UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA EN TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

CARACTERIZACIÓN DE UN ESQUEMA DE CODIFICACIÓN Y DECODIFICACIÓN CAÓTICO DE CANAL BAJO EL ESTÁNDAR LTE.

CARDONA DAVID
PEÑA GINETH

Bárbula, 18 de Agosto del 2015

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA EN TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

CARACTERIZACIÓN DE UN ESQUEMA DE CODIFICACIÓN Y DECODIFICACIÓN CAÓTICO DE CANAL BAJO EL ESTÁNDAR LTE.

TRABAJO ESPECIAL DE GRADO PRESENTADO ANTE LA ILUSTRE UNIVERSIDAD DE CARABOBO PARA OPTAR AL TÍTULO DE INGENIERO DE TELECOMUNICACIONES

CARDONA DAVID
PEÑA GINETH

Bárbula, 18 de Agosto del 2015

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA EN TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

CERTIFICADO DE APROBACIÓN

Los abajo firmantes miembros del jurado asignado para evaluar el trabajo especial de grado titulado "CARACTERIZACIÓN DE UN ESQUEMA DE CODIFICACIÓN Y DECODIFICACIÓN CAÓTICO DE CANAL BAJO EL ESTÁNDAR LTE.", realizado por los bachilleres CARDONA DAVID, cédula de identidad 18.533.085, PEÑA GINETH, cédula de identidad 20.393.234, hacemos constar que hemos revisado y aprobado dicho trabajo.

Firma
Prof. AHMAD OSMAN
TUTOR

Firma
Prof. CARLOS MEJÍAS
JURADO

Firma
Prof. ELIMAR HERNÁNDEZ
JURADO

Bárbula, 18 de Agosto del 2015

Dedicatoria

A mis padres por todo el esfuerzo que han hecho, por la paciencia que me han tenido, por todos los consejos dados y por compartir esto conmigo.

CARDONA DAVID

A mis padres por su amor y creer en mí en todo momento.
A mi hermana por ser mi mejor amiga.
A mi novio por su apoyo incondicional.

PEÑA GINETH

Agradecimientos

Gracias infinitas a mi familia por el apoyo que me brindaron durante toda la carrera. Agradecido con nuestro tutor, el Ingeniero Ahmad Osman, por todo el esfuerzo valioso dedicado a este trabajo. Un agradecimiento especial a mi novia Marbellys Ramos por toda la ayuda brindada y apoyo incondicional que hizo posible la culminación de este trabajo de grado. Por último, pero no menos importante, agradecido con el Ingeniero Carlos Mejías por todo el apoyo brindado.. . . David Cardona

Le agradezco a Dios por haberme acompañado a lo largo de mi carrera y por darme una familia maravillosa la cual me ha apoyado toda mi vida. Le doy gracias a mis padres por los valores que me han inculcado, y por haberme dado la oportunidad de tener una excelente educación en el transcurso de mi vida. A mi hermana por ser parte importante en mi vida y siempre estar pendiente de mí. Gracias a mi novio por su apoyo, amor y por ser mi amigo. Agradecida con los Ingenieros Ahmad y Carlos por dedicarnos parte de su tiempo para lograr culminar este trabajo. De igual manera, agradecida con toda mi familia en especial con mi tía Karla por prestarme su computadora durante todos los meses de este trabajo. Y finalmente a mis amigos Roxana, Miguel, Eduardo, Ricardo, los Luises, Kevin, Jorge, Moises,... Ustedes me acompañaron a lo largo de este proceso estuvieron pendientes de que todas las cosas me salieran bien, les agradezco la confianza que depositaron en mi para poder estar siempre en las buenas y las malas juntos. Gracias por confiar y creer en mí y haber hecho de mi etapa universitaria un trayecto de vivencia que nunca voy a olvidar... Gineth Peña

Índice general

Índice de figuras

Indice de tablas

Acrónimos

ADSL	**A**symmetric **D**igital **S**ubscriber **L**ine
ARQ	**A**utomatic **R**epeatre **Q**uest
AWGN	**A**dditive **W**hile **G**aussian **N**oise
BER	**B**it **E**rror **R**ate
CDMA	**C**ode **D**ivision **M**ultiple **A**ccess
DAB	**D**igital **A**udio **B**roadcasting
DVB-T	**D**igital **V**ideo **B**roadcasting- **T**errestrial
DVB-T2	**D**igital **V**ideo **B**roadcasting- **T**errestrial **2** Second Generation
ECG	**E**lectro**c**ardio**g**rama
FEC	**F**orward **E**rror **C**orrection
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
GRC	**G**NU **R**adio **C**ompanion
HSPA	**H**igh **S**peed **P**acket **A**ccess
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
IP	**I**nternet **P**rotocol
LCPD	**L**ow **D**ensity **P**arity **C**heck
LTE	**L**ong**T**erm **E**volution
MIMO	**M**ultiple **I**nput **M**ultiple **O**utput
OFDM	**O**rthogonal **F**requency **D**ivision **M**ultiplexing
OFDMA	**O**rthogonal **F**requency **D**ivision **M**ultiple **A**ccess
QAM	**Q**uadrature **A**mplitude **M**odulation
QPSK	**Q**uadrature**P**hase **S**hift **K**eying
RB	**R**esource **B**lock
RE	**R**esource **E**lement

RF	**R**adio **F**recuencias
SAE	**S**ystem **A**rchitecture **E**volution
SC-OFDMA	**S**ingle **C**arrier **O**rthogonal **F**recuency **D**ivision **M**ultiple **A**ccess
SISO	**S**oft-**I**nput **S**oft **O**utput
TDT	**T**elevisón **D**igital **T**errestre
UMTS	**U**niversal **M**obile **T**elecommunications **S**ystem
WCDMA	**W**ideband **C**ode **D**ivision **M**ultiple **A**cces
Wi-Fi	**Wi**reless - **Fi**delity
WIMA	**W**erner **I**cking **M**usic **A**ccess
3GPP	**3**rd **G**eneration **P**artnership **P**roject
4G	**4**ta. **G**eneración de Telefonía Móvil

CARACTERIZACIÓN DE UN ESQUEMA DE CODIFICACIÓN Y DECODIFICACIÓN CAÓTICO DE CANAL BAJO EL ESTÁNDAR LTE.

por

CARDONA DAVID y PEÑA GINETH

Presentado en el Departamento de Señales y Sistemas
de la Escuela de Ingeniería en Telecomunicaciones
el 18 de Agosto del 2015 para optar al Título de
Ingeniero de Telecomunicaciones

RESUMEN

La redundancia de bits implementada por los esquemas convencionales de codificación de canal que utiliza el estándar LTE, necesaria para permitir la detección y corrección de errores, no resulta eficiente desde el punto de vista de su capacidad, es decir, se utiliza más ancho de banda del necesario. De igual manera, el ruido que produce el canal de comunicación, puede alterar varios bits consecutivos y, de todas maneras, ser interpretada la información de manera errónea en el receptor. El presente trabajo de grado introduce un esquema de codificación alternativo, basado en la teoría del caos, con la finalidad de evaluar su comportamiento en el estándar LTE y compararlo con los esquemas de codificación utilizados actualmente.

Palabras Claves: LTE, Caos

Tutor: AHMAD OSMAN
Profesor del Departamento de Señales y Sistemas
Escuela de Telecomunicaciones. Facultad de Ingeniería

Capítulo I

Introducción

1.1. Motivación

La evolución más relevante que han desarrollado los estándares de telefonía móvil, ha sido la implementación de servicios multimedia los cuales requieren ancho de banda exigentes. Por ello, es importante alcanzar una elevada eficiencia espectral, que permita ofrecer servicio al mayor número de usuarios empleando un recurso limitado como es el espectro. De los esquemas de modulación presentes hasta la fecha, OFDM es el que presenta mejores características, ya que posee flexibilidad para la asignación de recursos, capacidad para obtener tasas de transmisión elevadas y su implementación es eficiente [1].

LTE es un estándar que aprovecha las ventajas ya mencionadas que proporciona la modulación OFDM para ofrecer servicios a un gran número de usuarios a la vez [2]. En dicho estándar se emplean fundamentalmente dos esquemas de codificación de canal para recuperar los errores producidos durante la transmisión: los códigos convolucionales y los Turbo códigos [1].

Recientemente, han surgido otras técnicas de codificación no convencionales, amparadas en sistemas dinámicos no lineales, que han podido extenderse hasta la codificación de canal. De lo anterior surge la vinculación con la teoría del caos,

que consiste en el estudio cualitativo del comportamiento aperiódico e inestable en sistemas dinámicos no lineales y deterministas [3].

Siguiendo en esta línea, se han desarrollado estudios que han apostado por el diseño de codificadores de canal basados en la teoría del caos, como por ejemplo, el trabajo realizado por Chen y Wornell [4], donde se obtuvieron resultados interesantes en el diseño de codificadores de canal usando sistemas caóticos discretos.

En otra investigación, Eduardo Morales, Oscar Rodríguez y Emiliano Blanco, estudiaron la codificación caótica, pero en este caso, con un algoritmo que permite la codificación de una señal de ECG real, utilizando para este proceso una señal caótica obtenida con el modelo de ecuaciones de Lorenz [3].

En el mismo orden de ideas, se tiene el trabajo "Channel coding in communications using chaos" [5], cuyo objetivo principal fue el diseño de un codificador de canal con corrección de errores implementando la teoría del Caos basado en el sistema de Lorenz.

En los métodos de codificación de canal utilizados actualmente, resulta importante destacar que su coste es el uso ineficiente del ancho de banda disponible; se requiere más velocidad (o tiempo) en la transmisión. De igual forma, en algunos casos, el ruido que produce el canal o medio de comunicación puede alterar el valor de varios bits seguidos y, de todas maneras, ser interpretada la información de manera errónea en el receptor. Ésto, aunado a la pérdida de potencia y tiempo requeridos para el reenvío de mensajes en los casos en los que haya errores en la transmisión, da paso a cuestionamientos acerca de la posibilidad de recurrir a nuevos métodos de codificación de canal [6].

Por lo expuesto hasta aquí, y como parte de la obtención de una posible solución al problema planteado, el presente trabajo de grado se orienta al desarrollo y estudio del comportamiento de un esquema de codificación de canal basado en la teoría del caos, en el estándar LTE, utilizando el software Gnuradio con la finalidad de comparar su comportamiento con el de los esquemas utilizados actualmente.

1.2. Objetivos

1.2.1. Objetivo General

Caracterizar un esquema de Codificación y Decodificación Caótica de Canal bajo el estándar LTE.

1.2.2. Objetivos Específicos

- Realizar una revisión bibliográfica acerca del funcionamiento y comportamiento de los esquemas de codificación de canal utilizados en los sistemas LTE, así como de los esquemas de codificación de canal basados en la teoría del caos.
- Simular los esquemas de codificación y decodificación de canal empleados en los sistemas LTE haciendo uso del software GNUradio.
- Simular un esquema de codificación y decodificación caótica de canal utilizando como herramienta el software GNURadio.
- Comparar el rendimiento entre el sistema de comunicación LTE convencional y el sistema de comunicación utilizando codificación y decodificación caótica de canal, en cuanto al BER.

1.3. Alcance

El propósito de este trabajo de grado se basó en la comparación de los esquemas de codificación de canal utilizados actualmente en el estándar LTE, con un esquema de codificación caótica de canal, utilizando para dicho objetivo el software GNUradio. Dicha comparación se realizó evaluando el comportamiento que tienen ambos esquemas en cuanto al BER, así como también la cantidad de bits de paridad necesarios para la transmisión, es decir, se tomó en cuenta el consumo de ancho de banda.

Se determinó los límites de la codificación caótica de canal en presencia de ruido AWGN y por los desvanecimientos de multitrayectoria tipo Rayleigh.

Capítulo II

Marco Conceptual

2.1. LTE

LTE (Long Term Evolution) es un estándar especificado por el proyecto de cooperación en sistemas de tercera generación (3GPP) en el Release 8, cuyo objetivo principal fue crear una arquitectura mucho más plana y sencilla, definida por el 3GPP como Evolución de la Arquitectura del Sistema (SAE, System Architecture Evolution), la cual está soportada totalmente en el Protocolo de Internet (IP, Internet Protocol), con un número limitado de nodos e interfaces que permiten disminuir los tiempos de señalización y procesamiento entre nodos [7]. Además, la tecnología LTE cuenta con un planificador de paquetes (packet scheduler) que cumple con funciones de asignación de dinámica de recursos, balanceo de carga, gestión de movilidad (mobility management), lo cual le permite adaptarse rápidamente a las condiciones variables del canal de radio por medio de la selección de diferentes esquemas de modulación y tasas de codificación.

LTE hace uso, en el enlace de bajada, de la técnica de Acceso Múltiple por División de Frecuencia Ortogonal (OFDMA, Orthogonal Frequency División Multiple Access) y en el enlace de subida de la técnica de Acceso Múltiple por División de

Frecuencia de Portadora Única (SC-FDMA, Single Carrier Frequency División Multiple Access) [8]. Cabe destacar también, que LTE opera con anchos de banda variables que van desde 1.25 hasta 20 MHz y velocidades de trasmisión de datos teóricos de hasta 100 Mbps para el enlace de bajada y 50 Mbps en el enlace de subida en un ancho de banda de 20 MHz. LTE emplea: Modulación por Desplazamiento de Fase en Cuadratura (QPSK, Quadrature Phase Shift Keying), Modulación en Amplitud y Cuadratura de 16 niveles (16 QAM, 16 Quadrature Amplitude Modulation), Modulación en Amplitud y Cuadratura de 64 niveles (64 QAM, 64 Quadrature Amplitude Modulation), codificación turbo, y sistemas de antenas de Entradas Múltiples y Salidas Múltiples (MIMO, Multiple Input Multiple Output), logrando de esta manera incrementar la eficiencia espectral y la capacidad del sistema [7].

2.1.1. Recursos de Transmisión de LTE en el Enlace de Bajada

Los recursos de transmisión de enlace descendente en LTE poseen dimensiones en el dominio de tiempo, frecuencia y espacio.

2.1.1.1. Dominio de la Frecuencia

En el dominio de la frecuencia, los recursos se agrupan en unidades de 12 subportadoras (lo que ocupa un total de 180kHz). Cada unidad de 12 subportadoras está contenida en una ranura (slot), al cual se denomina un bloque de recursos (RB, Resource Block).

La unidad más pequeña de los recursos, es el elemento de recursos (RE, Resource Element), que consiste en una subportadora de una duración de un símbolo OFDM.

