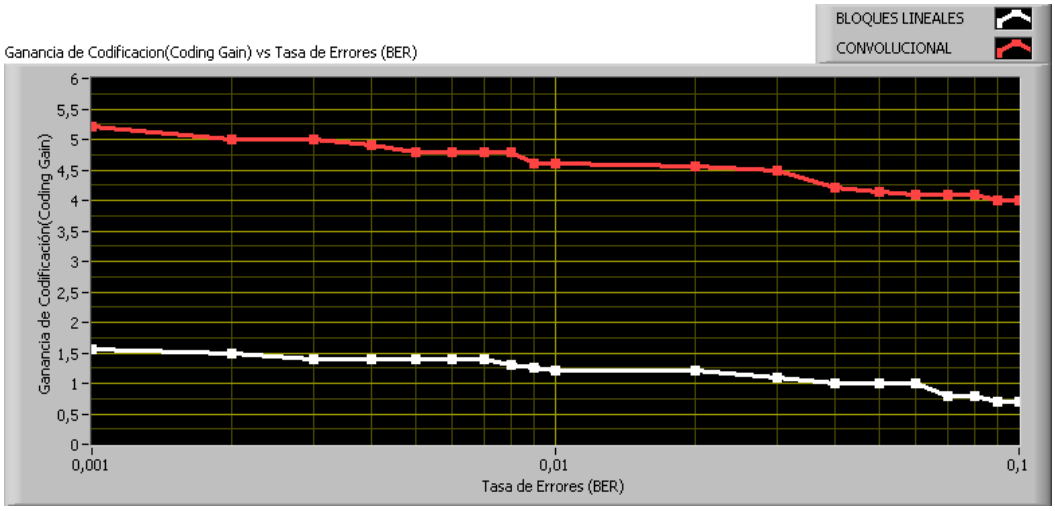


Fig. 4.55 Curvas de ganancia de codificación para codificadores lineales y convolucionales en un canal AWGN con modulación BPSK.

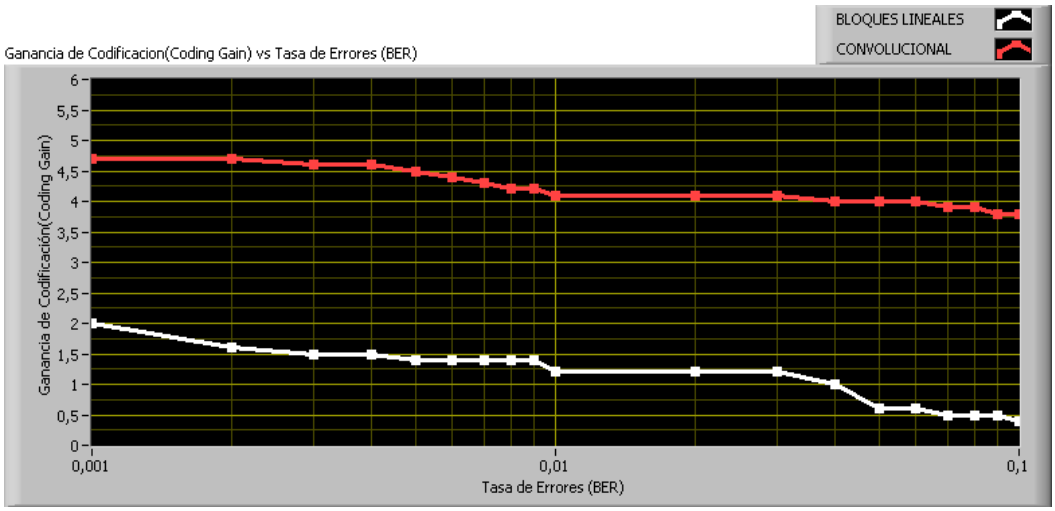


Fig. 4.56 Curvas de ganancia de codificación para codificadores lineales y convolucionales en un canal AWGN con modulación QPSK.

