Proyecto Final de Carrera

Implementación de técnicas de estimación y sincronización para sistemas OFDM

(Aplicación directa sobre la implementación de la capa física del estándar DVB -T)

Memoria del Proyecto Final de Carrera de Ingeniería de Telecomunicaciones

Realizada por **Marcos Alsinella Fernández**.
Dirigida por Jose A. López Salcedo

Bellaterra, 18 de Septiembre de 2008

Universitat Autònoma de Barcelona

etse
Escola Tècnica Superior d'Enginyeria

El sotasignat, Jose A. López Salcedo

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en Marcos Alsinella Fernández

I per tal que consti firma la present.

Signat: Jose A. López Salcedo

Bellaterra, 18 de Setembre de 2008

AGRADECIMIENTOS

Quiero dedicar unas líneas de esta memoria a dar las gracias a mis padres y a mi hermano por haberme apoyado y animado en todo momento, procurando lo mejor para mi y facilitándome las cosas en la medida de lo posible, no sólo durante la elaboración de este proyecto sino que a lo largo de toda mi vida. Os quiero.

A todos mis amigos, en especial mención a los más “pájaros”, por estar y haber estado ahí.

A mis amigos de la facultad, con quien he crecido y de los cuales siempre he recibido apoyo. Espero que el término "de la facultad" desaparezca para que se convierta simplemente en "mis amigos".

Por último agradecer a Jose L. Salcedo su apoyo durante la realización del proyecto, tanto profesionalmente como anímicamente.

A todos,
Muchas gracias.

Dedicado especialmente a Quim Obalat.

Dedicado con mucho cariño

a mis padres y a mi hermano

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE CÓDIGOS MATLAB®

LISTA DE SIGLAS

En este documento se utilizan muchas siglas para abreviar

ADSL	Asymmetrical Digital Subscriber Line	**ISI**	Inter Simbol Interference
AM	Amplitude Modulation	**ISO**	International Organization for Standarization
ASK	Amplitude Shift Keying	**ITU**	International Telecommunication Union
AWGN	Additive White Gaussian Noise	**LOS**	Line of Sight
BCH	Bose-Chaudlhuri-Hocquenghem code	**LP**	Low Priority bit stream
BER	Bit Error Rate	**LS**	Least Squares
BER	Bit Error Rate	**LTI**	Linear Time Invariant
CATV	Cable Television	**MCM**	MultiCarrier Modulation
COFDM	Coded FDM	**MCRB**	Modified Cramér-Rao Bound
DA	Data Aided	**MPEG-2**	Motion Picture Experts Group
DAB	Digital Audio Broadcasting	**NDA**	Non Data Aided
DAVIC	Digital Audio-Video Council	**NLOS**	Non-Line of Sight
DFE	Decision Feedback Equalizer	**OFDM**	Orthogonal Frequency Division Multiplexing
DFT	Discrete Fourier Transform	**PAPR**	Peak to Average Power Ratio
DTFT	Discrete Time Fourier Transform	**PES**	Elementary Stream Packet
DVB	Digital Video Broadcasting	**PRBS**	Pseudo Random Binary Sequence
DVB-T	DVB-T Terrestrial	**PSD**	Power Spectral Density
ES	Elementary Stream	**PSK**	Phase Shift Keying
ETSE	Escola Técnica Superior d'Enginyeria	**PU**	Program Unit
ETSI	European Telecommunications Standards Institute	**QAM**	Quadrature Amplitude Modulation
FDM	Frequency Division Multiplexing	**QPSK**	Quaternary Phase Shift Keying
FEC	Foward Error Correction	**RDSI**	Red Digital de Servicios Integrados
FFT	Fast Fourier Transform	**RS**	Reed-Solomon
FIFO	First_In, First_Out shift register	**RTPC**	Redes Telefónicas Públicas Conmutadas
FIR	Finite Impulse Response	**SFN**	Single Frequency Networks
FT	Fourier Transform	**SNR**	Signal to Noise Ratio
HDTV	High Definition Television	**TDL**	Tapped Delay Line
HFC	Hybrid Fiber-Coax	**TPS**	Transmission Parameter Signalling
HP	High Priority bit stream	**TS**	Transport Stream
ICI	Inter Carrier Interference	**UAB**	Universitat Autònoma de Barcelona
IDFT	Inverse Discrete Fourier Transform)	**UHF**	Ultra-High Frequency
IFFT	Inverse Fast Fourier Transform	**WLAN**	Wireless Local Area Network
ISBN	International Standard Book Number	**ZF**	Zero Forcing

Capítulo 1 INTRODUCCIÓN

1.1 BREVE INTRODUCCIÓN A LAS COMUNICACIONES SIN HILOS

El papel que desempeñan las comunicaciones sin hilos ha ido ganando terreno con el paso del tiempo. Para entender y dominar la teoría de las comunicaciones sin hilos ha sido esencial la obtención de conocimientos en el campo de las señales electromagnéticas, conocimientos que se han obtenido en un periodo relativamente corto de tiempo, que abarca los últimos 150 años.

