UNIVERSIDAD NACIONAL DEL CALLAO
ESCUELA DE POSGRADO

SECCIÓN DE POSGRADO DE LA FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

RENDIMIENTO DE LOS TURBOS CÓDIGOS CON DIFERENTES CONFIGURACIONES

TESIS PARA OPTAR EL GRADO ACADÉMICO DE MAESTRO EN CIENCIAS DE LA ELECTRÓNICA

MENCIÓN: TELECOMUNICACIONES

AUTOR : WILLIAMS FERNANDO ACOSTA SOLORZANO

CALLAO – PERU
2011

WILLIAMS FERNANDO ACOSTA SOLORZANO
INGENIERO ELECTRÓNICO
Reg. del Colegio de Ingenieros N° 92077

Héctor TRIGOSO Medina
00883645
Asesor

Id. Ejemplar: 39212

JURADO SUSTENTACIÓN DE TESIS DE MAESTRÍA

DR. JUAN HERBER GRADOS GAMARRA	Presidente
DR. MARCELO NEMESIO DAMAS NIÑO	Secretario
MG. FRANCO IVAN VELIZ LIZARRAGA	Miembro
MG. NICANOR RAŬL BENITES SARAVIA	Miembro
MG. HĘCTOR ARISTIDES TRIGOSO MEDINA	Asesor

Nº DE LIBRO 01

Nº DE ACTA : 11

FECHA : julio 20, 2011

RESOLUCION DE LA SECCION POSGRADO DE LA FIEE Nº 050-2011-DSPG-FIEE

DEDICATORIA

La presente Tesis la dedico primordialmente a mis padres Héctor Raúl Acosta Malpica y Elsa Solórzano Rojas, que con su apoyo permanente me impulsaron siempre a no renunciar nunca a mis sueños y aspiraciones personales.

A mi esposa Yolanda y mis hijos Christian y Leanh por haberme apoyado en este nuevo reto en mi vida y muy especialmente a mi asesor de tesis Msc. Héctor Arístides Trigoso Medina por la confianza y paciencia que me brindo y por estar dispuesto cada vez que lo necesitaba.

Un especial agradecimiento a los docentes de la Maestría en Telecomunicaciones de la Universidad Nacional del Callao , que con la mística de una enseñanza exigente, reconocida por todos, me entregaron las herramientas del conocimiento, necesarias para mi desenvolvimiento profesional.

ÍNDICE

PRÓLOGO

En el mundo de hoy la comunicación es una de las tecnologías presentes más importantes. Un problema con la comunicación es que interpretaciones equivocadas pueden ocurrir. Afortunadamente el lenguaje humano está construido de tal manera que si una letra está mal el lector descubrirá el error y lo corregirá, dependiendo del contexto, ó pidiendo una retransmisión, con un "Perdón". En el mundo de las computadoras sin embargo, la situación es un poco diferente. Las palabras están usualmente formadas por dos diferentes signos, uno y cero, comparado al lenguaje Inglés, el cual tiene 26 letras. Los ceros y unos son colocados en un único orden lo cual hace la palabra "inequivocable". Si un signo está mal puede ser mucho más difícil de traducir, ó en este contexto, decodificar el mensaje recibido al mensaje correcto el cual fue transmitido comparado con el idioma Inglés, el cual debiera tener muchas menos dificultades corrigiendo errores. Esto es debido a que el inglés es mucho más redundante que el desprotegido mensaje binario, y es por esto que muchos diferentes esquemas de códigos son construidos y usados en el mundo entero. Turbo Códigos son uno de los nuevos, presentados por primera vez en 1993 por Claude Berrou, Alain Glavieux and Punya Thitimajshima. Lo que sorprendió a la comunidad fue que se acercaban sorprendentemente al límite de Shannon. El cual será explicado más adelante. Los Turbo-códigos sin embargo, como muchos esquemas de codificación, son hechos para mejorar la capacidad de corrección de error del mensaje. Los Turbo-códigos son una clase de código de corrección lineal de alto desempeño que ha encontrado aplicación en comunicaciones satelitales en el espacio y otras áreas como comunicación celular, por ejemplo en 3G, la diferencia

entre la teoría de codificación y la criptografía será ahora definida. La Teoría de codificación es la ciencia de codificar datos de tal manera que cuando es enviada en un canal es muy probable que sea decodificada los mismos datos que fueron enviados. Veremos más adelante una corta introducción a la teoría de codificación. La Criptografía es la ciencia de codificar datos de tal manera que nadie excepto el receptor puede decodificar y leer los datos que fueron enviados a través de un canal. Así que si los datos son interceptados por un tercero, este no podrá leer los datos. Esta tesis explica alguna de las bases de los turbo-códigos.

RESUMEN

La presente Tesis de Maestría trata sobre el análisis del rendimiento de los códigos Turbo para distintas configuraciones de trabajo a modo tutorial, el cual representa una innovación total, tanto en su construcción como en su decodificación. Las consideraciones a tener en cuenta en el diseño del decodificador, han sido, obtener probabilidades de error (Pe) en el receptor lo más cercano a cero, las estrategias de codificación que se han utilizado para fines de comparación son a través de simulaciones continuas para verificar las prestaciones del sistema con la ayuda del utilitario Matlab 6.5.

Se ha presentado tanto el codificador como el decodificador en diagramas en bloques y mencionando los tipos de algoritmos de decodificación más conveniente SOVA y Log- MAP.

Como resultados finales se han obtenido 7 gráficos que hacen mención sobre probabilidad de error (Pe) vs relación señal a ruido (SNR), donde se puede verificar que variando los parámetros intrínsecos podemos obtener prestaciones optimas de los turbo códigos e Identificaremos el aporte que cada uno de esos parámetros contribuye para mejorar la performance.

Finalmente se puede concluir la validez del sistema a través del software y proponer una nueva configuración de Turbo códigos que tenga el mejor rendimiento a través de la combinación de los parámetros seleccionados.

ABSTRACT

This work deals with the performance of Turbo codes for different work Configurations as a tutorial, which represents a total innovation, both in its construction as in its decoding. The considerations to be taken into account in the design of the decoder have been:

A) Obtaining the error margin closest to zero (Pe) in the receiver, for a certain rank of signal to noise relation (SNR).

B) The coding strategies that have been used for purposes of comparison are through continuous simulations to verify the performance of the system with the help of Matlab 6.5 utility.

Both the encoder and the decoder have been presented in block diagrams and mentioning more convenient decoding SOVA and Log – MAP algorithm types.As final results,7 graphics that show probability of error (Pe) vs relation signal – to noise (SNR) have been obtained. Where you can verify that by varying the parameters, intrinsic parameters, we can utmost benefits from turbo codes and we will identify the contribution that each of these parameters makes to improve performance.

Finally we may conclude the validity of the this system through the software and propose a new configuration of Turbo codes that have better performance through a combination of the selected parameters.

CAPÍTULO I

PLANTEAMIENTO INICIAL DE LA INVESTIGACIÓN

1.1 Identificación del problema

Dentro del constante cambio en el mundo de la electrónica y en general del campo que nos competen este caso, las telecomunicaciones, es necesario que los profesionales del área de Ingeniera Electrónica de la Universidad Nacional del Callao 'UNAC' se encuentren en un proceso de actualización y de aprendizaje en lo concerniente al desarrollo de nuevas tecnologías y reglamentaciones.

Por otro lado un esquema de codificación de canal conocido como turbo códigos está ganando terreno por prometer niveles similares a los predichos por Claude Shannon, en cuanto a la confiabilidad en el transporte de los datos.

Este tipo de codificación ha sido propuesto por empresas encargadas de desarrollar productos de tercera generación 3G y 4G, dada la gran incidencia de ruidos e interferencias que experimentan estos dispositivos en su medio de transmisión.

1.2 Formulación del problema

El desarrollo de este trabajo de investigación por lo tanto no pretende inventar o desarrollar conceptos y tecnologías que han sido probadas e implementadas por una gran cantidad de Ingenieros a lo largo de todo el mundo, y que actualmente se encuentran en funcionamiento en una amplia gama de dispositivos inalámbricos, ya sea celulares, radios y productos de tercera generación (3G), etc. En el mundo de hoy la comunicación es una de las tecnologías presentes más

importantes. Un problema con la comunicación es que se pueden producir interpretaciones equivocadas. Afortunadamente el lenguaje humano está construido de tal manera que si una letra está mal el lector puede descubrir el error y corregirlo, dependiendo del contexto, ó pidiendo una retransmisión, con un Perdón. En el mundo de las computadoras sin embargo, la situación es un poco diferente. Las palabras están usualmente formadas por dos diferentes signos, uno y cero, comparado al lenguaje Inglés, el cual tiene 26 letras. Los ceros y unos son colocados en un único orden lo cual hace la palabra inequivocable. Si un signo está mal puede ser mucho más difícil de traducir, o en este contexto, decodificar el mensaje recibido al mensaje correcto el cual fue transmitido comparado con el idioma Inglés, el cual debiera tener muchas menos dificultades corrigiendo errores. Esto es debido a que el inglés es mucho más redundante que el desprotegido mensaje binario, y por esto muchos esquemas de códigos diferentes son construidos y usados en el mundo entero. Turbo Códigos son uno de los nuevos, presentados por primera vez en 1993 por Claude Berrou, Alain Glavieux and Punya Thitimajshima]. Lo que sorprendió a la comunidad fue que se acercaban sorprendentemente al límite de Shannon .

1.3 Objetivos de la investigación

1.3.1 Objetivo general

a.- Establecer métodos que permitan a los técnicos una toma de decisiones respecto a la mejor alternativa de los turbos codec.

b.- Mejorar la transmisión digital de gran capacidad.

c.- Alta ganancia de código, mejor rendimiento.

1.3.2 Objetivos específicos

Los objetivos específicos del presente trabajo son analizar el rendimiento de los turbo códigos con distintas configuraciones, estudiar los fundamentos matemáticos de los turbos códigos, su principio de funcionamiento, y su aplicación en tecnologías de telecomunicaciones existente. La validez de una plataforma de codificación decodificación turbo que permita simular la robustez de estos códigos.

1.4 Justificación

El desarrollo del proyecto se justifica por la necesidad de tener un diseño que se ajuste a las necesidades de la comunicación en el Perú. La carencia de este tipo de sistemas en el Laboratorio de la UNAC para fines educativos. Mejora del desempeño de este tipo de códigos y en consecuencia bajar costos en los equipos de comunicación y ayudar de esta manera a la población.

El problema de investigación es vulnerable porque es posible estudiar, diseñar y aplicar un sistema de telecomunicaciones nuevo. Se dispone de información técnica, herramientas de software, instrumentación. Una vez que ha sido positivamente demostrado los resultados de la utilización de este sistema, este

puede ser diseñado y distribuido en el mercado nacional e internacional.

1.5 Limitaciones y facilidades

1.5.1 Limitaciones

Si bien es cierto la utilización de esos algoritmos son de conocimiento y aplicación en pleno avance en países desarrollados; sin embargo, su utilización en nuestro país es todavía incipiente, y son muy pocas las empresas que con adecuado criterio logran implementarla con éxito. EL término limitaciones no se refiere a factores que obstaculizan el desarrollo de la investigación, sino a parámetros establecidos por el investigador para la mejor ejecución del proyecto de investigación. Podemos mencionar como limitante la disponibilidad de instrumental y recursos económicos, además de una política de integración de las universidades y de la industria que direccionen y canalicen las necesidades.

1.5.2 Facilidades

Para el diseño y simulación de los sistemas de codificación propuestos, se ha utilizado el programa Matlab 7.0 herramienta poderosa para el cálculo matemático y la validación del modelo propuesto. Las mayores facilidades son la disponibilidad de material humano que es, en la mayoría de los casos, obligado a emigrar al exterior, entre otros, por los motivos antes mencionados.

1.6 Hipótesis

En función del planteamiento del problema y de las interrogantes planteadas del problema, de los antecedentes técnicos, así como de los objetivos generales y específicos que se persigue, es que se plantea la siguiente hipótesis:

Se han identificado diferentes niveles de rendimiento en los Turbos códigos en diferentes configuraciones. Reconociendo que existen diversos parámetros de estos códigos que se pueden modificar para variar este parámetro.

Identificaremos el aporte que cada uno de esos parámetros para mejorar el rendimiento del sistema.

CAPÍTULO II

MARCO TEÓRICO

2.1 Antecedentes del estudio

Los servicios (ITU,1997a) a los que van a dar soporte estos novedosos sistemas, están basados en una disponibilidad de ancho de banda significativa por lo que se necesitan tasas de transmisión de datos altas. A estas premisas hay que añadir la particularidad del medio de transmisión ruidoso. El cual no es un medio controlable y estable, por el contrario, presenta una variabilidad importante que lleva a que las comunicaciones no siempre sean fiables entre dos puntos.

Para combatir estos inconvenientes, satisfaciendo las hipótesis de partida, estos sistemas requieren, entre otras soluciones, esquemas de codificación innovadores para poder dar servicio de calidad. La necesidad de esquemas de codificación potentes y robustos está claramente reflejada en los estándares definidos hasta ahora basado en W-CDMA dónde para satisfacer requisitos de calidad se está empleando un novedoso esquema de codificación denominado Turbo Códigos.

El nuevo esquema de codificación fue propuesto en 1993 por Berrou, Glavieux y Thitimajshima en la conferencia Internacional en comunicaciones en Ginebra (Berrou et al 1993). Defendía que una combinación de códigos convolucionales concatenados paralelamente y una decodificación por etapas e iterativa con criterio MAP en cada una de ellas, podría proporcionar comunicaciones fiables a una relación señal a ruido unas decimas por encima del límite teórico de Shannon (Shannon,1948). Este incremento en la ganancia de codificación respecto a los esquemas habituales tuvo gran impacto sobre los sistemas de radio de

comunicaciones de tercera generación, donde los requisitos ya mencionados en cuanto al ancho de banda estaban haciendo necesarios mejoras en los esquemas de codificación. La aplicación es inmediata desde el punto de prestaciones, lo que está por comprobar aún si las particularidades del medio de transmisión y la necesidad de implementar estos esquemas para servicios en tiempo real, nos llevan a unas prestaciones que aún no siendo las ideales, siguen siendo satisfactorias. Pero la propuesta de concatenar códigos no es tan reciente, los esquemas de codificación concatenados fueron propuestos por Forney (Forney, 1996) con el fin de lograr unas ganancias de codificación grandes por la combinación de dos códigos componentes relativamente simples. El esquema de codificación resultante es muy potente y está dotado de una estructura que permite una decodificación sencilla, como es el caso de la decodificación por etapas o la decodificación por etapas iterativas (Berrou).

Pero en 1993 con la presentación de la ponencia " Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes" (Berrout et al, 1993) dónde tomaron relevancia este tipo de esquemas de codificación. Previo al interés que despertó su aplicación a comunicaciones móviles, esta propuesta ha tenido una gran respuesta dentro de los círculos de investigación donde se han dedicado muchos esfuerzos a reproducir los resultados defendidos por los autores y a establecer los aspectos teóricos que definen el funcionamiento del esquema de codificación propuesto (Divsalar y Pollara,1995), (Hagenauer et al 1994), (Robertson,1994), (Benedetto y Montorsi,1996).

2.2 Bases teóricas

Hoy en día el problema que existe en las comunicaciones es enviar de manera confiable información libre, de errores a través de un canal de comunicación, aprovechando la capacidad máxima del canal de acuerdo al teorema de Claude Shannon, en donde se demostró teóricamente que para cualquier tasa de transmisión menor ó

igual a la capacidad de canal, existe un esquema de codificación que alcanza una probabilidad de error arbitrariamente pequeña, y por lo tanto se puede hacer la transmisión sobre el canal muy confiable y bajo una potencia de transmisión estrictamente necesaria.

Para enviar la información a través de un canal de comunicación se usan actualmente esquemas de codificación tanto en la fuente como en el canal. La codificación del canal tiene como tarea codificar la información adicionando redundancia al mensaje, de tal manera que ante la presencia de ruido en el canal se pueda detectar y corregir los errores generados por este; además con la codificación de canal se puede operar con transmisión de baja potencia, realizar transmisiones a largas distancias, mayor tolerancia a la interferencia, transmitir a altas tasas de datos y hacer posible el uso de pequeñas antenas.

Un sistema de codificación de canal además de corregir errores debe ser capaz de corregirlo tras fallas que se pueden presentar en la degradación del canal tales como: atenuación de la señal debido al medio, el ruido térmico, interferencia ínter simbólica, interferencia del múltiple usuario, la propagación multidireccional, y limitaciones de potencia [5].

Para el diseño de la codificación y modulación del canal se deben tener en cuenta cuatro factores [5]:

1. Probabilidad de error (P_e): este factor nos dice que tan confiable es la transmisión, es decir mide el desempeño de un sistema de comunicación digital.
2. Eficiencia espectral ó ancho de banda (WRs): esto mide la eficacia en gasto del ancho de banda, nos dice cuántos bits por segundo (R_s) pueden ser transmitidos en un ancho de banda dado *W*.
3. Tasa de relación señal a ruido (SNR), este parámetro mide cómo el esquema de codificación y modulación hace uso eficientemente de la potencia disponible.
4. Complejidad: Este factor está estrechamente relacionado con el costo del equipo.

En la codificación de canal se distinguen dos métodos:

- Corrección de error hacia atrás **BEC** (Backward error correction) ó **ARQ**. El cual solo emplea detección de errores, si un error es detectado se solicita retransmisión de todo el mensaje. Mientras este método tiene una baja complejidad en los esquemas de corrección de errores, se debe disponer de una comunicación dúplex, además la retransmisión trae como consecuencia retardos a nivel del tiempo, que en aplicaciones críticas como las de tiempo real no es aconsejable.
- Corrección de error hacia delante **FEC** (Forward error correction). En el Transmisor un codificador de FEC agrega redundancia a los datos en la forma de información de paridad de la siguiente manera: un codificador binario de FEC toma *k* bits a la vez y produce una salida (ó palabra de código) de *n* bits, donde n > k. Mientras que hay 2^n secuencias posibles de

n bits, solamente hay un subconjunto pequeño de ellos 2^k, que serán palabras de código válidas. El cociente **k/n** se llama la tasa de código.

- Entonces en el receptor, un decodificador de FEC puede explotar la redundancia de una manera tal que un número razonable de los errores de canal pueda ser corregido. Con un código FEC se logra tolerar más errores de canal, por lo que los sistemas codificados pueden permitirse funcionar con más baja transmisión de potencia, transmitir a largas distancias y tolerar más interferencia, hacer posible el uso de antenas más pequeñas, y se podría transmitir a una mayor velocidad para una potencia de transmisión dada. El método FEC solo requiere una comunicación simplex, este método es muy atractivo en sistemas de comunicación inalámbrica, ayudando a mejorar la eficiencia de la energía de los sistemas.

2.2.1 Bases epistémica

La epistemología pone en tela de juicio el conocimiento ya aceptado como valido por la comunidad científica, desde esa perspectiva se ha revisado lo anteriormente aceptado en telecomunicaciones en el traslado de información digital a través de los turbos códec, como son la información incorrecta por continuidad de ráfagas en el lado del receptor. La importancia de este punto reside en la apreciación subjetiva de los hechos ó acontecimientos por diferentes observadores, como es de conocimiento dos personas interpretan el mismo hecho de diferentes formas, una parte importante está constituida por el estado interno de nuestras mentes, el cual dependerá de nuestra educación cultural, nuestro conocimiento, nuestras expectativas.

2.2.2 Introducción a los códigos de Bloques

❖ **Códigos de bloque**

Estos códigos hacen la corrección de errores a nivel de bits tomando la información por bloques, dentro de ellos se encuentran los códigos Hamming, Golay, BCH, Reed Solomon (estos códigos como trabajan con símbolos de código m_arios , los hace eficientes para corregir errores de ruido en ráfaga).

❖ **Códigos Convolucionales**

Los códigos convolucionales trabajan serialmente, y tienen memoria, lo cual los hace comparablemente más eficientes que los códigos de bloque, por su simplicidad y capacidad de corrección.

❖ **Los Turbo códigos**

Los turbo códigos son un método de corrección de errores basado en los códigos convolucionales más intercalación y realimentación. Consiste en una estructura de codificación concatenada más un algoritmo iterativo, estos fueron introducidos en 1.993 por Berrou y Glavieux en la conferencia internacional de la IEEE en Ginebra Suiza. El esquema propuesto en dicho trabajo alcanzaba una probabilidad de error de bit de 10^{-5} usando una tasa de codificación de ½ sobre un canal de ruido blanco gausiano (AWGN) y modulación BPSK con una relación de energía promedio de bit sobre la densidad espectral de potencia de ruido ó (**N/Eb**) de 0.7 dB, lo cual está cercano al límite de Shannon que es 0.1dB. [6].

2.2.3 Definiciones preliminares

Por la naturaleza de este trabajo de investigación, será necesario un detallado trabajo de documentación y conocimiento de distintas áreas de conocimiento relacionadas con la codificación de canal y su simulación. Así, algunos de los aspectos que más esfuerzos han supuesto son los siguientes:

- **Sistemas de comunicaciones**

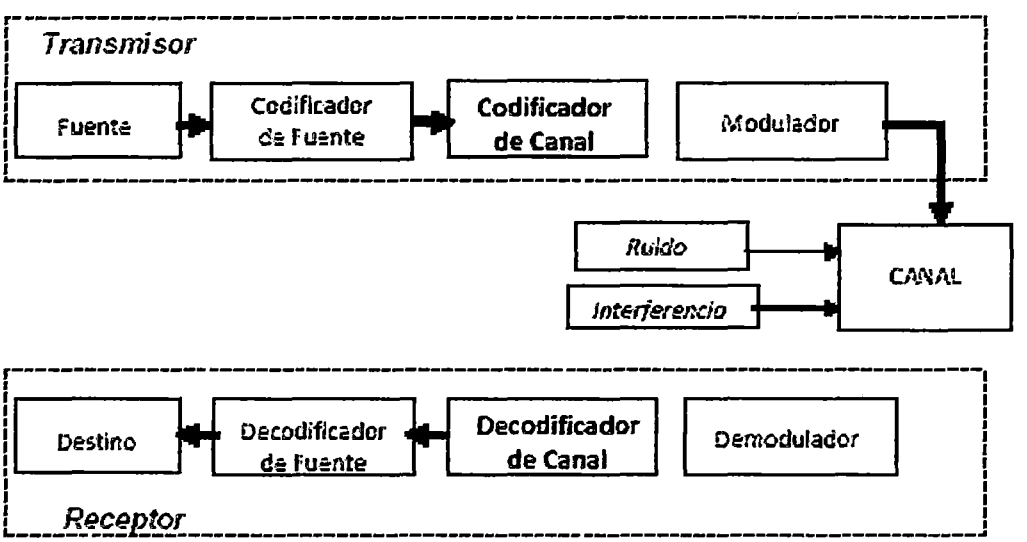
Un sistema de comunicaciones tiene la finalidad de transportar información de una parte (fuente) a otra (destino). La mayoría de los sistemas de comunicaciones modernos operan en el dominio digital, es decir, emplean una secuencia de símbolos de unos alfabetos finitos para representar la información. Transmisión de datos en formato digital que ofrece la posibilidad de emplear una gran variedad de técnicas de procesamiento de señal, que de otra forma no estarían disponible, incluyendo, por supuesto, codificación de corrección de errores. Un modelo de un sistema de comunicaciones es el que se muestra en la figura 2.1 cuyo bloque constitutivos son descritos brevemente.