Un bloque de recursos se compone por lo tanto de 84 elementos de recursos en el caso de la longitud de un prefijo cíclico normal, y 72 elementos de recursos en el caso del prefijo cíclico extendido.

2.1.1.2. Dominio del Tiempo

- En el tiempo existe una trama de 10ms.
- Ésta se divide en 10 subtramas de 1 milisegundo cada una.
- Cada subtrama de tiempo se divide en 2 slots de tiempo de 0.5ms.
- Cada slot tiene 7 símbolos OFDM si el prefijo cíclico es normal.
- Cada slot tiene 6 símbolos OFDM si el prefijo cíclico es extendido.

En la Tabla 2.1 se presentan los parámetros característicos en el enlace de bajada aparte de los ya definidos.

Tabla 2.1: Parámetros del Estándar LTE

Ancho de banda de Transmisión BW (MHz)	1.25	2.5	5	10	15	20
Espacio entre subportadoras (KHz)	15	15	15	15	15	15
Muestreo en Frecuencia Fs (MHz)	1.92	3.84	7.68	15.36	23.04	30.72
Longitud de la FFT	128	256	512	1024	1536	2048
Número de subportadoras útiles OT(Sin DC)	72	180	300	600	900	1200
Número de subportadora de guarda	55	75	211	423	635	847
Número de Subportadoras piloto	12	25	50	100	150	200
Resource Block	6	15	24	50	75	100
Prefijo Cíclico (CP) en muestras	9	18	36	72	108	144

2.1.2. OFDM (Multiplexación por División de Frecuencia Ortogonal)

Esta técnica consiste en enviar un conjunto de ondas portadoras en diferentes frecuencias, donde cada una transporta información, la cual es modulada en QAM o en QPSK. La multiplicación de portadoras ofrece robustez frente a la distorsión de

multirecorrido, una característica de ambientes urbanos con múltiplos obstáculos [9].

En OFDM, el espectro asociado a cada dato es una pequeña porción del ancho de banda total, el cual se divide en N subcanales. Cada subcanal se modula con un símbolo y se multiplexa en frecuencia. Ésto evita la necesidad de separar las portadoras por medio de bandas de guarda, y por lo tanto OFDM es altamente eficiente en el espectro, siendo el espaciamiento entre subportadora de 15kHz.

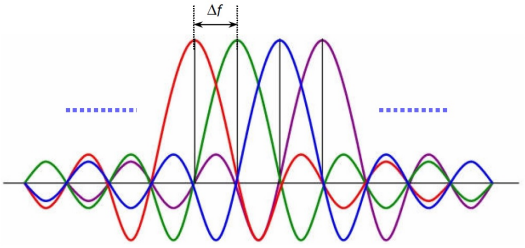

Figura 2.1: Principio de Ortogonalidad en OFDM.
[10]

Una gran cantidad de estándares hacen uso hoy en día de OFDM (Orthogonal Frequency Division Multiplexion). Claros ejemplos son algunos de los estándares de la familia DVB (Digital Video Broadcasting) como DVB-T (Digital Video BroadcastingTerrestrial) ó DVB-T2 (Digital Video BroadcastingsecondgenerationTerrestrial) , estándares para redes de área local inalámbrica como 802.11a , y estándares móviles como 4G LTE (fourthGeneration Long TermEvolution) [11].

El espaciamiento entre los sub canales en OFDM es tal, que pueden ser perfectamente separados en el receptor. Ésto permite una implementación de receptor de baja complejidad, lo que hace atractivo para OFDM de alta velocidad de transmisión de datos móviles, tales como la bajada LTE [9].

2.1.3. Codificación de Canal usada en LTE

Todo sistema de comunicación tiene un mismo objetivo: la transmisión de información desde un emisor hasta un receptor a través de un canal. Debido a la

naturaleza ruidosa de todo canal, el mensaje es distorsionado hasta llegar al receptor, por lo que en muchos casos se producen diferencias entre las secuencias de datos enviadas y las recibidas, estas diferencias son llamadas errores. Para solventar los errores en recepción se realiza la codificación de canal, cuyo objetivo es que el receptor sea capaz de detectar y corregir los errores producidos durante la transmisión. La codificación de canal consiste en introducir redundancia, de forma que sea posible reconstruir la secuencia de datos original de la forma más fiable posible [12]

En LTE, se emplean fundamentalmente dos esquemas de codificación de canal para recuperar los errores producidos durante la transmisión: los códigos convolucionales y los turbo códigos, siendo los primeros empleados para codificar información de control, puesto que requieren unos algoritmos de decodificación más sencillos desde un punto de vista computacional, mientras que los turbo códigos se utilizan para codificar información de usuario, ya que alcanzan una mayor eficiencia pese a ser más complejos que los códigos convolucionales [1].

2.1.3.1. Codificador convolucional de Canal

Los códigos convolucionales inicializan la memoria del codificador con los últimos bits del bloque que se está codificando, lo que provoca que el estado inicial y final del trellis sea el mismo. El valor inicial del registro de desplazamiento del codificador debe ser fijado con los valores correspondientes a los 6 bits de información en el stream de entrada para que los estados iniciales y finales en el registro de desplazamiento sean los mismos [13].

Para que el codificador se inicialice en cero, se agregan unos símbolos iguales a cero, estos símbolos adicionales se denominan Tail-biting. De esta manera el codificador asocia una, y sólo una secuencia codificada con cada secuencia de información.

En un sistema LTE, donde la tasa de código es de 1/3, este esquema de codificación convolucional, usa un codificador con una longitud 7. Los últimos 6 bits son utilizados para definir la cantidad de estados posibles. La salida de datos es

codificada con el codificador iniciado con el Tail-Biting, luego, los 6 bits del final del bloque de datos son puestos al inicio para ser usados como ráfagas de bits. Estas ráfagas empujan los bits que se quedaron en la etapa del bloque FEC previo [1] (2.2).

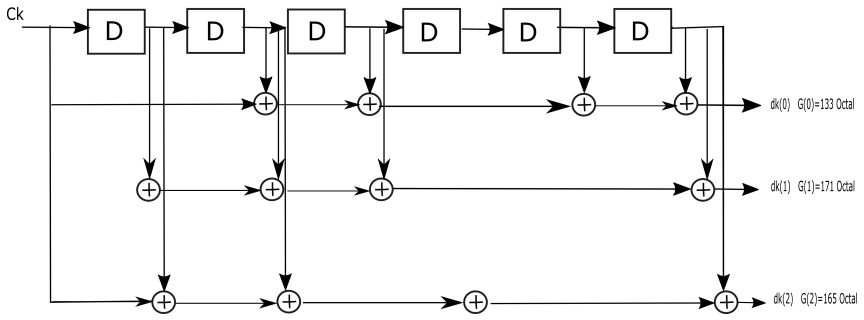

Figura 2.2: Codificador Convolucional Tail- Biting.

Las salidas del codificador $d_k^{(0)}, d_k^{(1)}, d_k^{(2)}$, corresponden con las primeras, segundas y terceras ráfagas de paridad [13].

Se pueden utilizar L registros cuyas salidas se suman de forma que se obtiene una palabra del código para cada entrada. Con la entrada de una nueva palabra a codificar las anteriores se desplazan al siguiente registro y se obtiene una nueva palabra codificada.

El proceso de decodificación consiste en la aplicación del algoritmo clásico de Viterbi [1].

2.1.3.2. Turbo Códigos

Los Turbo Códigos son un nuevo esquema de codificación, pertenecientes a la familia de códigos correctores de errores convolucionales. Ésta codificación posee un rendimiento en términos de tasa de error de bit (BER) muy próximos al conocido límite de Shannon, limite matemático teórico de la máxima tasa de transferencia en entornos ruidosos [12].

El codificador turbo implementa dos codificadores convolucionales , el primero codifica la secuencia de entrada (de longitud m bits), mientras que el segundo

codifica la secuencia de entrada permutada por el interleaver. De esta forma, cada codificador presenta a su salida unos bits de paridad, de longitud n/2. Finalmente la palabra código a enviar al canal es conformada con los bits originales sin codificar (m bits), y las dos secuencias de paridad (n/2 bits cada una) producidas por cada codificador convolucional, por lo que la palabra código presenta una longitud de m+n bits, con lo que la tasa del código es:

$$R = \frac{m}{m+n}$$

Este cociente representará la información que existe de la palabra código transmitida, dando así una visión de la proporción de bits de paridad frente a bits de información [12]. En el caso de LTE, la tasa de codificación utilizada por los Turbo Códigos es de 1/3 [14].

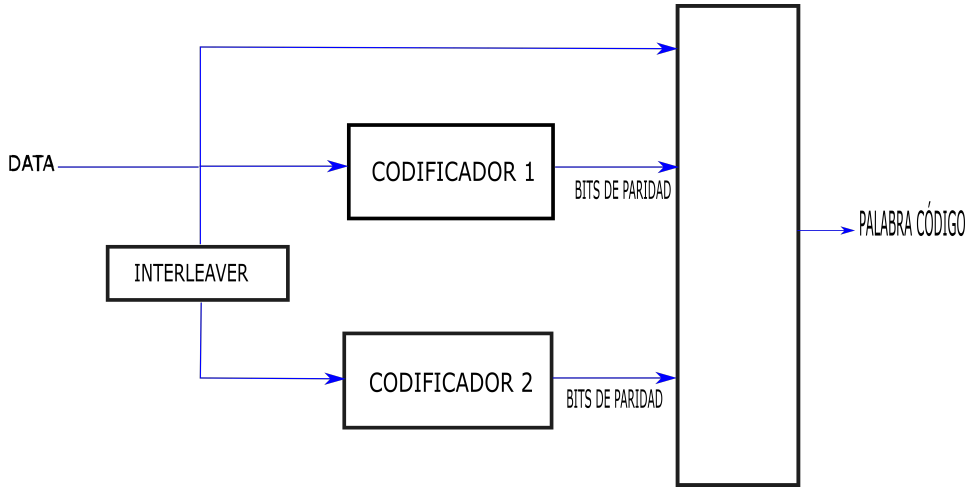

Figura 2.3: Codificador Turbo.

La posibilidad de corregir errores se consigue añadiendo al mensaje original unos bits de redundancia. La fuente digital envía la secuencia de datos al codificador, encargado de añadir dichos bits de redundancia. A la salida del codificador obtenemos la denominada palabra código. Esta palabra código es enviada al receptor y éste, mediante el decodificador adecuado y aplicando los algoritmos de corrección de errores, obtendrá la secuencia de datos original.

La decodificación se realiza con dos decodificadores SISO (Soft Input Soft Output), uno para cada una de las versiones codificadas de los datos, los cuales se van pasando iterativamente las estimaciones soft de los bits de información codificados [1].

2.1.4. Modelo de Canal Multitrayecto utilizado en el enlace de bajada en LTE

Al utilizar el canal de radio frecuencia como medio de transmisión no guiado y no controlado en un sistema de comunicaciones, los esquemas de modulación deben combatir todos los fenómenos que en él se presentan, en particular, el desvanecimiento de multitrayectoria. Este fenómeno, cuya característica usualmente es variante en el tiempo, es producido por la velocidad de movimiento del transmisor, del receptor o por la disposición variante de los objetos presentes en el canal, causantes de una reflexión, refracción, dispersión y retardo de la onda transmitida. Adicionalmente, el efecto Doppler provoca una disfuminación de la señal de recepción en el dominio frecuencial (dispersión Doppler).

La Unión Internacional de Telecomunicaciones (ITU, International Telecommunications Union) ha cuantificado el fenómeno de multitrayectoria como [15]:

- Peatonal Extendido A 5 Hz (EPA5, Extended Peatonal A 5 Hz).
- Urbano Típico Extendido 70 Hz (ETU, Extended Typical Urban 70 Hz).
- Vehicular Extendido A 70 Hz (EVA, Extended Vehicular A 70 Hz).
- Urbano Típico Extendido 300 Hz (ETU, Extended Typical Urban 300 Hz).

Tabla 2.2: Velocidad de desplazamiento promedio para la frecuencia de operación definida y el corrimiento Doppler.

Canal	Corrimiento Doppler [Hz]	Velocidad [Km/H]
EPA5	5	2.16
EVA70	70	30.24
ETU70	70	30.24
ETU300	300	129.6

2.2. Codificación Caótica de Canal

2.2.1. Teoría del Caos

Antes de comenzar a analizar las bases de esta teoría, resulta importante profundizar en el significado del caos. Proviene del griego **khaos**, que significa abismo o abismo abierto, por lo tanto es común pensar que el caos es aquello que carece de orden, algo confuso e incoherente. Sin embargo, para la teoría del caos, el enfoque difiere de los conceptos mencionados, ya que el caos es la misma esencia del orden [16].

La teoría del caos puede definirse como el campo de estudio de la conducta de los sistemas dinámicos y no-lineales. Un sistema dinámico puede ser representado mediante un modelo matemático que describe su variación en el tiempo, en dicho modelo matemático suelen aparecer ecuaciones diferenciales. En los sistemas no-lineales, no hay relación sencilla entre causa y efecto; un cambio en una de las variables puede afectar el valor de otra de manera tal que diverja radicalmente de lo que sería predecible [17].

En las dos últimas décadas, el interés por los fenómenos caóticos ha ido en aumento. Se han publicado diversas revistas (Chaos, International Journal of Bifurcation and Chaos, Physical Review E, Physica D) en las que se han descrito ejemplos de esta naturaleza, entre los que se destacan las reacciones químicas, circuitos eléctricos, mecánica celeste, ecología, economía, vibraciones mecánicas, láseres, y un largo etcétera [18]. Los principios de esta teoría se han utilizado también para describir y explicar fenómenos naturales y artificiales, como por ejemplo, para prevenir

ataques epilépticos, para el estudio del comportamiento de los estados financieros, para modelar sistemas de producción, etc.

2.2.2. El Sistema Caótico de Lorenz

Un oscilador caótico se puede controlar aplicando pequeñas perturbaciones a las variables del sistema, dichas perturbaciones harán que su evolución temporal siga una trayectoria deseada [19]. El sistema caótico de Lorenz, en particular, ha resultado ventajoso a la hora de investigar la idea del control de su dinámica desarrollando ésta técnica [5].