En 1864 James Clerk Maxwell formuló la teoría de las ondas electromagnéticas, de las que surgieron las conocidas Leyes de Maxwell referidas a la propagación de ondas electromagnéticas. En 1887, Albert Michelson y Edward Morley demostraron físicamente la existencia de las ondas electromagnéticas mediante un experimento que pretendía medir la velocidad de la luz. Un año más tarde, en 1888, Heinrich Hertz construyó el primer generador de ondas electromagnéticas. Éste último, junto a Maxwell y Oliver Lodge, llevó a cabo proyectos pioneros en el desarrollo de sistemas de comunicaciones sin hilos cuya distancia entre transmisor y receptor no superaba los 150 metros.

En poco tiempo, durante el periodo de 1895 a 1905, se mejoraron las técnicas de comunicación sin hilos, siendo Guglielmo Marconi quien inventó un aparato para transmitir ondas de radio a larga distancia. En 1901 se produjo la primera transmisión de banda a banda del océano Atlántico, desde Inglaterra a Canadá. En 1906, Reginald Fessenden consiguió realizar con éxito la primera radiodifusión de voz y música mediante la técnica de modulación en amplitud, AM (*Amplitude Modulation*). Pasados unos años, en 1927, se produjo la primera difusión de televisión en dos países a la vez. Los laboratorios Bell Labs y John Baird lo realizaron en Nueva York e Inglaterra respectivamente y paralelamente.

Tal y como ha sucedido a lo largo de la historia, la guerra ha sido una excusa para desarrollar y probar nuevas técnicas de comunicaciones, entre otras cosas. Durante la Segunda Guerra Mundial aparecieron las primeras técnicas de transmisión mediante espectro ensanchado, útiles para camuflar mediante encriptación los mensajes de voz dentro de señales con aspecto ruidoso.

En el 1946 apareció el primer sistema de telefonía móvil público en 5 ciudades americanas. Finalmente, se introdujo en Europa el primer sistema de telefonía móvil digital.

Todos estos hechos [1], además de otros importantes que no se han nombrado en el párrafo anterior, han hecho posible que en la actualidad podamos gozar de diferentes sistemas de comunicaciones sin hilos como, por ejemplo, los sistemas de comunicación móvil, la televisión por satélite o la conexión a Internet sin hilos.

La transmisión de señales a través del espacio libre presenta dificultades esenciales que necesitan solucionarse para obtener una transmisión de calidad, como puede ser el problema de eliminar las interferencias o fenómenos relacionados a la propagación. Por estas razones aparecen las técnicas de modulación digital, que son esenciales para la gran mayoría de los sistemas de comunicación digital, ya sea un sistema de telefonía, un sistema celular de comunicaciones o bien un sistema de comunicaciones satelital. Dichos inconvenientes hacen que el campo de las modulaciones digitales sea una línea de investigación muy atractiva.

Desde hace un par de décadas, la investigación y el desarrollo de técnicas de modulación han sido tareas muy activas y se han obtenido resultados muy prometedores [2]. Entre estas técnicas se encuentra la modulación multiportadora OFDM, modulación que se estudia en este proyecto junto con diferentes técnicas de estimación y sincronización que ayudan a paliar los efectos problemáticos inherentes a este esquema de modulación y a los sistemas de transmisión sin hilos en general.

1.2 MARCO DEL PROYECTO

Este proyecto ha sido desarrollado en colaboración con el Departamento de Telecomunicación e Ingeniería de Sistemas de la Escola Tècnica Superior d'Enginyeria (*ETSE*) de la Universitat Auntònoma de Barcelona (*UAB*) y se encuentra enmarcado dentro del campo de investigación relativo al procesamiento de datos e implementación de algoritmos de estimación y sincronización de sistemas de comunicaciones sin hilos.

1.3 OBJETIVOS DEL PROYECTO

El objetivo inicial de este proyecto fue la implementación de una plataforma software de simulación en Matlab® que recogiera la arquitectura de un sistema de transmisión OFDM que permitiera la implementación y el estudio de técnicas de estimación y sincronización.

La esencia del objetivo ha continuado siendo la misma pero, finalmente, se ha aplicado esta plataforma directamente sobre el esquema de transmisión del estándar de capa física DVB-T, que utiliza el sistema OFDM como esquema de modulación de datos. La decisión de ampliar el área de estudio ha sido debida al interés de realizar el proyecto sobre una base de aplicación real. Este hecho ha implicado una etapa de documentación importante dentro del periodo de realización del proyecto puesto que trabajar con estándares no es una tarea fácil, por lo que conseguir implementar satisfactoriamente un sistema de transmisión completo y fiel a un estándar se puede considerar un logro. Dichos estándares recogen las especificaciones referentes a la arquitectura del sistema transmisor pero no detallan el diseño de los bloques que lo forman ni la arquitectura del sistema receptor por lo que se ha tenido que utilizar bibliografía extra que se encuentra fuera del objetivo inicial del proyecto.