La fuente de información genera mensajes que deben ser trasmitidos al receptor. Estos mensajes pueden ser analógicos ó digitales, dependiendo del tipo de fuente por ejemplo, la voz trasmitida durante una conversación telefónica es representada por una señal de tiempo y amplitud continuos generada por un micrófono, que junto con el parlante constituye una fuente analógica. En un sistema de comunicaciones digitales, los mensajes emitidos por una fuente analógica son convertidos a una señal digital, usualmente representada por secuencias de dígitos binarios ó bits. Esta operación es realizada por el codificador de fuente, el cual busca representar el mensaje de entrada con tan poco bits, este proceso es llamado compresión de datos, proceso durante el cual

se reduce la redundancia presente en los mensajes de la fuente, ó en forma ideal se prescinde completamente de ella.

El siguiente bloque en el modelo de comunicaciones es el codificador de canal, mientras que el codificador de fuente es seleccionado con respecto a una fuente de información particular, el codificador de canal se elige generalmente con respecto al canal por el cual los mensajes van a ser transmitidos. El propósito del codificador de canal es darle un formato a los mensajes transmitidos, de tal manera de incrementar su inmunidad al ruido y a la interferencia introducida por el canal. Esto se realiza insertando redundancia controlada en el mensaje a ser trasmitido, permitiendo al receptor detectar y, posiblemente, corregir errores. El problema de esta tesis pertenece a esta clase de códigos, que son códigos de canal.

Fig. 2.1 Diagrama de bloques de un sistema de comunicaciones



Fuente: grafico tomado de [9] Divsalar, D, and F. Pollara

El canal sirve como un medio de unión por el cual los mensajes codificados son transmitidos. En muchas situaciones practicas, por ejemplo en radio

comunicaciones, el canal es de ondas electromagnéticas, sin embargo, este no puede ser empleado para transmitir directamente las secuencias de dígitos binarios. Más bien las secuencias digitales deben ser convertidas en formas de ondas adecuadas a las características específicas del canal, tarea que es efectuada por el modulador. Las ondas de salida del modulador son afectadas entonces en una forma que depende de las características del canal. Estas ondas dadas son la entrada del demodulador, cuya función es la inversa del modulador, convertir la onda recibida en una secuencia discreta en el tiempo, Idealmente idéntica a la que entro al modulador. Aquí es donde el decodificador de canal entra en acción; su papel es contrarrestar las perturbaciones inducidas en el canal, valiéndose de la redundancia introducida por el codificador de canal. La secuencia estimada aquí es trasformada en la salida estimada de la fuente en el decodificador de fuente, el cual envía este estimado al destino. Cuando la fuente es analógica, se debe efectuar una conversión digital/analógica (D/A) .En un sistema bien diseñado, el estimado debe ser una fiel reproducción de la salida de la fuente, excepto cuando el canal sea muy ruidoso.

❖ **Atenuación**

Causada generalmente por la absorción de energía y las multitrayectorias de la señal en el medio de propagación. En el caso de comunicaciones satelitales la atenuación se manifiesta como una pérdida de potencia en la señal transmitida, como función de la distancia, conocida como el efecto de pérdida de espacio libre. En el caso de comunicaciones móviles, la atenuación no es fija, y depende de la combinación del movimiento y de las numerosas reflexiones de la señal de radio transmitida, causadas por el ambiente de propagación entre la antena transmisora y la receptora. Tales canales de comunicaciones se conocen canales de

desvanecimiento por multitrayectoria.

❖ Capacidad de canal

Shannon introdujo el concepto de capacidad del canal C, que es la máxima velocidad a la cual la información podrá ser trasmitida confiablemente en un canal de comunicaciones dado. Entendiéndose entonces, que la capacidad del canal es una medida de la cantidad de información que puede ser transportada entre la entrada X y la salida Y de un canal, la medida de capacidad esta relacionada con la definición matemática de información. La información mutua promedio entre X e Y, está dada por la ecuación 2.1.

$$I(X,Y) = \begin{cases} \sum x \sum y\, p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} & \text{para un canal discreto} \\ \int p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} & \text{para un canal continuo} \end{cases} \qquad (2.1)$$

donde p(x) y p(y) son las funciones de densidad de probabilidad marginales de X e Y respectivamente y p(x,y) es la función densidad de probabilidad unida de X e Y. Entonces la capacidad del canal se define como el valor máximo de I(X,Y), maximizado sobre la distribución de entrada p(x), así

$$C = \frac{max}{p(x)} I(X,Y), \qquad (2.2)$$

medida en bits por transmisión, La importancia de la capacidad de canal C queda claramente establecida por el teorema de codificación ecuación (2.2) para

canales contiguos y ruidosos con limitaciones de potencia promedio , éste establece lo siguiente:

Teorema (Codificación para canales ruidosos de shannon) A cada canal se le puede asociar una capacidad de canal **C**, de tal manera que existen códigos de control de errores tales que la información puede ser trasmitidas sobre el canal a velocidades menores que **C** con una probabilidad de error de bits (BER) arbitrariamente baja.

S= Potencia de señal

N= Potencia de ruido

B= Ancho de banda

N= No.B (**No** , densidad espectral de potencia)

$$C = B \log_2(1 + \frac{S}{NoB}) \qquad (2.3)$$

¿Qué sucede si aumentamos el ancho de banda?

Veamos:

$$\lim_{B\to\infty} C = \lim_{B\to\infty} \frac{NoB}{S} . \frac{S}{N0} \lim_2(1 + \frac{S}{NOB}) \qquad (2.4)$$

$$\lim_{B\to\infty} \frac{S}{No} log2\left[(1 + \frac{S}{NoB})^{\frac{NoB}{S}}\right]$$

Si $z = \frac{S}{NoB}$

$$\frac{S}{NO} \lim_{z\to 0} \log_2[(1+z)^{1/z}] = \frac{S}{NO} \log_2 \lim_{Z\to 0}[(1+z)^{1/z}] = \frac{S}{No} \log_2 e = 1{,}44 \frac{S}{N0} \qquad (2.5)$$

Es decir que por más que aumentamos el ancho de banda, la capacidad de transmisión no crece indefinidamente, sino que tiene un límite.

Supongamos esto en la condición **Rb ≤ C**

Rb , ⟶ bits / seg

B , ⟶ Hz

Eb ⟶ joule / bits

No ⟶ w / Hz

S = E b.Rb (2.6)

N = No B

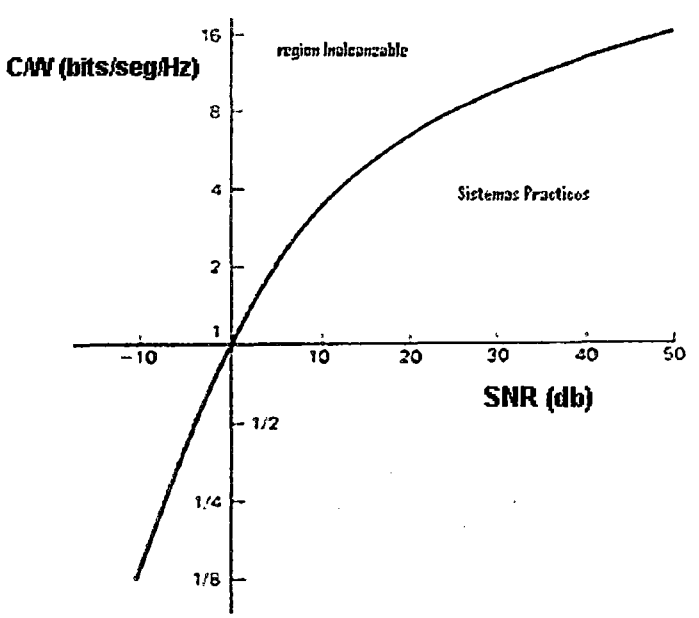
$Rb \leq B \log_2(1 + \frac{EbRb}{NoB})$ estos son los sistemas posibles (2.7)

$$\frac{Rb}{B} \leq \log_2(1 + \frac{EbRb}{NoB})$$

$$2^{R/B} \geq 1 + \frac{Eb\ Rb}{No\ B} \quad (2.8)$$

$\frac{Eb}{No} \geq \frac{2^{R/B}-1}{Rb/B}$ estos son los posibles sistemas como se muestran en la siguiente figura 2.2.

Fig.2.2 .Capacidad del canal normalizado versus SNR del canal (el ancho de banda se indica con W en lugar de B)., tomado de [12].



Fuente: tomado de [12].

❖ **Medida de performance**

Para un ancho de banda W fijo aumenta con un incremento en la potencia de la señal transmitida. Por otro lado si P es fijo, la capacidad se puede incrementar, aumentando el ancho de banda W. Cuando W $\rightarrow \infty$, la capacidad del canal se aproxima a su valor asintótico (capacidad de canal de ancho de banda infinito), hallado como:

$$C_{\infty} = \frac{P}{\eta o} \log_2 e \qquad (2.9)$$

La potencia de entrada al canal se puede expresar como $P = E_b R_b$, donde E_b es la energía transmitida por bit de información y R_b es la velocidad de información en bits/s. Combinando el teorema de codificación de canal con la velocidad de canal de ancho de banda infinito se tiene que $R_b < \frac{P}{No} \log_2 e$, lo que nos lleva a un requerimiento fundamental para la realización de las comunicaciones:

$$\frac{Eb}{No} > \frac{1}{\log_2 e} = \ln 2, \qquad (2.10)$$

Donde ln es el logaritmo natural. De la figura 2.2 se desprende que no es posible diseñar un sistema de comunicaciones con una probabilidad de error arbitrariamente baja si $E_b / N_o < -1.6$ db. Por otro lado, mientras que $E_b / N_0 > -1{,}6$ db, es posible en teoría lograr una probabilidad de error arbitrariamente baja, usando códigos suficientemente largos.

La ecuación 2.10 se trata de un límite inferior de la relación señal a ruido (SNR).

En términos de la relación energía por bits con respecto a la densidad espectral de ruido (E_b/N_o), requerida para lograr una comunicación libre de error. Sin embargo, los supuestos asumidos para derivar este límite son demasiado

optimistas para la mayoría de las situaciones prácticas. Primeramente el ancho de banda del canal es limitado, segundo se permite a la palabra de código transmitida ser infinitamente larga, lo que introducirá un retardo infinito en la transmisión. Como es de suponerse los sistemas de comunicaciones son diseñados para trasmitir en un tiempo limitado. Además este límite es derivado asumiendo una entrada continua, sabiéndose que los sistemas de comunicaciones digitales emplean esquemas de modulación con alfabetos finitos. Debido a esas suposiciones la (E_b/N_o) requerida es en realidad mayor que -1,6 db. Existen otros límites teóricos que dan cotas inferiores más exactas, tomando en consideración algunas de esas suposiciones, como por ejemplo el límite de paquete esférico y el límite esférico tangencial.

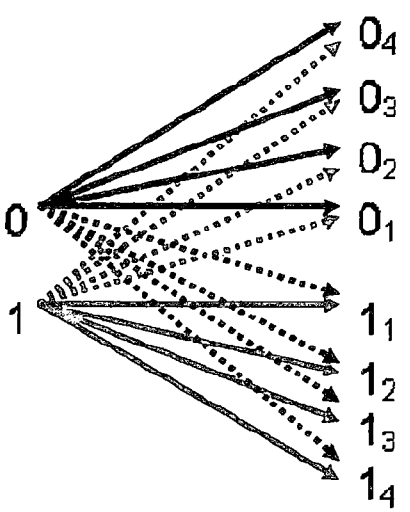
- **Decisión Soft vs Decisión Hard**

Es muy importante tener en consideración estos dos conceptos, ya que de ellos depende que tengamos una mejor o peor implementación de nuestro sistema. Si el demodulador (en un sistema de transmisión que podría ser el de la figura 1.1) realiza una decisión hard (esto quiere decir, a su salida tenemos un "0" ó un "1", y nada más) y entrega el resultado al decodificador corrector de errores (ó decodificador de canal), este no tendrá ninguna información adicional sobre el canal; no sabrá si estamos muy seguros de que ese valor es correcto ó por el contrario, que el valor verdaderamente obtenido estaba muy próximo al umbral establecido para decidir que bit era y por consiguiente, de que podemos habernos equivocado con una cierta probabilidad más o menos importante.

Sin embargo, si utilizamos una decisión soft en el demodulador, la información que pasaremos al decodificador de canal será mayor, ya que dispondremos de 2^n

valores para codificar la información obtenida, donde n es el número de bits utilizados en la cuantización de dichos valores recibidos. Así, por ejemplo, si utilizamos 3 bits en cuantización, dispondremos de 8 valores para representar dos posibles transmisiones:'0' o '1'. De esta manera, si el valor recibido se cuantificó con un '**14**', según la figura 2.3, podemos decir que el valor recibido es un '1' con una probabilidad muy elevada. Por el contrario si recibimos un '**01**', nos decantaremos por pensar que se había transmitido un '0' lógico pero no estaremos muy convencidos de ello.

Fig. 2.3. Valores recibidos con decisión soft en un canal discreto sin memoria



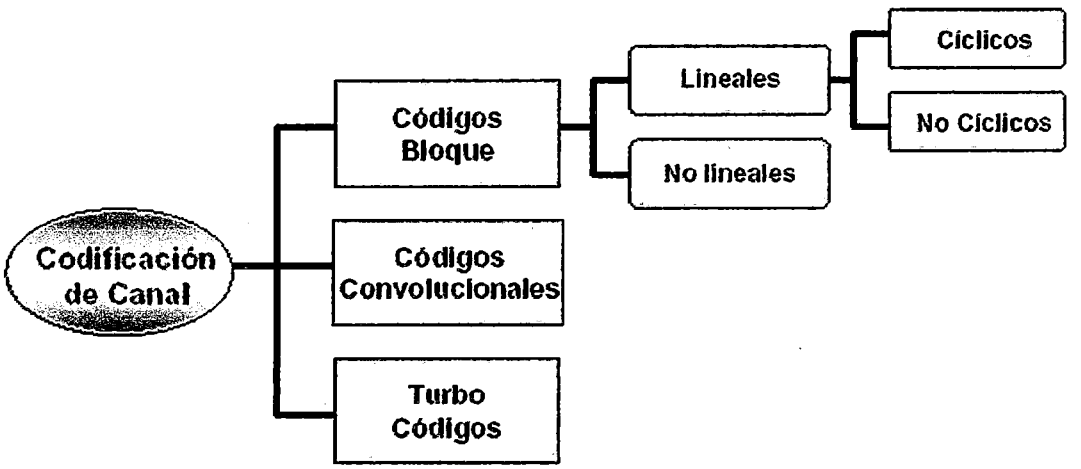
Gracias a esta información añadida que obtenemos al utilizar decisión soft podremos alcanzar mejores resultados en la tasa de error con los códigos correctores de errores que utilicen esta propiedad

2.3 Tipos de codificación de canal

Los códigos correctores de errores ó de codificación de canal pueden dividirse en código bloque y códigos convolucionales. Puede definirse una tercera clase de códigos, llamados Turbo códigos, pueden ser considerados como una sub-clase de los códigos bloque ó de los códigos convolucionales como se muestra en la figura 2.4. Esto es así porque los Turbo Códigos, al igual que los códigos de bloque, necesitan que todo bloque de información a codificar y decodificar esté presente para comenzar cualquiera de estos dos procesos. Sin embargo, no obtiene los bits de paridad a partir de un sistema de ecuaciones (como los código bloque), sino a partir de un registro de estado, como los códigos convolucionales. Es más, utiliza un mínimo de dos códigos convolucionales simples RSC para ello. Nosotros la clasificaremos como una tercera clase de códigos independientes surgida de una mezcla de las dos anteriores.

Cabe mencionar que los códigos bloque pueden dividirse a su vez en lineales y no lineales. y dentro de los lineales podemos dividirlos en cíclicos y no cíclicos. Los más utilizados son los lineales, ya que gracias a esta propiedad es más sencillo implementar la codificación y la decodificación. Y dentro de ellos los más utilizados son los cíclicos, ya que esta propiedad nos permitirá realizar implementaciones tanto software como hardware.

Fig. 2.4 clasificación de los códigos correctores de errores



Fuente: tomado internet Borja Álvarez Feíto

❖ **Ruido blanco gaussiano**

Puede tomar diferentes formas. Las fuentes más común son el calor en el transceptor y en el medio de propagación. El modelo más comúnmente asumido de ruido es el modelo de ruido aditivo blanco gaussiano (AGWN), por sus siglas en inglés Additive white Gaussian noise).

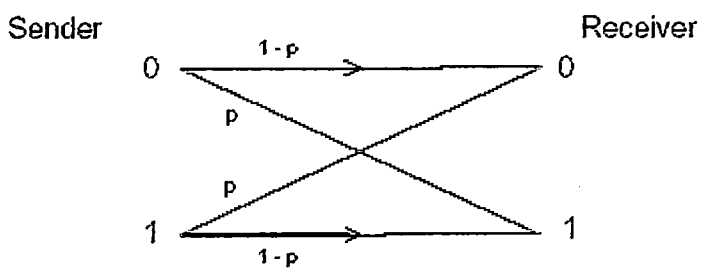
Cuando se estudian y se comparan los códigos de canal es conveniente considerar al modulador y al demodulador como parte del canal. Si este canal emplea solo dos símbolos (0 y 1) y realiza detecciones duras en el receptor, se trata de un canal simétrico binario, como el que se ve en la Fig. 2.5. Este es un canal compuesto con entradas y salidas discretas en el tiempo, caracterizado por las posibles entradas, conjunto X= {0,1}, las posibles salidas, conjunto Y = {0,1} y las probabilidades de transición que relacionan las posibles salidas con las posibles entradas. Si en la secuencia binaria transmitida, el ruido y otras perturbaciones del canal causan errores estadísticamente independientes, con

probabilidades promedio p, las probabilidades de transición son:

$$P(Y=0 \setminus X=1) = P(Y=1 \setminus X=0) = p$$
$$P(Y=1 \setminus X=1) = P(Y=0 \setminus X=0) = 1-p \qquad (2.11)$$

En general estas probabilidades de transición pueden variar con el tiempo, y pueden estar también correlacionadas de un tiempo a otro. Además, las salidas

Fig. 2.5 Canal binario simétrico



del canal no dependen necesariamente de la entrada actual solamente, sino que pueden depender de entradas previas. En tales casos se dicen que el canal tiene memoria. Los canales encontrados en sistemas de comunicaciones móviles son ejemplos de canales que normalmente deben ser modelados como variantes en el tiempo y con memoria. Estos efectos surgen de la combinación del movimiento y de las numerosas reflexiones de las ondas de radio transmitidas, causadas por el ambiente de propagación entre las antenas transmisoras y receptoras.

Un modelo que se adapta con exactitud a varios sistemas de comunicaciones y que es el más comúnmente usado, entre otras razones, por su simplicidad y su facilidad de análisis en diversos contexto, es el canal de ruido blanco aditivo gaussiano (AWGN), si bien este modelo no se adapta exactamente al comportamiento de un canal de comunicaciones móviles, si lo hace para los de

comunicaciones por satélite y para los terrestre en línea de vista, es por eso que se adoptará este modelo en el desarrollo de esta tesis, ya que por tratarse de un estudio del funcionamiento de los turbos códigos no es conveniente añadir complicaciones al canal que difunden su comprensión.

Este modelo de ruido blanco gausiano (AWGN), es aditivo ya que la salida del canal Y se obtiene añadiendo a la entrada X la variable aleatoria gaussiana G, así:

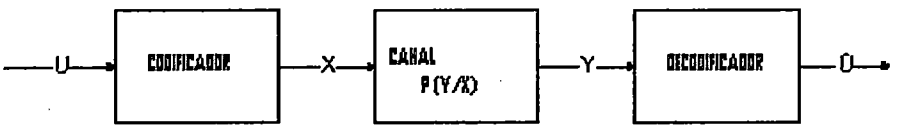
$$Y = X + G \tag{2.12}$$

Donde G es una variable aleatoria gaussiana con media cero y varianza σ^2. Para un símbolo de entrada dado X= **x**, la salida del canal Y es una variable aleatoria gaussiana con media **x** y varianza σ^2, así:

$$p(y\backslash x) = \frac{1}{\sqrt{2pi\sigma}} \cdot e^{\frac{-(y-x)^2}{2\sigma^2}} \tag{2.13}$$

Donde p(.) denota una función densidad de probabilidad de una variable aleatoria. La porción blanca del AGWN se refiere a la densidad espectral del ruido, qué se asume tiene el valor constante N_o entre las frecuencias 0 a $+\infty$. Esto es claramente una imposibilidad física, porque se requeriría que la fuente de ruido tenga energía infinita; pero esta suposición brinda buenos resultados cuando se aplica al análisis de sistemas de comunicaciones de banda limitada.