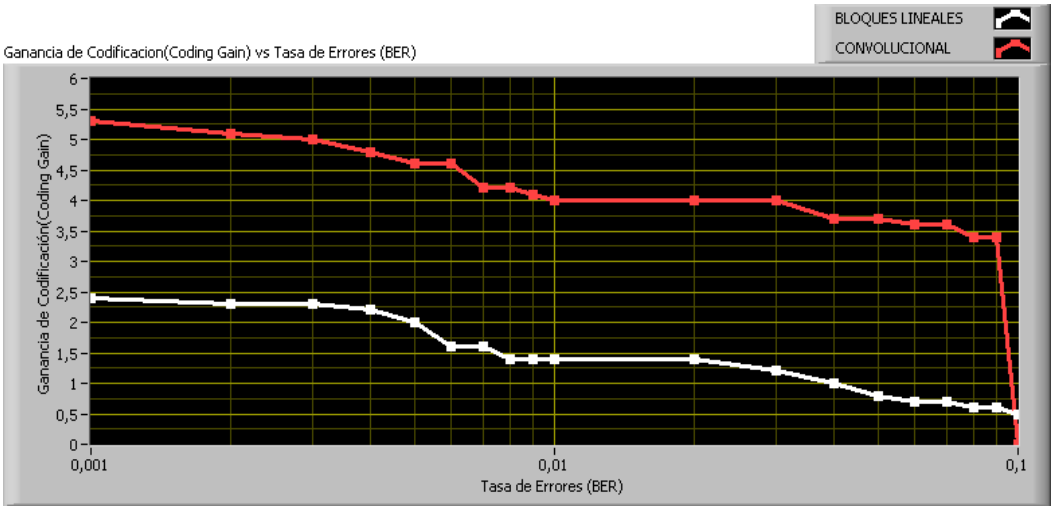


Fig. 4.57 Curvas de ganancia de codificación para codificadores lineales y convolucionales en un canal AWGN con modulación 16QAM.

Para culminar con nuestro análisis se observa en la figura 4.58, 4.59 y 4.60 curvas de BER vs SNR de los algoritmos utilizados en un canal RAYLEIGH con las respectivas 3 modulaciones. En todos los casos se pueden observar que los sistemas de comunicación OFDM que utilizan codificación convolucional tienen un mejor desempeño en cuanto respecta a las tasas de errores. Podemos comparar una vez más, que debido al multicamino producido por el canal Rayleigh el BER aumenta con respecto a los valores obtenidos en el canal AWGN.

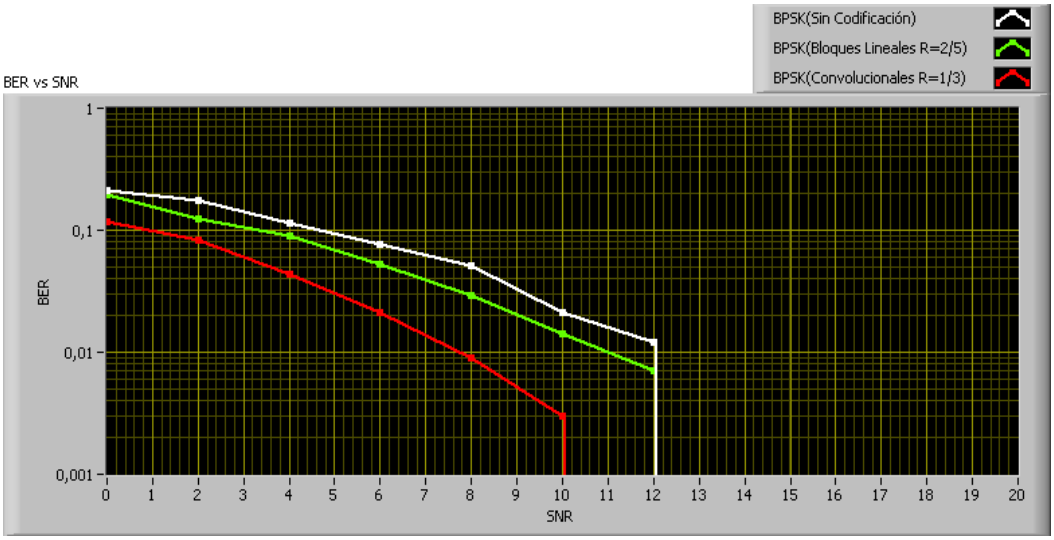


Fig. 4.58 Curvas BER vs SNR para un canal RAYLEIGH usando los diferentes algoritmos implementados con modulación BPSK.