Tras estudiar el sistema de transmisión OFDM, a pesar de que ofrece múltiples ventajas frente a otros esquemas de modulación, se ha podido determinar que posee problemas inherentes importantes a su estructura, como es la necesidad de disponer de buena sincronización. Por ese motivo, resulta interesante estudiar las posibles soluciones aplicables e implementables en el esquema del receptor OFDM, que son, en definitiva, los bloques de estimación y sincronización de tiempo y de frecuencia. De hecho, la implementación de algoritmos de estimación y sincronización es un objetivo del proyecto y se ha realizado de manera que se aprovecha la estructura de la señal DVB-T.

Aprovechando que el sistema DVB-T es un sistema de transmisión que utiliza canales inalámbricos se ha estudiado las características de éste y su influencia negativa sobre la señal transmitida. Para disponer de un sistema de transmisor completo se ha considerado oportuno estudiar e implementar un algoritmo de estimación de canal que aprovecha la estructura de la señal OFDM y un algoritmo de ecualización que aprovecha las facilidades que ofrece el esquema OFDM.

Para poder evaluar el estimador de canal y el ecualizador implementados se ha tenido que disponer de un modelo de canal inalámbrico para simular el paso de los datos por un entorno hostil que varía la amplitud y la fase de la señal, entre otros efectos , añadiendo así mayor problemática en la correcta recepción. Se debe comentar que el campo de estudio del modelado de canales es muy amplio y, por tanto, existen múltiples métodos de modelado que pueden resultar más o menos fieles, según el caso. Este es un punto que no entra dentro de los objetivos principales del proyecto. Aún así, bajo la necesidad de obtener una respuesta de canal para poder evaluar el algoritmo estimador de canal, se ha hecho un pequeño estudio referente a este ámbito. Como resultado se ha podido obtener el modelado de canal de diferentes entornos, más o menos parecidos a la realidad pero siendo este último punto el menos importante.

Puesto que existen numerosas posibilidades de estudio dentro del ámbito de la transmisión utilizando el esquema de transmisión DVB-T y, más concretamente, el sistema de modulación OFDM, se ha tomado como objetivo el implementar una plataforma software con Matlab® que resulta eficiente en cuanto a coste computacional y que ofrece múltiples opciones de simulación gracias a su estructura. Dicha estructura permite que el sistema pueda ser ampliado con más algoritmos, ya sean de estimación como de codificación, permitiendo así poder ser utilizado en próximos proyectos para el estudio de áreas más concretas (como podría ser el caso del modelado de canal y su influencia sobre la señal transmitida). Para que esto último sea posible, se ha creído conveniente e importante documentar, de manera sencilla pero suficiente, la estructura del simulador y sus posibilidades con el objetivo de poder ser aprovechado y ampliado.

La elaboración de este proyecto final de carrera ha requerido el estudio profundizado de numerosas materias, muchas de las cuales muchas eran nuevas para el proyectista, que engloban los requisitos para poder implementar un sistema transmisor en software. Entre ellas está la codificación de canal y su decodificación, la modulación OFDM y su demodulación, algoritmos de estimación y sincronización, herramientas de estudio de las prestaciones del sistema y, por último, modelado de canal. A esto se le suma la necesidad de familiarizarse con el entorno Matlab®, profundizando en las grandes posibilidades que ofrece, para desempeñar la tarea de implementación de una plataforma de simulación de estas características.

Por todos estos motivos, se cumple el objetivo principal de este proyecto final de carrera: obtener el título de Ingeniero de Telecomunicaciones.

1.4 ESTRUCTURA DEL DOCUMENTO

El contenido expuesto en esta memoria se divide en 7 capítulos. Tras el capítulo introductorio, el segundo pretende enmarcar el proyecto elaborado dentro de la aplicación principal a la que va dirigido el estándar DVB-T, la televisión digital. A pesar de que no es un capítulo fundamental para la correcta comprensión del proyecto elaborado, se ha creído conveniente introducir al lector dentro del sistema de transmisión de datos de voz e imagen que, de hecho, son los datos que se transmiten utilizando dicho estándar. El tercer capítulo recoge los fundamentos de la modulación OFDM haciendo hincapié en sus condiciones de trabajo y sus ventajas e inconvenientes. Para explicar el contenido del proyecto realizado se ha utilizado una estructura que pretende facilitar la comprensión del mismo, por lo que los siguientes tres capítulos están muy relacionados pero debidamente separados. El capítulo 4 presenta el contenido referente a la arquitectura del sistema transmisor DVB-T, el capítulo 5 presenta el contenido referente al modelo de canal utilizado para transmitir la señal y el capítulo 6 presenta el contenido referente a la arquitectura del sistema transmisor implementado. Estos tres capítulos siguen la misma estructura, que consiste en presentar bloque a bloque cada uno de los bloques implementados que conforman el sistema en cuestión, ofreciendo información referente a sus objetivos, su funcionamiento, su relación con el estándar DVB-T y, dirigido a aquellas personas interesadas, el código implementado (con la intención de facilitar la comprensión del mismo si se encuentra enmarcado en un apartado que explica sus fundamentos). Cuando es el caso oportuno se muestran las simulaciones, y comentarios, pertinentes del sistema en estudio. Como el sistema transmisor y el sistema receptor tienen muchas partes complementarias, en los bloques del capítulo 6 en dónde esto ocurre la extensión de las explicaciones se ve reducida pero referenciada a los bloques del transmisor. Para finalizar, se presenta un capítulo que recoge conclusiones y propuestas para proyectos futuros.