Fig. 2.6 Modelo de comunicaciones donde el canal, el modulador y demodulador son reemplazados por un canal de tiempo discreto



Un modelo de comunicaciones simplificado, donde el modulador, demodulador y canal de ondas es reemplazado por un canal de tiempo discreto se muestra en la Fig.2.6 en este modelo se ha logrado las simplificaciones reemplazando la fuente y el codificador de fuente con una secuencia de símbolos binarios. Estos símbolos son variables aleatorias independientes e idénticamente distribuidas, con igual probabilidad de ser 0 ó 1. Esto corresponde a una situación ideal en la que al codificador de fuente se le ha removido toda la redundancia presente en la secuencia de información original. El resultado de estas simplificaciones es un modelo que es frecuentemente usado en estudios y comparaciones de códigos de canal, y es el que se usa en este trabajo de investigación.

2.4 Codificación de canal

Los códigos de corrección de errores ó de canal surgen a finales de la década de 1940, con el trabajo pionero de Shannon, Hamming y Golay. En su Teoría matemática de las comunicaciones Shannon definió los conceptos teóricos de la codificación, basado en rigurosas demostraciones matemáticas que sentaron las bases de lo que hoy se conoce como Teoría de la información. En su trabajo dejó por sentado que es posible lograr comunicaciones con una muy pequeña probabilidad de error (tan pequeña como se desee), si se utiliza algún esquema de codificación, con una velocidad de transmisión **r** menor que la capacidad del canal '**C**' . Sin embargo no estableció ningún método práctico para lograr esta

capacidad. Desde entonces se ha dado un gran esfuerzo por diseñar estos códigos que operan lo más cercano posible a los límites predichos por este gran matemático.

Las funciones de codificación y decodificación involucran las operaciones aritméticas de adición y multiplicación modulares realizadas en las palabras de código. Estas operaciones matemáticas se realizan de acuerdo a las convenciones del campo algebraico que tiene como sus elementos los símbolos contenidos en el alfabeto que se va a usar. Aquí algunas definiciones y ejemplos de estas.

2.5 Programación en Matlab

Matlab es el programa utilizado para la realización de las simulaciones y para el diseño del codificador y decodificador turbo. Así, se hace completamente necesario el estar familiarizado con este software tanto para la creación del código como para el análisis de los resultados y la generación.

2.6 Codificación de señales

2.6.1 Código de bloque

Al mismo tiempo que Shannon definía los límites teóricos de la comunicación confiable, Hamming y Golay estaban desarrollando los primeros esquemas prácticos de códigos correctores de errores. Su trabajo dio nacimiento a una nueva floreciente rama de la matemática aplicada conocida como teoría de la codificación. Normalmente se atribuye a Richard Hamming haber descubierto el primer código corrector de errores en 1948 [14], el cual lleva hoy su nombre. A

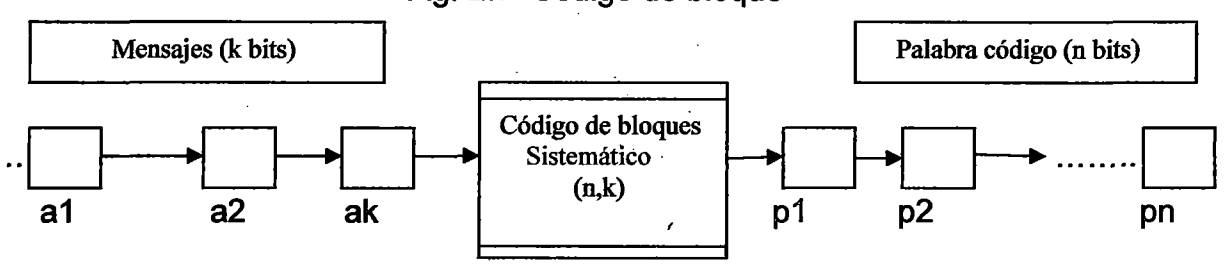
este se sumaron los códigos Golay, Reed-Muller, cíclicos, también conocidos como CRC (códigos de redundancia cíclica), BCH y Reed Solomon.

Un código de bloque C consiste en un conjunto de T vectores de longitud fija C_i,= $i = 0, \ldots\ldots, T-1$, llamados palabras de código, cada uno de la forma C= (C_0,C_1,.....C_{n-1}), que toma valores de un campo de Galois de q elementos, denotado por GF(q). Se dice entonces que el código C es q-ario. Cuando el alfabeto consiste de dos elementos, 0 y 1, se trata de un código binario y los elementos de cualquier palabra de código se llaman bits. La longitud de una palabra de código es el número de elementos en el vector y se denota por n. Existen entonces,q^n posibles palabras de código en un bloque de longitud n. De estas q^n palabras de código se seleccionan T= q^K palabras de código, siendo (k<n), para formar el código C.

Un código de bloque es aquel en el que se pueden enviar 2^k mensajes distintos, de k bits cada uno, a los que se añaden (n-k) bits redundantes que se generan a partir de los k bits del mensaje siguiendo una regla predeterminada.

Un código de bloques es sistemático si los primeros (últimos) k bits son el mensaje y los restantes (n-k) son el chequeo.la figura 2.7 muestra el proceso de codificación de bloques sistemática. Al bloque de n bits se le llama palabra de código, el código ASCII seria un código (8,7).

Fig. 2.7 Código de bloque



Un codigo de bloques es lineal cuando la suma módulo 2 de dos palabras de codigo es tambien otra palabra de código.

Existen dos formas de implementar un código de bloques. La primera de ellas consiste en almacenar en una memoria las 2^k palabras de código correspondientes a los mensajes de información. Este primer método se vuelve inviable cuando los valores de k se hacen grandes, puesto que los requisitos de memoria y tiempo de busqueda se vuelven inaplicables. Por ejemplo, un código (127,92) tiene un total de $2^{92} \sim 5.10^{27}$ palabras de código.

El segundo y mas empleado metodo para obtener una palabra de codigo lineal es:

$$C = D(1,K) \times G(k,n) \qquad (2.14)$$

Donde **D** es un mensaje (vector de k elemntos) y **G** es la matriz generadora (con k filas y n columnas). Al multiplicar ambas (modulo 2) se obtiene la palabra de código **C** de n elementos.

La forma general de G, para un código sistemático,es :

$$G = \begin{bmatrix} \leftarrow g1 \rightarrow \\ \leftarrow g2 \rightarrow \\ \cdots \\ \cdots \\ \cdots \\ \leftarrow gn \rightarrow \end{bmatrix} = \begin{bmatrix} g11\ g12 \ldots\ldots\ g1n \\ g21\ g22 \ldots\ldots\ g2n \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ gk1\ gk2 \cdots\cdots\cdots\ gkn \end{bmatrix}$$

Donde Pk,n-k es la matriz de generacion de paridad e $\mathbf{I_k}$ es la matriz identidad. De esta forma, al multiplicar por D, los k primeros (últimos) elementos de C, son los mismos que los de D.

- **Definición (Distancia de Hamming)** Se representa por d_{ij}, donde $i \neq j$, es el número de elemnetos ó posiciones correspondiente en las que difieren dos palabras de código. En el caso binario puede ser hallada tomando la suma módulo 2 de dos palabra de código y contando el número de unos en el resultado.

- **Definición (Distancia mínima)** la distancia mínima d_{min} de un código es la menor de las distancias de Hamming entre cualesquiera dos palabras de código.

$$dmin = \frac{min}{i \neq j} dij \qquad (2.15)$$

Un código con distancia mínima d_{min} es capaz de corregir todas las palabras de código con **t** ó menos errores, donde:

$$t = \left[\frac{dmin-1}{2}\right] \qquad (2.16)$$

- **Definición (peso de Hamming)** De una palabra de código, w_1,es la distancia de Hanming entre ellas mísmas y la palabra de código todos ceros.

$$\mathbf{wi = di_0} \qquad (2.17)$$

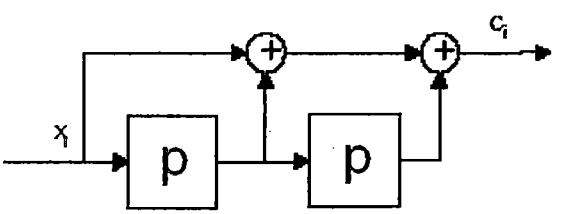
El peso de Hamming se puede determinar simplemente contando el número de elementos no nulos en una palabra de código. Para códigos lineales, la distancia mínima es el menor peso de Hamming de todas las palabras de código, excepto la palabra de código todos ceros.

$$dmin = \underset{t>0}{min}\{wi\} \qquad (2.18)$$

2.6.2 Códigos Cíclicos

Un código cíclico es un subconjunto de la clase de los códigos lineales, en el que cada corrimiento cíclico de una palabra de código produce una palabra de código valida. Esto es 'C' es ciclico si para cada palabra de código C= ($c_{n-1}, c_{n-2}, \ldots\ldots c_1, c_0$) ε C.

Fig. 2.8 codificador cíclico para g(p) = p^2 + p +1



Esto puede ser generalizado en que cada corrimiento cíclico de C, es una palabra de codigo. La matriz generadora de un código cíclico puede ser expresada de la forma:

$$G = \begin{bmatrix} go & g1 \ldots\ldots gn-k & 0 \ldots\ldots 0 \\ 0 & go \quad g1 \ldots\ldots gn-k & \ldots\ldots 0 \\ & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\ & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\ 0 \ldots 0 \ldots\ldots go \ldots g1 & \ldots\ldots\ldots & gn-k \end{bmatrix} \qquad (2.19)$$

Además de su representacion matricial un código un código puede ser especificado en forma compacta por polinomios. La palabra de codigo C = $\{Cn-1, cn-2, \ldots\ldots c1co\}$ puede ser expresada como polinomio C(p) de grado ≤ n -1, de la forma C(p) = cn-1$p^{n-1}p^{n-2}p^{n-3}p^{n-4}$ + ...c1p + c_o. El polinomio generador

de un codigo ciclico (n,k) es g(p) = p^{n-k} + $g_{n-k-1}p^{n-k-1}$+.....+g1p +1. Asimismo, el mensaje puede ser expresado como un mensaje polinomico

X(p) = xk-1p^{k-1}+$x_{k-2}p^{k-2}$ +.....+ x1p + xo, donde$[xk-1xk-2\ldots\ldots\ldots\ldots x1xo]$ representa los k bits de informacion C(p) = x(p)g(p). La multiplicacion de polinimios es equivalente a la convolucion de los coeficientes del polinomio,y entonces la operación de codificacion puede ser expresada como

$$c_i = \sum_{j=0}^{n-k}(xi-j).gi \qquad (2.20)$$

La convolución de secuencias de valor discreto puede ser implementada por una red lineal de registros de desplazamiento,y asi el decodificador para códigos cíclicos es extremadamente simple. Un ejemplo de un codificador cíclico que usa registros de desplazamiento lineales se muestra en la figura 2.8 para el pòlinimio generador g(p) = p^2 + p +1 .

2.6.3 Códigos sistematicos

Se dice que un código es sistematico si el mensaje Xm esta contenido en la palabra de código Cm. La matriz generadora de un código sistematico (n,k) puede ser particionada como:

$$G = [I \mid P] = \begin{bmatrix} 1\ 0\ \ 0\ldots 0 \\ 0\ 1\ 0..\ 0 \\ \ldots\ldots\ldots\ldots \\ 0\ 0\ 0\ 0\ 1 \end{bmatrix} \begin{bmatrix} P11\ P12\ldots\ldots\ldots P1n-k \\ p21\ p22\ldots\ldots\ldots p2n-k \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ pk1\ pk2\ldots\ldots\ldots pkn-k \end{bmatrix} \qquad (2.21)$$

Donde Ik es la matriz identidad k x k y P es una matriz k x (n-k). La matriz comprobadora de paridad asociada con un codigo sistematico es

$$H = [P^T \mid I\, n-k] \tag{2.22}$$

donde P^T es la traspuesta de P. La matriz comprobadora de paridad brinda una manera eficiente de computar la distancia minima, que es el menor número de columnas en H que suman cero.

Dado que cualquier palabra de código del codigo (n,k) es ortogonal a cada fila de la matriz H, se tiene

$$C_m \cdot H^T = 0 \tag{2.23}$$

donde 0 es un vector fila con n - k elementos todos ceros, y C_m es una palabra de codigo del código (n,k), ya que la ecuacion 2.23 se cumple para cada palabra de código del código (n,k), tambien se tiene que

$$GH^T = 0, \tag{2.24}$$

donde 0 es ahora una matriz k x (n-k) con todos sus elementos ceros.

Aprovechando la propiedad de la ecuacion 2.23 y definiendo a la palabra de código recibida a la salida del demodulador como **Y**,se puede afirmar que para tener una comunicación sin error se debe cumplir que $\mathbf{Y}H^T = 0$. Pero **Y** dificilmente es igual a C_m, ya que el canal afecta a la palabra de codigo, añadiendole,un vector de error binario **e**. Entonces **Y** puede ser expresado como

$$Y = C_m + e \tag{2.25}$$

Es asi como el producto YH^T nos lleva a

$$YH^T = (C_m + e)H^T$$

$$= Cm H^T + e H^T$$

$$e H^T = S \qquad (2.26)$$

donde S es un vector de dimension (n-k) que se conoce como el sindrome del patron de error. Los componentes de S son cero cuando los elemnetos de **Y** corresponden a los de una palabra de codigo válida, y son no cero cuando no corresponden. Es posible, entonces asociar a cada sindrome un patron de error, el mas problable, de tal manera que la decodificacion coniste en, una vez calculado el sindrome S, verificar el patron de error **e** asociado mas aparente, y restarlo de la palabra recibida **Y.** Dado que son códigos binarios la sustracción modulo 2 es identica a la adiccion modulo 2, esta operación se puede expresar como

$$Cm = \mathbf{Y} \oplus em \qquad (2.27)$$

El siguiente es un ejemplo de codificación y decodificación de un código lineal sistemático de bloque (7,4)

Ejemplo 2.1.1 Código de bloque (7,4)

$$G = \left.\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}\right\} = \{I4|P\} \qquad (2.28)$$

Una palabra de código típica se expresaría de la forma

$$Cm = \{xm1\ xm2\ xm3\ xm4\ cm5\ cm6\ cm7\}$$

Donde los $\{xmj\}$ representan los cuatro bits de información y los $\{cmj\}$ representan los tres bits comprobadores de paridad, dados por

$$\begin{aligned} Cm_5 &= Xm_1 + Xm_2 + Xm_3 \\ Cm_6 &= Xm_2 + Xm_3 + Xm_4 \\ Cm_7 &= Xm_1 + Xm_2 + Xm_4 \end{aligned} \tag{2.29}$$

De acuerdo a la ecuación 2.22 se tiene para este código sistemático (7,4), que la matriz H es de la forma

$$H = \left(\begin{array}{cccc|ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right) \tag{2.30}$$

El producto $Cm H^T$ lleva a estas tres ecuaciones

$$\begin{aligned} Xm_1 + Xm_2 + Xm_3 + Cm_5 &= 0 \\ Xm_2 + Xm_3 + Xm_4 + Cm_6 &= 0 \\ Xm_1 + Xm_2 + Xm_4 + Cm_7 &= 0 \end{aligned} \tag{2.31}$$

2.7 Códigos Convolucionales

Dado que los constituyentes que forman los codificadores de los Turbo Códigos son Códigos Convolucionales, en este apartado explicaremos detenidamente estos códigos correctores de errores que durante años han sido ampliamente utilizados en multitud de aplicaciones. y es que la propiedad de poder utilizar información soft del canal en tiempo real ha sido la gran ventaja sobre los códigos de bloque y la que ha hecho posible su utilización en muchos sistemas de comunicaciones. Así pues, presentamos a continuación dichos códigos, y más

concretamente los binarios, es decir, aquellos cuyos datos de entrada pueden adquirir los valores {0, 1}.

En el proceso de codificación, los códigos de bloque segmentan la información en bloques de longitud k y los mapean en palabras de código de longitud n. Ambas longitudes k y n son fijas, Esto hace que la palabra de código deba ser recibida completamente para poder decodificarla. Esto puede producir retardos excesivamente grandes cuando la longitud de bloque es larga, lo que resulta intolerable para algunos sistemas. Además, los códigos de bloque requieren una sincronización de marco muy precisa.

2.7.1 Estructura del codificador

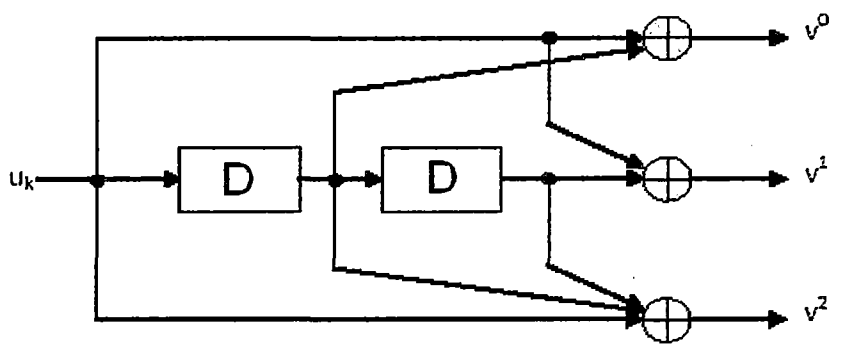
Como ya se ha comentado previamente, la codificación convolucional no necesita disponer de todo un bloque de información a codificar para comenzar dicho proceso. Es más, cada vez que recibimos un bits de información podemos codificarlo teniendo en cuenta únicamente el estado del codificador en ese instante, por lo que vemos que es un tipo de codificación que depende de los bits transmitidos en instante anteriores.

Este hecho no quiere decir que a la hora de codificar finalmente dividamos la información a transmitir en bloques cuya longitud nos sea adecuada, ya que esto facilitara ambos procesos: la codificación y la decodificación.

El proceso de codificación introduce la redundancia aprovechando las características bien conocidas de los registros de estados (LSR, Linear Shift Register). En ellos nos basta conocer la entrada y el estado del codificador para automáticamente obtener la salida. Con tal finalidad un conjunto de m biestable nos almacenara los valores de las m entradas anteriores, obteniendo así un total

de 2^m estados posibles. La secuencia de bits codificada se genera como suma en modulo dos de las diferentes uniones establecidas en el codificador. Un ejemplo es que se muestra en la figura 2.9 donde se observan los biestables y las sumas xor efectuadas.

Fig.2.9 Ejemplo de codificador Convolucional de memoria 2



Existen una serie de parámetros de interés que definen y caracterizan un código convolucional, y son lo siguiente:

- **Tasa de codificación(r)** : Relación entre los bits de entrada (k) y los de salida (n). Suele expresarse en forma de fracción: **k/n**. El codificador del ejemplo anterior tiene una tasa de 1/3, ya que por cada bit de entrada (u_k) tenemos tres salidas (v^o, v^1, v^2).

- **Memoria máxima (m):** El número de biestables del decodificador define la memoria de dicho codificador. Pero el máximo número de biestable dentro de un mismo registro de estado nos define la memoria máxima. Para el caso en el

que **r = 1/n**, es decir, tengamos una sola entrada, los conceptos de memoria y memoria máxima coinciden

- **Constraint Length (k):** Nos indica el número máximo de bits de los cuales depende la salida. Según esta definición: k = m + 1.

Si describimos el proceso de codificación de forma matemática observamos que dicho proceso consiste en realizar tantas convoluciones como salidas tengamos (de ahí el nombre de estos códigos).Por lo que podemos escribir para cada una de las salidas:

$$v_k^n = \sum_{i=0}^{m} gik(\hat{}n).u_{k-1} \qquad (2.32)$$

Donde n representa cada una de las salidas del codificador, k es el instante en el que entra el dato uk, y $g_{i,k}^{(n)}$ representa el polinomio formado por cada una de las conexiones del registro de estados para la salida n. $g_{i,k}^{(n)} = \left\{g_{i,k}^{(n)}\right\}$ son los generadores de los codificadores, expresados en forma octal

❖ Representaciones del codificador

Existen varias formas equivalentes de representar la codificación realizada por un código convolucional.las más utilizadas son:

1.- Forma matricial o polinómica

2.- Diagrama de estados

3.- Diagrama de caminos (Trellis Diagram Representation)

❖ Representación matricial o polinómica

Para tener una representación matricial ó polinómica completa tenemos dos posibilidades: ó bien describimos los polinomios generadores (ó matriz de conexiones) de cada una de de las salidas, ó bien la matriz generadora del código que nos indica las conexiones que se efectúan en cada uno de los biestables.

❖ Polinomios generadores o matriz de conexiones

Describir los polinomios generadores de cada una de las salidas es muy sencillo, ya que solo consiste en asignar los valores '1' o '0' a cada una de las conexiones del LSR dependiendo de si existen ('1') o ('0'), incluyendo la del bit de entrada y teniendo en cuenta que esta es la conexión de menor peso. Así, cada uno de los polinomios tendrá un grado máximo igual a la memoria (m), y vendrá dado según la siguiente expresión:

$$g^i = go^i + g_1^i.\mathrm{D} + \ldots\ldots\ldots\ldots + g_m^i.D^m$$

Donde g_f^j nos indica el valor de la conexión j para la salida i, y tomara el valor '1' si existe esa conexión físicamente y '0' si no existe. Según la estructura del codificador de la figura 2.9 tenemos los siguientes polinomios:

$g^0(\mathrm{D}) = 1 + \mathrm{D}$ $\qquad$ $g^1(\mathrm{D}) = 1 + \mathrm{D}^2$ $\qquad$ $g^2(\mathrm{D}) = 1 + \mathrm{D} + \mathrm{D}^2$

Todos los polinomios del codificador suelen expresarse en forma octal, y cuando mostramos todos los polinomios generadores en forma de matriz estamos hablando de la matriz de conexiones (G). En nuestro ejemplo:

$$G = \{ g^0, g^1, g^2 \} = \{ 6,5,7 \}$$

2.7.2 Matriz generadora del código (G)

La representación de la matriz generadora G (mismo símbolo que para la matriz de conexiones), al igual que los polinomios generadores de cada salida, muestra las conexiones hardware de los LSR que intervienen en las sumas modulo_2 que se efectúan. La diferencia se encuentra en que ahora nos fijaremos en las conexiones efectuadas a la entrada o salida de un mismo biestable para todas las salidas.