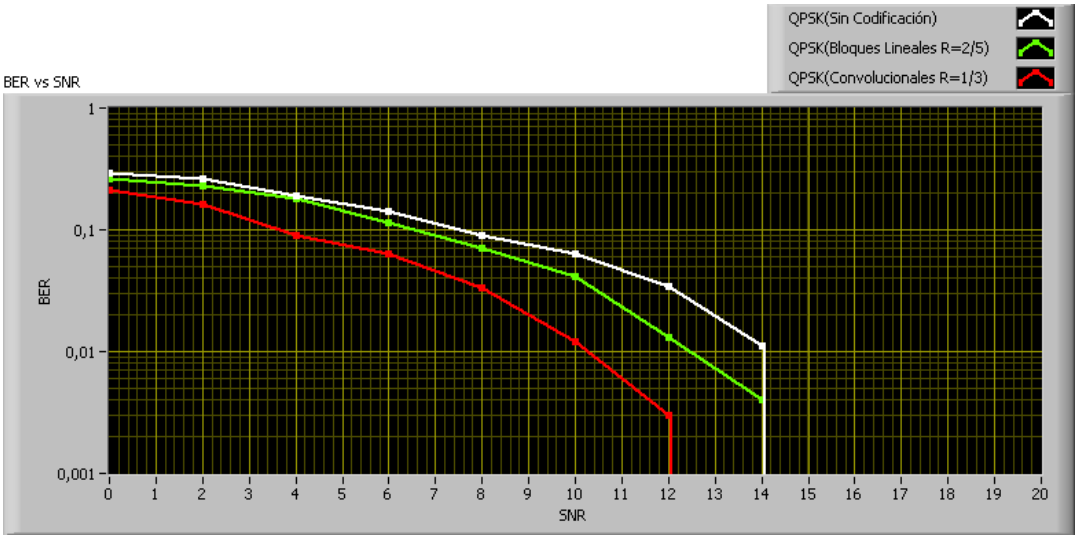


Fig. 4.59 Curvas BER vs SNR para un canal RAYLEIGH usando los diferentes algoritmos implementados con modulación QPSK.

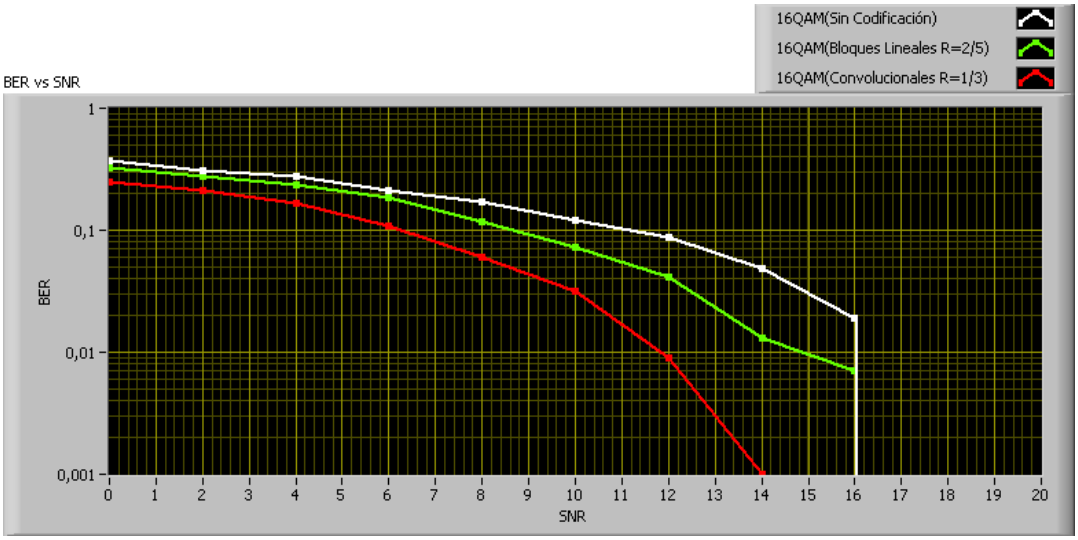


Fig. 4.60 Curvas BER vs SNR para un canal RAYLEIGH usando los diferentes algoritmos implementados con modulación 16QAM.

Al igual que en el canal AWGN se muestran en las figuras 4.61, 4.62 y 4.63 curvas de ganancia de codificación para los algoritmos implementados usando un canal Rayleigh y modulación BPSK,QPSK y 16QAM. Se puede observar que se requiere en promedio 2 dB menos de potencia de transmisión para obtener una tasa de errores entre 0,001 y 0,1 si usamos códigos de bloques lineales. Mientras que con codificación convolucional se requieren en promedio 4 dB menos de potencia de señal para obtener el mismo rango de tasa de errores. Como se observa los valores de ganancia se asemejan a los obtenidos en AWGN dándonos a entender la robustez, eficacia e importancia del uso de ellos, más aún cuando se experimenta con canales que introducen condiciones adversas a nuestro mensaje.