Para complementar el trabajo se presentan 4 anexos. El primero es un manual de usuario para el uso del simulador implementado, ya sea para realizar simulaciones con lo que está implementado como para ampliar el mismo, dando información de su estructura y funcionalidades de una manera sencilla e intuitiva. El segundo anexo explica las herramientas utilizadas para el estudio de prestaciones del sistema simulado, sea cual sea su configuración. El tercer anexo recoge el contenido referente al modelado estadístico del canal de transmisión al cual se hace referencia en ocasiones. Debido a que el modelado de canal no es uno de los objetivos principales del proyecto, se ha determinado que la ubicación de este contenido debe presentarse en un anexo. Para finalizar, en el último anexo se presentan fundamentos de la codificación para la corrección de errores a los que se hace referencia en ciertas ocasiones ya que es una herramienta que utiliza el estándar de transmisión DVB-T para mejorar las prestaciones del sistema. La implementación de algoritmos de detección y corrección de errores tampoco es un objetivo del proyecto por lo que, nuevamente, se ha determinado que la ubicación referente a este tipo de información debe estar en un anexo.

Tanto las explicaciones extraídas de bibliografía como las figuras, tablas, ecuaciones y códigos están debidamente referenciadas.

Llegados a este punto, para finalizar el capítulo, únicamente queda el desearle una entretenida y lucrativa lectura. Que lo disfrute.

Capítulo 2 CAMINO HACIA EL DVB-T

En este capítulo se pretende enmarcar el proyecto desarrollado dentro de su campo de operación. A pesar de que el estudio del contenido de éste no resulta indispensable para la elaboración ni el entendimiento del proyecto, resulta interesante conocer los orígenes del estándar implementado y de los datos tratados, sus ventajas e inconvenientes, la problemática a la que se enfrenta y, en definitiva, resulta interesante obtener un punto de vista general antes de entrar en materia. Es por este motivo que se ha decidido incluir un capítulo en el que se empieza hablando de la televisión digital, sus diversos sistemas de transmisión y el formato de sus datos para finalizar en la explicación de uno de los organismos reguladores de este servicio más importantes del mundo, el DVB, y, en concreto, del estándar de transmisión que dicho proyecto ocupa.

2.1 LA TELEVISIÓN DIGITAL

La televisión es quizás, junto con la radio, el medio de comunicación de masas más extendido (sólo en Europa supera los 140 millones de usuarios). Un sistema de televisión, en general, se divide en dos partes: la red de producción y la red de transporte. La red de producción es la parte del sistema en la que se generan los contenidos, mientras que la red de transporte es la que hace llegar dichos contenidos a usuarios de la red [3]. En este proyecto nos centramos en esta última parte de transporte.

La televisión digital, en comparación con su homóloga analógica, ofrece una cantidad de ventajas derivadas de la naturaleza binaria de la señal, destacando las más importantes:

- Ofrece mayor calidad de sonido e imagen
- El mantenimiento de la calidad es extremo a extremo.
- Tiene mayor capacidad y flexibilidad de envío de la información.
- Ofrece servicios interactivos de altas prestaciones adaptados a cualquier entorno.
- Integra servicios y ofrece otros nuevos que son interactivos.

Los proveedores son los encargados de aportar a la red valor para los usuarios. Existen dos tipos: proveedores de contenidos que suministran canales de televisión y radio, y los proveedores de servicios. Los primeros suelen transmitir sus contenidos a través de un enlace de satélite con el centro de emisión del operador de televisión digital o bien mediante una red de cable a la que ambos están conectados. En cuanto a la provisión de servicios, las tecnologías empleadas son la RTPC (*Redes Telefónicas Públicas Conmutadas*), RDSI (*Red Digital de Servicios Integrados*), Frame relay (comunicación mediante retransmisión de tramas) o Internet [3].