En general y para un código convolucional de tasa de codificación k/n las representación matricial G consistirá en una matriz de k filas, donde cada una de ellas contendrá toda la información de cada uno de los LSR del código, cumpliendo (según la notación de la figura 2.9)

$$V = u.G \qquad (2.34)$$

Pero tal y como habíamos mencionado anteriormente, nos interesan los casos en que k=1, por lo que G pasara a ser una matriz de una sola fila, y en ellas tendremos tantos elementos como conexiones posibles, es decir, la memoria del codificador mas uno (m +1):

$$G= [G0, G1, G2 \ldots\ldots\ldots Gm] \qquad (2.35)$$

Cada Gi representa a las conexiones establecidas en cada una de las salidas para el biestable i. Por lo que también las podemos expresar como:

$$Gi = [g_i^o, gi_i^1, g_i^2, g_i^3 \ldots\ldots, g_i^{m-1}] \quad (2.36)$$

Donde g_i^j representa la conexión i de la salida j del LSR, y cuyo valor será '1' si existe la conexión para el biestable i-esimo de dicha salida y '0' en caso contrario. Normalmente esta expresión suele realizarse de forma polinómica, por lo que pasamos a tener:

$$G(D) = Go + G_1D + \ldots\ldots\ldots\ldots + Gm.D^m \quad (2.37)$$

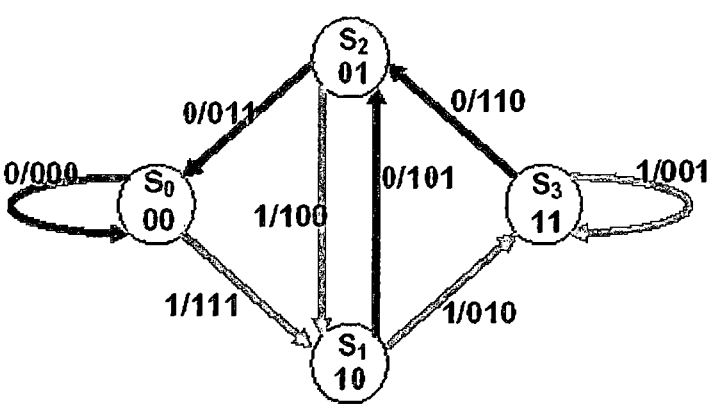
Así, volviendo al caso del codificador de la figura 2.9 tendremos la siguiente matriz generadora

$$G(D) = [1\ 1\ 1] + [1\ 0\ 1].D + [0\ 1\ 1].D^2$$

2.7.3 Diagrama de estados

Esta es una técnica más utilizada para describir el funcionamiento de un código convolucuional. Consiste en mostrar gráficamente las transiciones entre los diferentes estados teniendo en cuenta el estado actual y el bit de entrada. Para el ejemplo de la figura 2.9 tenemos el siguiente diagrama.

Fig . 2.10 Diagrama de estado del Codificador de la figura 2.9



En el diagrama se observa que la información del estado del codificador se encuentra en los círculos. En este caso, al ser el codificador de memoria 2 tenemos cuatro estados posibles S_0=00, S_1=10, S_2=01,y S_3= 11. Cada nuevo bit de entrada causa una transición entre estados, descrita mediante las líneas unidireccionales que unen dichos círculos. La información incluida en cada trayectoria de transición, denotada como u/v, representa el bit de entrada ***u*** y la palabra codificada de salida ***v*** teniendo en cuenta el estado S_i en el que nos encontramos. En este caso tenemos un codificador de rate 1/3, por lo que por cada bit de entrada tendremos 3 de salida (tal como lo indica la fig.2.9).

❖ Diagrama de caminos (TRELLIS)

Esta forma es básicamente una re-descripción del diagrama de estados, pero es muy utilizada debido a que hace más fácil y entendible el proceso de decodificación. Consiste en mostrar todas las posibles transiciones entre estados para cada iteración temporal (puede entenderse como una expansión temporal del diagrama de estados). El trellis siempre comienza en el estado cero (So), debido a que también en la práctica cada vez que se comienza la transmisión de un nuevo bloque debe partirse del estado cero, para así comenzar posteriormente la decodificación desde un estado conocido. Normalmente una leyenda acompaña al trellis para mostrar las transiciones de estados y sus correspondiente entradas y salidas (u /v) de forma general.

He aquí el trellis y la leyenda del codificador de la figura 2.9.

Fig. 2.11. Trellis del codificador de la figura 2.9

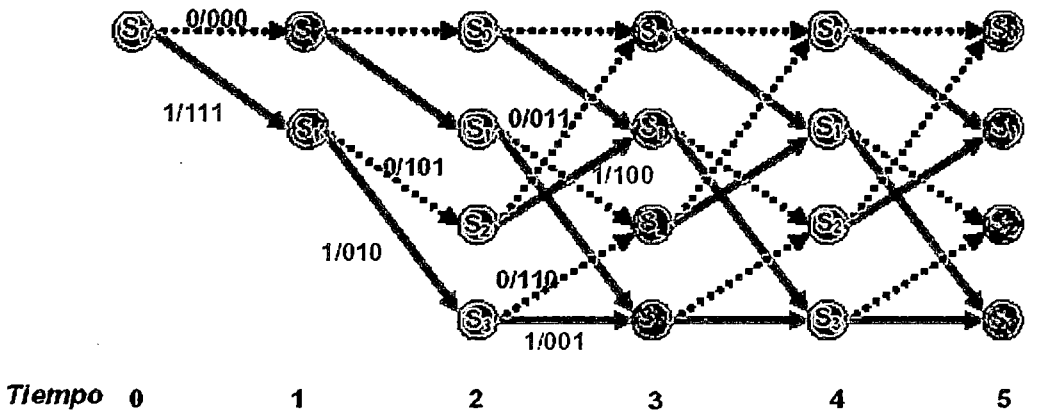
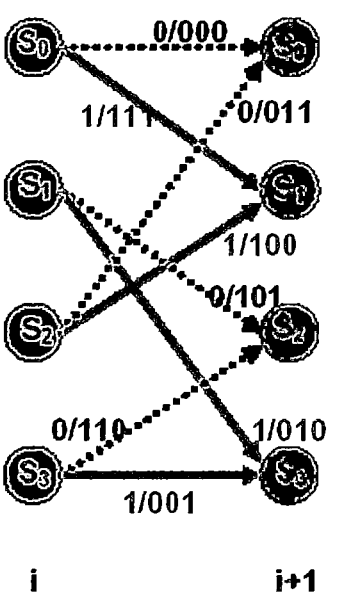


Fig. 2.12. Leyenda del codificador de la figura 2.9.



❖ Terminación del trellis

Ya se ha comentado que la codificación convolucional puede codificar cada dato de entrada de forma instantánea únicamente conociendo el estado actual del registro de estado (y el valor de entrada). Por tanto no necesita disponer de un bloque completo para comenzar a codificar. En la práctica nos será de utilidad ir transmitiendo conjuntos de bloques de información de forma separada, ya que así podremos finalizar cada uno de estos conjuntos en un estado conocido, que por comodidad será el estado cero. Para ello lo único que tendremos que hacer es añadir al final de cada bloque a transmitir tantos ceros como la memoria del codificador.

❖ Punturing o perforación

Muchas veces estaremos utilizando un determinado código con una tasa de codificación k/n (que normalmente por comodidad será de tasa 1/n), pero quizás nos interesara transmitir a una tasa menor. En este caso tendremos dos posibilidades:

Cambiar de codificador o bien realizar un puncturing.

El proceso del puncturing consiste en aumentar la tasa de codificación de un código simplemente no transmitiendo algunos de los bits ya codificados. Para ello seguiremos un orden establecido que vendrá representado por una matriz P, donde los bits que finalmente serán transmitidos vienen representados por unos y los que no por ceros. Así por ejemplo, si estamos trabajando con un código de tasa ½ y nos interesa pasar a una tasa de 2/3 podríamos hacerlo utilizando la siguiente matriz de puncturing.

$$>>P = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

donde cada una de las filas representa cada una de las salidas del codificador. Esta matriz viene a decirnos que de la primera salida transmitiremos todos los bits codificados, mientras que la segunda fila solo uno de cada dos será enviado a través del canal.

2.8 Distancia de los códigos convolucionales

Las distancias de un código de canal son muy importantes, ya que reflejan la capacidad de dicho código para detectar y corregir errores. Existen dos conceptos básicos a tener en cuenta a la hora de hablar de las distancias de un código: peso y distancias de Hamming. El peso de Haming de una secuencia, $W_H(x)$, es el número de posiciones distintas de cero dentro de la secuencia **x**, $d_H(x_1,x_2)$, es el número de posiciones en las que ambas secuencias difieren. Así:

$$d_H(x_1,x_2) = W_H(x_1,x_2) \tag{2.38}$$

La distancia mas inportante de los códigos convolucionales (C) es la distancia libre, que se define como la mínima distancia de Hamming entre dos secuencias :

$$dd_{free} = \min_{v1,v2 \in C}\{dH(v1,v2)\} = \min_{v \in C/0}\{wH(v)\} \tag{2.39}$$

donde la segunda igualdad proviene de la linealidad del cúdigo, ya que la diferencia entre dos secuencias codificadas es tambien una secuencia

codificada. Su inportancia deriva en que esta distancia nos indica la capacidad correcta del codigo convolucional. Podemos llegar a corregir una secuencia erronea (*e*) si :

$$Wh(e) < \frac{dfree}{2} \qquad (2.40)$$

Asi, cuanto mayor sea la distancia libre del código, mejores seran las prestaciones y la capacidad correctoras de dicho código.

2.8.1 RSC vs NSC

Existen varios tipos de codigos convolucionale, auque especialmente dos de ellos poseen un gran interes practico: los códigos convolucionales no sistematico (NSC,Non-Sistematic Convolutional) y los códigos Convolucionales Sistematico y recursivos (RSC,Recursive Systematic Convolutional).

Se dice que un código es sistematico (SC,Systematic Convolutional) si los bits de información que entran al codificador son tambien transmitidos a travez del canal. Si esto no sucede el código sera no sistematico, como ocurre en los códigos convencionales Fig.2.9 .

En general, los codigos NSC presentan mejores prestaciones (menor BER) a altas SNR's (Berrou) debido a que poseen una mayor distancia libres que los RSC. A bajas SNR's normalmente suele suceder lo contrario. Sin enbargo, pese a esta peor prestacion de los códigos SC con altas SNR's, se nos plantea muy interesantes debido a que cuando generamos el codificador del Turbo Código y utilicemos (normalmente) dos códigos convolucionales para ello, solo tendremos que enviar los bits sistematicos del primero de ellos, ya que son los mismos para ambas (aunque entrelazados). Disminuiremos así el overhead que provoca la

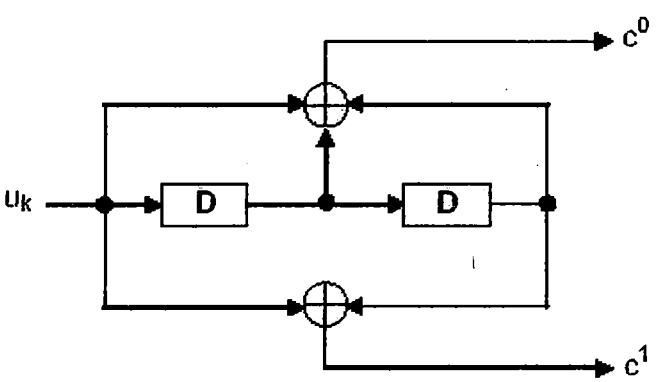
codificacion de canal . Pero además, dentro de la familia de codigos SC encontramos los recursivos (RSC), dónde una de sus salidas realimenta a la entrada. Estos código son interentes porque, por una parte, conseguimos igualar las propiedades de distancia libre respecto los NSC (por lo que conseguiremos resultados similares para altas SNR's, donde los SC empeoraban), y, ademas,a bajas SNR's se comporta como los SC, mejorando las prestaciones de los NSC.

En general y tal como explica Berrou [3] los codigos RSC mejoran las prestaciones de sus equivalentes NSC a cualquier SNR para rates mayores a 2/3. Sin enbargo, dependiendo de la aplicación (particualmente trabajando con turbo códigos) puede interesarnos utilizar códigos parcialmente sistematicos para redurcir el efecto del floor, fenomeno que se explicara más adelante, donde estudiaremos a fondo las justificaciones del redimiento de los turbos codigos.

2.8.2 Obtencion de un RSC a partir de un NSC

Supongamos el codigo NSC de la siguiente figura :

Fig. 2.13 Codificador NSC con m = 2 y rate = ½

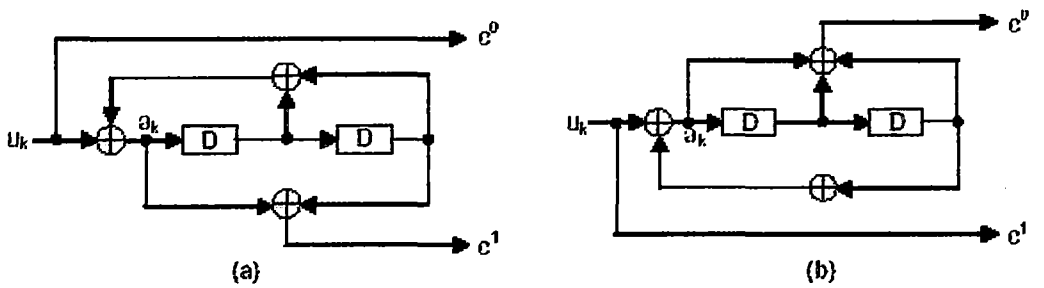


donde tal y como ya sabemos cada una de las salidas es el resultado de realizar la siguiente convolucion:

$$C_k^n = \sum_{i=0}^{n} g_{i,k}^{(n)} \cdot u_{k-1} \text{ , con } g_{i,k}^{(n)} = \{0,1\} \qquad (2.41)$$

Tal y como se ha explicado, los RSC se obtienen de los NSC realimentado la entrada con unas de sus salida. Por otra parte al ser sistematico la entrada sera directamente transmitida a travez del canal. Siguiendo estas instruciones podemos generar los dos RSC que mostraremos a continuacion:

Fig. 2.14 Dos codificadores RSC asociados al NSC de la figura 2.13



donde ahora incorporamos una nueva variable, **ak**, que corresponde con la verdadera entrada al registro de estados y cuyo valor es el que sigue:

$$ak = uk + \sum_{i=0}^{n} \gamma i \; a_{k-i} \qquad (2.42)$$

y que Berrou mostro en [3] que posee las mismas propiedades estadísticas que u_k para los NSC, es decir, adquiere ambos valores $\{0,1\}$ con la misma probabilidad; por lo que se demuestra que ambos códigos (RSC y NSC) poseen la misma distancia libre d_{fre}. En esta expresión aparece Y_i , cuyo valor será $g_i^{(1)}$ si $c_k^o = u_k$ (Fig. 2.14 a), ó bien $g_i^{(2)}$ si $c_k^1 = u_k$.(Fig 2.14 b)

2.8.3 Expresión Polinómica de los RSC

Tal y como hemos podido apreciar, la matriz generadora de conexiones para un código NSC tiene la forma:

$$G_{NSC} = [g1(D), g2(D), \ldots\ldots, gn(D)] \qquad (2.43)$$

Para el caso de los códigos RSC la matriz de conexiones tendrá la forma:

$$G_{NSC} = \left[1, \frac{g2(D)}{g1(D)}, \ldots\ldots \frac{gn(D)}{g1(D)}\right] \qquad (2.44)$$

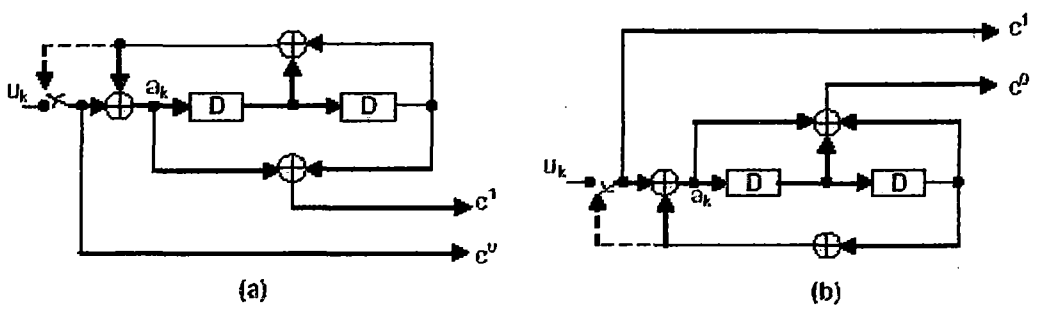
Donde el **1** representa la salida del codificador que adquiere como valor la propia entrada , $g_1(D)$ el polinomio que realimenta a la entrada y $g_j(D)$, para j = 2,.....,n, las restantes salidas del RSC.

❖ Terminación del trellis

En este caso para dejar el codificador en el estado cero no nos servirá introducirle m ceros, ya que al disponer de realimentación no necesariamente el estado final será el cero. Lo que necesitamos por tanto es que el bit que verdaderamente esta en el registro de estados, es decir, a_k, es el resultado de realizar la suma modulo 2 entre le bit de entrada (u_k) y la realimentación. Por lo tanto, si queremos que el resultado de esta suma sea cero, ambos bits sumados tienen que coincidir, por lo que los bits de terminación en los códigos RSC deben introducirse los bits que

conforman el registro de estados en ese momento. Este proceso se realizara mediante la realimentación, que en este caso también será el dato de entrada, como muestra la línea de puntos de los codificadores de la figura 2.15 .

Figura 2.15. Codificadores RSC con hardware preparado para la terminación



❖ Puncturing

Todo lo que se explico anteriormente (Punturing ó perforación) sobre el puncturing en los códigos NSC sigue siendo totalmente valido para los RSC. Pero eso es así, teniendo en cuenta que para que el código sea sistemático los datos de entrada deben enviarse a través del canal. Por tanto la única restricción que existirá en los códigos RSC a la hora de realizar el puncturing es que todos los bits sistemáticos deben ser enviado, pudiéndose aplicar cualquier matriz de puncturing sobre los bits de paridad.

2.9 Proceso de decodificación, Algoritmo de Viterbi

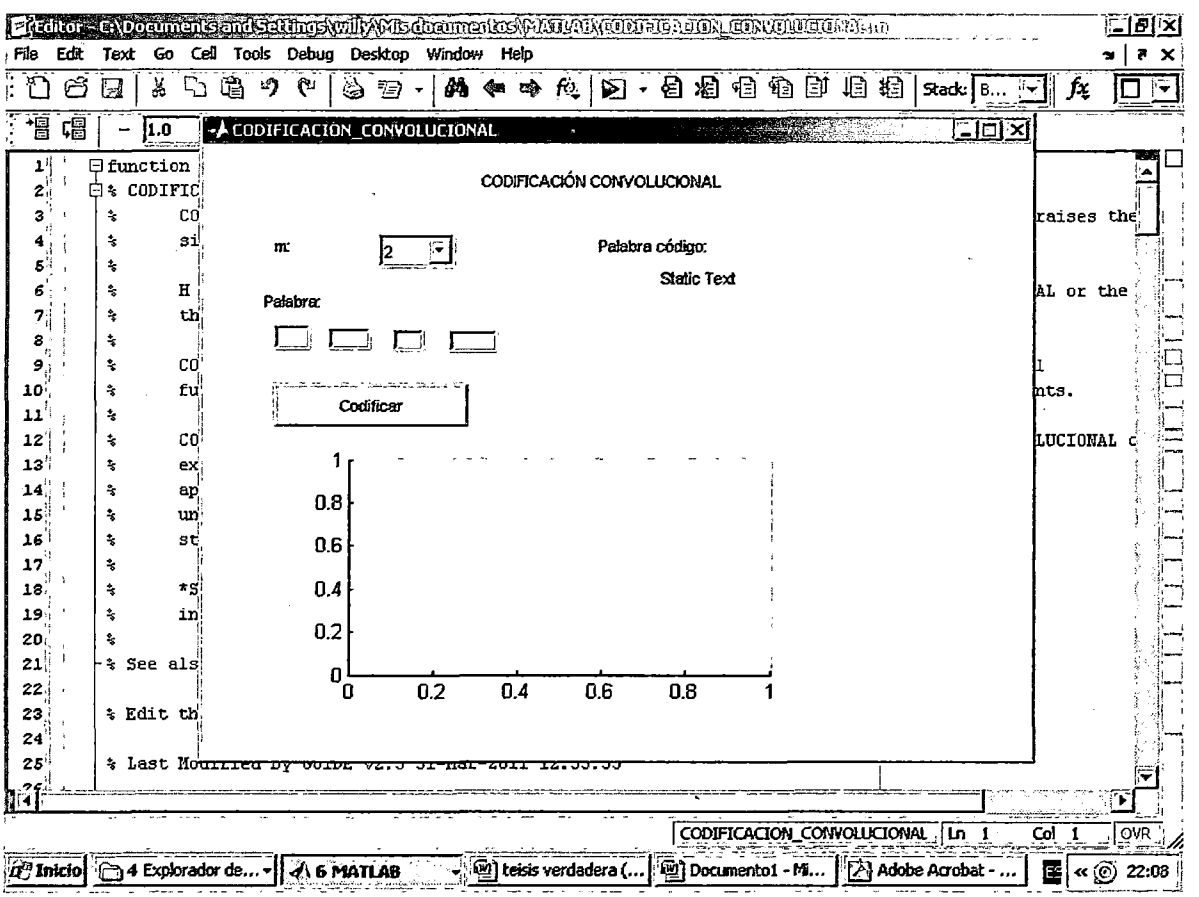
Existen varios algoritmos basados en el trellis para decodificar códigos convolucionales. Entre ellos destaca el que propuso A.J.Viterbi en 1967 [4]y que lleva su mismo nombre. Este es el decodificador optimo que maximiza la probabilidad de la estimación de la secuencia recibida (máxima verosimilitud, ML,Maximun Likelihood),ó lo que es lo mismo, minimiza la probabilidad de error

de la trama. Y se impuso por encima de otros algoritmos también importantes como MAP (algoritmo optimo que obtiene el símbolo mas probablemente transmitido, ó equivalentemente, minimiza la probabilidad de error de símbolo), y que fue durante mucho tiempo olvidado debido a su alta complejidad de implementación, aunque como veremos tendrá una gran importancia en la decodificación de los Turbos Códigos

2.10 Ejemplos de aplicación

Aplicación Nº 1

Es un programa que realiza la Codificación Convolucional de una palabra introducida por el usuario.