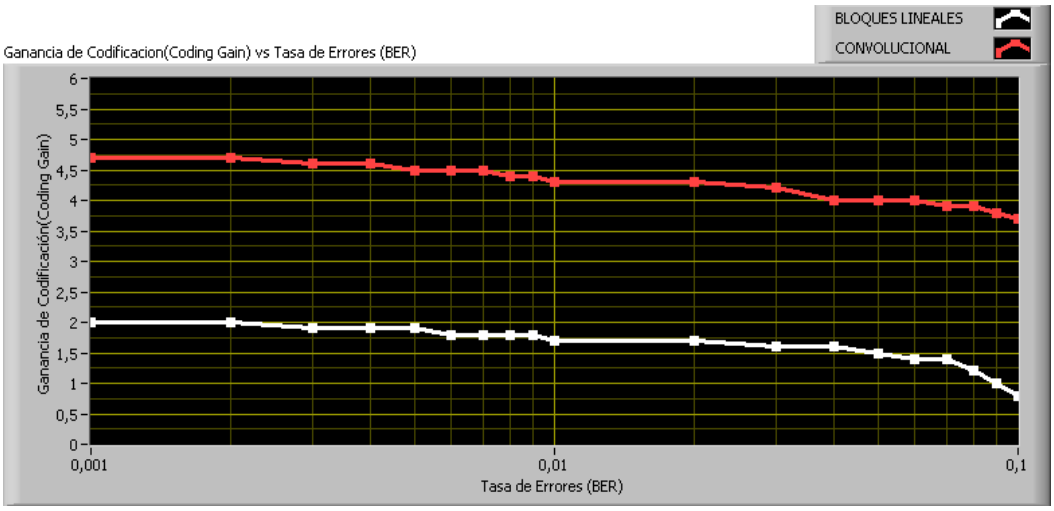


Fig. 4.61 Curvas de ganancia de codificación para codificadores lineales y convolucionales en un canal RAYLEIGH con modulación BPSK.

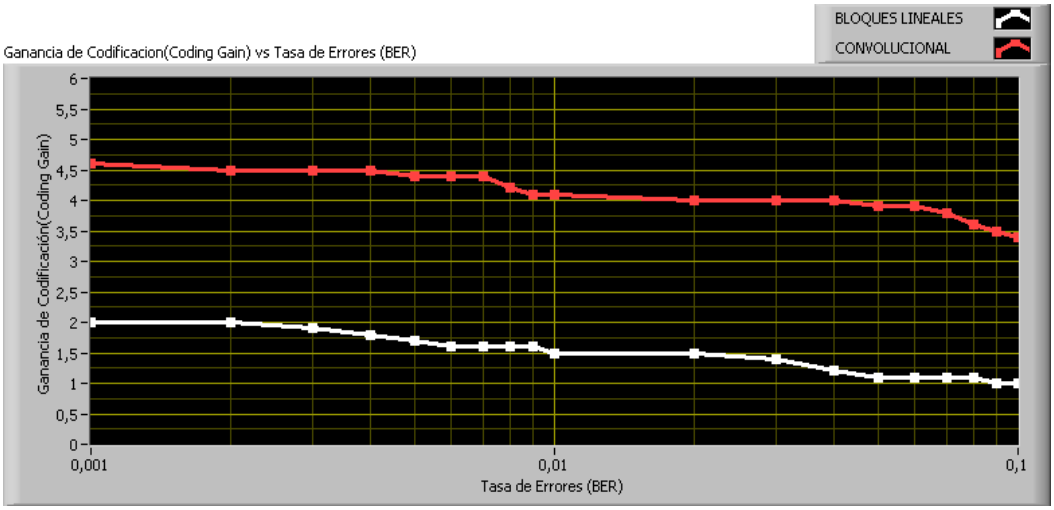


Fig. 4.62 Curvas de ganancia de codificación para codificadores lineales y convolucionales en un canal RAYLEIGH con modulación QPSK.