Las ecuaciones que describen el Sistema Caótico de Lorenz se definen como:

$$\dot{x} = \sigma(x - y) \tag{2.1}$$

$$\dot{y} = rx - y - xz \tag{2.2}$$

$$\dot{z} = bz - xy \tag{2.3}$$

Al asignarles los valores estándar a los siguientes parámetros:

$$\sigma = 10; r = 28,\ b = \frac{8}{3}$$

El sistema de Lorenz genera un atractor caótico, en un espacio de estado tridimensional. El diagrama de estado (x, y, z) realiza una trayectoria aparentemente aleatoria, formando 2 lóbulos. Inicialmente, rodeando un lóbulo un par de veces y luego saltando al otro durante unos ciclos, luego de vuelta [19].

La representación del sistema caótico de Lorenz cuando se tiene una proyección en el plano XZ, se muestra en la figura (2.4).

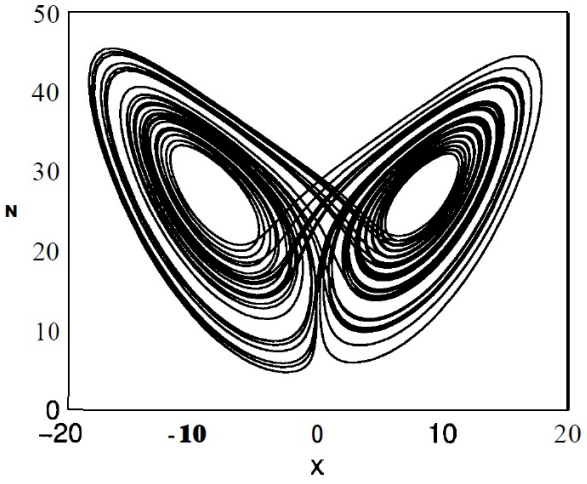

Figura 2.4: El Sistema Caótico de Lorenz
[19]

2.2.3. Las Superficies de Poncairé

Para hacer posible el control del sistema caótico de Lorenz, se definen 2 superficies de Poincaré con la finalidad de hacer posible el proceso de aprendizaje explicado en la siguiente sección, así como el control del sistema caótico de Lorenz.

Las 2 Superficies de Poncairé vienen dadas por:

$$y = +\sqrt{b\,(r-1)}, |x| \geqslant \sqrt{b\,(r-1)} \tag{2.4}$$

$$y = -\sqrt{b\,(r-1)}, |x| \geqslant \sqrt{b\,(r-1)} \tag{2.5}$$

Cuando el sistema cruza la superficie con $y = -\sqrt{b\,(r-1)}$, se dice que se genera el símbolo **A** y cuando el sistema cruza con la superficie $y = +\sqrt{b\,(r-1)}$ se dice que se genera el símbolo **B**.

Entonces, existe una relación directa entre la evolución en el tiempo del sistema y la cadena de símbolos resultantes de los cruces sucesivos de las superficies de Poincaré. Lo anterior es lo que se define como dinámica simbólica del sistema [5].

Como el sistema es determinístico, hay una correspondencia uno a uno entre el punto donde las coordenadas de estado del sistema cruzan con una de estas superficies y la futura secuencia de n-símbolos, $S_1, S_2, ..., S_n$ generada después del cruce [5].

S_1 representa el cruce actual con la superficie.

S_n representa la superficie que será atravesada $n-1$ oscilaciones más tarde.

Cuando el sistema **corre libre**, la evolución temporal a largo plazo del diagrama de estados da lugar a una cadena de símbolos con apariencia aleatoria.

Sin embargo, existe una técnica de control que permite generar una cadena de símbolos deseados que contengan un mensaje. Tal control es ejecutado aplicando pequeñas perturbaciones a una variable del sistema, obligando a su trayectoria a cruzar las superficies de Poincaré en puntos específicos que den lugar a una dinámica simbólica en particular [5].

Hay que remarcar que en este contexto, **Control** significa la habilidad de guiar al sistema para que genere una cadena de símbolos deseados que llevan un mensaje (en vez de una secuencia con apariencia caótica, como lo hacen los sistemas sin control) [5].

2.2.4. Esquema de Codificación Caótica de Canal

Brevemente, lo que se plantea a continuación se trata de un esquema de codificación caótico de canal diferente a los que utilizan convencionalmente por los sistemas de comunicaciones en la actualidad. Será desarrollado en 3 secciones muy importantes: Aprendizaje, Codificación Caótica y Decodificación Caótica.

2.2.4.1. Aprendizaje

Antes de hacer las perturbaciones al sistema es necesario realizar un proceso de aprendizaje, cuyo resultado nos dirá cómo exactamente deben realizarse dichas

perturbaciones para que su dinámica simbólica genere un mensaje deseado. En líneas generales, este proceso comprende los siguientes pasos:

- Primero se especifica el valor de n, que representa el tamaño de las secuencias de símbolos con las que se quiere trabajar.
- Una vez que el sistema caótico de Lorenz y el valor de n son definidos, se establecen las condiciones iniciales para el sistema, las cuales son (t,x,y,z) = (0,0,1,0). Una vez establecidas, se ejecuta el sistema dejándolo correr libremente con ayuda de un simulador durante un periodo de tiempo suficiente para que se generen las 2^n diferentes secuencias posibles.
- Se determinan los valores que toma la variable Z(t) en los cruces con las superficies de Poincaré y las cadenas de n-símbolos generados posteriormente por el sistema.
- Se asocia un rango de valores de Z (de aquí en adelante referidos como bin) a cada una de las diferentes secuencias de n-símbolos.
- Una vez que se identifican los bins, se calcula un promedio de todas las Z en el mismo bin, lo cual se denota como Zmean. Es importante destacar, que se puede calcular este promedio, ya que todos los valores de Z pertenecientes a un mismo bin, generan una misma secuencia.
- El valor de Zmean asociado a cada bin, determina completamente una secuencia de n-símbolos. Cuando esta asociación es hecha para todos los bins, el proceso de aprendizaje es completado. En este momento se obtiene toda la información que el codificador caótico requiere.

2.2.4.2. Codificación Caótica

Dado un mensaje a trasmitir, el codificador caótico aplica una codificación diferencial a dicho mensaje, con la finalidad de generar una secuencia de símbolos.

La salida del codificador caótico estará definida por los valores de Zmean necesarios para que el receptor pueda generar la secuencia de símbolos correspondiente al mensaje.

Para una mejor comprensión, el proceso de codificación caótica será explicado a través de un ejemplo al finalizar esta sección.

Codificación Diferencial:

El procedimiento descrito a continuación es referido como Codificación Diferencial en el contexto de comunicación digital, donde el problema de la decodificación de una señal de información consiste en pulsos con una notación de fase desconocida. Por lo tanto, la información no es extraída de la fase absoluta de los pulsos, sino de la diferencia de fase entre los pulsos sucesivos.

El procedimiento usado para aplicar la codificación diferencial es el siguiente:

Dado un mensaje binario

m = 001110101101001101100

- Se asigna el bit **0** al cambio de símbolo en 2 cruces sucesivos con las superficies de Poncairé **(AB ó BA)**.
- Se le asigna el bit **1** a la repetición del mismo símbolo **(AA ó BB)**.

De esta forma, el mismo mensaje puede ser representado por cualquiera de las siguientes secuencias de símbolos:

ABAAAABBAAABBABBBAAAABA empezando por **A**

BABBBBAABBBAABAAABBBBAB empezando por **B**

Esto es útil para evitar problemas relacionados con la generación de secuencias de bits complementarias.

Cálculo de los Valores de Zmean a transmitir:

Para generar los valores de Zmean que representen el mensaje que se quiere transmitir, se necesita una de las 2 secuencias anteriores que puede generar el codificador diferencial (dependiendo de si empieza por **A** ó por **B**).

Se asigna un valor de n, el cual representará la manera en que la secuencia simbólica será dividida y trabajada posteriormente.

En este momento se debe aclarar que, las secuencias de n-símbolos generadas en el proceso de aprendizaje producto de los cruces con las superficies de Poncairé, son almacenadas junto con los valores de Zmean asociados a cada una de ellas, para luego ser usados en el codificador caótico.

Es aquí, donde el aprendizaje es de gran importancia, ya que las secuencias simbólicas generadas del mensaje a transmitir son comparadas con las secuencias almacenadas del aprendizaje.

En pocas palabras, cada secuencia de n-símbolos a transmitir, se corresponde con un valor de Zmean almacenado en el codificador caótico (obtenido del proceso de aprendizaje). De esta manera, se van obteniendo los valores de Zmean que irán a la salida del codificador caótico.

Ejemplo:

En el proceso de aprendizaje, los valores almacenados correspondientes a los cortes con las superficies de Poncairé cuando el sistema corre libre, para n = 3, fueron los siguientes:

Secuencias = BAA, AAB, ABB, BBB, BBA, BAB, ABA, AAA

Los valores de Zmean asociados a dichas secuencias son:

Zmean = 13.1959, 16.9518, 11.4379, 20.7618, 17.7182, 14.8283, 16.6336, 20.5722

Supongamos que el mensaje que se quiere transmitir es:

m = 0011101011010011011100

Aplicando codificación diferencial, el mensaje queda como:

ABAAAABBAAABBABBBAAAABA

El siguiente paso, correspondiente al método de codificación caótica según [5], es ir construyendo secuencias de símbolos de tamaño n. La primera secuencia corresponde a los primeros 3 (n) símbolos del mensaje codificado diferencialmente, la segunda secuencia se construye tomando el segundo, tercero y cuarto símbolo

del mensaje codificado diferencialmente, la tercera secuencia es definida de igual manera con 3 símbolos, pero ahora empezando por el tercer símbolo del mensaje codificado diferencialmente, (tercer, cuarto y quinto símbolo), y así sucesivamente. Las secuencias de n-símbolos generadas con n = 3 correspondientes al mensaje, son las siguientes:

ABA, BAA, AAA, AAA, AAB, ABB, BBA, BAA, AAA, AAB, ABB, BBA, BAB, ABB, BBB, BBA, BAA, AAA, AAA, AAB, ABA

Comparando estas secuencias con las del aprendizaje, se obtiene a la salida del codificador los valores contenidos en la Tabla 2.3:

Tabla 2.3: Salida del Codificador Caótico

Zmean	Secuencia
20.5722	AAA
16.9518	AAB
16.6336	ABA
13.1959	BAA
20.5722	AAA
20.5722	AAA
16.9518	AAB
11.4379	ABB
17.7182	BBA
13.1959	BAA
20.5722	AAA
16.9518	AAB
11.4379	ABB
17.7182	BBA
14.8283	BAB
11.4379	ABB
20.7618	BBB
17.7182	BBA
13.1959	BAA
20.5722	AAA
20.5722	AAA
16.9518	AAB
16.6336	ABA
13.1959	BAA
20.5722	AAA

Cabe señalar, que los últimos 2 valores de Zmean son enviados para añadir redundancia a los dos últimos símbolos (B y A).

2.2.4.3. Decodificación Caótica

En la decodificación caótica, es donde se realizan las perturbaciones a la trayectoria del sistema de Lorenz, necesarias para obtener los valores de Zmean que generen la forma de onda a la salida, correspondiente con el mensaje deseado.

El decodificador caótico recibe todos los valores de Zmean provenientes del codificador. Luego, el sistema de Lorenz empieza a correr utilizando los mismos valores de parámetros y condiciones iniciales utilizados por el sistema implementado en el codificador caótico. Cuando la trayectoria del sistema cruza con una de las superficies de Poincaré, se calcula un delta, restando el primer valor de Zmean recibido con el valor que tiene la variable Z al momento del cruce. Posteriormente, se perturba al sistema con el delta calculado, consiguiendo que dicha superficie sea atravesada con el valor de Zmean recibido. Este mismo procedimiento se repitirá para todos los valores de Zmean que llegaron al receptor.

De esta manera, el mensaje recibido (codificado diferencialmente) se puede observar en los picos de la variable X(t) del sistema perturbado. Asumiendo que los picos positivos representarán al símbolo **B** y los negativos al símbolo **A**. Para finalizar, el decodificador hace el proceso inverso del codificador diferencial, es decir, convierte la secuencia de símbolos codificada, a los bits que representan el mensaje.

De forma general, el esquema de comunicación propuesto, puede verse como una división del algoritmo de control en 2 partes:

- En el transmisor, se toma el mensaje y se computan los valores de Zmean necesarios para generarlo. Esta información pasa al receptor convencionalmente, es decir, usando las técnicas de comunicación estándar.
- En el receptor, se aplican las perturbaciones en el sistema de Lorenz y se observa la evolución en el tiempo de la variable X(t) para recuperar el mensaje.

2.2.5. Protección de errores

El método de codificación caótica presentado, puede proveer protección contra errores debido a que los valores de Zmean(k) son altamente redundantes. El comportamiento determinístico del sistema permite predecir, a partir del valor de la variable Z(t) en cualquier cruce con las superficies de Poincaré, los símbolos que serán generados en los n-1 cruces siguientes. Una pequeña perturbación aplicada en el k-ésimo cruce con una de las superficies de Poincaré, modifica el símbolo S_{k+n-1} generado por el sistema, pero los símbolos $S_k \ldots S_{k+n-2}$ son los mismos que habrían sido generados si no se hubiera aplicado perturbación alguna.

Recordando que Zmean(k) es el valor central del bin asociado con la secuencia de n-símbolos generada después del k-ésimo cruce con la superficie de Poincaré, se tienen bins más grandes para n menores, es decir, los bins son formados por más valores de Z que producen la misma evolución del sistema caótico y que por lo tanto, generan la misma secuencia de símbolos. Por consiguiente, así exista un error, el valor de Z recibido puede estar todavía dentro del bin asociado con la misma secuencia de n-símbolos que Zmean(k) y aún se puede recuperar la información sin error.

Pero más importante aún, es que aunque el valor de Z recibido no pertenezca al mismo bin que Zmean(k), es probable que pertenezca a un bin vecino, el cual está asociado a una secuencia de símbolos que sólo difiere en los últimos símbolos. Ahora, si el siguiente valor de Z es recibido sin errores (Zmean(k + 1)), éste definirá los símbolos $S_{k+1} \ldots S_{k+n-1}$. Por lo tanto, el error puede ser evitado.

Veamos ésto último con un ejemplo para n = 6:

Supongamos que queremos transmitir la secuencia de símbolos **AAAABBABABBABAABB**

De la Tabla 2.4 se ve que para que el sistema caótico en el receptor pueda generar los primeros 6 símbolos **AAAABB** ($S_1, S_2, S_3, S_4, S_5, S_6$) , el valor de Zmean(1) que debe recibir es **20.1416**. Pero si ocurre un error a causa del ruido presente en el canal, y llega **21.0905**, la secuencia generada después de la perturbación será **AAAAAB**.