Este programa esta implementado con un ciclo for para ir recorriendo la posición en la palabra, y dentro de este ciclo se uso un ciclo if para hacer las operaciones según la memoria elegida m:

```
Desde i=longituddelmensajehasta1
s = [ mensaje (i), y]
y = s(1:m -1)
si m==2
o1 =s(1)

O2 = bitxor(s(1),s(2))
O3 = []
Si  m==3
O1 = bitxor(s(1),s(3))
O2 = bitxor(s(1),s(2))
O3 = []
Si  m==4
O1 = bitxor(s(1),s(2))
O2 = s(4)
O3= bitxor(s(1),s(3))
Fin
```

Elementos del programa

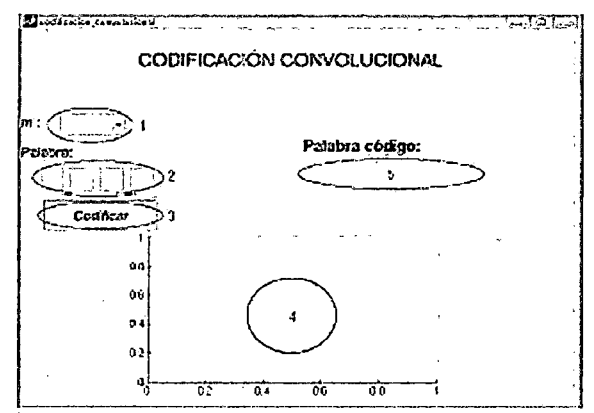


Fig. Elementos de codificación

1.- **m** es donde se elige la memoria que va a tener la codificación: 2,3,ó 4

2.- Palabra. Es donde se introduce la palabra a codificar

3.- Codificar. Es el botón que ejecuta la codificación

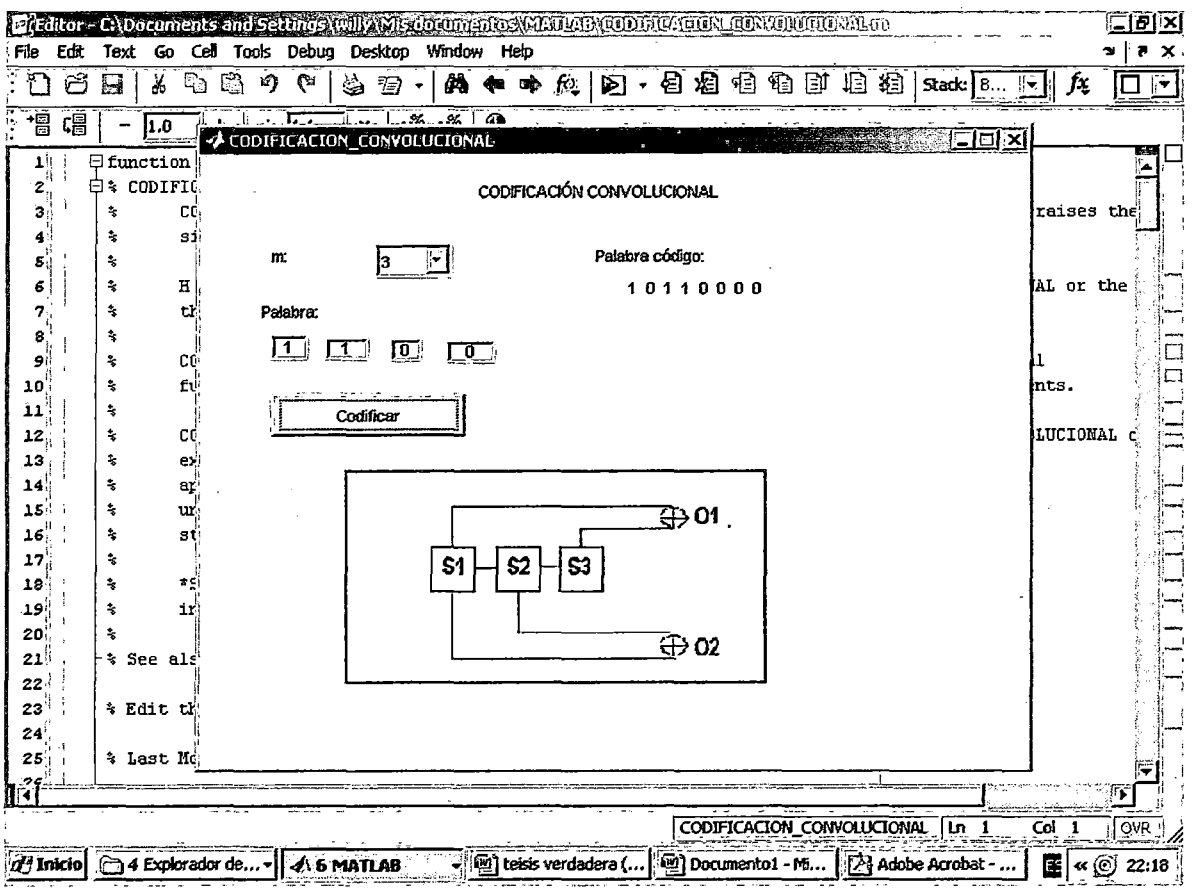
4.- Dibujo del codificador. Es donde se muestra el dibujo del codificador

5.- Palabra código. Es donde se muestra la palabra codificada

Introduciendo en nuestro ejemplo el código **1 1 0 0**, tenemos la palabra codificada

1 0 1 1 0 0 0 0

Figura 2.16 código convolucional



APLICACIÓN Nº 2

Simulación de un sistema de comunicación digital

Presentamos el entorno grafico que simulara un sistema de comunicación digital, el tema de nuestro interés es calcular el BER, define la calidad de un sistema de comunicación digital.

Descripción de la simulación del sistema de comunicación digital

El bloque Bernolli Binary, genera en forma aleatoria bits que va a ser transmitidos, estos provienen de cualquier proceso de conversión análogo digital y en esencia, esta señal binaria es la que nos interesa para la transmisión y para la calidad del sistema.

Los parámetros que tienen son los siguientes: se setea a 0.5 la posibilidad que el software cree un cero 0 ó 1 binario, se coloca como máximo una secuencia de 10^{+06} bits, el software generara un numero de bits menor o igual a dicho valor, se cogen dichos bits a razón de 1 bits por segundo y se coloca a un valor de 3 las tramas, es decir 3 bits por trama ya que se considera una codificación convolucional con una secuencia compuesta por 3 registro de desplazamiento, la trama elegida de 3 bits por trama son las que van a entrar al codificador.

Se coloca el codificador convolucional FEC ½ , es decir por cada bits de entrada el codificador genera 2 bits de salida, al tener como entrada la trama de 3 bits, codifica a 6 bits, los cuales llevan información del bit a codificar y de los dos anteriores a el.

Luego del codificador convolucional, se agrega un bloque que elimina redundancia en los bits y realiza un mapeo de 6 bits, lleva un vector por defecto (**110110**), si la longitud del vector de entrada es menor que la longitud del vector por defecto, se repite el primer vector y se completa con los bits correspondientes

para llegar a los 4 bits, el simulink realiza dicho mapeo, adicionalmente reduce el ancho de banda.

Modualdor BPSK, con fase inicial de 0°, llevando una muestra por símbolo.

Se simula un canal de ruido blanco, aditivo y gaussiano, el ruido blanco es debido al ruido térmico, tiene una energía de bits frente al ruido expresada por Es / No y una potencia en la señal de entrada expresada en vatios, se fija a un periodo de símbolo inicialmente a 1 seg. La relación señal a ruido viene expresada por SNR, se probaran con los valores mínimos permisibles, para poder demodular la señal.

El bloque realiza el mapeo de 4 bits a 6 bits, usa como es lógico un vector idéntico ya que al restablecer los bits usa lógica con or-exclusivo, esto devuelve a los 6 bits que entran al decodificador, el cual mediante el algoritmo de Viterbi los regresa a los 3 bits de la trama inicial.

Algoritmo de decodificación de 2 a 1, es decir por cada 2 bits de entrada el algoritmo devuelve 1 bits, para reducir la posibilidades de errores y corrección de los mismos.

Simulación N°1

Tabla de valores asignados 2.1

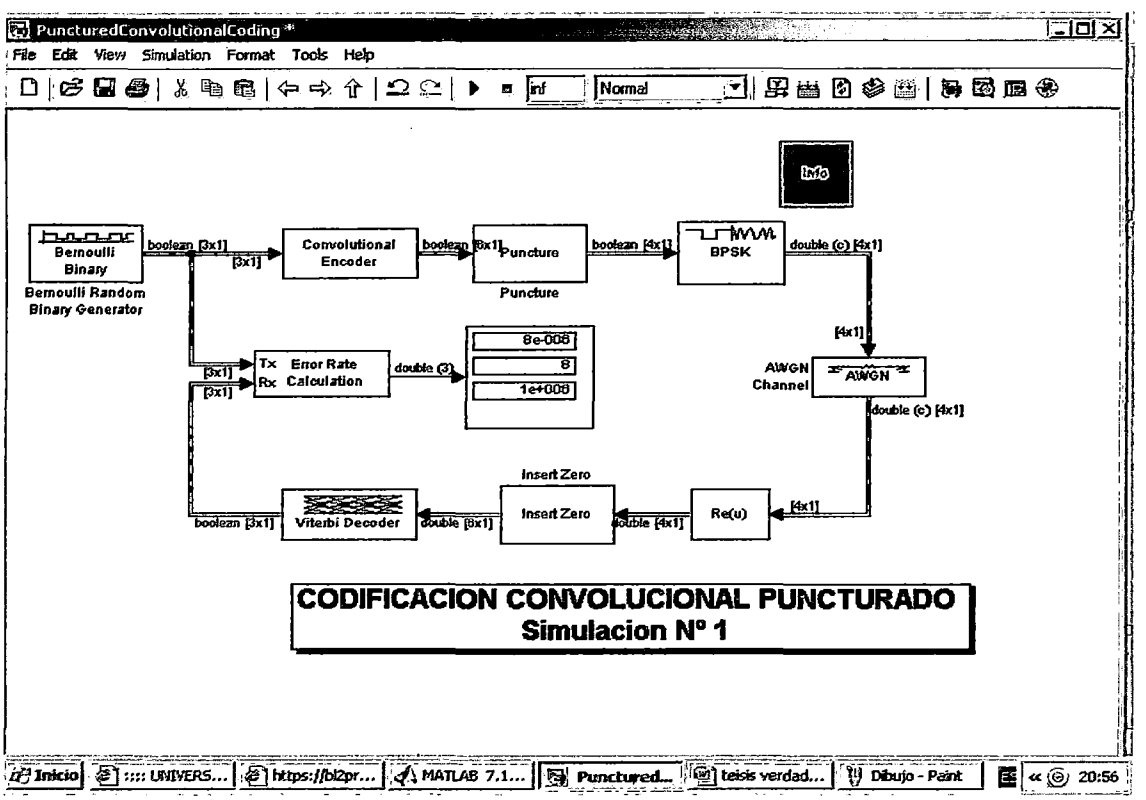
SNR(db)	60
Es / No(db)	6

Tabla de valores obtenidos 2.2

Bits Tx	1E+006	
BER	8E-006	
Bits con errores	(Bits Tx)* BER	8

La tabla Nº 2.1 muestra los valores usados en la simulación Nº 01, los valores de la tabla 2.2 son el resultado de la simulación respectiva, el esquema de simulación se presenta en la siguiente figura.

Fig.2.17 Esquema de la simulación Nº 01



Al correr la simulación se tiene un BER igual 8E-006, esto define una excelente calidad en el sistema de comunicación digital, ya que nos interesa cual es la probabilidad de que los bits transmitidos lleguen con errores, en este caso, al transmitir 1000000 bits (un millón de bits), existe la posibilidad de que 8 de ellos lleguen con error, mostrando una optima calidad de enlace.

Simulación Nº 2

Tabla de valores asignados 2.3

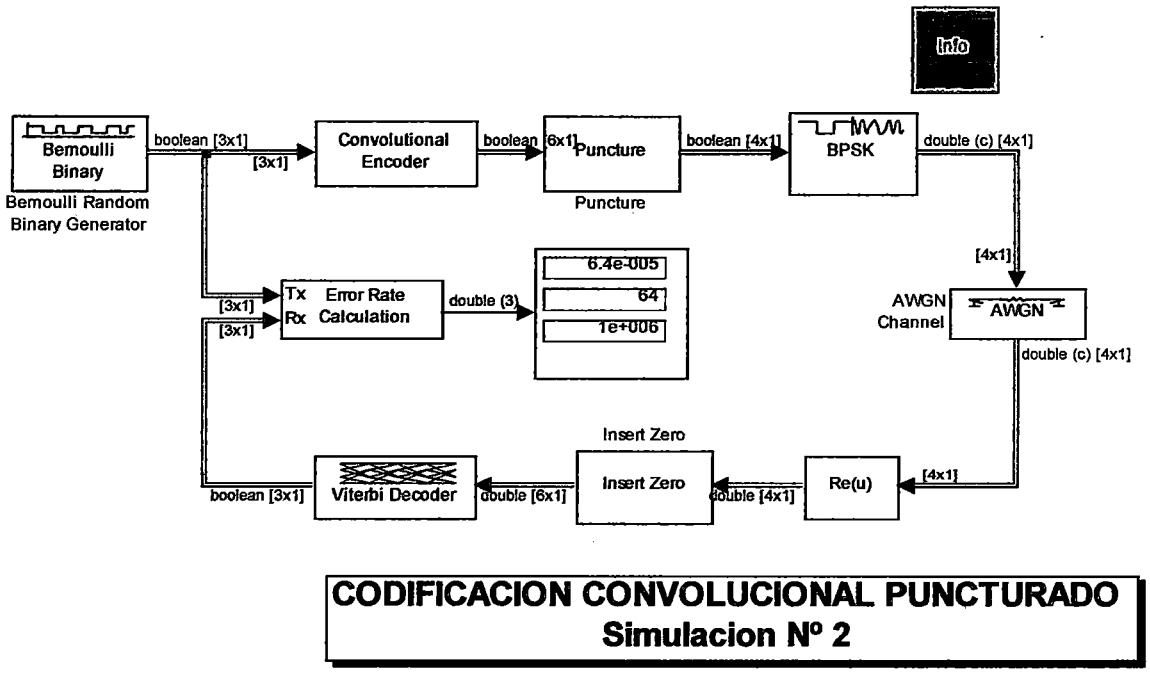
SNR (db)	50
Es / No (dB)	5

Tabla de valores obtenidos 2.4

Bits Tx	1E +006	
BER	6.4E-005	
Bits con errores	64	64

La tabla Nº 2.3 muestra los valores usados en la simulación Nº 02 , los valores de la TABLA 2.4 son el resultado de la simulación respectiva, el esquema de simulación se presenta en la figura .

Fig 2.18 Esquema de simulación 02



Simulación 3

Tabla de valores

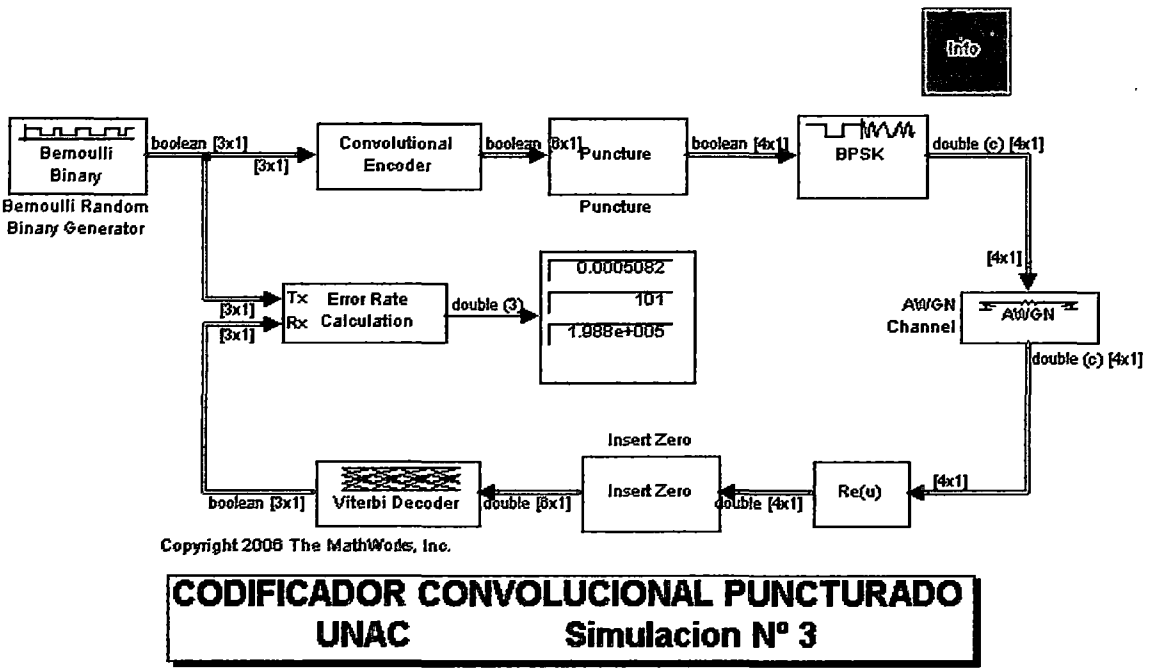
Tabla de valores asignados 2.5

SNR (db)	40
Es / No (db)	4

Tablas de valores obtenidos 2.6

Bits Tx	1.988E+005	
BER	0.0005082	
Bits con errores	(Bits Tx) * BER	101

Fig.2.19 Esquema de simulación 03



Observamos que el sistema de comunicaciones digital se cuelga al contabilizar alrededor de 100 bits errados.

Simulación 4

Tabla de valores asignados 2.7

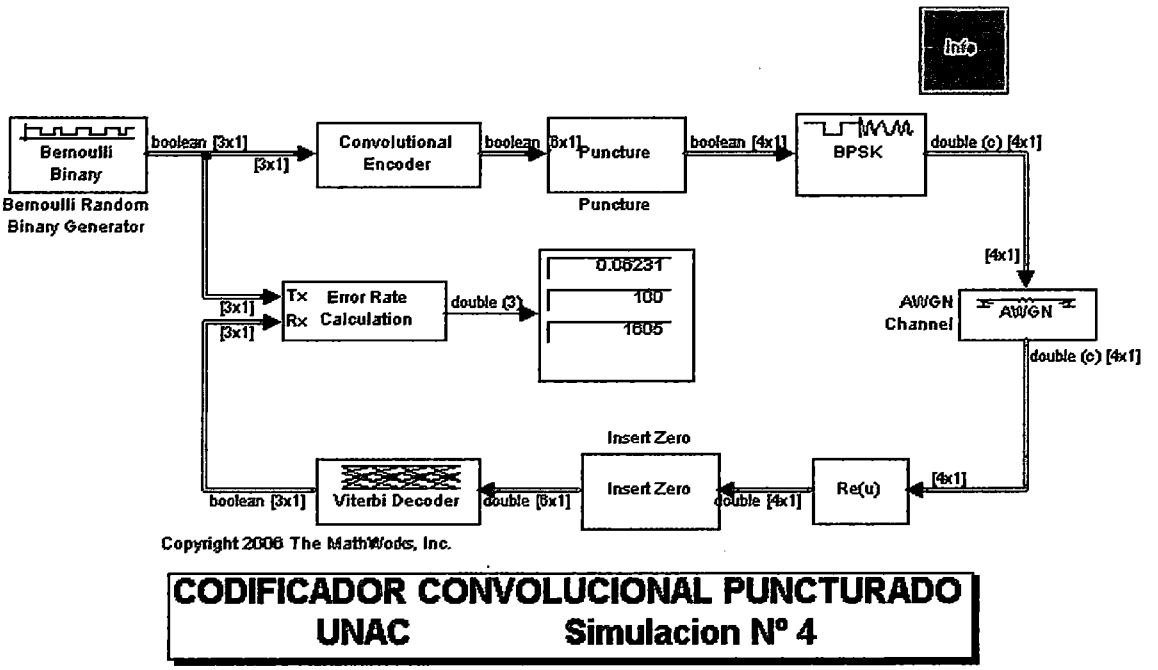
SNR	20
Es / No (db)	2

Tabla de valores obtenidos 2.8

Bits Tx	1605	
BER	0.06231	
Bits con errores	(Bits Tx)*BER	100

La tabla Nº 2.7 muestra los valores usados en la simulación Nº 04 , los valores de la TABLA Nº 2.8 son el resultado de la simulación respectiva, el esquema de simulación se presenta en la figura.

Fig. 2.20 Esquema de simulación Nº 04



Notamos que un valor de SNR= 20 db, es insuficiente para tener una buena calidad de enlace en el sistema de comunicaciones digital, el sistema contabiliza 100 bits errados y automáticamente pasa a un estado de alerta o alarmas de desconexión o caídas de servicio.

CAPITULO III

METODOLOGÍA

3.1 Relación entre las variables de la investigación

Para optimizar el sistema, se necesita que los turbos códigos un nuevo sistema de comunicación digital transmita con el mínimo error posible. Para efectos de nuestro estudio debemos tener en cuenta las siguientes variables.

Variables Dependientes:

1.- Tipo de entrelazadores

2.- Elección de los códigos (RSC).

3.- Puncturado

4.- Algoritmo de decodificación

5.- Longitud de la trama

6.- Nº de iteraciones

7.- Algoritmo de codificación

8.- Tasa del código

Variables independientes:

1.- Peb (probabilidad de error de bit)

2.- SNR (relación señal a ruido)

3.2 Tipo de Investigación

Durante la etapa de diseño del nuevo sistema de codificación turbo códigos basado en los códigos convolucionales, se realizara usando como herramientas matemáticas para el modelado, simulación con el software Matlab 7.0.

y el tipo de Investigación que utilizaremos será transversal científico experimental.