Ya en el operador de televisión digital, el centro de emisión captura las señales de los proveedores de contenidos y el servidor de aplicaciones integra los servicios de los proveedores. Recogida esta información se codifica y, en caso necesario, se encripta, convirtiéndola a un formato adecuado a la red de televisión y, posteriormente, la multiplexa para constituir la señal transmitida.

La señal resultante, emitida por el operador, llega al terminal de usuario, que la decodifica, la demodula y la presenta en la pantalla de la televisión del usuario.

Existen tres modos de hacer llegar la señal de televisión digital a los usuarios: por cable, por satélite y vía terrestre, además de hacerlo mediante ADSL (*Asymmetrical Digital Subscriber Line*). Existen ciertas implicaciones en el uso de un modo u otro de transmisión que, resumidamente, son:

Tabla 2.1 Medios de televisión digital

	Satélite	Cable	Terrestre
Implantación	Fácil y rápida	Difícil y costosa	Fácil y rápida
Cobertura	Continental	Local	Local
Ancho de banda	Gran capacidad	Limitado	Limitado
Contenidos	No permite información local	Información local	Información local
Canal de retorno	Limitado	Ilimitado	Limitado

A continuación se realiza una breve descripción de cada uno de los modos principales de transmisión para conocer de manera breve sus principios de funcionamiento y sus ventajas y limitaciones.

2.1.1 Televisión por satélite

En el caso de la televisión vía satélite el repetidor empleado es un satélite que orbita alrededor de la Tierra. Por su parte, el segmento terreno lo constituyen la estación emisora y las instalaciones receptoras de los usuarios. Los contenidos se generan en el estudio de televisión y se envían al transmisor. Una vez allí, la señal se modula en un canal de satélite y se amplifica como paso previo a su envío al satélite para su distribución.

Dado que los niveles de potencia recibida en un satélite son particularmente bajos (del orden de *pW* [3]), las estaciones terrenas deben diseñarse para soportar relaciones portadora a ruido mínimas. Por esta razón, el receptor suele consistir en una antena parabólica de gran diámetro altamente lineal y un amplificador de bajo ruido.

En cuanto a los receptores de usuario, encontramos dos tipos: individual y colectivo. Básicamente, cualquier instalación receptora se divide en tres partes:

- Antena: es la encargada de captar la señal del satélite, del acierto en su elección depende la calidad de la señal recibida.
- Unidad externa: su misión es disminuir la frecuencia recibida del satélite para su distribución a la unidad interior vía cable coaxial.
- Unidad interior: es la última etapa de la instalación receptora y se encarga de enviarla al receptor de televisión.

Además, intervienen otros elementos como procesadores y amplificadores de frecuencia intermedia, conversores y elementos de distribución.

2.1.2 Televisión por cable

La televisión por cable CATV (*Cable Television*) es una evolución de las antiguas instalaciones de video comunitario en las que un único canal de televisión era distribuido a todas las viviendas de un mismo edificio. Las redes CATV permiten la distribución de señales de satélite a través de una red de cable. Este nuevo sistema ofrecía muchos más canales de televisión que el tradicional basado en la radiodifusión terrestre, por lo que se inició así una nueva forma de plantear el negocio de la televisión a través de nuevos tipos de programación basados en canales especializados (noticias, deportes, películas) de pago y abriendo el camino hacia la industria de la televisión multicanal. Sin embargo, este tipo de redes se desplegaba sobre una red de cable coaxial con largas cascadas de amplificadores según una topología de árbol-rama, con las consecuencias en cuanto a fiabilidad y calidad de la señal que ello implica. Para solucionar estos problemas, y gracias a la disminución de las tecnologías ópticas, surgieron las redes híbridas fibra-coaxial, HFC (*Hybrid Fiber-Coax*). En una red HFC podemos distinguir las siguientes partes:

- Cabecera: constituye el centro de operaciones en el que se llevan a cabo procesos tales como la captación de señales y su procesamiento.
- Red de distribución: construida con fibra óptica, lleva la señal hasta el punto en el que comienza la red de distribución coaxial.
- Red de acceso: se divide en dos tramos. El primer tramo es la parte del nodo óptico y, con topología árbol-rama, lleva la señal hasta la red de abonado a través de cable coaxial. El segundo tramo, también de cable coaxial, transporta la señal hasta el equipo del usuario.

2.1.3 Televisión terrestre

En los sistemas terrenos, la señal de televisión se propaga por el aire y viaja desde los estudios de producción hasta las instalaciones de usuario, ya sean individuales o colectivas.

En general, la distribución de la señal de televisión se lleva a cabo mediante una red en la que se emplean satélites y enlaces terrenos. El satélite constituye un medio de interconexión entre una serie de centros reemisores que cambian la modulación de la señal y la envían por la red de distribución terrestre. Ésta última está formada por un conjunto de radioenlaces entre las antenas emisoras y las antenas receptoras de las instalaciones de usuario.