Se ve entonces que dicha secuencia sólo difiere de la secuencia correcta, en el último símbolo. Incluso si el error es mayor y llega el valor de **22.135**, la secuencia que se genera después de la perturbación difiere en los últimos 2 simbolos de la secuencia deseada. Para este ejemplo supongamos que el valor de Zmean recibido fué **21.0905** y se genera un error en el símbolo S_5.

El valor de Zmean(2) correspondiente al segundo cruce con las superficies de Poincaré que debe transmitirse para generar la secuencia **AAABBA** ($S_2, S_3, S_4, S_5, S_6, S_7$) es 19.1062. Si dicho valor llega sin errores al receptor, se corrige entonces el símbolo S_5.

Tabla 2.4: Valores de Zmean para ejemplo con n = 6

Zmean	Secuencia
.	.
.	.
22.135	AAAAAA
21.0905	AAAAAB
20.1416	AAAABB
.	.
.	.
19.1062	AAABBA
.	.
.	.

Capítulo III

Procedimientos de la investigación

3.1. Etapa 1: Simulación del esquema de comunicación en el estándar LTE.

Esta etapa fue dedicada a realizar simulaciones en el software GNURadio, con el objetivo de analizar su comportamiento, conocer las ventajas y limitaciones que presenta el software.

Para poder simular el esquema de comunicación en el estándar LTE, fue necesario analizar de manera separada el esquema de codificación de canal y el esquema de modulación que se emplea en dicho estándar.

LTE fundamentalmente emplea 2 esquemas de codificación de canal en su transmisión: Convolucional y Turbo Códigos. Siendo los primeros empleados para codificar información de control, mientras que los turbo códigos se utilizan para codificar información de usuario, ya que alcanzan una mayor eficiencia.

Para el esquema de modulación y demodulación OFDM, se estudió el comportamiento que tienen los bloques disponibles en las librerías de GNURadio diseñados para este tipo de modulación, los cuales son: OFDM Mod y OFDM Demod, así como también OFDM Transmitter y OFDM Receiver. Considerando que estos bloques no cumplen con todos los requerimientos del estándar, se tomó la decisión de

desglosar lo bloques OFDM mod y respectivamente el OFDM demod, de manera de ajustarlo a las características que posee el estándar LTE.

Una vez ya caracterizados los esquemas de codificación de canal, se procedió a simular el esquema de comunicaciones en el estándar LTE.

3.1.1. Simulación del Esquema de Codificación de Canal utilizado en el Estándar LTE.

Simulación del Esquema de Codificación de Canal Convolucional.

Fuente

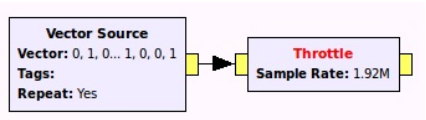

Figura 3.1: Fuente de bits.

Vector Source: permite transmitir un vector de bits, el cual es especificado por el usuario en la opción Vector.

Throttle: Se coloca a la salida de la fuente para que la computadora no utilice todos sus recursos. El valor del Sample Rate va a depender de la frecuencia de muestreo en la que se esté trabajando. En nuestro caso, los valores de Sample Rate se especifican en la tabla (2.1).

Codificador:

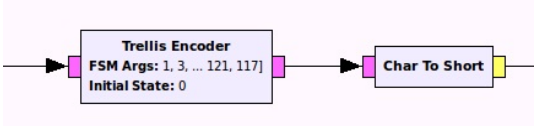

Figura 3.2: Codificador Convolucional.

Trellis-encoder: Este bloque tiene como parámetros básicos, dos parámetros específicos. El primero, denominado FSM args, permite especificar la máquina de estados que se utiliza para codificar y el segundo parámetro es el estado inicial.

Debe señalarse que el parámetro FSM args se puede especificar de 2 maneras:

- Colocando la ruta y el nombre del archivo con la especificación de la máquina de estados.
- Especificando el número de bits que entran al codificador, el número de bits que se obtienen a su salida y entre corchetes los polinomios generadores con base decimal.

Char to Short: Cambia la variable de tipo Char a Short.

Decodificador:

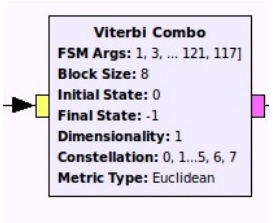

Figura 3.3: Decodificador: Viterbi Combo.

Viterbi Combo: Tiene como entrada, en este caso, elementos de tipo short. Implementa un decodificador Viterbi sobre los símbolos, para una máquina de estado que se especifica como argumento, al igual que el largo del bloque. Este bloque es la combinación del bloque *Trellis Metrics* y *Viterbi*, pero carga menos el PC, por lo que es recomendable utilizar el bloque *Viterbi Combo* en lugar de los dos anteriores.

Bloques empleados para visualizar la señal:

Error Rate: Este bloque se usa, en este caso, para comparar la señal transmitida con la señal decodificada, es decir, mide el BER entre dichas señales.

QT GUI Number Sink: Para mostrar el resultado calculado por el **Error Rate** en una interfaz gráfica, es necesario utilizar este bloque en serie.

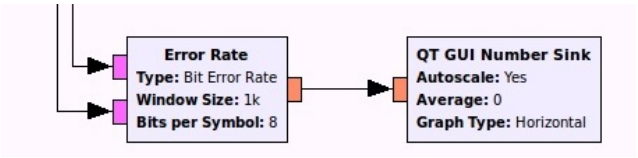

Figura 3.4: Tasa de Error Binario (BER)

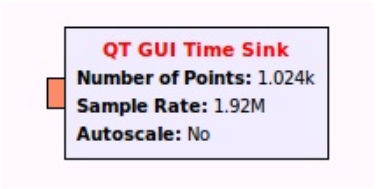

Figura 3.5: Visualizador de la Señal en el Dominio Temporal

QT GUI Time Sink: Es utilizado para visualizar la señal en el dominio temporal, o incluso obtener la constelación activando el modo XY. Es útil para los sistemas de comunicaciones digitales. Entre los parámetros que caracterizan al osciloscopio se tiene la tasa de muestreo, que es el número de veces por segundo que se mide la señal analógica, este parámetro se va a ajustar de acuerdo a la necesidades del estándar.

QT GUI Frecuency Sink: Tiene la misma función que el analizador de espectros. Los parámetros configurables que posee son: la frecuencia de muestreo, el número de puntos que se muestran, el tamaño de la FFT, ventana FFT,y el promedio FFT.

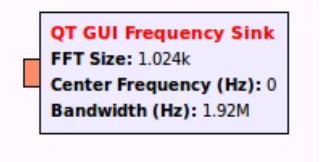

Figura 3.6: Visualizador de la Señal en el Dominio de la Frecuencia

Simulación del Esquema de Codificación de Canal empleando Turbo Códigos

Etapa de codificación:

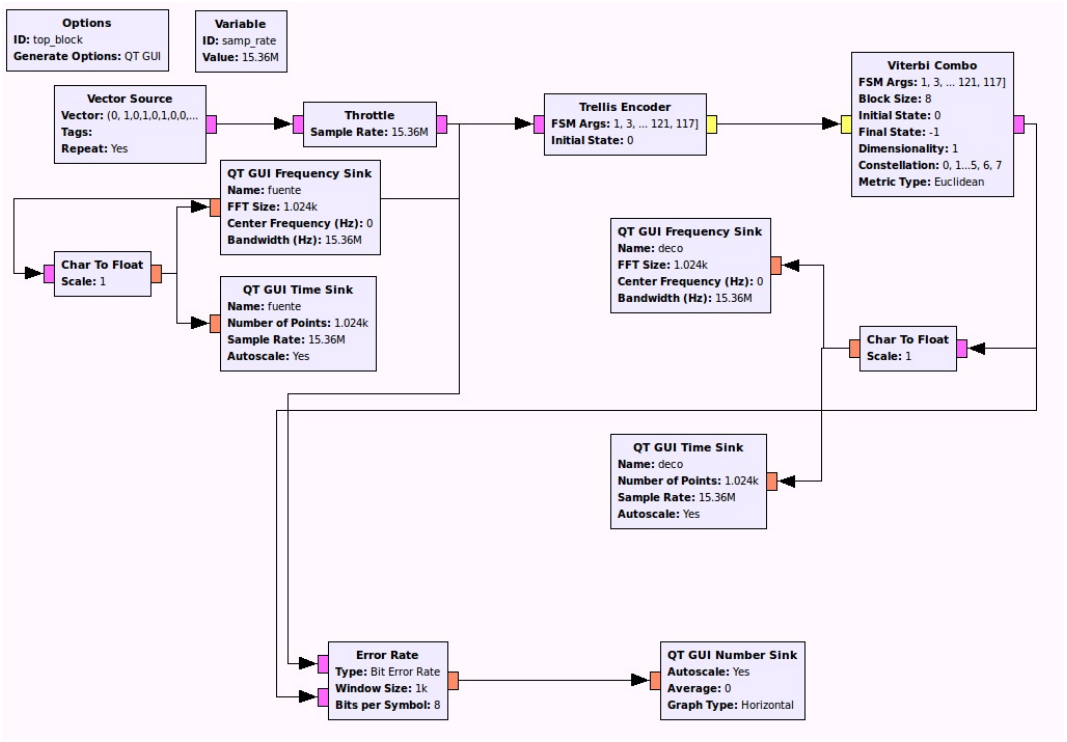

Figura 3.7: Método de Codificación y Decodificación de Canal Convolucional usado en el estándar LTE

La etapa de codificación está compuesta por los siguientes 4 bloques fundamentales:

- **Throttle:** Cumple la misma función de la explicada en la sección (3.1.1).
- **Stream to Tagged Stream:** etiqueta una cantidad de bytes. Es necesario crear esta etiqueta ya que el siguiente bloque necesita conocer la cantidad de bytes con los que va a trabajar.
- **Repack bits:** Es el bloque fundamental para que el codificador funcione. Ya que regula la cantidad de bits. Al trabajar con un codificador de tasa 1/3, se necesita que a su entrada haya 1 bit. Por lo mismo, se configura para que a la salida de este bloque salga 1 bit por byte, si la cantidad de bits es mayor, la simulación se cancelará.

- **PCCC Encoder:** Los FSM permiten especificar las máquinas de estados que se utilizarán para codificar, en este parámetro va la ruta y el nombre del archivo con la especificación de la máquina de estados. El archivo utilizado para este parámetro se obtuvo de los archivos que sumistra GNURadio en sus librerías. El segundo parámetro es el estado inicial. Es importante resaltar que como cualquier codificador digital, sólo acepta entradas discretas. El parámetro Blocklength deben ser números $0, 1, 2, ..., X - 1$, donde X es el tamaño del alfabeto de entrada de los del fsm. La cardinalidad de la salida del PCCC es el producto de las cardinalidades de las salidas de los dos FSMs que se están utilizando.

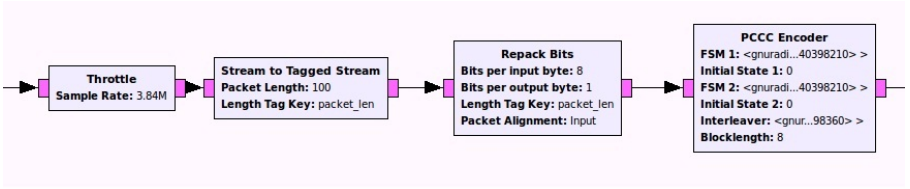

Figura 3.8: Etapa de la Codificación usando Turbo Códigos

Etapa de decodificación:

Para la decodificación de la señal, sólo se necesitan 3 bloques en GNURadio.

- **Chart toFloat:** es un cambio de variable de tipo char a tipo float para poder enviar la señal al decodificador.
- **PCCC Decoder Combo:** los FSM deben ser los mismos utilizados en el *PCCC Encoder*. Como la entrada del bloque es compleja, se debe establecer en 1 la dimensionalidad. En el parámetro Metric Type se especifica la distancia Euclideana entre el símbolo recibido y los símbolos de la constelación o una distancia Hard entre el símbolo recibido y los de la constelación. Esta distancia Hard vale 0 para el símbolo más cercano de la constelación y 1 para los restantes.
- **Repack bits:** realiza la función inversa al bloque Repack bits del codificador, para que a la salida se tenga un byte de 8 bits.

En la figura (3.9) se muestra los bloques empleados en GNURadio para la etapa de decodificación.

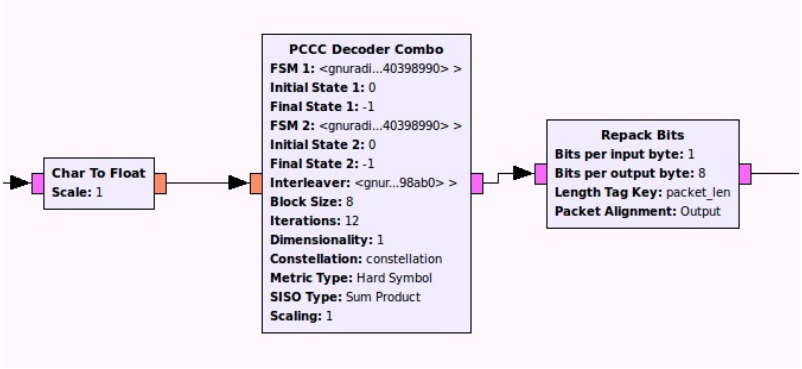

Figura 3.9: Etapa de la Decodificación empleando Turbo Códigos

3.1.2. Esquema de Modulación OFDM utilizado en el Estándar LTE.

Es importante acotar que los parámetros usados en todas las simulaciones, cumplen con lo establecido en el estándar.

Los posibles esquemas de modulación del Estándar LTE en el enlace descendente son: QPSK, 16-QAM y 64-QAM, y para el uplink: QPSK y 16-QAM, y la 64-QAM dependiendo de la capacidad del terminal móvil. Es importante resaltar que sólo se trabajará con el enlace descendente, ya que para uplink se utiliza otro tipo de modulación, específicamente SC-FDMA.

Transmisor: está compuesto básicamente por 3 etapas, las cuales de manera general son: el mapeo de símbolos, el modulador OFDM y el prefijo cíclico.

Figura 3.10: Transmisor OFDM.

Sin embargo, antes de entrar en detalles con el transmisor, se debe señalar cómo está estructurado un paquete de datos en GNURadio. Esta información será de

gran utilidad en el receptor, ya que le indicará al mismo, el tamaño de la cabecera, evitando así errores de sincronización.