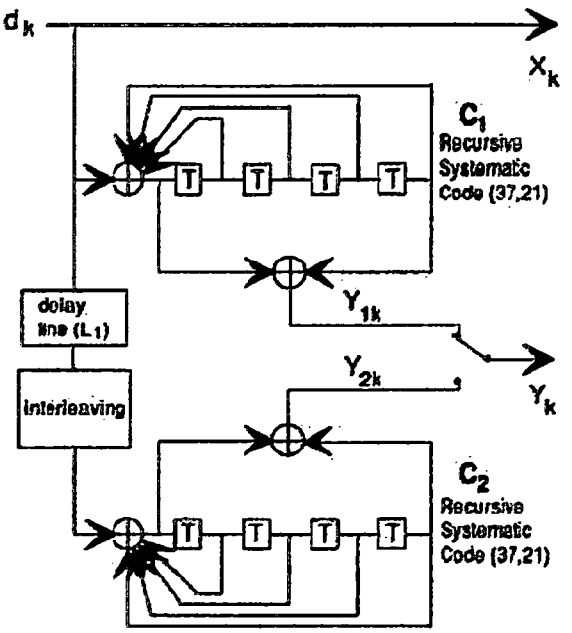
3.3 Diseño de la Investigación

3.3.1 Turbo códigos

Corría el año 1993 cuando en una conferencia del IEEE sobre comunicaciones en Ginebra (Suiza), dos ingenieros electrónicos franceses Claude Berrou y Alanin Glavieux del departamento de electrónica del Ecole National, hicieron una fabulosa aclamación: habían inventado un esquema de codificación digital que podría llevar a un eventual transmisión sin errores y potencias de transmisión eficaces, más de lo que la mayoría de los expertos podían llegar a pensar. Defendían que con este esquema de codificación y haciendo uso de la misma potencia de transmisión se podía llegar a conseguir el doble de velocidad que con otros esquemas de codificación, ó viceversa, con la mitad de potencia en transmisión se conseguían los mismos ratios de transmisión.

Este nuevo esquema de codificación era llamado Turbo Códigos y pertenece a la familia de códigos correctores de errores convolucionales. Esta codificación posee un rendimiento en términos de tasa de error de bits (BER) muy próximos al límite de Shannon [1]. Limite matemático teórico de la máxima tasa de transferencia en entornos ruidosos.

Fig. 3.1. Codificador Turbo con CSR



Fuente: tomado de Borja Álvarez Feíto

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 el ratio del código es:

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

Este cociente representara el tanto por uno de información que existe en la palabra código trasmitida, dando así una visión de la proporción de bits de paridad frente a bits de información.

Una vez que son transmitidos los bits por el canal, estos son expuestos al ruido pudiendo llegar erróneos ó no. La señal analógica en recepción es muestreada y cuantificada en un determinado rango, de forma que el valor más bajo de ese rango indicara que lo recibido será un cero, y el valor más alto del rango que lo recibido es un 1. Por lo que el proceso de cuantificación es modelable mediante una función de densidad de probabilidad por lo que los números negativos representan un 0 y los positivos un 1.

Fig.3.2 Codificador Turbo en bloques

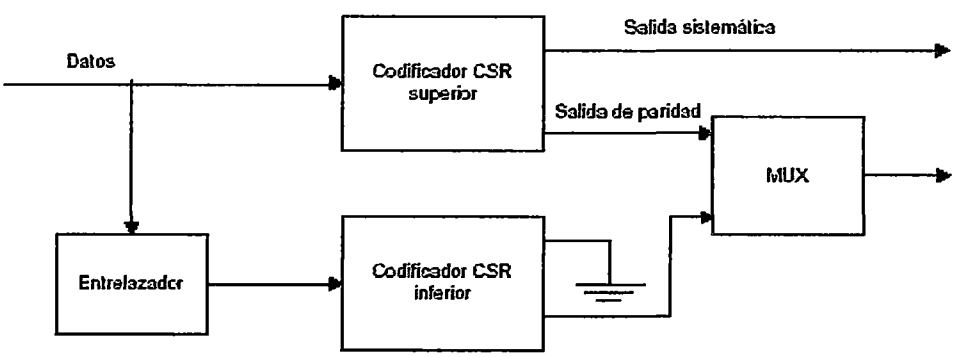
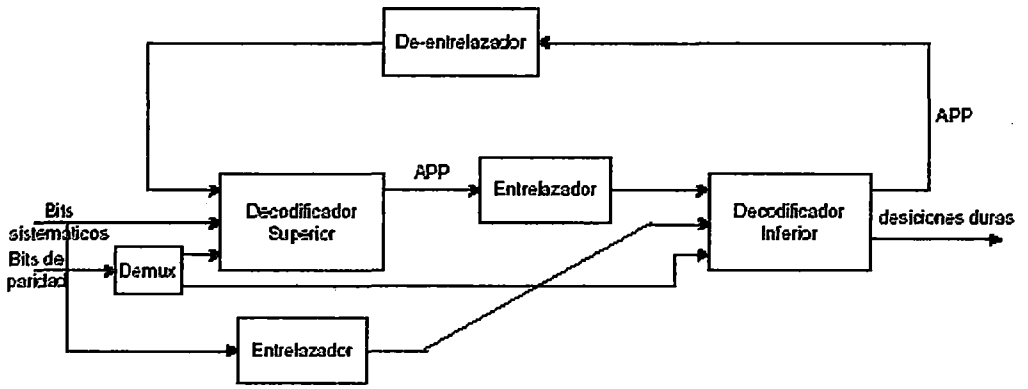


Fig. 3.3 Decodificador en bloques



En la decodificación se produce el proceso que se observa en la figura (3.3). El primer decodificador recoge la cuantificación llevada a cabo para las secuencias de paridad del primer codificador y la cuantificación de los bits de datos; mientras que el segundo decoder hace lo propio con las secuencias de los bits de datos permutados y la secuencia de paridad del segundo codificador. Los decodificadores intercambian información entre si repetidamente, y al cabo de un cierto número de iteraciones (normalmente de 4 a 10) cada uno entrega una secuencia de bits decodificada. Finalmente se suman las secuencias proporcionadas por cada decodificador mas la correspondiente con los datos originales, proporcionando una salida no binaria, sino cuantificada, conociéndose como decisión suave de decodificación. Seguidamente, una decisión dura decide entre 0 y 1.

Existen múltiples algoritmos que optimizan estas decodificaciones, lo más comúnmente usados son los algoritmo de Viterbi y los algoritmo recursivos MAP, que son basados en decisiones suaves llamados SISO (Soft Input,Soft - Output).

Las aplicaciones en las que se pueden encontrar esta codificación son múltiples, como en la tecnología UMTS, el estándar de televisión por satélite DVB-S, comunicaciones espaciales, etc.

3.3.2 Entrelazador

El entrelazador mostrado en el transmisor de la figura 3.2 se ha introducido con el fin de aumentar la distancia libre [1] del diagrama de trellis resultante al tiempo que se esparcen los posibles errores en ráfaga en la fuente de datos de esta manera, el entrelazador cumple la función de evitar que se generen palabras de códigos de bajo peso (medidos como distancia de Haming) simultáneamente en todas las salidas codificadas. Esto último facilita la detección y corrección de errores, ya que ningún camino de cada codificador tendrá un peso menor a la distancia libre, y cuanto mayor sea esta, mayor será la confiabilidad de la decisión.

En el receptor, su función es diferente: es el encargado de poner en fase los bits sistemáticos recibidos, no afectados por el interleaver del transmisor, y la redundancia del segundo codificador, afectada por el interleaver del transmisor. Adicionalmente, debido a que es necesario poner en fase las informaciones extrínsecas intercambiadas por los decodificadores de manera de ser interpretada correctamente, es el interleaver el encargado de cumplir esta función.

Gracias a la presencia del interleaver en el receptor, es posible que ambos decodificadores ejecuten el mismo código de detección independientemente de la secuencia transmitida, sin necesidad de contemplar en la programación de los mismos la existencia y naturaleza de los interleavers. La naturaleza del interleaver

puede ser muy variada; debido a que su función es reordenar los datos, esta operación puede ser realizada de diferentes maneras.

- Formando una matriz con el bloque de datos, leyéndola fila a fila y escribiéndola columna a columna, efectuando una simple permutación, denominado interleaver de bloque
- Reordenando los datos de acuerdo a un algoritmo convolucional, utilizando registros de desplazamiento y multiplexores, denominado interleaver convolucional ó mutiplexado
- En forma pseudoaleatoria, denominado interleaver pseudoaleatorio

Los diversos tipos de interleavers son utilizados bajo distintas condiciones y relaciones de compromiso. En general, un interleaver de bloque tiene mejor performance que uno aleatorio para pequeños tamaños de bloques, es decir menor a 100 bits, invirtiéndose lo antedicho para bloques de grandes tamaños; adicionalmente, el interleaver aleatorio siempre tiene mejor factor de dispersión de errores, denominado factor de spreading, que el interleaver de bloques. En general, la performance de los interleavers pseudoaleatorios mejora al aumentar el tamaño del mismo, en perjuicio del retardo introducido por el almacenamiento y procesamiento de los datos.

3.3.3. Puncturado

La salida sistemática del turbo codificador $c^{(0)}$ se toma del codificador CRC superior. Las dos salidas de paridad $c^{(1)}$ y $c^{(2)}$ se toman de las salidas de paridad de los codificadores CRS superior e inferior respectivamente.Losa tres trenes de salida se multiplexan para formar la palabra de código (despreciando la perdida de tasa fraccional debido a la terminación de trellis) es 1/3. Así como los códigos convolucionales, esta tasa puede ser incrementada por puncturado .En particular, un turbo código de tasa 1/2 puede ser obtenido de un turbo código de tasa 1/3 usando la siguiente matriz de puncturado.

$$P_M = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

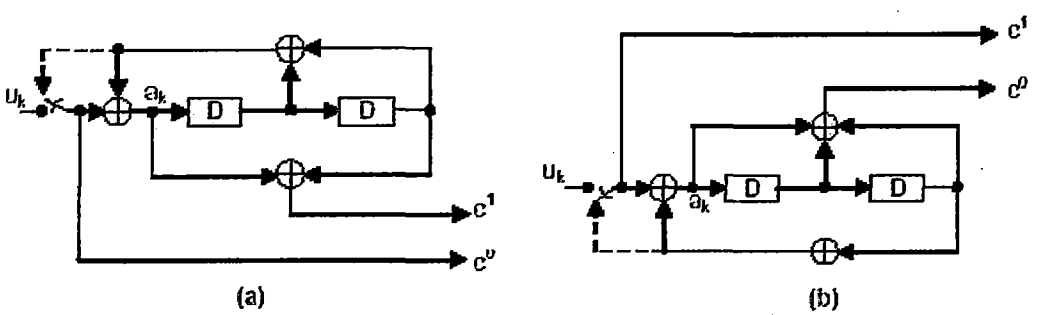
En general como se ha podido apreciar, y se explico en el capitulo precedente, el puncturado es el proceso por el cual se borran sistemáticamente ciertos bits del tren de salida de un codificador. El número de bits borrados determina la nueva tasa del código.

❖ Terminación de Trellis

En este caso para dejar el codificador en el estado cero no nos servirá introducirle m ceros, ya que al disponer de realimentación no necesariamente el estado final será el cero. Lo que necesitamos por tanto es que el bit que verdaderamente entra al registro de estados, es decir, ak, sea cero. ak es el resultado de realizar la suma modulo 2 entre el bit de entrada (uk) y la realimentación. Por tanto, si queremos que el resultado de esta suma sea cero, ambos bits sumados tienen

que coincidir, por lo que los bits de terminación en los códigos RSC deben ser lo de registro de estados. Así, cuando se acabe de codificar los bits del bloque a transmitir deben introducirse los bits que conforman el registro de estados en ese momento. Este proceso se realizara mediante la realimentación, que en ese caso también será el dato de entrada, como muestra la línea de puntos de los codificadores de la figura 3.4.

Fig. 3.4 Codificador RSC con hardware preparado para la terminación



3.3.4 Algoritmo de Decodificación

El número de iteraciones tiene gran impacto en la performance debido a que la mayoría de los algoritmos de decodificación son iterativos. Este concepto de decodificación iterativa se explicara en los resultados finales.

3.3.5 Decodificación Iterativa

Debido a la presencia del entrelazador, la decodificación óptima (máxima likelihood) es increíblemente compleja y por ende impracticable. Sin embargo un algoritmo sub-optimo de decodificación iterativa se presento en [24], el cual ofrece una buena performance a una mucha menor complejidad. La idea de esta estrategia de decodificación es descomponer el problema de decodificación en conjunto, en dos problemas más pequeños (decodificar cada uno de los códigos constituyentes) con soluciones optimas localmente y compartir información de una manera iterativa. El decodificador asociado a cada uno de los códigos constitutivos esta modificado de tal manera que produce salidas suaves en la forma de probabilidades a posteriores (APPs) de los bits de datos. los dos decodificadores están colocados en cascada como se muestra en la figura 3.3 tal que el decodificador inferior recibe la salida suave del decodificador superior. Al final de la primera de la primera iteración, la salida suave del decodificador inferior alimenta al decodificador superior y es usado como información a priori durante la siguiente iteración. La decodificación continúa en una forma iterativa hasta que se obtiene la performance deseada.

3.3.6 Turbo Decodificación

Se ha mostrado anteriormente que los turbos códigos pueden alcanzar una gran performance a bajas SNR debido principalmente a la baja multiplicidad de las palabras de código de distancia mínima. Sin embargo, el algoritmo usado para decodificar los turbos códigos no se ha descrito hasta ahora. Se vera que presentara una introducción al proceso de turbo decodificación. Se mostrara que

un turbo decodificación de dos módulos de entrada y salida suaves que trabajan juntos de una forma iterativa.

3.4 Metódica de cada momento de la investigación

De una forma general, las actividades a realizarse serán: Identificación del problema, selección de la información necesaria, selección de los instrumentos de colecta de datos, definición del procedimiento y métodos de obtención de los datos, resultados e informe final.

Los datos resultantes serán obtenidos por simulación con el software MATLAB.

3.5 Operacionalización de variables

- Pe =f(Tipo de interleaver)
- Pe = f(RSC)
- Pe = f(Algoritmo de Decodificación)
- Pe = f(Longitud de trama)
- Pe = f(tasa de codificación)

3.6 Población y muestra

Por el tipo de investigación no corresponde determinar el tamaño de la muestra Debido a que no se ha determinado el tamaño de la muestra, no corresponde utilizar las técnicas e instrumentos para recolección de datos.

3.7 Técnica e Instrumentos de recolección de datos

Como mencionamos en ítem anterior, la principal herramienta utilizada será un software de simulación matemático MATLAB 7.0 versión Académica desarrollado MATWORK .Es un producto comercial, pero la versión académica es liberada para fines académicos.

3.8 Procedimientos de recolección de datos

Los datos fueron colectados por cada experiencia de simulación

3.9 Procesamiento estadístico y análisis de datos

No corresponde utilizar procesamiento y análisis de datos.

CAPÍTULO IV

RESULTADOS

4.1 Resultados parciales

No aplicable, porque el objetivo es obtener resultados finales propios del simulador.

4.2 Resultados finales (Esquema de simulación)

4.2.1 Influencia de la longitud de la trama

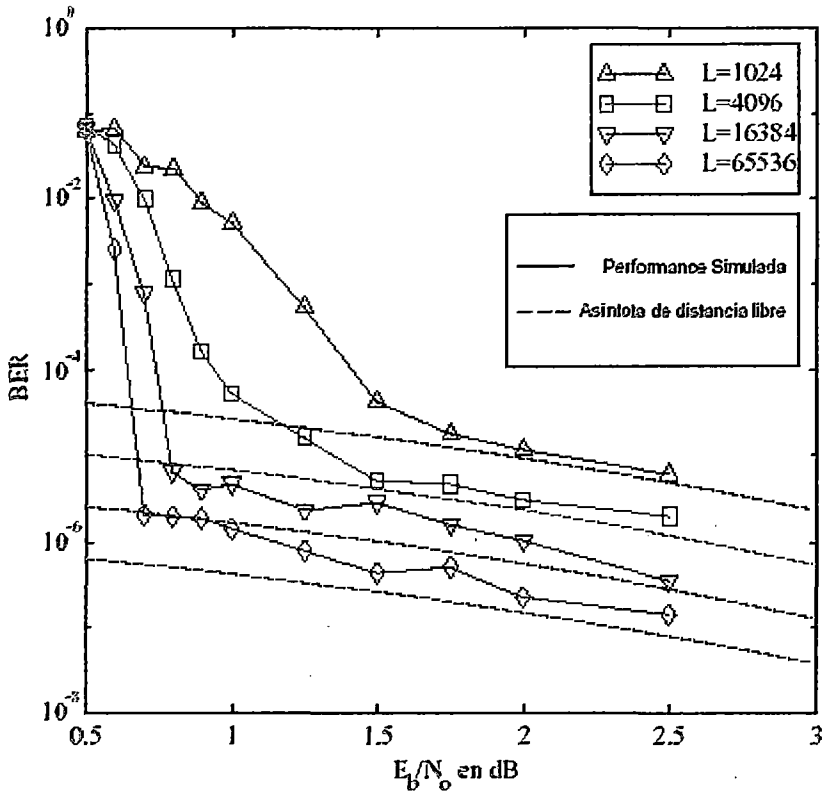
En la siguiente ecuación

$$Pb = \frac{WdminNdmin}{k} Q\left(\sqrt{\frac{dmin.2rEb}{NO}}\right)$$ ver demostración [15]

la longitud del mensaje k aparece en el denominador del coeficiente principal. La longitud del mensaje esta relacionada con la longitud del entrelazador por k = L - Kc +1, y para entrelazadores suficientemente largos, k ~ L. Así, la BER asintótica de un turbo código es inversamente proporcional a la medida de su entrelazador. La asíntota de distancia libre de un turbo código puede ser de de esta manera, bajada, incrementando la medida del entrelazador, ó inversamente, elevada, disminuyendo su medida. La figura 4.1 muestra la performance asintótica y simulada para cuatro diferentes tamaños de entrelazador: L : 1024, L = 16,384 ; y L = 65,536. Se ejecutan 18 iteraciones de decodificación log-MAP, y se fijan 40

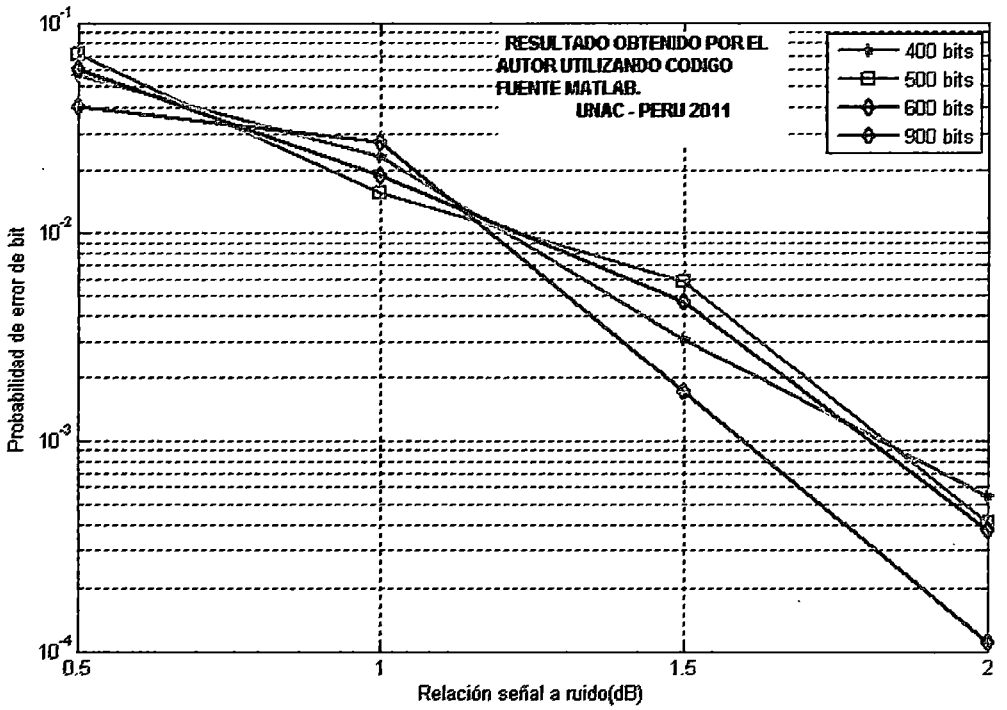
palabras de código erróneas para cada valor de Eb /No. Nótese como la performance de un turbo está fuertemente relacionado con el tamaño del entrelazador. De hecho, el tamaño del entrelazador es uno de los factores que mas afectan la performance de los turbos códigos.

Fig.4.1 Performance asintótica y simulada de un turbo código de tasa r = ½ y longitud de contención Kc = 5 con varias medidas de entrelazadores



Fuente : Valenti, Matthew C. Iterative Detection and Decoding for Wireles Communications, Virginia, 1990

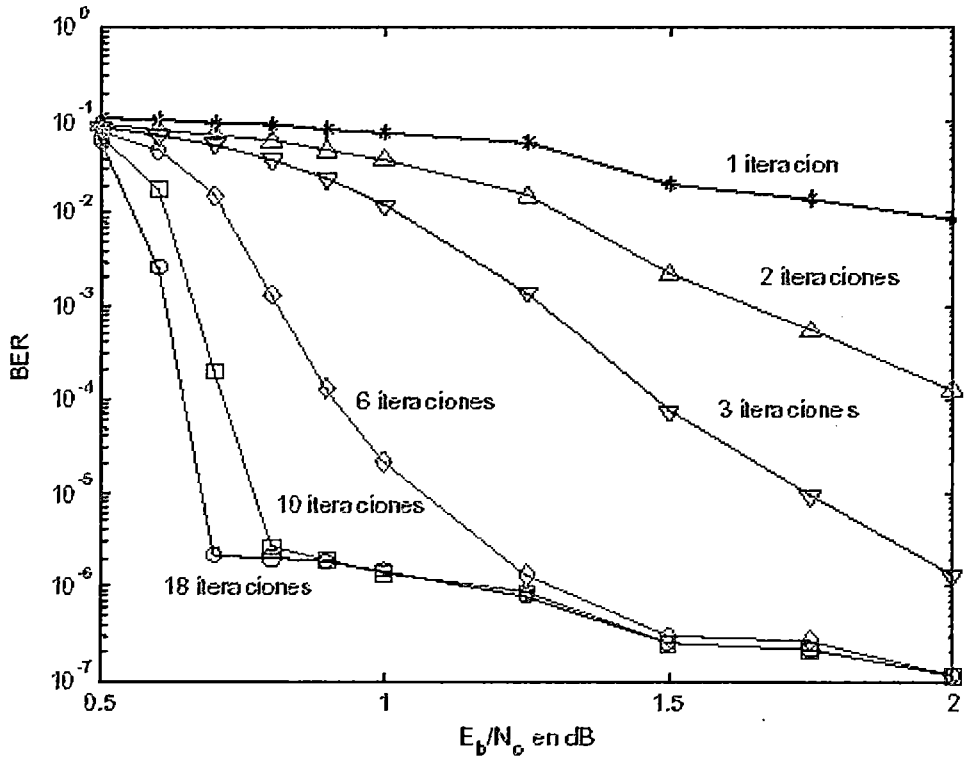
Fig.2.4 Performance del turbo código con el tamaño de la trama



4.2.2 Influencia del número de iteraciones

Como ya se menciono anteriormente el numero de iteraciones es muy importante en el diseño de turbo códigos, ya que como se puede apreciar en la figura 4.2 la performance del turbo código original mejora en función de que aumenta el número de iteraciones.