Las primeras instalaciones de recepción de televisión eran instalaciones individuales, pues no eran muchos los hogares que contaban con aparatos de televisión. Cada usuario disponía de una antena receptora y del resto de elementos necesarios para hacer llegar la señal hasta el receptor. Sin embargo, esta situación cambiaría cuando el poder adquisitivo de los usuarios aumentó, incrementándose también el número de receptores. Surgieron así las instalaciones colectivas.

En general, un sistema de recepción de televisión se puede dividir en tres grandes partes:

- El sistema captador de señales: se encarga de recibir las señales procedentes de los emisores y reemisores. Se deben situar en el exterior, en un lugar con buena visibilidad, como el tejado o la azotea del edificio, y suelen ser del tipo Yagui o de panel. Forman parte del sistema captador las antenas, cuya misión es captar las señales, y os preamplificadores, que aumentan el nivel de la señal de salida de antena introduciendo el menor ruido posible.

- La cabecera: adecua las señales de salida del sistema captador para que sea recibida por el usuario con las condiciones de calidad requeridas. Se suele ubicar cercano al sistema captador y está formado por componentes activos (amplificadores, conversores y moduladores) y por componentes pasivos (filtros, mezcladores, ecualizadores y atenuadores)

Finalmente, la red de distribución, a través de todo el edificio o vivienda, lleva la señal a las tomas de usuario. Se compone de conectores, repartidores, derivadores, tomas y cajas de paso y cable coaxial. En instalaciones individuales, además, existen antenas interiores que se conectan directamente a la entrada del receptor. También es posible emplear amplificadores directamente en el mástil de la antena y amplificadores interiores de vivienda.

2.2 EL FORMATO DE DATOS MPEG-2

En 1988 se formó el grupo MPEG (*Moving Picture Expert Group*) de la ISO (*International Organization for Standarization*) con el objetivo de acordar una norma de compresión para señales de video con las correspondientes señales de audio asociadas, con vistas a su almacenamiento y recuperación en equipos digitales adecuados.

MPEG ha desarrollado estándares que forman la plataforma base para el de la transmisión de señales de vídeo, entre ellos:

- MPEG-1: normaliza el formato del flujo de datos codificados así como los procesos de codificación y decodificación. Este estándar fue diseñado originalmente como un método de almacenamiento en CD, cinta de audio digital, unidades de cinta, disco duro de ordenador y discos ópticos. Sin ir más lejos, MPEG-1 puede considerarse como una generalización de la norma ITU (*International Telecommunication Union*) H.261. Las características más importantes de esta norma son:

    - Las velocidades de bit óptimas para vídeo pueden oscilar entre 1 y 3 Mbps, mientras que en audio pueden variar desde 64 hasta 448 Kbps.

    - La velocidad de transmisión óptima de la trama de transporte es de 1.5 Mbps y para el vídeo digital es de 1.2 Mbps.

    - En MPEG-1 la calidad subjetiva del sonido es similar al que entrega una señal de radio modulada en FM, y en cuanto a imagen es similar al sistema VHS. En este entorno de calidad se ha desarrollado la industria de multimedia de gran consumo y algunos entonos profesionales de edición de video, producción, video interactivo y video-juegos.

- MPEG-2; pretende crear un estándar genérico de arquitectura abierta, con aplicaciones a campos tan diversos como sistemas de distribución de televisión por cable, enlaces de cámaras autónomas para noticias, telefonía visual, sistemas de video-vigilancia, conexiones digitales de televisión terrestre y vía satélite, etc. Este estándar ha nacido de las ventajas que supone una norma única para la compresión de TV digital, derivado hacia la firma de un acuerdo de entendimiento para la adopción y normalización en equipos.

2.2.1 Imagen digital

En general, una imagen está compuesta por un conjunto de elementos de imagen que reciben el nombre de *píxeles*. Cada uno de estos *píxeles* contiene información de luminancia (brillo de la imagen) y de crominancia (color). Por motivos que se escapan al objetivo de este proyecto, la información de crominancia se transmite de forma de dos señales de diferencia de color que reciben el nombre de *Cr* (rojo) y *Cb* (azul). En la mayoría de los casos, existe una fuerte dependencia entre elementos de imagen vecinos. La televisión analógica ignora este hecho y transmite todos los *píxeles*. Por el contrario, la televisión digital explota este hecho para comprimir la señal y aprovechar el ancho de banda de manera más eficiente (codificación de fuente).

El primer paso para convertir la señal a un formato digital es muestrearla. En este sentido, conviene tener en cuenta que el ancho de banda de las señales de crominancia es la mitad que el de la señal de luminancia, ya que el ojo humano es menos sensible a las diferencias de color que a las de brillo. Por esta razón, la luminancia se muestrea a 13.5 MHz mientras que en la croma la frecuencia de muestreo es de 6.75 MHz.