Por lo tanto, un paquete consta de los siguientes elementos:

- Un preamble: se utiliza para la detección, la sincronización (en tiempo y frecuencia) y la estimación inicial del estado del canal.
- Una cabecera: es de longitud fija y almacena información sobre el paquete, como su longitud (más importante), pero potencialmente otra información, tal como el número de paquete, su destinatario etc.
- La payload: al colocarla en cero, se autoajusta la carga útil.
- Una suma de comprobación, por lo general un valor CRC, para validar el contenido del paquete.

Volviendo al transmisor, se explica a detalle cómo se plantea cada una de las etapas, utilizado los bloques disponibles en GNUradio.

Symbol Mapping: Es el mapeo o modulación digital tanto de la trama de datos como de la cabecera, que más adelante será multiplexada. Este bloque va acompañado de:

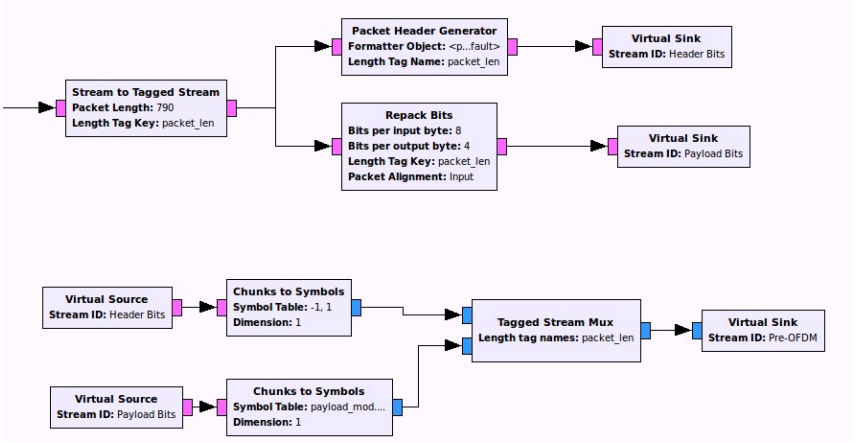

Figura 3.11: Mapeo de Símbolos.

- **Stream to Tagged Stream:** Convierte un flujo regular de datos en un flujo de datos etiquetado. Este bloque tiene como objetivo agregar etiquetas de longitud, en intervalos regulares. La longitud se asigna en Bytes. Un paquete de datos consta de N bytes. Existen bloques en GNURadio que necesitan conocer el tamaño del paquete para su funcionamiento, como es el caso del *Packet Header Generator*, *Repack Bits*, *Tagged Stream Mux*, entre otros. Si no se incluye este bloque al esquema, la simulación arrojará errores porque no hay manera de saber el límite de los paquetes.

- **Packet header generator:** A la entrada se tiene un flujo de datos etiquetados y a la salida se tiene un flujo de datos etiquetados que contiene una cabecera. Básicamente, genera una cabecera de tamaño 29 bits; los bits 0-11 contienen el tamaño de los N bytes etiquetados, los bits 12-27 tienen la cantidad bits totales de la cabecera y el bit 28 es un bit de paridad.

- **Repack bits:** Empaqueta bits de un flujo de entrada en flujo de bits de salida. Ningún bit es descartado en este bloque, el rango de bits que se puede empaquetar está en el intervalo [1-8], se lee desde el bit menos significativo y se empieza a copiar también desde el menos significativo. Si una etiqueta de datos es detectada, el bloque asume una secuencia marcada, si el número de bits relevantes a la entrada de este bloque no es múltiplo del número de bits de entrada, se procede a rellenar con ceros hasta completar. Para el caso del estándar LTE, por cada 8 bits de entrada, salen 2, 4 o 6 bits a la salida dependiendo de qué modulación se vaya a usar (QPSK, 16QAM ó 64QAM).

- **Virtual Sink:** Funciona como un nodo de finalización o sumidero, se le proporciona un nombre al bloque.

- **Virtual Source:** Funciona como una fuente, los datos de esta fuente provienen del *Virtual Sink* que contenga el mismo nombre.

- **Chunks to Symbols:** Mapea un flujo de datos de acuerdo a la constelación indicada. Para este caso, se utilizó la constelación de Grey para QPSK, 16QAM y 64QAM y la dimensión (D) tiene que ser 1.

- **Tagged Stream Mux:** Combina el flujo de datos que tiene a su entrada. Para este esquema, en la primera entrada se coloca la cabecera y en la segunda entrada la payload. A la salida se tiene un único flujo de datos, el cual está compuesto primero por la cabecera y luego por la carga útil.

OFDM Modulation: En esta etapa, se toman los símbolos escalares modulados y se transforman en vectores, los cuales serán la entrada para una transformada rápida inversa de Fourier. También se asignará la cantidad y posición de las portadoras de datos y piloto.

La trama (header + payload) es modulada por un esquema OFDM bajo el estándar LTE, el cual asignará, dependiendo del ancho de banda, la cantidad de portadoras disponibles (FFT), las portadoras que llevarán la información (OccupiedCarrier), las portadoras pilotos y como se modularán esas portadoras pilotos (PilotCarriers and Pilot symbols). Todos estos parámetros mencionados son configurados siguiendo las especificaciones del estándar LTE 2.1. Por último, se creará un preámbulo (preamble) el cual se utiliza para la detección, la sincronización y la posible estimación del canal. Este preamble viene dado por dos SyncWords cuya longitud debe ser del mismo tamaño de la FFT y su construcción viene dada por el trabajo [20].

La sincronización se basa en la búsqueda de un símbolo de entrenamiento con dos mitades idénticas en el dominio del tiempo, que permanecerán igual después de pasar a través del canal, excepto que habrá una diferencia de fase entre ellas causada por el desplazamiento de frecuencia de la portadora. Para el caso de GNU-Radio se utilizan 2 símbolos o palabras. En nomenclatura de OFDM, un símbolo se refiere a todas las sub portadoras (FFT). El primer símbolo está compuesto por dos mitades y se construyen igual (en orden de tiempo); mediante la transmisión de una secuencia de pseudoruido (PN) en las frecuencias pares, mientras que los ceros se usan en las frecuencias impares. Dicho de otra forma, en cada frecuencia par, uno de los puntos de una constelación QPSK es transmitido. Con el fin de mantener la energía de la señal aproximadamente constante para cada símbolo de los componentes de la frecuencia de este símbolo de entrenamiento, se multiplica

por un factor de $\sqrt{2}$ en el transmisor, o los cuatros puntos de la constelación QPSK se toman de una constelación mayor, como la 64 QAM, de manera que se puedan utilizar los puntos con mayor energía. Los datos transmitidos no serán confundidos con el inicio de la trama, ya que los datos reales deben contener frecuencias impares. El segundo símbolo se construye con puntos de una constelación BPSK, tanto en las frecuencias pares como en las impares. Es importante destacar que, en la construcción de las 2 palabras de sincronización, la cantidad de puntos a usar de las constelaciones es igual al número de subportadoras de datos y el resto se completan con ceros.

La etapa de la modulación OFDM se compone de 2 bloques:

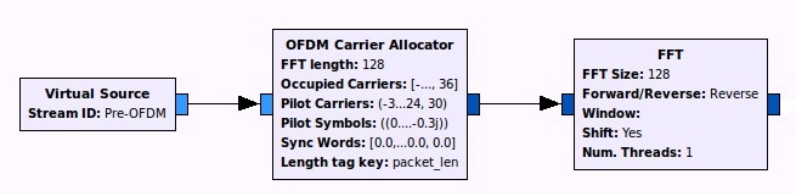

Figura 3.12: Modulación OFDM.

- **OFDM Carrier Allocator:** Realiza la modulación OFDM en el dominio de la frecuencia de valores complejos y añade portadoras pilotos. Este bloque convierte símbolos complejos modulados en vectores, los cuales son la entrada para aplicar la IFFT en un transmisor OFDM. El parámetro Pilot Symbols permite ubicar los símbolos pilotos en las portadoras. Si este parámetro no es configurado por defecto, se coloca como cero. El número de la FFT (FFT Length), la posición de las portadoras con datos a utilizar (Occupied Carriers), el lugar en que se ubican las portadoras pilotos (Pilot Carriers), los símbolos como se van a modular las portadoras pilotos (Pilot Symbols) debe coincidir con el número de portadoras pilotos. Las palabras de sincronización a usar (Sync Word) normalmente son 2, y cada una es del tamaño de la FFT. Por último se debe especificar el nombre del flujo de datos al cual se le vaya a aplicar este proceso (Lenght Tag Key).

- **FFT:** Su función es aplicar la transformada inversa de Fourier a un flujo de vectores. Para lograr esto, se tiene que especificar en el bloque la opción Reverse.

Prefijo Cíclico: Para completar la modulación se agregará la longitud del prefijo cíclico que viene dada por el estándar. En la figura (3.13) se muestra los bloques empleados en GNURadio.

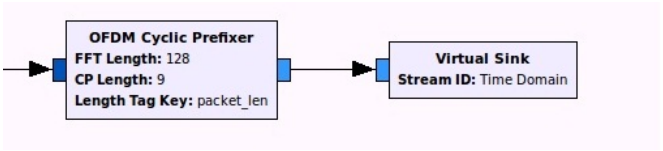

Figura 3.13: Prefijo Ciclico.

- **OFDM Cyclic Prefixer:** Añade un prefijo cíclico y realiza una conformación de impulsos de símbolos OFDM. A la entrada se tienen símbolos OFDM en el tiempo luego de aplicar la IFFT. Se puede aplicar el prefijo cíclico a un flujo de datos etiquetados. Hay que hacer notar que, la inclusión del prefijo cíclico soluciona el problema de dejar un espacio de guarda vacío entre portadoras; en su lugar, se llena el espacio de guarda con una extensión cíclica del símbolo OFDM.

Canal: La información ya modulada se transmite por un canal que simula el AWGN y de igual manera pasará por un simulador de modelo de desvanecimiento tipo Rayleigh que pondrá la trama modulada bajo los efectos de la frecuencia Doppler que puede presentar el medio, dicho simulador de desvanecimiento está modelado por el siguiente trabajo [15]. En la figura (3.14) se muestra los bloques empleados en GNURadio.

- **Channel Model:** Este bloque implementa un simulador básico de modelo de canal que puede ser utilizado para ayudar a evaluar el desempeño de un esquema de comunicación en presencia de ruido. Permitiendo al usuario ajustar la tensión de la fuente de ruido AWGN y efectos de Multitrayecto.

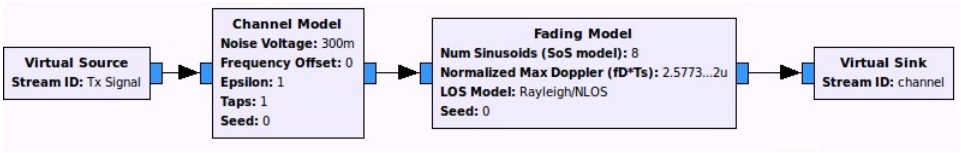

Figura 3.14: Canal.

- **Fading Model:** Este bloque implementa el método descrito en el [21] para simular los efectos de desvanecimientos Rayleigh. En el parámetro Normalized Max Doppler se introducen las frecuencias especificadas en la tabla 2.2, de acuerdo a las necesidades del usuario.

  Receptor: Para la realización del receptor, la señal a demodular requiere pasar por varias etapas, entre las cuales están, quitar el prefijo cíclico, demodular la señal OFDM y retirar los símbolos mapeados en el transmisor (demodulación digital), como lo muestra la siguiente figura (3.15).

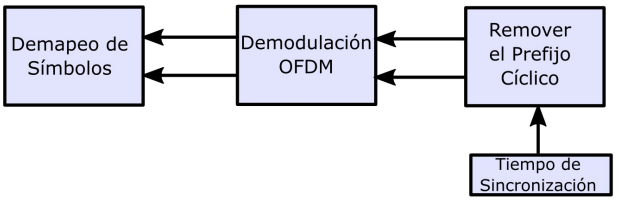

Figura 3.15: Receptor OFDM.

Recordando lo que se mencionó en (3.1.2), en el receptor es necesario también explicar en detalle el proceso de **Sincronización, detección y demultiplexación** para que se pueda detectar la trama de manera eficiente.

La demodulación se inicia con el proceso de sincronización, el cual detecta primero las Sync Word para mandar una señal de reloj al demultiplexador, que junto con la trama modulada y otra señal que indica el tamaño de la cabecera, procede a separar la cabecera de la payload. En la figura (3.16) se muestran los bloques empleados en GNURadio.

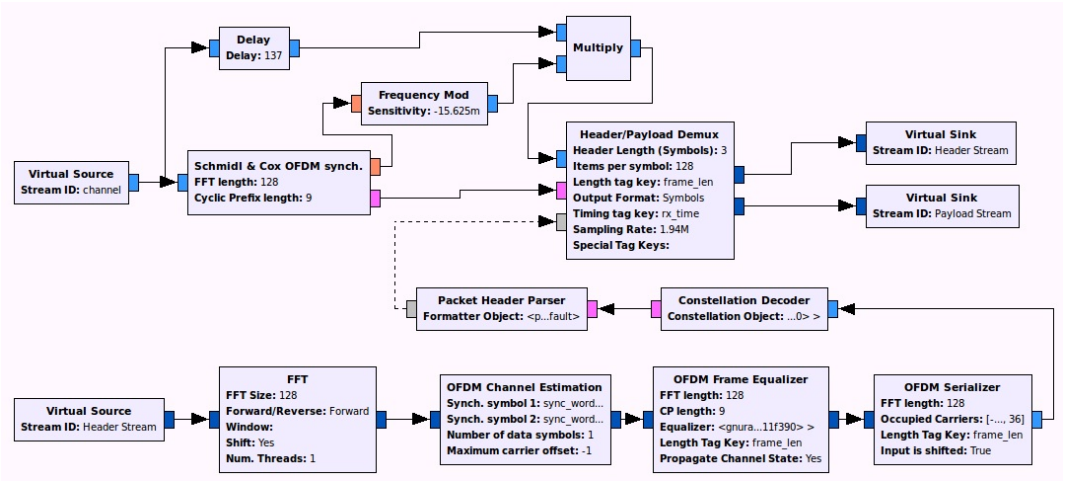

Figura 3.16: Sincronización, Detección y Demultiplexación.