Fig.4.2 Performance del turbo código original en función del numero de iteraciones.

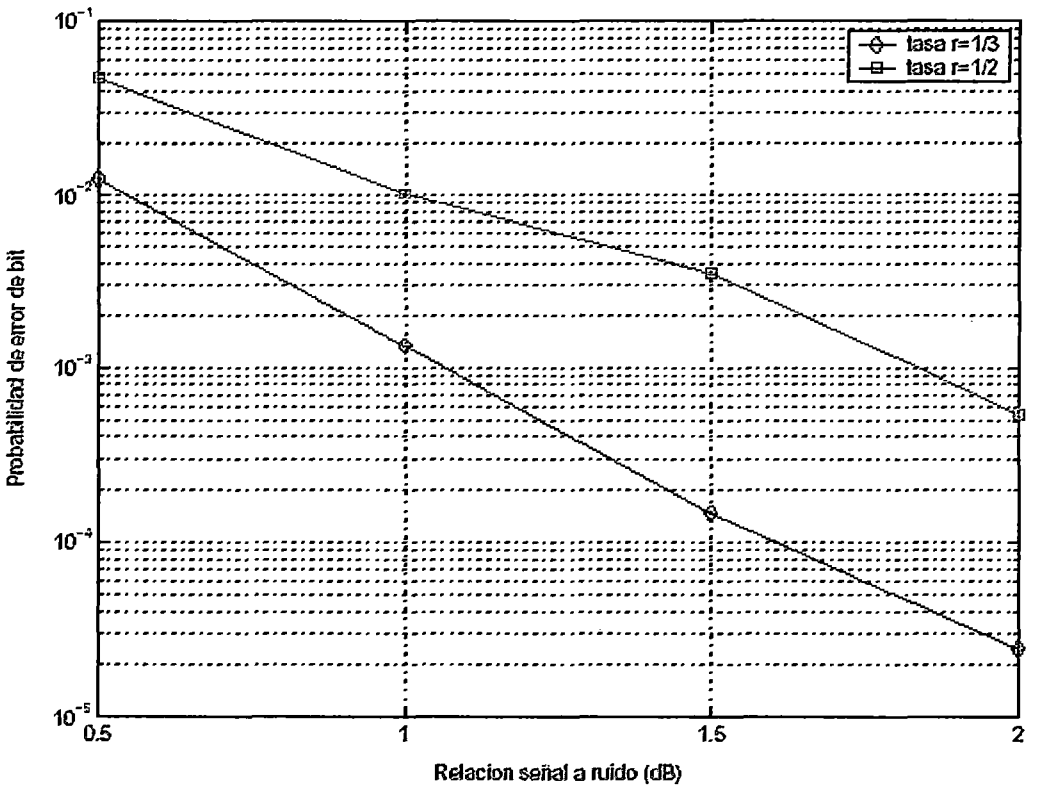


4.2.3 Influencia de la tasa del código

Para estas simulaciones se emplearan tasas de código de r = ½ y r = 1/3. Como ya se menciono anteriormente los códigos de tasa r = ½ se logran puncturado los bits de paridad de los codificadores constituyentes de los de tasa r = 1/3. La performance de los códigos de tasa r= 1/3 comparada a la de los códigos de tasa r= ½ se muestra en la figura 4.3 donde se puede apreciar que a medida que aumenta la tasa del código, la probabilidad de error del bit (BER) disminuye.

La figura 4.3 muestra la performance simulada para dos diferentes tasas de código: r= ½ y r = 1/3. Se ejecutan 9 iteraciones de decodificación log-MAP,y se fijan 15 palabras de código errónea para cada valor de Eb/No.

Fig .4.3 Performance en función de la tasa del código, código de trama de 400 bits. Se fijan 15 palabras de código erróneas para cada valor de Eb / No.



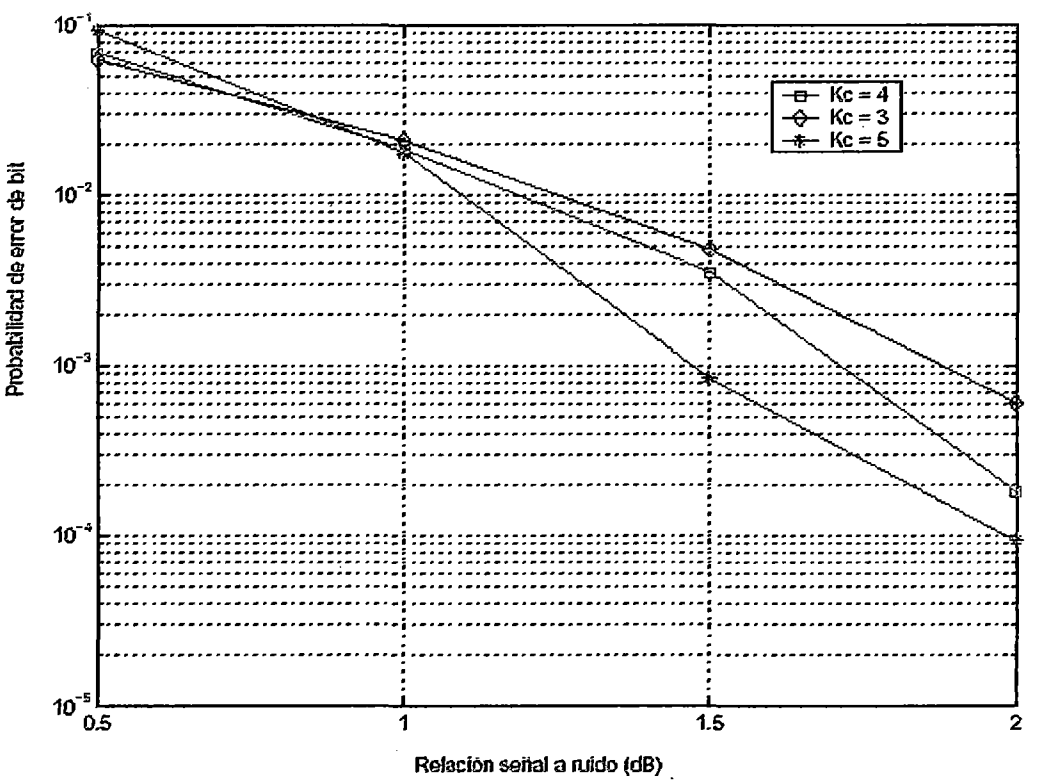
4.2.4 Influencia del código generador

En la figura 4.4 se puede apreciar que al aumentar la longitud de contención para iguales relaciones señal a ruido la probabilidad de error de bit (BER), disminuye. Se puede apreciar que cuando se aumenta la longitud de contención de Kc=3 a Kc=4 y de Kc= 4 a Kc = 5, mejora la performance de los turbos códigos, aunque en valores pequeños (aproximadamente 0.3 db).Sin embargo, la complejidad se incrementa en un factor de 2. Esto se debe a que al aumentar la longitud de contención, como se desprende de lo dicho en la sección anterior 1.2, los bits de los que depende dicha salida aumentan también, por lo tanto el número de operaciones que se deben realizar tanto en el codificador, se incrementan,

proporcionándole al sistema una mayor carga de procesamiento y requiriendo, por tanto, un mayor tiempo para efectuar estas operaciones. Este incremento en la complejidad es de más o menos dos veces entre Kc= 3 y Kc=4, y entre Kc= 5, por lo que se recomienda que, dada la poca mejora lograda, la longitud de contención optima sea de un valor máximo de Kc= 4.

Cabe señalar que las simulaciones se efectuaron con una longitud de trama L= 400,9 iteraciones de decodificación Log-Map y 15 palabras de código erróneas para cada valor de Eb/No.

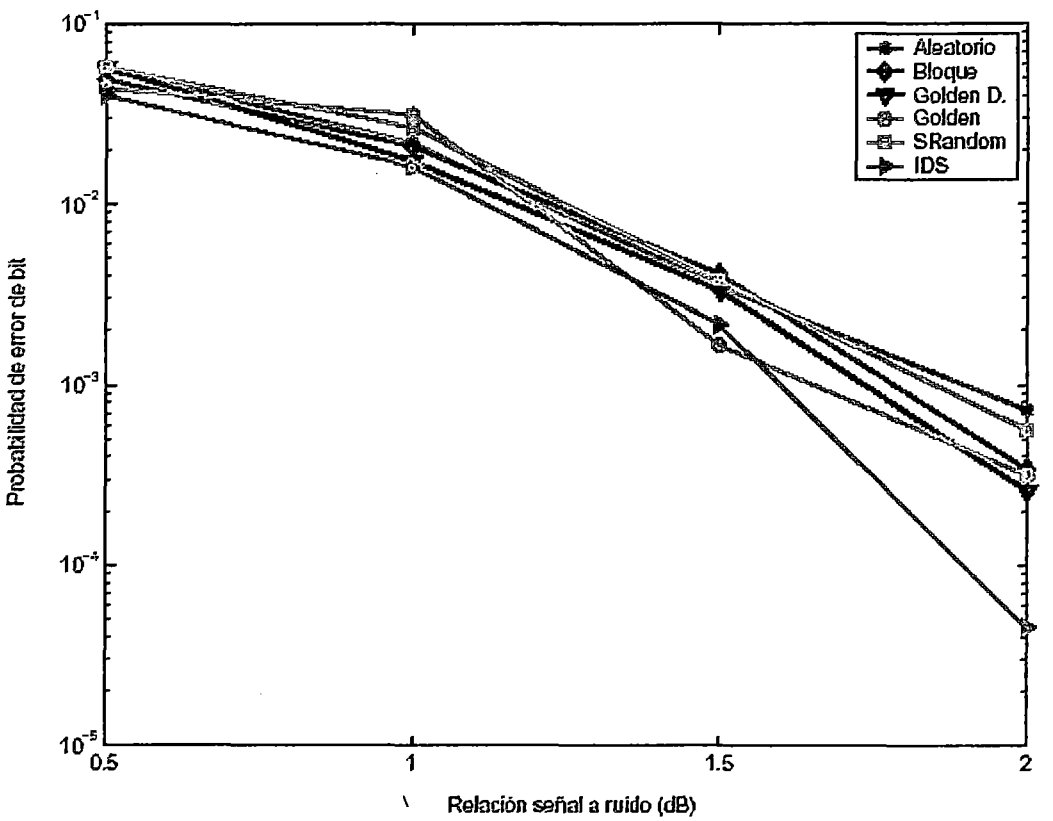
Figura 4.4 Performance en función de la longitud de contención, código de trama de 400 bits.



4.2.5 Influencia del diseño del entrelazador

Se ha simulado el desempeño para diversos tipos de entrelazadores como se puede apreciar en la figura 4.5 donde se han empleado 9 iteraciones de decodificación Log-MAP y 15 palabras de código erróneas para cada valor de Eb /No. Se puede que la performance de los turbos códigos es afectada por el entrelazador empleado.

Figura 4.5: Performance en función del entrelazador, código de trama de 400 bits



El rendimiento del entrelazador y su contribución a la performance de los turbo códigos depende de dos aspectos fundamentales: el espectro de distancia del código y la correlación entre los datos de entrada y salida suave de cada decodificador correspondiente a sus bits de paridad. En este sentido se pueden diseñar entrelazadores donde se maximicen estos dos aspectos, logrando, por lo tanto, un desempeño óptimo.

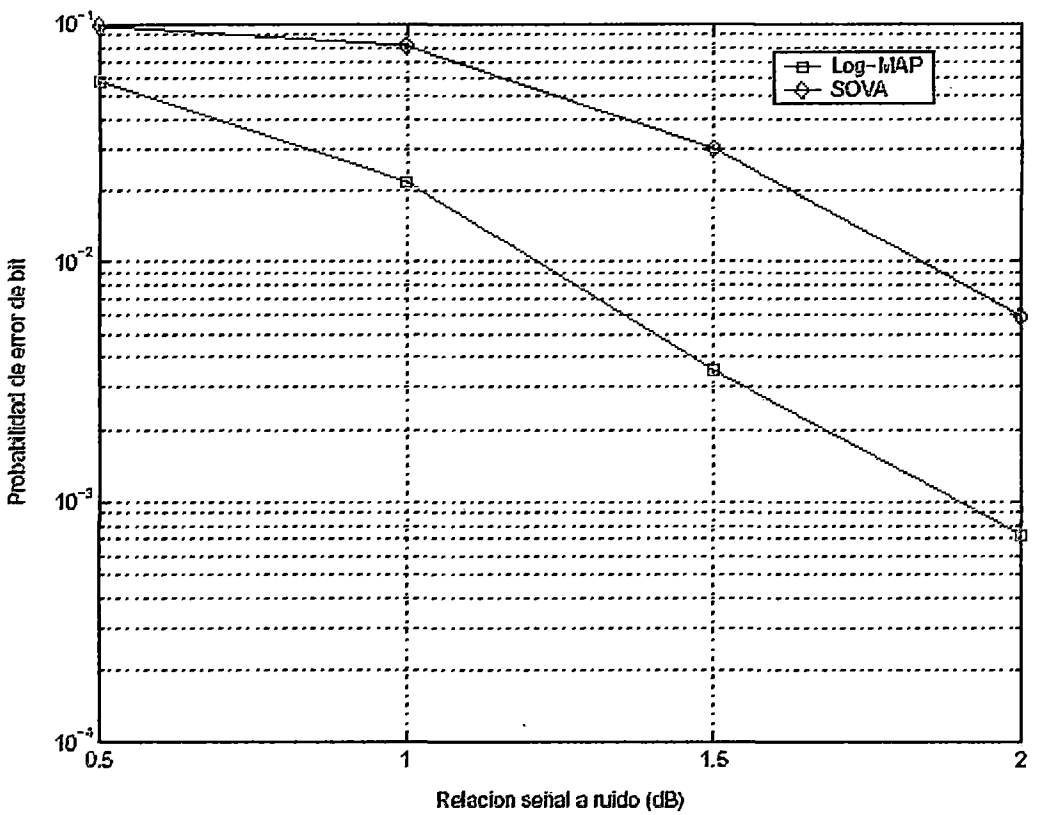
Para el caso de los entrelazadores para tramas largas la mayoría de los entrelazadores aleatorios funcionan adecuadamente. Esto concuerda con los resultados obtenidos por Berrou y su equipo de investigadores franceses.

4.2.6 Comparación entre el algoritmo de decodificación SOVA y LOG-Map

Se puede apreciar que para el algoritmo Log-Map se producen valores menores de probabilidad de error de bit. Por lo tanto logrando una mejor performance como se puede apreciar en la figura 4.6 .Para la simulación se han empleado 9 iteraciones y 15 palabras de código erróneas para cada valor de Eb/No. Empleando para ambos casos código generadores $g^{(0)} = p^3 + p^2 + p + 1$ y $g^{(1)} = p^{(3)} + p + 1$

Si bien el algoritmo SOVA tiene un desempeño inferior al Log-MAP, su implementación es más sencilla, y su tiempo de cálculo es menor, por lo que puede usarse para sistemas que requieren una mayor velocidad de transmisión y tengan un menor costo de implementación por consumir menos recursos computacionales.

Fig. 4.6 : Performance en función del algoritmo de decodificación, código de trama de 400 bits.



CAPÍTULO V

DISCUSIÓN DE RESULTADOS

Contrastación de hipótesis con los resultados

Se puede analizar según los resultados finales que variando los parámetros del codificador obtenemos el rendimiento esperado el cual logra comprobar la hipótesis de partida. La valides del sistema está sujeto a la implementación del código fuente Matlab 7.0.

Contrastación de resultados con otros estudios similares

El futuro de las redes inalámbricas de banda ancha pasa por la nueva tecnología Wimax, capaz de ofrecer grandes velocidades de acceso en un amplio radio de acción. En comunicaciones radio de altas prestaciones la codificación de canal y las técnicas de corrección de errores que se aplican son cruciales para un alto rendimiento de la red.

La tecnología Wimax (IEEE 802.16) recoge como codificación opcional los turbos códigos, con lo que las prestaciones aumentaran.

CONCLUSIONES Y RECOMENDACIONES

El propósito de esta tesis fue dar una introducción general a la teoría de códigos y a los turbos códigos específicamente, explicando la característica que influyen en su rendimiento.

Sobre la teoría de códigos se puede decir que todavía es posible seguir mejorando el rendimiento de los esquemas de codificación, acercándolo aun mas a los límites de capacidad de canal predichos por Shannon. Los turbos códigos originales demostraron estar a solo 0.7dB del límite de Shannon, derribando el mito que ya no era posible seguir mejorando el rendimiento de los códigos prácticos, que se tenían hasta entonces.

Esta performance superior de los turbos códigos es alcanzada solo cuando la longitud del entralazador,o equivalentemente la trama, es muy larga, en el orden de varios miles de bits.

Una de las mayores revoluciones en cuanto a la introducción de los turbos códigos es el proceso de decodificación iterativa, por medio de la cual dos decodificadores constituyentes se turnan para decodificar el mensaje recibido. En cada nuevo intento cada decodificador emplea la salida del otro decodificador. Si es adecuadamente formulado y con la suficiente relación señal a ruido, el proceso de decodificación iterativa es exitoso, logra refinar las salidas del decodificador hasta que los dos codificadores convergen en una misma decisión.

La razón por la cual se emplea la decodificación iterativa se asocia a la todavía grande complejidad de emplear decodificación Maximum Likelihood (ML). La decodificación iterativa provee una decodificación eficiente de un código complejo, a costa de ser un proceso sub optimo comparado con la decodificación Maximum Likelihood.

Finalmente se puede concluir que los turbos códigos, por su gran performance son los códigos que están ya marcando la pauta en los sistemas de comunicaciones de tercera generación y con aplicaciones en muchas ramas de la ciencia, por lo que entenderlos y estar listos para emplearlos en sistemas prácticos de comunicación es una tarea urgente.

Se recomienda en un futuro analizar cómo se comporta los turbos códigos frente a tramas cortas y que aplicaciones presenta los turbos códigos en telecomunicaciones o electrónica.

REFERENCIALES

[1]Andersen,J.D." Turbo Coding for Deep Space Applications ", Proccedings of the 1995 IEEE International Symposium on Information Theory,36 (Setiembre 1995

[2]Benedeto,S.and G.Montorsi. " Performance Evaluation of Turbo Codes, " Electronics Letters,31(3): 163-165(Febrero 1995).

[3]Benedeto,S. G. Montorsi. " Unveiling turbo codes: Some results on parallel concatenated coding schemes," IEEE Transsaction on Information Theory, 42(2):409-428(Marzo 1996).

[4]Bose, R.C and D.K Ray – Chaudhuri. " Further results on error correcting binary group codes, " Informations and control , 3:68-79(marzo 1960).

[5]Bose,R.C and D,K. –Chaudhuri. " On a class of error correcting binary group codes," Informations and control,3:68-79(Marzo 1990).

[6]C.Berror,A. Glavieux and P. Thitimajshima," Near Shannon Limit Error Correcting Coding and Decoding: Turbo-Codes," Proceeding of IEEE ICC 93, 1064 – 1070 (1993).

[7]Collins,O,M." The subtleties and intracies of buildings a constraint length 15 convolutional decoder, " IEEE Transactions on Communications,40:1810-1819 (Diciembre 1992)

[8]Divssalar,D.and F.Pollara. " Multiple turbo codes, "Proc.,IEEE MILCOM, 279-285(Noviembre 1995).

[9]Divsalar,D, and F.Pollara. " Turbo codes for deep – space communications", the Telecommunications and Data Adquisitions Progres Report,42-120:29-39 (febrero 1995)

[10]Divsalar,D.and F.Pollara." Turbo codes for deep-sapace communications," The Telecommunications and Data Adquisitions Progress Report,42-120:29-39

[11]Elias,P." Coding for noisy channel," IRE Conv.Record,4:37-47(1995)

[12]Foney,G.D." The Viterbi algorithm," Proc., Proccedings of the Institute of Electrical and Electronic Engineers,37:657(1949).

[13]Golay,M.J.E."Notes on digital codings," Proceedings of the Institute of Electrical and Electronic Engineers,37:657(1949).

[14]Hamming,R.W." Error Detecting and Correcting Codes," Bell sys.Tech.J., 29:147-160(1950)

[15]Herzberg,H." Multilevel Turbo Coding with Short Interleaver," IEEE Jounal on Select Areas in Communications,16(2):303-309(Febrero 1998).

[16] Desarrollo de un esquema de codificación basado en los turbos códigos definidos en el estándar 3GPP(S-UMTS).