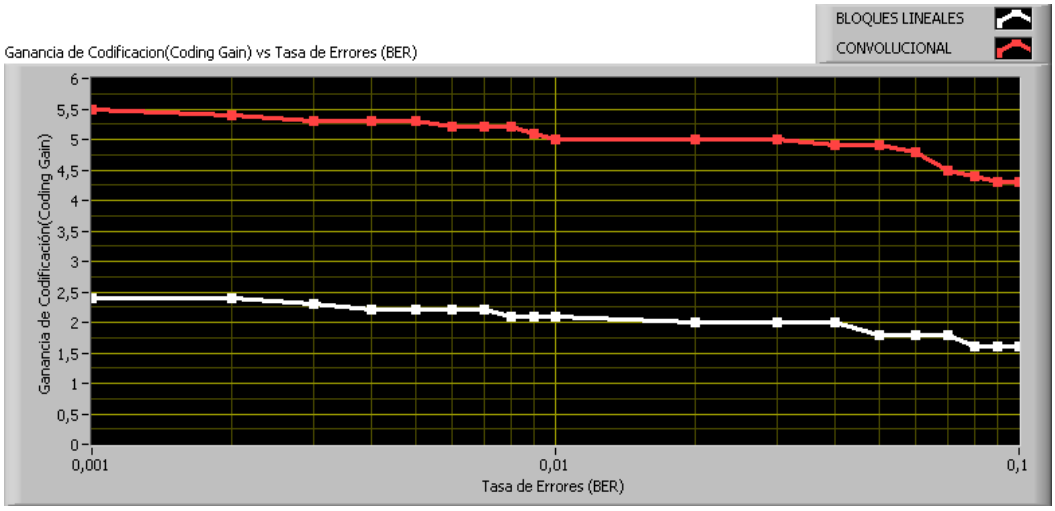


Fig. 4.63 Curvas de ganancia de codificación para codificadores lineales y convolucionales en un canal RAYLEIGH con modulación 16QAM.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

1. Los algoritmos de codificación y decodificación de canal son de vital importancia en todo sistema de comunicación; especialmente cuando necesitamos que el receptor obtenga un mensaje tan idéntico como el transmisor. Un codificador de canal garantizara tasas de errores menores frente a un sistema que no hace uso de ellos.

2. El tipo de modulación es un factor que influye en el cálculo del BER. Mientas la constelación tenga mayor cantidad de regiones de máxima verosimilitud existe mayor probabilidad que el receptor se equivoque al momento de decidir a favor de un símbolo. De aquí que la modulación

16QAM tiene tasas de errores mayores frente a un sistema que usa QPSK o BPSK.

3. Un codificador de bloque lineal basa su algoritmo en operaciones algebraicas de matrices, suma y multiplicación en módulo 2, creando palabras de código sistemáticos que permiten identificar los bits de mensaje y los bits de redundancia. El receptor emplea estas palabras de código para la detección y corrección de errores; sin embargo silos errores ocurrieran en ráfagas, el decodificador podría decidir a favor del símbolo erróneo. Es por ello que en escenarios en donde existe multitrayectoria y pérdidas debido a la ausencia de línea de vista la probabilidad de error aumenta.

4. Un codificador convolucional genera palabras de código basadas en los bits presentes y pasados dentro de un registro de desplazamiento. Por esta razón, posee tasas de errores más bajas en comparación con la codificación de bloques lineales; por otro lado la complejidad de este tipo de codificadores es mayor ya que implica la implementación de máquinas de estado finito.

5. Se puede observar que al utilizar codificación de canal existe una ganancia de codificación, esto es, para obtener una misma tasa de errores el nivel de señal a ruido es menor en comparación con un sistema que no utiliza codificación de canal.

6. Además de los factores a considerar para la decisión de uso entre un esquema de codificación de canal y otro, mencionado en los párrafos anteriores como complejidad, ganancia de codificación y tasa de errores obtenida, es necesario recordar que el uso de esquemas de corrección de errores disminuye la velocidad de transmisión de datos o throughput. Es necesario tener un compromiso entre complejidad, BER, ganancia de codificación, tipo de modulación y throughput.

7. El desempeño de un sistema de comunicación codificado sobre canales de interferencia intersimbólica y de ruido blanco aditivo gaussiano, es muy similar cuando se trabaja sobre la modulación OFDM, debido a las características propias de este tipo de esquemas como son introducción de null tones, intervalos de guarda y portadoras ortogonales. Sin embargo, sobre canales con desvanecimiento plano, Rayleigh, el sistema presenta vulnerabilidades que podrían afectar los niveles de error obtenidos. En este caso, se puede considerar el uso de modulación

adaptiva incluyendo tipos de modulación como QPSK o BPSK según la calidad del enlace.

RECOMENDACIONES

1. Debido a que el uso de los equipos USRP a nivel mundial es reciente, se sugiere consultar información sobre su fabricación, funcionamiento y resultados obtenidos en diversas referencias previo a su experimentación con los mismos.

2. Los códigos de canal presentan vulnerabilidades a los errores en ráfagas que puedan ser introducidos por el canal, incrementando las tasas de error. Esto puede ser mitigado con ayuda de un bloque de *interleaver* o intercalador aplicado a las palabras de código, dado una regla de correspondencia conocida por el transmisor y el receptor.

3. La codificación convolucional es la base para la implementación de codificación turbo que es uno de los esquemas de corrección de errores más cercanos al límite de Shannon y presentes en sistemas de comunicaciones de tercera y cuarta generación, el presente documento pretende ser el punto de partida a utilizarse dentro de una investigación profunda en el estudio de la codificación de canal en sistemas

contemporáneos, por lo que se sugiere la experimentación sobre esta rama.