El principal problema de la transmisión de vídeo es el ancho de banda que requiere. En efecto, las exigencias de ancho de banda características de la señal de vídeo sin comprimir se encuentran en el rango de los 100 a 240 Mbps y en algunos casos, como en la televisión digital de alta definición, puede alcanzar 1 Gbps. Por esta razón, es necesario emplear algún algoritmo de compresión que reduzca el ancho de banda de la señal a la vez que el resultado se mantiene dentro de unos parámetros de calidad aceptables.

La compresión de cualquier señal, en general, se consigue eliminando la información de redundancia que, en el caso de flujo de video, se encuentra en la misma trama o en tramas próximas.

2.2.2 Codificación de la señal MPEG-2

Tal y como se ha comentado, transmitir la señal tal cual exige anchos de banda enormes. Por ello, se aprovecha la redundancia (espacial o temporal) para comprimir la señal (codificación de fuente). Dos ejemplos son la codificación secuencial (cuando existen diversas secuencias idénticas, se transmite una secuencia única seguida del número de veces que se repite) y la codificación diferencial (aprovecha el parecido entre muestras próximas para transmitir únicamente las diferencias entre ellas). De esta manera, a partir del valor de una muestra o conjunto de muestras es posible predecir el valor de la siguiente.

2.2.3 El sistema de transporte MPEG-2

El sistema de transporte MPEG-2 integra, en un único flujo, las señales de audio, video y otros datos auxiliares. A cada una de estas señales se les asigna un flujo elemental ES (*Elementary Stream*) del que se obtendrán paquetes de información que, añadidos a la información adicional, formarán la señal múltiplex o programa. Todo este proceso queda recogido en la siguiente figura.

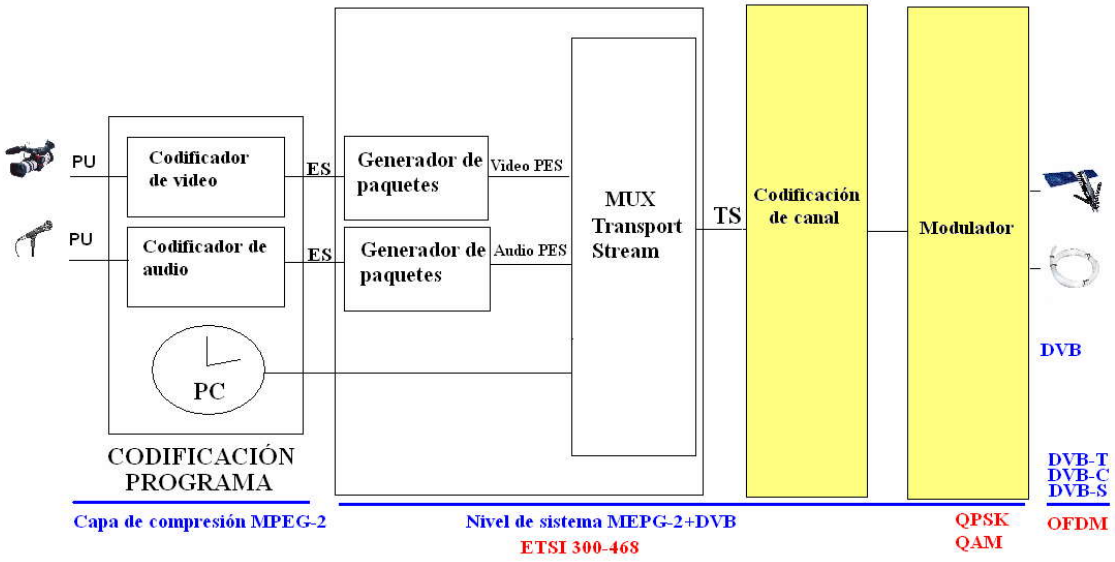


Fig. 2.1 Sistemas de transporte MPEG-2.

En el proyecto que se presenta con esta memoria se ha trabajado a nivel de flujo de transporte, directamente con la señal que resulta de la salida del multiplexador presentado en la Fig. 2.1, es decir, directamente con los paquetes del flujo de transporte, TS (*Transport Stream*). En el siguiente apartado se explica brevemente el origen de cada TS.

2.2.4 Jerarquía MPEG

En la Fig. 2.1 se presenta un diagrama de bloques del sistema de transporte MPEG, en el que se puede observar el flujo de datos recorriendo el esquema desde la generación de los datos de audio y video de la aplicación (Capa de aplicación OSI número 4) hasta su transmisión (Capa física OSI).

Se parte de las señales de audio y video, muestreadas y cuantificadas, denotadas como Unidades de Presentación o programa, PU (*Program Unit*).

Estas dos cadenas PU pasan por un bloque codificador de fuente MPEG-2 (Véase apartado 2.2.2), en cuya salida se obtienen las unidades de datos comprimidas denotadas como Cadenas Elementales, ES (*Elementary Stream*), tanto de audio como de video. Una cadena ES no es más que la salida en bruto de un codificador, y contiene no más que la información necesaria para que un decodificador pueda formar el video o el audio original. La sintaxis de la señal comprimida está rígidamente definida, de forma que asegure que todos los decodificadores puedan usarla.