ANEXO A
MATRIZ DE CONSISTENCIA
Título: "RENDIMIENTO DE LOS TURBOS CODIGOS CON DIFERENTES CONFIGURACIONES"

PROBLEMAS	OBJETIVOS	HIPOTESIS	VARIABLES	MÉTODOS
General Se han identificado diferentes niveles de rendimiento en los Turbos códigos en distintas configuraciones. Reconociendo que existen diversos parámetros de estos códigos que se pueden modificar para variar este rendimiento. Identificaremos el aporte que cada uno de esos parámetros para mejorar la performance.	**Objetivo general** El objetivo general del presente trabajo de investigación es : 1.- Identificar los parámetros que afectan el rendimiento de los Turbos códigos **Objetivos específicos** 1.- En qué medida cada uno de esos parámetros afectan el rendimiento de los turbos códigos. 2.-Diseño, simulación y verificación de codificador y decodificador de los Turbos códigos. 3.- Proponer una configuración de Turbos códigos que tenga el mejor rendimiento a través de la combinación de los parámetros seleccionados.	Es posible mejorar el rendimiento de los Turbos códigos a través de la manipulación de los parámetros que constituyen parte de estos códigos.	**Variables Dependientes:** 1.- Tipo de entrelazadores 2.- Elección de los códigos (RSC). 3.- Puncturado 4.- Algoritmo de decodificación 5.- Longitud de la trama 6.- Nº de iteraciones 7.- Algoritmo de codificación 8.- Tasa del código **Variables independientes:** - Peb (probabilidad de error de bit) - SNR (relación señal a ruido)	**General** La metodología empleada consiste en el Modelamiento de un sistema de comunicaciones, diseñado bajo códigos convolucionales y usando simulación de los Turbos códigos en Matlab 7.0 **Específicos** Modelar matemáticamente los códigos convolucionales Modelar los entrelazadores y los códigos RSC

ANEXO B

Glosario de términos

- **UIT** : Unión internacional de telecomunicaciones
- **CDMA** : Acceso múltiple por división de códigos
- **ARQ** : Corrección de error hacia atrás
- **FEC** : Corrección de error hacia adelante
- **AWGN** : Ruido blanco gaussiano
- **No** : Densidad espectral de potencia de ruido
- **Eb** : Energía de bits
- **D / A** : Conversor analógico / digital
- **C** : Capacidad del canal
- **Rb** : Tasa de bits
- **SNR** : Relación señal a ruido
- **dij** : Distancia de Hamming
- **Wi** : Peso de Hamming
- **G** : Matriz generadora de códigos
- **P** : Matriz de paridad
- **S** : Entropía
- **RSC** : Códigos sistemáticos recursivos
- **NSC** : Códigos sistemáticos no recursivos

ANEXO C

Programa principales

1.- La posibilidad de seleccionar entre diversos tipos de entrelazadores y realizar los cálculos y graficas con cada uno de ellos

2.- los valores de relación señal a ruido han sido modificados

3.- La grafica del numero de bits errados por trama trasmitida de la iteración final para cada uno de los valores de la relación señal a ruido (Eb No db)

4.- La grafica final de las curvas de probabilidad de error de bits versus relación señal a ruido para los diferentes valores de relación señal a ruido.

Turbo_syms_demo.m

```
% Este script simula el sistema de turbo codigos clasico
% Simula códigos convolucionales concatenados paralelos.
% Se asumen dos codificadores componentes CSR (Convolucionales Sistemáticos
% Recursivos) de tasa 1/2.
% El primer codificador es terminado con bits de cola. Los Bits (Info + cola)
% son 'mezclados' y pasados al segundo codificador, mientras el
% segundo codificador es dejado abierto sin bits de cola.
% Bits aleatorios de info. se modulan en +1/-1, y se transmiten en canal AWGN.
% Los entrelazadores se pueden seleccionar de diversos tipos.
%
% Se emplea el algoritmo Log-MAP sin cuantizacion o aproximacion.
% Usando ln(e^x+e^y) = max(x,y) + ln(1+e^(-abs(x-y))), Se puede simplificar
% el Log-MAP con una tabla de busqueda para la funcion de correccion.
% Si se usa la aproximacion ln(e^x+e^y) = max(x,y), se tiene MAX-Log-MAP.

clear all

% Escribe los mensajes del Command Window en un archivo de texto
diary turbo_logmap.txt

% Escoger el algoritmo de decodificacion

dec_alg = input(' Ingrese el algoritmo de decodificacion. (0:Log-MAP, 1:SOVA)  default 0
');
if isempty(dec_alg)
  dec_alg = 0;
end

% Longitud de la trama
L_total = input(' Ingrese la longitud de trama (= info + cola, default: 400)  ');
```

```
if isempty(L_total)
  L_total = 400;   % bits de infomación mas bits de cola
end

% Generador del codigo
g = input(' Ingrese el generador de codigo: ( default: g = [1 1 1; 1 0 1 ] )     ');
if isempty(g)
  g = [ 1 1 1;
       1 0 1 ];
end
%g = [1 1 0 1; 1 1 1 1];
%g = [1 1 1 1 1; 1 0 0 0 1];

[n,K] = size(g); m = K - 1; nstates = 2^m;

%puncture = 0, puncturado tasa 1/2;
%puncture = 1, no puncturado tasa 1/3
puncture = input(' Escoja puncturado / no puncturado (0/1): default 0     ');
if isempty(puncture)
   puncture = 0;
end
 % Tasa del codigo
rate = 1/(2+puncture);
%Amplitude de desvanecimiento Fading; a=1 en canal AWGN
a = 1;
% Escoja el tipo de entrelazador
method = input(' Escoja el entrelazador (0:Golden, 1:G. Dithered, 2:Aleatorio, 3:Bloque,
4:Bloque rev, 5:odd-even, 6:3gpp, 7:SRandom) default 2    ');
if isempty(method)
  method = 2;  % entrelazador aleatorio
end
 % Número de iteraciones
niter = input(' Ingrese número de iteraciones por cada trama: default 5      ');
if isempty(niter)
  niter = 5;
end
% Numero de errores de trama a contar como criterio de parada
ferrlim = input(' Ingrese el número de errores de trama para terminar: default 15      ');
if isempty(ferrlim)
  ferrlim = 15;
end

EbN0db = input(' Ingrese Eb/N0 en dB : default [0.5 1 1.5 2.0 2.5 3.0]    ');
if isempty(EbN0db)
  EbN0db = [0.5 1 1.5 2.0 2.5 3.0];
end
 fprintf('\n\n-------------------------------------------------------\n');
if dec_alg == 0
  fprintf(' === Decodificador Log-MAP === \n');
else
```

```
  fprintf(' === Decodificador SOVA === \n');
end
fprintf(' Longitud de trama = %6d\n',L_total);
fprintf(' Generador de codigo: \n'); for i = 1:n
   for j = 1:K
      fprintf( '%6d', g(i,j));
   end
   fprintf('\n');
end
 if puncture==0
  fprintf(' Puncturado, tasa del codigo = 1/2 \n');
else
  fprintf(' No puncturado, tasa del codigo = 1/3 \n');
end
 if method == 8
   fprintf(' Entrelazador IDS \n');
elseif method == 7
   fprintf(' Entrelazador SRandom \n');
elseif method == 6
   fprintf(' Entrelazador 3gpp \n');
elseif method == 5
   fprintf(' Entrelazador odd-even \n');
elseif method == 4
   fprintf(' Entrelazador de bloque reverso \n');
elseif method == 3
   fprintf(' Entrelazador de bloque \n');
elseif method == 2
   fprintf(' Entrelazador aleatorio \n');
elseif method == 1
   fprintf(' Entrelazador Golden Dithered \n');
else
   fprintf(' Entrelazador Golden \n');
end
 fprintf(' Numero de iteraciones =  %6d\n', niter);
fprintf(' Errores de trama para terminar = %6d\n', ferrlim);
fprintf(' Eb / N0 (dB) = '); for i = 1:length(EbN0db)
   fprintf('%10.2f',EbN0db(i));
end
fprintf('\n----------------------------------------------------\n\n');
 fprintf('+ + + + Por favor, espere mientras se da el resultado. + + + +\n');
 for nEN = 1:length(EbN0db)
  en = 10^(EbN0db(nEN)/10);  % convierte Eb/N0 de unidades db a numeros normales
  L_c = 4*a*en*rate;  % valor de fiabilidad del canal
  sigma = 1/sqrt(2*rate*en);  % desviacion estandar del ruido AWGN

% Aclarar los contadores de error de bit y de error de trama
  errs(nEN,1:niter) = zeros(1,niter);
  nferr(nEN,1:niter) = zeros(1,niter);

  nframe = 0;   % resetear el contador de tramas transmitidas
```

```
  while nferr(nEN, niter)<ferrlim
    nframe = nframe + 1;
    x = round(rand(1, L_total-m));    % info. bits
   if method == 8
   alpha = load('IDS256_1317'); % distribucion de bits en el entrelazador IDS
elseif method == 7
   alpha = load('imapSrandom256'); % dist. de bits en el entrelazador Srandom
elseif method == 6
   alpha = entrelazador_3gpp(L_total) +1; % dist. de bits en entrelazador 3gpp
elseif method == 5
   alpha = odd_even_interleaver(L_total-m,1,2,m); % bits en entrelazador odd_even
elseif method == 4
   e = sort(randperm(L_total));
   alpha = back_block(e, 50, 8); % bits en entrelazador de bloque reverso
elseif method == 3
   e = sort(randperm(L_total));
   alpha = interleave(e, 50, 8); % bits en entrelazador de bloque
elseif method == 2
   [temp, alpha] = sort(rand(1, L_total));   % bits en el entrelazador aleatorio
elseif method == 1
   alpha = goldenint(L_total,1); % bits en el entrelazador Golden Dithered
else
   alpha = goldenint(L_total,0);  % bits en el entrelazador Golden
end

   en_output = encoderm( x, g, alpha, puncture ) ; % salida del codificador (+1/-1)
 r = en_output+sigma*randn(1,L_total*(2+puncture)); % bits recibidos
% demultiplexacion para obtener entrada para decodificadores 1 y 2
    yk = demultiplex(r,alpha,puncture);
 % Poner escala a los bits recibidos
    rec_s = 0.5*L_c*yk;
 % Inicializar informacion extrinsica
    L_e(1:L_total) = zeros(1,L_total);
  for iter = 1:niter
% Decodificador Uno
      L_a(alpha) = L_e;  % info a priori.
      if dec_alg == 0
        L_all = logmapo(rec_s(1,:), g, L_a, 1);  % info completa.
      else
        L_all = sova0(2*rec_s(1,:), g, L_a, 1);  % info completa.
      end
      L_e = L_all - 2*rec_s(1,1:2:2*L_total) - L_a;  % info extrinseca.
 % Decodificador Dos
      L_a = L_e(alpha);  % info a priori.
      if dec_alg == 0
        L_all = logmapo(rec_s(2,:), g, L_a, 2);  % info completa.
      else
        L_all = sova0(rec_s(2,:), g, L_a, 2);  % info completa.
      end
      L_e = L_all - 2*rec_s(2,1:2:2*L_total) - L_a;  % info extrinseca.
```

```
% Estima los bits de información.
      xhat(alpha) = (sign(L_all)+1)/2;

% Numero de errores de bit en la iteracion presente
      err(iter) = length(find(xhat(1:L_total-m)~=x));
% Cuenta errores de trama para la iteracion presente
      if err(iter)>0
        nferr(nEN,iter) = nferr(nEN,iter)+1;
      end
    end   %iter

% Grafica el número de bits errados por cada trama transmitida de la
% iteracion final para cada uno de los valores de la relacion señal
% a ruido (EbN0db)
  % stem(nframe, err(niter))
  % xlabel('Trama transmitida');
  % ylabel('Número de bits errados');
  % hold on;

% Número total de errores de bits para todas las iteraciones
    errs(nEN,1:niter) = errs(nEN,1:niter) + err(1:niter);
 if rem(nframe,3)==0 | nferr(nEN, niter)==ferrlim
% Tasa de error de bit (BER)
      ber(nEN,1:niter) = errs(nEN,1:niter)/nframe/(L_total-m);
% Tasa de error de trama (FER)
      fer(nEN,1:niter) = nferr(nEN,1:niter)/nframe;

% Muestra resultados intermedios en proceso
fprintf('************** Eb/N0 = %5.2f db **************\n', EbN0db(nEN));
fprintf('Longitud de Trama = %d, tasa 1/%d. \n', L_total, 2+puncture);
fprintf('%d tramas transmitidas, %d tramas erradas.\n', nframe, nferr(nEN, niter));
fprintf('Tasa de Error de Bit (de la iteracion 1 a la iteracion %d):\n', niter);
for i=1:niter
  fprintf('%8.4e   ', ber(nEN,i));
end
fprintf('\n');
fprintf('Tasa de error de trama (de la iteracion 1 a la iteracion %d):\n', niter);
for i=1:niter
  fprintf('%8.4e   ', fer(nEN,i));
end
fprintf('\n');
fprintf('************************************************\n\n');

 % Grabar resultados intermedios
      save turbo_sys_demo EbN0db ber fer
    end

  end    %while
 % figure
```

```
end       %nEN

% Elaboracion de los graficos. Esta parte es variada de acuerdo a lo que se
% necesite

BER = ber(1:length(EbN0db), niter);
semilogy (EbN0db,BER,'rs-')
grid on; xlabel('Relación señal a ruido(dB)');
ylabel('Probabilidad de error de bit');

diary off
```

Trellis.m

```
function [next_out, next_state, last_out, last_state] = trellis(g)
%

[n,K] = size(g);
m = K - 1;
max_state = 2^m;

%
for state=1:max_state
  state_vector = bin_state( state-1, m );

%
  d_k = 0;
  a_k = rem( g(1,:)*[0 state_vector]', 2 );
  [out_0, state_0] = encode_bit(g, a_k, state_vector);
  out_0(1) = 0;

%
  d_k = 1;
  a_k = rem( g(1,:)*[1 state_vector]', 2 );
  [out_1, state_1] = encode_bit(g, a_k, state_vector);
  out_1(1) = 1;
  next_out(state,:) = 2*[out_0 out_1]-1;
  next_state(state,:) = [(int_state(state_0)+1) (int_state(state_1)+1)];
end

%
last_state = zeros(max_state,2);
for bit=0:1
for state=1:max_state
    last_state(next_state(state,bit+1), bit+1)=state;
    last_out(next_state(state, bit+1), bit*2+1:bit*2+2) ...
      = next_out(state, bit*2+1:bit*2+2);
end
end
```

logmapo.m

```
function L_all = logmapo(rec_s,g,L_a,ind_dec)
%
%
L_total = length(rec_s)/2;
[n,K] = size(g);
m = K - 1;
nstates = 2^m;        %

%
[next_out, next_state, last_out, last_state] = trellis(g);

Infty = 1e10;

%
Alpha(1,1) = 0;
Alpha(1,2:nstates) = -Infty*ones(1,nstates-1);

%
if ind_dec==1
  Beta(L_total,1) = 0;
  Beta(L_total,2:nstates) = -Infty*ones(1,nstates-1);
elseif ind_dec==2
  Beta(L_total,1:nstates) = zeros(1,nstates);
else
  fprintf('ind_dec is limited to 1 and 2!\n');
end

%
for k = 2:L_total+1
for state2 = 1:nstates
    gamma = -Infty*ones(1,nstates);
    gamma(last_state(state2,1)) = (-rec_s(2*k-3)+rec_s(2*k-2)*last_out(state2,2))....
        -log(1+exp(L_a(k-1)));
    gamma(last_state(state2,2)) = (rec_s(2*k-3)+rec_s(2*k-2)*last_out(state2,4))....
        +L_a(k-1)-log(1+exp(L_a(k-1)));

if(sum(exp(gamma+Alpha(k-1,:)))<1e-300)
      Alpha(k,state2)=-Infty;
else
      Alpha(k,state2) = log( sum( exp( gamma+Alpha(k-1,:) ) ) );
end
end
   tempmax(k) = max(Alpha(k,:));
   Alpha(k,:) = Alpha(k,:) - tempmax(k);
end

%
for k = L_total-1:-1:1
```

```
for state1 = 1:nstates
    gamma = -Infty*ones(1,nstates);
    gamma(next_state(state1,1)) = (-rec_s(2*k+1)+rec_s(2*k+2)*next_out(state1,2))....
        -log(1+exp(L_a(k+1)));
    gamma(next_state(state1,2)) = (rec_s(2*k+1)+rec_s(2*k+2)*next_out(state1,4))....
        +L_a(k+1)-log(1+exp(L_a(k+1)));
if(sum(exp(gamma+Beta(k+1,:)))<1e-300)
Beta(k,state1)=-Infty;
else
      Beta(k,state1) = log(sum(exp(gamma+Beta(k+1,:))));
end
end
  Beta(k,:) = Beta(k,:) - tempmax(k+1);
end

%
for k = 1:L_total
for state2 = 1:nstates
    gamma0 = (-rec_s(2*k-1)+rec_s(2*k)*last_out(state2,2))....
        -log(1+exp(L_a(k)));
    gamma1 = (rec_s(2*k-1)+rec_s(2*k)*last_out(state2,4))...
        +L_a(k)-log(1+exp(L_a(k)));
    temp0(state2) = exp(gamma0 + Alpha(k,last_state(state2,1)) + Beta(k,state2));
    temp1(state2) = exp(gamma1 + Alpha(k,last_state(state2,2)) + Beta(k,state2));
end
  L_all(k) = log(sum(temp1)) - log(sum(temp0));
end
```

Sova0.m

```
function L_all = sova(rec_s, g, L_a, ind_dec)
%
L_total = length(L_a);
[n,K] = size(g);
m = K - 1;
nstates = 2^m;
Infty = 1e10;

%
delta = 30;

%
[next_out, next_state, last_out, last_state] = trellis(g);

%
for t=1:L_total+1
for state=1:nstates
    path_metric(state,t) = -Infty;
end
```

```
end

%
path_metric(1,1) = 0;
for t=1:L_total
  y = rec_s(2*t-1:2*t);
for state=1:nstates
    sym0 = last_out(state,1:2);
    sym1 = last_out(state,3:4);
    state0 = last_state(state,1);
    state1 = last_state(state,2);
    Mk0 = y*sym0' - L_a(t)/2 + path_metric(state0,t);
    Mk1 = y*sym1' + L_a(t)/2 + path_metric(state1,t);

if Mk0>Mk1
      path_metric(state,t+1)=Mk0;
      Mdiff(state,t+1) = Mk0 - Mk1;
      prev_bit(state, t+1) = 0;
else
      path_metric(state,t+1)=Mk1;
      Mdiff(state,t+1) = Mk1 - Mk0;
      prev_bit(state,t+1) = 1;
end

end
end

%
if ind_dec == 1
  mlstate(L_total+1) = 1;
else
  mlstate(L_total+1) = find( path_metric(:,L_total+1)==max(path_metric(:,L_total+1)) );
end

%
for t=L_total:-1:1
  est(t) = prev_bit(mlstate(t+1),t+1);
  mlstate(t) = last_state(mlstate(t+1), est(t)+1);
end

%
for t=1:L_total
  llr = Infty;
for i=0:delta
if t+i<L_total+1
      bit = 1-est(t+i);
      temp_state = last_state(mlstate(t+i+1), bit+1);
for j=i-1:-1:0
        bit = prev_bit(temp_state,t+j+1);
        temp_state = last_state(temp_state, bit+1);
```

```
end
if bit~=est(t)
        llr = min( llr,Mdiff(mlstate(t+i+1), t+i+1) );
end
end
end
  L_all(t) = (2*est(t) - 1) * llr;
end
```

encoderm.m

```
function en_output = encoderm( x, g, alpha, puncture )
%

[n,K] = size(g);
m = K - 1;
L_info = length(x);
L_total = L_info + m;

%
input = x;
output1 = rsc_encode(g,input,1);

%

y(1,:) = output1(1:2:2*L_total);
y(2,:) = output1(2:2:2*L_total);


%
for i = 1:L_total
  input1(1,i) = y(1,alpha(i));
end
output2 = rsc_encode(g, input1(1,1:L_total), -1 );
y(3,:) = output2(2:2:2*L_total);

%


if puncture > 0    %
for i = 1:L_total
for j = 1:3
        en_output(1,3*(i-1)+j) = y(j,i);
end
end
else%
for i=1:L_total
     en_output(1,n*(i-1)+1) = y(1,i);
if rem(i,2)
%
```

```
        en_output(1,n*i) = y(2,i);
else
%
        en_output(1,n*i) = y(3,i);
end
end
end

%
en_output = 2 * en_output - ones(size(en_output));
```

encode_bit.m

```
function [output, state] = encode_bit(g, input, state)
%
[n,k] = size(g);
m = k-1;

%
for i=1:n
  output(i) = g(i,1)*input;
for j = 2:k
    output(i) = xor(output(i),g(i,j)*state(j-1));
end;
end

state = [input, state(1:m-1)];
```

demultiplex.m

```
function subr = demultiplex(r, alpha, puncture);

L_total = length(r)/(2+puncture);

%
if puncture == 1       %
for i = 1:L_total
    x_sys(i) = r(3*(i-1)+1);
for j = 1:2
        subr(j,2*i) = r(3*(i-1)+1+j);
end
end
else%
for i = 1:L_total
     x_sys(i) = r(2*(i-1)+1);
for j = 1:2
        subr(j,2*i) = 0;
```

```
end
if rem(i,2)>0
        subr(1,2*i) = r(2*i);
else
        subr(2,2*i) = r(2*i);
end
end
end

%
for j = 1:L_total

%
  subr(1,2*(j-1)+1) = x_sys(j);
%
  subr(2,2*(j-1)+1) = x_sys(alpha(j));
end
```

Programas adicionales correspondientes a los entrelazadores

Odd_even_interleaver

```
%odd_even_interleaver.m

function alpha_rsc = odd_even_interleaver(DLength,TerminateF,NEncoders,m)

% Esta funcion implementa en entrelazador odd-even(impar_par)
% alpha_rsc = modd_even_interleaver(DLength,TerminateF,NEncoders,m)

if TerminateF
   index_odd = [1:2:DLength+1];
   index_even = [2:2:DLength+m];
else
   index_odd = [1:2:DLength];
   index_even = [2:2:DLength];
end% if TerminateF
Lodd = length(index_odd); Leven=length(index_even) ; for i1=1:NEncoders-1
   [dummy,alpha_tmp1(1,:)]=sort(rand(1,Lodd));
    [dummy,alpha_tmp2(1,:)]=sort(rand(1,Leven));
if (Lodd>Leven)
       alpha_rsc(i1,:)=...
          reshape([index_odd(alpha_tmp1);index_even(alpha_tmp2),NaN],...
          1,2*Lodd);
else
       alpha_rsc(i1,:)=...
          reshape([index_odd(alpha_tmp1);index_even(alpha_tmp2)],1,2*Lodd);
end% if (Lodd>Leven)
end% for i1= 1:NEncoders-1

if(Lodd>Leven)
   alpha_rsc = alpha_rsc(:,1:end-1);
end% if (Lodd>Leven)
```