- **Delay:** Retrasa la entrada cierto número de muestras. Un retraso positivo inserta ceros al principio del flujo de datos y un retraso negativo descarta muestras. Para este caso, el delay tiene que ser igual al número de la FFT, más el número de muestras de prefijo cíclico (CP). Si no se usa esta cantidad de muestras para el retraso, se producen errores de sincronización.

- **Schmidl and Cox Synchronisation for OFDM:** A la entrada se tienen muestras complejas. Y se tienen dos salidas; en la salida 0 (morado) se tiene el desplazamiento en frecuencia escalado por la duración del símbolo OFDM, a la salida 1 (naranja) se tiene el comienzo del primer símbolo OFDM sin las palabras de sincronización.

- **Frequency Mod:** Multiplica señales, desplaza en frecuencia.

- **Header/Payload Demux:** Separa la cabecera y la paylaoad del flujo modulado. En la primera entrada (azul) se recibe la señal modulada en flujo de muestras de datos que provienen del dispositivo receptor. Este bloque descarta todas las muestras hasta que llegue la señal del trigger, que es la segunda entrada (morado). Cuando la señal del trigger es detectada, el demultiplezador copia la header y el preambler en la primera salida (header). El bloque entonces espera hasta recibir un mensaje en la tercera entrada (Gris). Este

mensaje debe ser del tipo PMT dictionary. Luego la payload se copia en la segunda salida (payload).

Demodulación OFDM:

Luego de que la señal es separada, se aplica el proceso inverso a la modulación. Para esta parte hay que demodular la cabecera y la trama por separado. En la figura (3.17) se muestran los bloques empleados en GNURadio para lograr esto.

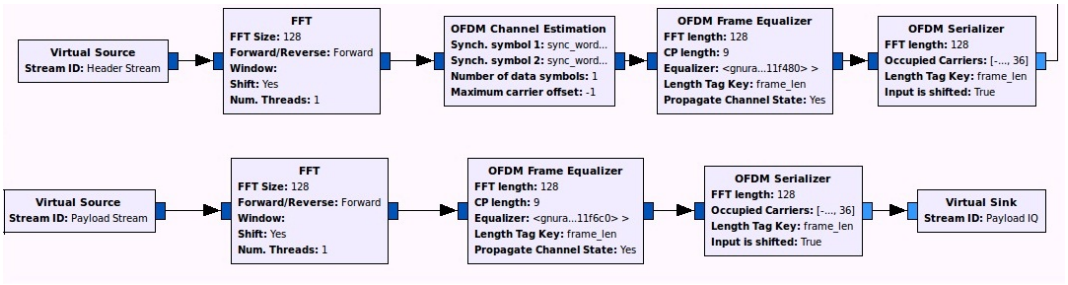

Figura 3.17: Demodulacion OFDM.

- **OFDM Channel Stimation:** Estima el canal y la frecuencia de desplazamiento. A la entrada se tienen símbolos OFDM (en el dominio de la frecuencia). El primero, o los primeros dos símbolos son esperados para la sincronización de símbolos, los cuales son usados para estimar la frecuencia de desplazamiento. A la salida se tiene el flujo de bits sin la palabra de sincronización.

- **OFDM Frame Equalizer:** Elimina la frecuencia de desplazamiento. Realiza la ecualización de una trama OFDM etiquetada. A la entrada de este bloque se conecta un flujo de símbolos OFDM etiquetados, a la salida se tiene el mismo flujo de entrada pero ecualizado y con la frecuencia corregida. La etiqueta que contiene el desplazamiento en frecuencia no se elimina.

- **OFDM Serializer:** Realiza la operación inversa al *OFDM CarrierAllocator*, remueve los símbolos pilotos a su salida como un flujo de datos etiquetado. Se puede especificar dos diferentes etiquetas para este montaje, una para la cabecera o la payload, o una para las dos al mismo tiempo.

- **HConstellation Decoder:** Decodifica puntos complejos de una constelación de una trama de bits, basado en el mapeo de Grey.
- **Packet Header Parser:** Es el bloque inverso al *Packet Header Generator*. La diferencia es que la cabecera se analiza como un diccionario y no como un flujo de bits.

Symbol Demapping: Se demodula la trama de bits de información. En la figura (3.18) se muestra los bloques empleados en GNURadio.

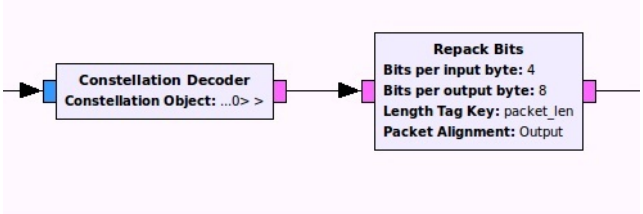

Figura 3.18: Symbol Demapping.

- **Constellation Decoder:** Decodifica puntos complejos de una constelación de una trama de bits, basado en el mapeo de Grey.
- **Unpack K bits:** Se encarga de empaquetar K bits (aunque el nombre confunda). Para este montaje se usa este empaquetador de bits para que vayan pasando de 8 bits en 8 bits, tanto para la señal original, como para la señal demodulada, con el fin de que entren al mismo tiempo al bloque para el cálculo del BER. Si no se usa este bloque, las señales entran al bloque del cálculo del BER de manera desfasada.Por ejemplo, si entran dos bytes y k = 8 Entrada = [0xf5, 0x08] Salida = [1,1,1,1, 0,1,0,1, 0,0,0,0, 1,0,0,0].

3.1.3. Simulación del Esquema de Comunicación Convencional en el estándar LTE

Se tomaron los 2 esquemas de codificación de canal (Convolucional y Turbo códigos) y se realizaron las simulaciones correspondientes a las especificaciones del

estándar LTE. Ésto, con la finalidad de comparar posteriormente su rendimiento con el obtenido al emplear la codificación caótica de canal en dicho estándar.

Para ésta etapa, se realizaron las simulaciones para los anchos de banda de transmisión 1.25, 2.5 y 5 MHz. En cuanto a la modulación digital, se tomaron las modulaciones utilizadas por el estándar (QPSK, 16 QAM Y 64 QAM), así mismo se consideró un canal de ruido Gaussiano y la dispersión de multitrayecto en la frecuencia de 5Hz que simula el efecto Doppler, para el estudio del desempeño del esquema de comunicación.

3.2. Etapa 2: Creación de los Bloques Caóticos y Simulación de un Esquema de Codificación Caótica de Canal en el estándar LTE.

Para poder realizar esta etapa, fue necesario aprender los lenguajes de programación que utiliza la plataforma GNURadio, los cuales son C++ y Python. Una vez familiarizados con dichos lenguajes, se desarrollaron los códigos que dieron lugar a la elaboración de los nuevos bloques necesarios para la simulación y estudio de un esquema de codificación caótica de canal.

El esquema de codificación caótica de canal fue desarrollado en 3 secciones muy importantes: Aprendizaje, Codificación Caótica y Decodificación Caótica.

Una vez creados los bloques de codificación de canal caótico, se evaluó el desempeño que tienen los mismos en el estándar LTE en cuanto al BER.

3.2.1. Proceso de creación de los bloques Caóticos

A continuación, se describirá el proceso desarrollado para la creación de los bloques codificador caótico de canal y decodificador caótico de canal en GNURadio, necesarios para la realización de las simulaciones y la posterior comparación con los esquemas de codificación de canal convolucional y con Turbo códigos.

Resulta importante señalar que para que dichos bloques puedan ser creados correctamente, es necesaria la generación de diversas carpetas y archivos que necesitan ser escritos con una sintaxis específica. Es por esto, que para usuarios que están empezando en este ámbito, este proceso puede resultar bastante complicado. Sin embargo, existe una herramienta llamada gr modtool que simplifica significativamente este proceso creando los archivos necesarios para el óptimo funcionamiento de los bloques.

Inicialmente, se ejecutó el comando `gr_modtool newmod Caotico` para añadir el módulo de nombre *Caotico*. Es aquí donde podremos ubicar los dos bloques creados.

Luego de creado el módulo, se procedió a acceder a la carpeta correspondiente al modulo creado, la cual tiene por nombre *gr-Caotico*, utilizando el comando *cd gr-Caotico*. Posteriormente se utilizó el comando `gr_modtool add Codificador if` para crear el bloque codificador caótico de canal. El sufijo *if* en el último comando ejecutado, define los tipos de variables que utiliza el bloque a la entrada y a la salida; integer (entero) a la entrada y float (flotante) a la salida.

Una vez emitido este último comando, se nos preguntó el tipo de bloque que se requería. Este parámetro define las características que tiene un bloque en cuanto a la relación *entrada-salida*. Para nuestro propósito, se especificó que el bloque requerido era de tipo general.

De igual manera, se nos solicitó que introdujéramos los parámetros necesarios para controlar el procesamiento del bloque de acuerdo a nuestras exigencias. En nuestro caso, los parámetros utilizados fueron *Tamaño de símbolo* y *Tamaño de entrada* , que especifican el tamaño de las secuencias de los símbolos y el tamaño del vector de entrada, respectivamente. Dichos parámetros podrán ser variados por el usuario por medio de la interfaz del bloque, dependiendo de sus requerimientos al momento de realizar una simulación.

En este punto, la herramienta gr modtool generó, entre otros, 3 archivos, cuya correcta edición resulta de vital importancia para el correcto funcionamiento de

nuestro bloque. Los archivos generados son: `Caotico_Codificado_if.xml`, `Codificador_if_impl.cc` y `Codificador_if_impl.h`. A continuación, se realizará una breve descripción del propósito de cada uno de ellos.

- En el archivo `Caotico_Codificador_if.xml`, es en donde se especificó el nombre del bloque (mediante el cual se podrá localizar en la librería de GNURadio); los parámetros de entrada definidos en el párrafo anterior, junto con su tipo de variable; y además, el tipo de variable que se manejará en la entrada y la salida del bloque (en este caso, integer y float respectivamente).
- El archivo `Codificador_if_impl.cc`, resulta ser el más importante, por lo tanto se explicará de manera más detallada.
    - En la primera parte se tiene lo que se denomina el constructor. Aquí se definió la cantidad de puertos de entrada y salida del bloque, así como el tamaño que maneja cada una. Además, en esta parte se declararon las variables que dependen de los parámetros de entrada del bloque, para que puedan ser utilizados en el código de C++ y en las funciones que se mencionarán más adelante. De igual manera, es en esta parte donde se hizo uso de los argumentos `set_output_multiple()` y `set_relative_rate()`, cuya función es especificar la cantidad de ítems a la salida y la relación entre ítems de salida e ítems de entrada, respectivamente.
    - Luego se tiene lo que se denomina el destructor, que consta de 2 funciones; la función *forecast* y la función *general work*:
        - La función *forecast* es nuestra manera de establecer el número de elementos a la entrada necesarios para producir un elemento a la salida. El argumento utilizado aquí se denota `ninput_items_required()`.
        - La función *general work* es la que juega el papel central en la creación de un nuevo bloque. Es donde se lleva a cabo el proceso de convertir un stream de entrada, en un stream de salida. El código de C++ responsable del procesamiento de la señal dentro del bloque, además del ajuste de los argumentos "`consume_each()`" y "`return()`",

pertenecen a esta función. El argumento "`consume_each()`"le dice al sistema, cuántos ítems se han consumido del stream de entrada y el argumento "`return()`"le dice al sistema cuantos ítems de salida se han producido.

- Por último, se editó el archivo "`Codificador_if_impl.h`", en donde se declararon las variables que dependen de los parámetros de entrada para que puedan ser usadas en el código correspondiente al procesamiento de señal, en el archivo "`Codificador_if_impl.cc`".

En cuanto al bloque decodificador caótico, el proceso realizado fue similar, con la obvia diferencia en cuanto a los nombres de los 3 archivos generados y su respectiva edición; cada uno utiliza diferentes valores fijados en los argumentos del archivo .cc.

3.2.2. Simulación del Esquema de Codificación caótica de canal

En esta sección se realizaron las simulaciones del esquema de codificación caótica de canal para los siguientes tres valores de n (tamaño de símbolo): 2,3 y 6. Así mismo, se simuló para las frecuencias de muestreo 1.92, 3.84 y 7.68 MHz, correspondientes a los anchos de banda de 1.25, 2.5 y 5 MHz del estándar LTE.

De los bloques utilizados para realizar ésta simulación, sólo 3 no han sido presentados. Se procederá a describirlos a continuación.

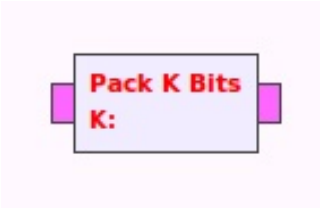

Figura 3.19: Pack K Bits.

Pack k bits: El objetivo de éste bloque en el presente esquema, es tomar 1 bit a la vez para que vayan ingresando de esta manera al codificador caótico.

El siguiente paso será presentar los bloques creados que permitirán realizar las simulaciones correspondientes a esta etapa.

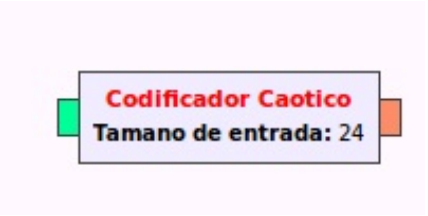

Figura 3.20: Codificador Caótico.

Codificador Caotico: Como se mencionó anteriormente, los parámetros de entrada de este bloque son: "`Tamaño de simbolo`", y "`Tamaño de entrada`". El primero, corresponde al tamaño de las secuencias de símbolos que se van a ir generando, cuyo valor correspondiente de Zmean, es lo que ira a salida del codificador. Es decir, que si el usuario fija este valor en 3, los posibles valores que se van a transmitir dependiendo de la secuencia de n-símbolos que se vayan generando, serán los que presentaron en la sección (2.2.4.2). El segundo parámetro de entrada va a depender, en este caso, del tamaño del vector a la entrada.

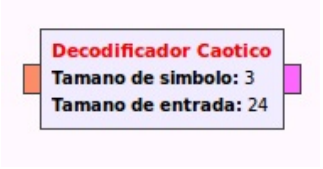

Figura 3.21: Decodificador Caótico.

Decodificador Caótico: Los valores de los parámetros seteados en este bloque, son y representan lo mismo que en el codificador caótico.

Una vez programados los módulos que definen la codificación de canal caótica, se realizaron las conexiones necesarias con la finalidad de simular el esquema de codificación caótico y así poder evaluar el comportamiento que presenta el mismo en cuanto a BER.