4. Para comprobar el correcto funcionamiento de los esquemas de codificación de canal se sugiere la simulación e implementación de canales con desvanecimiento plano cuyos resultados nos servirán para determinar el desempeño del sistema y realizar futuras comparaciones.

BIBLIOGRAFÍA

[1] S. Haykin, Sistemas de Comunicación, Mexico: Limusa Wiley, 2002.

[2] A. Goldsmith, Wireless Communications, California: Cambridge University Press, 2005.

[3] W. Tomasi, Sistemas de comunicaciones electrónicas, México: Pearson Educación, 2003.

[4] L. Jimenez, J. Parrado, C. Quiza y C. Suárez, «Modulación Multiportadora OFDM,» *Ciencia, Investigación, Academia y Desarrolo,* pp. 30-34, 2001.

[5] S. (. E. L. Online), «Revistas Bolivianas,» [En línea]. Available: [http://www.revistasbolivianas.org.bo/img/revistas/ran/v3n4/tabla09_5.gif] . [Último acceso: 10 Junio 2013].

[6] Wikitel, «Wikitel.info,» [En línea]. Available: http://wikitel.info/images/thumb/e/ee/Distribucion_OFDM.JPG/600px-Distribucion_OFDM.JPG. [Último acceso: 15 08 2013].

[7] C. P. Vega, «Modulación COFDM,» de *Transmisión de Televisión*.

[8] A. Jara, «Tesis Multiplexación por división de frecuencia ortogonal codificada COFDM».

[9] D. Garro, Tesis Multiplexación por división de frecuencias ortogonales OFDM, Universidad de Costa Rica, 2012.

[10] T. Rappaport, Introduction to Wireless Communication Systems.

[11] M. Nentwig, «DSPRelated,» [En línea]. Available: http://www.dsprelated.com/showarticle/32.php. [Último acceso: 20 11 2013].

[12] B. Pattan, «microwaves&rf,» [En línea]. Available: http://mwrf.com/systems/understanding-terrestrial-multipath-fading-

phenomena. [Último acceso: 01 12 2013].

[13] C. Torres y C. Paez, Análisis de un sistema de comunicaciones afectado por los desvanecimientos plano y lento tipo Rayleigh, 2008.

[14] D. Vallejo, Estudio y Simulación de Turbocódigos utilizando el algoritmo MAP y SOVA, Quito: Escuela Politécnica Nacional, 2011.

[15] Shannon, THE FUNDAMENTAL THEOREM FOR A NOISELESS CHANNEL, 1948.

[16] M. M. Amaya, Análisis del decodificador de turbocódigos con el empleo de ventanas deslizantes y algoritmo MAX-log-MAP, Tijuana, México: Centro de Investigación y desarrollo digital, 2011.

[17] J. Proakis, Digital Communications, Mc Graw Hill.

[18] B. Sklar, Digital Communications, EUA: Prentice Hall.

[19] M. Mezoa, «connexions,» [En línea]. Available: http://cnx.org/content/m36906/latest/. [Último acceso: 10 09 2013].

[20] Viterbi, Tomado de Principles of Digital Communication and Coding.

[21] R. Heath, Digital Communications, National Technology & Science Press, 2012.

[22] H. P. Hsu, Signals and Systems, EUA: Schaum, 1995.

[23] J. Monter y P. León, Tesis, Universidad Nacional Autónomo de México.

[24] K. Pahlavan y A. Levesque, Wireless Information Networks, Canada: John Wiley, 2005.

[25] A. Viterbi, «Convolutional Codes and their performance in communications systems,» IEEE Transactions on Communication Techonology, vol. 19, nº 5, pp. 751-772, 1971.

[26] V. Sharma, A. Shrivastav, A. Jain y A. Panday, «BER performance of OFDM-BPSK,-QPSK,- QAM over AWGN channel using forward Error correcting code,» International Journal of Engineering, Research and

Applications, vol. 2, nº 3, pp. 1619-1624, 2012.

[27] P. Malode, «Performance of Linear Block Coded OFDM system in BER and PAPR under different channels,» *International Journal of Applied Information Systems,* vol. 3, nº 4, pp. 6-12, 2012.

[28] S. Joshi, «Coded-OFDM in Various Multipath Fading Environments,» *IEEE Transactions,* vol. 3, nº 7, pp. 127-131, 2010.