El esquema de codificación caótica de canal completo, se puede visualizar en la figura (3.22).

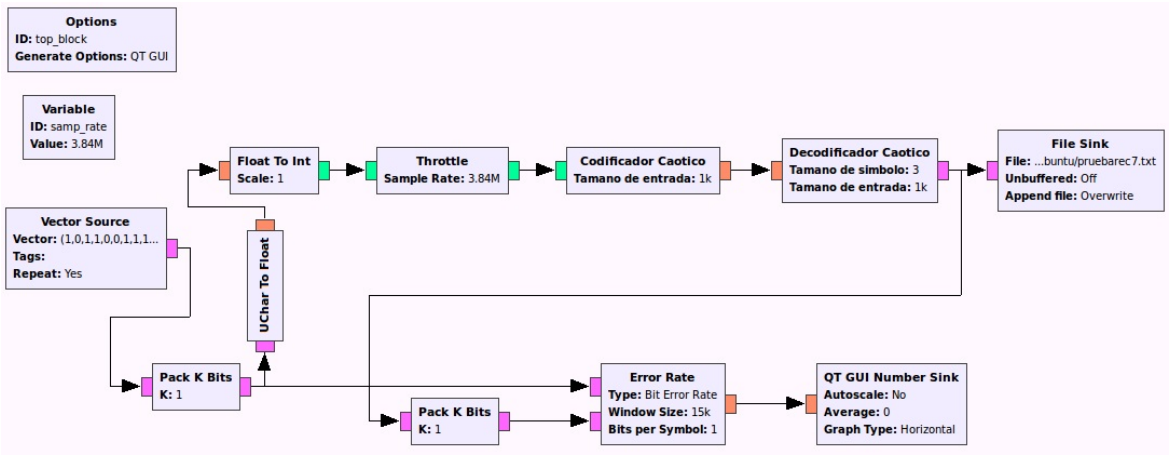

Figura 3.22: Esquema de Codificación Caótica de Canal.

A continuación se describirá lo que sucede en el esquema anterior, no sin antes aclarar, que no se transmiten sólo 24 bits; el vector va repitiéndose hasta que el usuario decida detener la simulación. En este caso, se decidió tomar un File Sink en el receptor para tener una idea de cuántos bits son transmitidos.

- En este caso, el vector de entrada es de tamaño 24 y el tipo de variable de salida es byte. De aquí se ve la necesidad de usar el bloque Pack K Bits; los elementos del vector de entrada se van a ir separando de a 1 bit al pasar por este bloque para luego ir entrando al codificador. Si no se usara este bloque, cada elemento a la entrada del codificador, entraría como su equivalente a 8 bits. Por ejemplo, si el elemento es 1, entraría la secuencia 00000001 al codificador.

- En el codificador, se irán tomando de 24 en 24 bits. Con los primeros 24 bits que entren, se generará una secuencia de símbolos de tamaño 25, producto de la codificación diferencial. Luego, esa secuencia se va a ir dividiedo de 3 en 3 (en caso de que el tamaño de símbolo sea 3) de acuerdo al procedimiento explicado en el Capítulo 2, para finalmente, comparar cada una de las secuencias generadas, con las 2^n secuencias almacenadas en el bloque y así decidir qué valores de Zmean van a ser transmitidos.

- Una vez que se tienen todos los valores de Zmean correspondientes a cada secuencia de símbolos generada, estos son transmitidos al receptor. Este último calcula las perturbaciones que se necesitan aplicar al sistema de Lorenz a partir de los valores de Zmean que recibe.
- Luego de que el receptor aplica las perturbaciones al sistema de Lorenz, observa la evolución de la variable X(t); cuando se tienen picos positivos, se genera el símbolo **B** y si se tienen picos negativos, se genera el **A**.
- Por último, el receptor aplica la decodificación diferencial, y como en este caso no hay canal de transmisión que genere ruido, la secuencia de bits a la salida coincide con la secuencia de bits que se tenía a la entrada (proveniente del vector de entrada).

3.2.3. Simulación de un Esquema de Comunicación con Codificación Caótica de Canal en el estándar LTE.

Después de realizar la codificación caótica de canal, fue necesario evaluar el comportamiento que tiene la misma en el estándar LTE. Agregándole a éste esquema, la etapa de la modulación OFDM que se presentó anteriormente en la sección 3.1.2.

Para este caso se tomaron las mismas consideraciones en cuanto a las especificaciones del estándar LTE que se tomaron para las simulaciones realizadas de la codificación de canal convolucional y con Turbo códigos en dicho estándar, en cuanto a ancho de banda y modulación digital.

3.3. Etapa 3: Comparación entre los esquemas convencionales de Codificación y Decodificación de Canal, con el esquema diseñado, en el estándar LTE.

Luego de simulados ambos esquemas de comunicación en el estándar LTE, se realizó una comparación para así poder determinar cuál esquema presenta un mejor comportamiento en cuanto BER y a consumo de ancho de banda. Ésta etapa fue desarrollada cuando ya se tuvieron ambos esquemas caracterizados.

Para tal fin, se realizaron gráficas de BER vs SNR para cada ancho de banda y para cada esquema de modulación digital, de manera de visualizar el comportamiento de la codificación de canal convolucional y de la codificacion caótica de canal para los 3 valores de n, en una sóla gráfica. Lo mismo se hizo para realizar las comparaciones entre el comportamiento de los Turbo códigos y de la codificación caótica de canal para los 3 valores de n considerados.

Capítulo IV

Análisis, interpretación y presentación de los resultados

4.1. Etapa 1: Simulación de los Esquemas de Codificación de Canal utilizados en el Estándar LTE y del Esquema de Codificación Caótica de Canal.

- Inicialmente, se evaluó el comportamiento tanto de los esquemas de codificación de canal utilizados en el estándar LTE, como de los bloques caóticos creados en GNURadio. Para el caso del Codificador convolucional y de los Turbo códigos, se utilizaron los bloques disponibles en la librería de GNURadio, tomando en cuenta los valores de fsm establecidos por el estándar, lográndose recuperar la señal original en el receptor sin errores, resultado que era el esperado, ya que para ninguno de los esquemas de codificación se consideró el uso de canal que incorporara ruido. Así mismo, se comprobó para todos los esquemas evaluados, que ninguna de las frecuencias de muestreo establecidas por dicho estándar afectan el resultado; se recibió perfectamente la señal para los 4 valores planteados.
- Al incorporar los bloques descritos en la sección 3.1.2 correspondientes al esquema de modulación especificado por el estándar LTE, se obtuvieron los

resultados esperados para los esquemas estudiados, es decir, un aumento de la cantidad de bits erróneos recibidos a medida que se disminuía el nivel de SNR (dB).

Con las figuras 4.1 y 4.2 se puede verificar lo mencionado anteriormente. Ya que ambos resultados se obtuvieron utilizando el mismo esquema de modulación y de codificación caótica de canal.

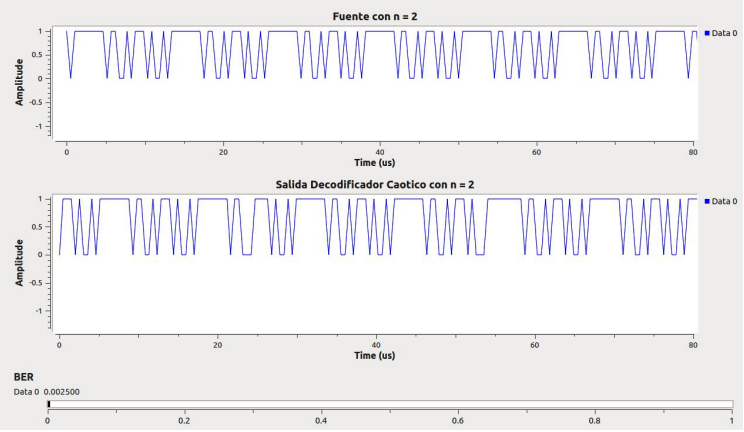

Figura 4.1: Comparación entre la señal proveniente de la fuente y la señal obtenida a la salida del decodificador caótico con Esquema de Modulación de 16 QAM y un Ancho de Banda de 1.25 MHz cuando se tiene un SNR de 16.7 dB.

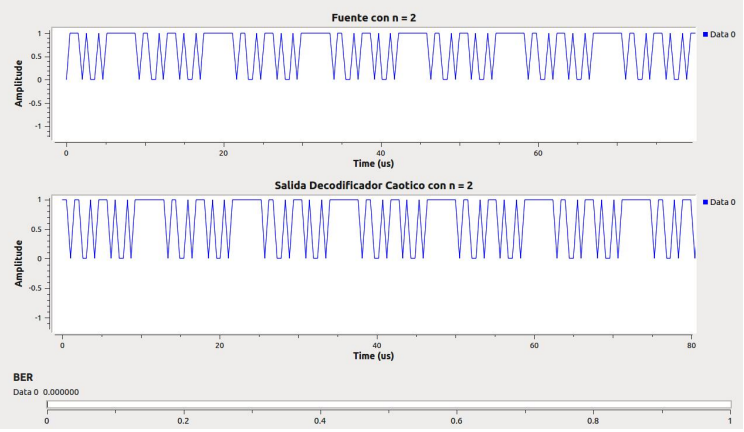

Figura 4.2: Comparación entre la señal proveniente de la fuente y la señal obtenida a la salida del decodificador caótico con Esquema de Modulación de 16 QAM y un Ancho de Banda de 1.25 MHz cuando se tiene un SNR de 19.86 dB.

Es importante mencionar que la figura 4.2 no presenta errores en la recepción, sin embargo, existe un desfasaje de 0.000005 s. debido a la cantidad de bloques empleados para poder recuperar la señal original.

4.2. Etapa 2: Comparación entre el Esquema de Comunicación con Codificación de canal Convolucional y los Turbo Códigos, con el Esquema de Comunicación con Codificación Caótica de Canal, en el estándar LTE.

4.2.1. Consumo de Ancho de banda

En cuanto al consumo de ancho de banda, éste método de codificación caótica de canal basado en el sistema de Lorenz presenta ventajas con respecto al esquema de codificación de canal convolucional y a los Turbo códigos (cuando se utiliza la tasa de de codificación fijada por el estándar LTE). Lo anterior puede comprobarse al analizar, en primer lugar, la característica que presenta la relación entrada−salida del bloque codificador caótico creado. Dicho bloque acepta variables del mismo tamaño a su entrada y a su salida (entero y flotante, respectivamente), es decir 32 bits. Por lo tanto, recordando que para este caso las simulaciones fueron realizadas tomando como fuente un vector de 24 elementos, se tienen $24*32\mathtt{bits}$ a la entrada del codificador caótico y a su salida, $(24+\mathtt{n})*32$ bits, es decir, la relación conseguida es $\frac{24}{24+\mathtt{n}}$. Ahora, recordando que en el estándar LTE, la tasa de codificación de canal es $\frac{1}{3}$, se comprueba que cualquiera de los casos de codificación caótica presenta menos consumo de ancho de banda que el que presenta los esquemas de codificación de canal convolucional y los Turbo códigos.

4.2.2. Carga computacional

Con el uso del codificador caótico creado se pudieron obtener los resultados de las simulaciones de manera más rápida que con el uso de los Turbo ódigos, presentando los primeros, una duración de 1 minuto y medio aproximadamente, mientras que el tiempo de simulación al utilizar Turbo códigos fué, en promedio, de 9 minutos. Por otra parte, el tiempo de simulación obtenido con el uso de los bloques correspondientes a la codificación convolucional, fúe de sólo 1 minuto, siendo ésta entonces, la que consume menos carga computacional. Las simulaciones realizadas en esta investigación se realizaron con una PC con procesador AMD Phenom(tm) II X6 1055T de 2.8GHz y memoria RAM 4 GB, esta información es de interés ya que los tiempos en las simulaciones varían dependiendo de la capacidad de procesamiento que tenga la máquina.

En este punto resulta importante destacar, que existe una manera de reducir aún más la duración de las simulaciones al emplear los bloques caóticos creados. Esto puede lograrse prescindiendo de el sistema cáotico de Lorenz en el codificador, almacenando sólo los valores de Zmean obtenidos del proceso de aprendizaje y sus respectivas secuencias.

4.2.3. BER Vs SNR(dB)

4.2.3.1. Ensayo de verificación para el cálculo del SNR(dB)

Para ésta sección, inicialmente se procedió a realizar una gráfica de BER Vs. SNR(dB) de un esquema de modulación digital sencillo, como lo es el de QPSK, con el objetivo de validar el procedimiento utilizado para el cálculo de SNR(dB) utilizando bloques disponibles en la librería de GNURadio.

En la figura 4.3 se observa la gráfica de BER Vs. SNR(dB) realizada a partir de los valores obtenidos en GNURadio.

Al comparar dicha gráfica con la mostrada en la figura 4.4 [22], se aprecia que la técnica utilizada para el cáculo de SNR(dB), es correcta.

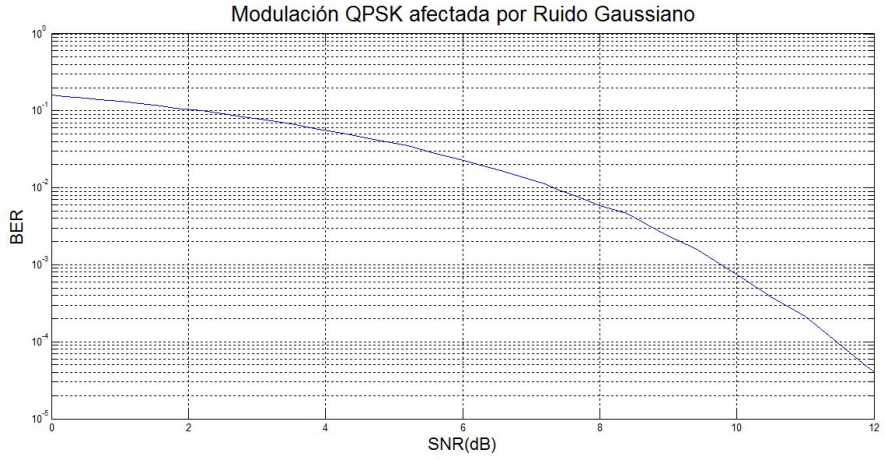

Figura 4.3: Gráfica de BER vs SNR(dB) del Esquema de Modulación de QPSK en ambiente gaussiano, obtenida a partir de GNURadio.