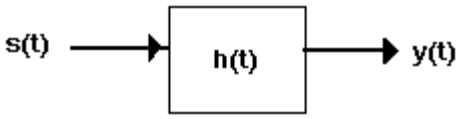
[29] Recommendation ITU-R m.1225. Guidelines for evaluation of radio transmission technologies for IMT-2000

ANEXOS

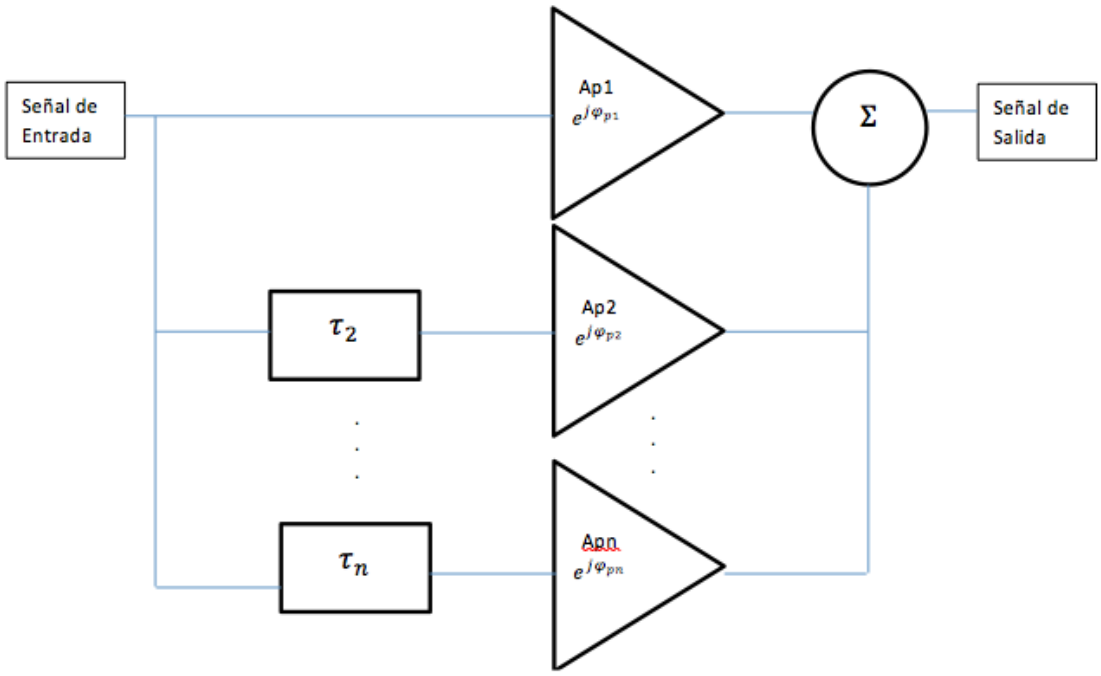
ANEXO 1

RESPUESTA DE UN CANAL CON DESVANECIMIENTO

El canal de comunicaciones con desvanecimientos puede ser modelado como un sistema lineal:



Donde h(t) se conoce como la respuesta al impulso del canal, el cual podemos esquematizarlo de la siguiente manera:



Siendo τ_n: $Retardos$ y $Apne^{j\varphi_{pn}}$: $Ganancia\ y\ Fase\ de\ los\ retardos$

Siguiendo dicho esquemático, se puede representar a la función h(t) de la siguiente forma:

$$h(t) = \sum_{p=1}^{N} a_p e^{j\emptyset_p} \delta(t - \tau_p)$$

De modo que la señal y(t) recibida en el receptor es:

$$y(t) = s(t) * h(t) = s(t) * \sum_{p=1}^{N} a_p e^{j\emptyset_p} \delta(t - \tau_p)$$

$$y(t) = \sum_{p=1}^{N} a_p e^{j\emptyset_p} (s(t) * \delta(t - \tau_p))$$

$$\boxed{y(t) = \sum_{p=1}^{N} a_p e^{j\emptyset_p} s(t - \tau_p)}$$

Donde los parámetros a_p, φ_p y τ_p: dependen de la respuesta del canal, siendo:

$$a_p: Magnitud\ de\ los\ pulsos\ de\ retardo$$

$$\varphi_p: Fase\ de\ los\ pulsos\ de\ retardo$$

$$\tau_p: Retardo\ de\ cada\ pulso$$

El perfil de retardos del canal de interferencia intersimbólica ISI y Rayleigh ha sido configurado siguiendo el formato de la tabla que se muestra a continuación:

$a_i(dB)$	$\tau_i(s)$
a_1	τ_1
a_2	τ_2
a_3	τ_3
a_4	τ_4
a_5	τ_5

Considerando: $\tau = 1\ us, 2us, \dots, n\ us; n \in \mathbb{Z}^+$

Se debe convertir las amplitudes de los retardos, esto es:

$$a_i(Amplitud) = 10^{\frac{-a_i(dB)}{20}}$$

De modo que cada uno de los coeficientes p correspondiente a la magnitud de los retardos, es obtenido de acuerdo a una variable aleatoria como sigue:

$$a_p = \frac{a_i \#f\ v.a.}{\sqrt{\sum_{i=1}^{N} {a_i}^2}}$$

Siendo el término **#f v.a.**, un número que es función de una variable aleatoria que puede ser Rayleigh, Rician Nakagami, entre otras.

PARÁMETROS PARA UN CANAL ISI

Channel Response	
Componente Compleja	**Equivalente en Magnitud y Fase**
$a_1 + jb_1$	$a_{p1}e^{j\varphi_1}$
$a_2 + jb_2$	$a_{p2}e^{j\varphi_2}$
$a_3 + jb_3$	$a_{p3}e^{j\varphi_3}$

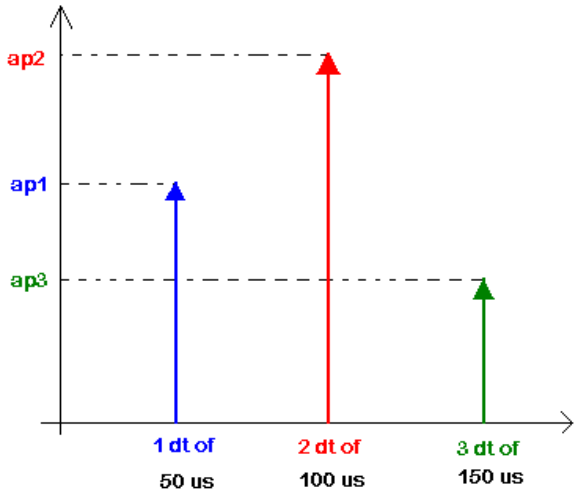
Se puede observar de la tabla adjunta que los retardos de cada uno de los pulsos que forman parte de la respuesta impulso del canal ISI es constante, siendo éste dependiente del *oversample factor* y del parámetro dt tomado del cluster *complex waveform*.

Ejemplo:

Oversample factor: 20

dt=2,5us

50us



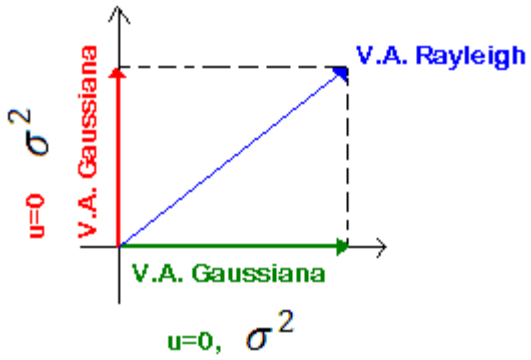
PARÁMETROS PARA UN CANAL RAYLEIGH

Channel Response	
$\boldsymbol{a_i(dB)}$	$\boldsymbol{\tau_i(s)}$
a_1	τ_1
a_2	τ_2
a_3	τ_3
a_4	τ_4

Los parámetros para la respuesta del canal han sido tomados del estándar ITU-P1225 que corresponden a un canal indoor de oficina tal como lo indica el anexo 2.

Se considera una función de densidad de probabilidad Rayleigh, la cual es obtenida a partir de la suma en fase y en cuadratura de dos variables

aleatorias gaussianas de media cero y varianza $\sigma^2 = No/2$, como se muestra a continuación:



Siendo la media y varianza de la variable aleatoria Rayleigh, de la siguiente forma:

$$\mu_R = \sqrt{\frac{\pi}{2}}\sigma_g$$

$$\sigma_R^2 = \frac{4-\pi}{2}\sigma_g^2$$

ANEXO 2

PARÁMETROS DE RETARDOS PRODUCIDOS POR EL MULTICAMINO EN UN AMBIENTE INDOOR DE OFICINA BASADOS EN EL ESTÁNDAR ITU-P1225 [29].

Tap	Channel A		Channel B		Doppler spectrum
	Relative delay (ns)	Average power (dB)	Relative delay (ns)	Average power (dB)	
1	0	0	0	0	Flat
2	50	–3.0	100	–3.6	Flat
3	110	–10.0	200	–7.2	Flat
4	170	–18.0	300	–10.8	Flat
5	290	–26.0	500	–18.0	Flat
6	310	–32.0	700	–25.2	Flat

ANEXO3

DIAGRAMA DE BLOQUE EN LABVIEW DEL CANAL RAYLEIGH