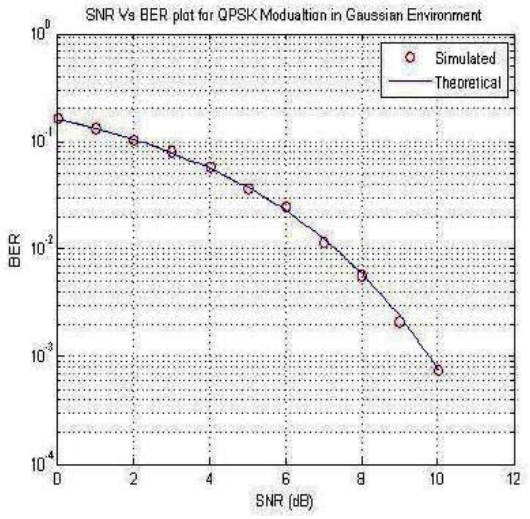

Figura 4.4: Gráfica de BER vs SNR(dB) con Esquema de Modulación de QPSK en ambiente gaussiano.

4.2.3.2. Turbo Códigos Vs. Codificación Convolucional

Después de realizar las simulaciones para los anchos de banda de 1.25, 2.5 y 5 MHz, se obtuvo que los Turbo códigos presentaron un mejor desempeño que el obtenido con el esquema de codificación convolucional, como se muestra en la

figura 4.5. Recordando que los Turbo códigos son los que más se acercan al límite de Shannon.

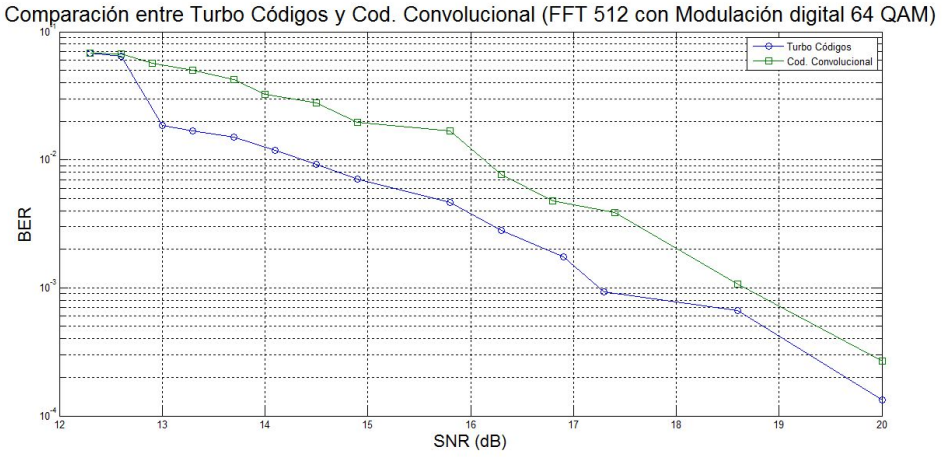

Figura 4.5: Comparación entre Turbo Códigos y Codificación Convolucional con una FFT de 512 y Modulación digital de 64 QAM.

4.2.3.3. Codificación Convolucional Vs. Codificación Caótica

De todas las gráficas de BER Vs. SNR(dB) obtenidas, la que mejor desempeño mostró para la codificación caótica de canal, con respecto a la codificación convolucional, fue la resultante del estudio en el ancho de banda de 5 MHz (Tamaño de FFT de 512) al utilizar una modulación digital de 64 QAM.

A partir de la gráfica correspondiente a la figura 4.6, se tiene que para el rango de SNR comprendido entre 16 y 20 dB, aproximadamente, la codificación convolucional permite la transmisión de la información con una mayor efectividad en cuanto a la cantidad de errores obtenidos a causa del canal de transmisión, que los obtenidos haciendo uso de la codificación caótica. Sin embargo, al ir disminuyendo el nivel del SNR a partir del rango mencionado, se tiene que para los 3 valores de n, la codificación caótica presenta un mejor comportamiento que la codificación convolucional.

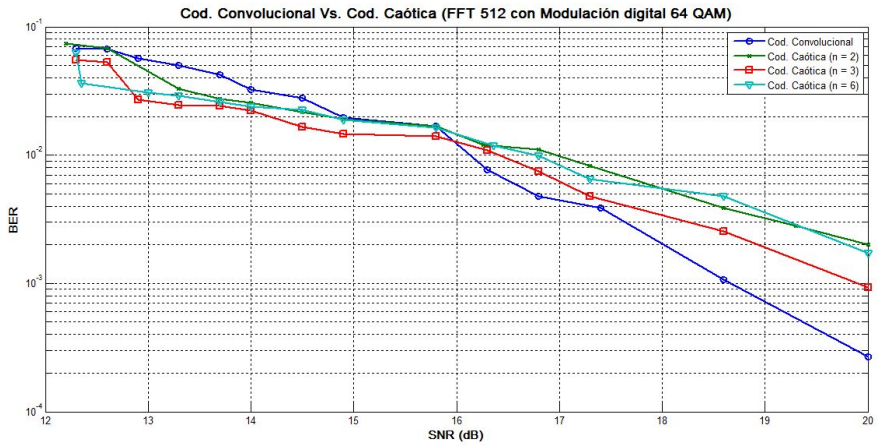

Figura 4.6: Comparación entre Codificación caótica de canal y Codificación Convolucional con FFT de 512 y Modulación digital de 64 QAM.

4.2.3.4. Turbo Códigos Vs. Codificación Caótica

Como ya se mencionó en la sección anterior, el caso en el que se obtuvo un mejor comportamiento de la codificación caótica de canal, fue en el ancho de banda de 5 MHz. De la gráfica correspondiente a la figura 4.7, se observa que cuando se utilizó la modulación digital de 64 QAM, la codificación caótica de canal, para n = 3 y 6, presentó menos errores en la comunicación que los Turbo códigos en el rango de 12 a 12.6 dB. A partir de dicho rango, los Turbo códigos tuvieron un mejor comportamiento en cuanto a BER.

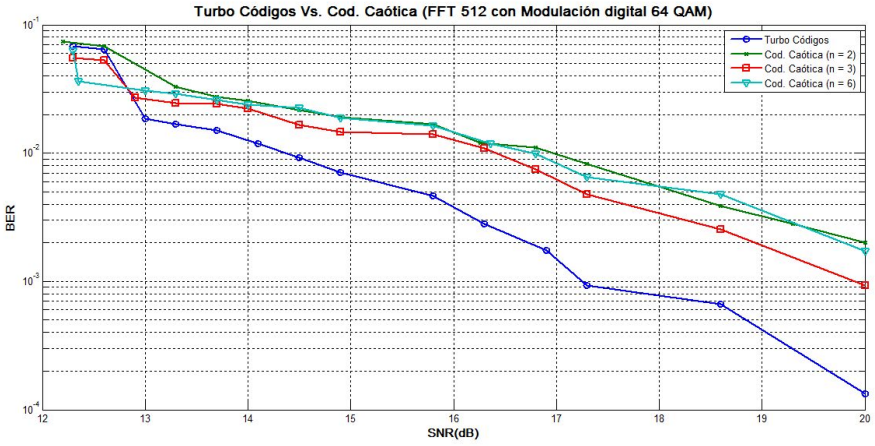

Figura 4.7: Comparación entre Codificación caótica de canal y Turbo Códigos con FFT de 512 y Modulación digital de 64 QAM.

Capítulo V

Conclusiones y recomendaciones

5.1. Conclusiones

- La finalidad de esta investigación, fue brindar un estudio comparativo entre la codificación de canal utilizada convencionalmente por el estándar LTE y la aplicación del desarrollo de nuevas tecnologías de codificación de canal basadas en la tería del caos, cuyo rendimiento no había sido estudiado en el estándar LTE hasta el momento.
- Aunque con el método de codificación de canal basado en la teoría del caos se obtuvo puntos en los cuales presentó un mejor comportamiento en cuanto a BER que los Turbo códigos, éste último resultó, en general, menos suceptible a errores durante la transmisión para todos los anchos de banda y esquemas de modulación digital considerados.
- El uso del esquema de codificación caótica de canal (utilizando el método explicado) basado en el sistema de Lorenz, permite un aprovechamiento más eficiente del ancho de banda que el que se tiene con el uso de los Turbo códigos y la codificación de canal convolucional.
- Aún cuando de todas las pruebas realizadas, el tiempo de simulación resultó menor al utilizar los bloques caóticos creados, que al utilizar los bloques

correspondientes a la codificación de canal con Turbo códigos, es posible optimizar el código desarrollado en el bloque codificador caótico con el fin de reducir la carga del CPU y de obtener los resultados en un período menor de tiempo. Esto se hace prescindiendo del sistema caótico de Lorenz, almacenando sólo las 2^n secuencias posibles (para los tres valores de n), junto con los valores de Zmean que las generan.

- Aunque el objetivo de este método de codificación de canal basado en la teoría del caos, no es el de brindar seguridad a la transmisión, el hecho de transmitir sólo los valores de las perturbaciones que deben ser aplicadas al sistema en el receptor (en lugar de las señales caóticas) para posibilitar la recuperación de la información, resulta ventajoso. Lo anterior se debe a que el receptor necesita conocer los parámetros exactos del sistema, así como las condiciones iniciales consideradas por el transmisor, para poder reconstruir la señal caótica que contiene el mensaje.

- A pesar de que no se hayan conseguido resultados favorables en cuanto a la presencia de errores en la transmisón con el uso de este método de codificación caótica de canal en particular, no quiere decir que se deba descartar por completo la aplicación de la teoría del caos para este fin. De hecho, a partir de los resultados obtenidos, podemos afirmar que es posible implementar caos matemático como método pseudoaleatorio para codificadores de canal en estándares de comunicaciones actuales. Queda de parte de futuros estudios ó simulaciones, encontrar un método cuyos resultados permitan establecer un ambiente ajustado a su desempeño.

5.2. Recomendaciones

Realizar estudios comparativos con otros esquemas de codificación caótica, que puedan competir con los esquemas de codificación de canal usados actualmente por el estándar LTE, en términos de BER. Así mismo, emplear otras herramientas computacionales para dichas comparaciones con el fin de obtener resultados para un rango mayor de SNR.

Estudiar la teoría del caos en el diseño de permutadores pseudoaleatorios en los Turbo códigos.

Estudiar las principales técnicas de diseño de codificadores de canal para estándares de comunicaciones inalámbricas de última generación.

Apéndice A

Códigos desrrollados en esta Investigación

Los códigos desarrollados en esta investigación se encuentran almacenados en el CD del proyecto.

Referencias Bibliográficas

[1] Jorge Ortín Gracia. Cuaderno red de cátedras telefónica. 2012.

[2] Christopher Wong Matos. Análisis y diseño de una red 3gpp lte en el departamento de cusco. 2011.

[3] Eduardo Barbará Morales, Oscar E Rodríguez Ramírez, and Emiliano Alba Blanco. Algoritmo de codificación de señales electrocardiográficas mediante el modelo caótico de lorenz.

[4] Brian Chen and Gregory W Wornell. Analog error-correcting codes based on chaotic dynamical systems. *Communications, IEEE Transactions on*, 46(7):881–890, 1998.

[5] Inés P Mariño, Luis López, and Miguel AF Sanjuán. Channel coding in communications using chaos. *Physics Letters A*, 295(4):185–191, 2002.

[6] Fernando Corteggiano, Marcelo Gioda, Esteban Carranza, M Luciana Medina, Diego Bazán, and Gerardo Di Claudio. Simulación y análisis del algoritmo de codificación turbo en sistemas de comunicaciones móviles: un estudio comparativo. In *XI Congreso Argentino de Ciencias de la Computación*, 2005.

[7] Ramon Agusti C. Lte: Nuevas tendencias en comunicaciones móviles. 2010.

[8] Harri Holma and Antti Toskala. *LTE for UMTS-OFDMA and SC-FDMA based radio access*. John Wiley & Sons, 2009.

[9] Jesús Raúl Carballo Guzmán, Miguel Ángel Contreras Silva, et al. Diseño de propuesta para la integración del sistema umts con el sistema lte avanzado e ims en las operadoras de servicio móvil celular en venezuela. 2013.

[10] Telesystem Innovations Inc. Lte in a nutshell: The physical layer. 2010.

[11] Jeanette Wannstrom. Lte-advanced. *Third Generation Partnership Project (3GPP)*, 2012.

[12] Juan Valera Requena. Diseño y desarrollo del interleaver para el decodificador turbo código wimax. 2008.

[13] Network TSGRANGRA. Evolved universal terrestrial radio access (e-utra); multiplexing and channel coding. *3rd Generation Partnership Project (3GPP), vol. TS*, 36, 2014.

[14] Telesystem Innovations. Lte in a nutshell. *White paper*, 2010.

[15] Víctor Manuel Quintero Flórez, Jameson Samir Zúñiga Muñoz, and Fernando Darío Córdoba Muñoz. Análisis del desempeño a nivel físico del enlace de bajada de la evolución a largo término (lte). *Gerencia Tecnológica Informática*, 12 (34):75–90, 2013.

[16] R Cisneros, N Escamilla, and D Plascencia. Sistemas caóticos aplicados en telecomunicaciones. *Instituto Politécnico Nacional, Escuela Superior de Ingeniería Mecánica y Eléctrica, México, Proyecto de Investigación SIP*, 200091457, 2010.

[17] Alicia D'Anjou D'Anjou, C Sarasola, and Francisco Javier Torrealdea Folgado. Caos determinista. *Sigma: revista de matemáticas= matematika aldizkaria*, (26): 149–161, 2005.

[18] Joan Josep Solaz-Portolés, Departament de Didàctica de les Ciències, and Experimentals i Socials. Un poco más sobre la teoría del caos y los fractales. *AÑO 16, NÚMERO 62, OCTUBRE—DICIEMBRE 2011 EN ESTE NÚMERO*, page 1988.

[19] Grupo de Fisica No Lineal and Inés Pérez Mariño. Synchronization and control of chaotic systems. spatio-temporal structures and applications to communications. 1999.

[20] Timothy M. Schmidl and Donald C. Cox. *Robust Frequency and Timing Synchronization for OFDM*.

[21] Amirhossein Alimohammad, Saeed Fouladi Fard, Bruce F Cockburn, and Christian Schlegel. Compact rayleigh and rician fading simulator based on random walk processes. *Communications, IET*, 3(8):1333–1342, 2009.

[22] Anuj Sharma2 Lokendra Singh1. Compressive modulation in digital communication using qpsk. *International Journal of Engineering Trends and Technology (IJETT)*, 2014.