Para propósitos prácticos, las cadenas elementales ES conteniendo audio o video deben ser partidas en paquetes, PES (*Elementary Stream Packet*). Estos paquetes son identificados mediante cabeceras que contienen marcas de tiempo para sincronización.

La especificación de sistema de MPEG-2 describe como los paquetes de datos comprimidos ES de audio y video pueden ser multiplexados junto con otros datos para formar una simple cadena de datos, según la aplicación:

- La cadena de programa, PS (*Programme Stream*), fue diseñada pensando en el almacenamiento y reproducción de un simple programa a partir de un dispositivo de almacenamiento digital, eventualmente libre de ruido.

- La cadena o flujo de transporte, ya mencionada y denotada como TS, fue pensada para la entrega simultánea de múltiples programas sobre canales con ruido.

El caso que abarca el sistema DVB-T es el de cadenas de transporte TS. A partir de estos datos empieza la función de la capa física de transmisión, parte implementada en el proyecto y presentada en este documento. Estos datos TS pasarán por unos bloques de codificación de canal y de modulación para ser transmitidos y obtener una señal óptima en recepción. La estructura de datos de las cadenas TS se presenta en la siguiente figura [4]:

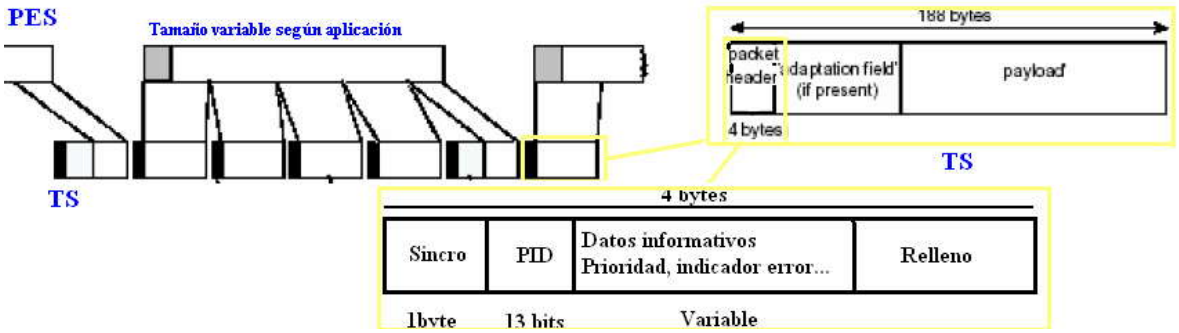


Fig. 2.2 Estructura de datos de entrada al sistema de la capa física de transporte

2.3 EL ORGANISMO REGULADOR DVB

En Europa se han creado dos grupos de trabajo, el DVB (*Digital Video Broadcasting*), que partiendo del ámbito exclusivamente europeo tiene actualmente ámbito mundial, y el DAVIC (*Digital Audio-Video Council*) también de influencia mundial, para establecer los criterios técnicos a seguir. El presente documento se centra en el primero de éstos, puesto que la implementación del algoritmo de estudio se refiere a uno de sus estándares.

El proyecto DVB ha definido un marco técnico adecuado para los sistemas de televisión digital tanto en sistemas por satélite como de cable y transmisiones terrestres además de servicios de sistemas de información, sistemas comunes de encriptado e interfaces de acceso condicional.

DVB ha adoptado el estándar de codificación de video y audio a sus necesidades. Las razones de esta elección son varias. Por un lado, la estructura de paquetes de DVB simplifica la tarea de sincronización y permite el transporte de todos los componentes en una misma trama binaria. En segundo lugar, se trata de un sistema idóneo para el almacenamiento, ya que la propia trama binaria incluye su información de presentación para posibilitar el ordenamiento de las mismas. Finalmente, DVB permite la introducción de información de servicio de una forma flexible para audio, video y datos.

Uno de los requisitos de partida de los sistemas DVB es la reutilización de elementos comunes, tal y como ilustra la Fig. 2.3, de manera que sea posible aprovechar pequeñas economías de escala, ya que muchos sistemas utilizan gran cantidad de estos elementos comunes. La Tabla 2.2 muestra el conjunto de algunos de los estándares del DVB.

Tabla 2.2 Estándares DVB

Denominación	Descripción
DVB-S	Transmisión de televisión digital vía satélite
DVB-C	Transmisión de televisión digital por cable
DVB-CS	Transmisión en redes SMATV
DVB-MS	Distribución punto a multipunto basada en DVB-C para frecuencias menores a 10 GHz
DVB-MS	Distribución punto a multipunto basada en DVB-C para frecuencias superiores a 10 GHz
DVB-T	**Transmisión de televisión digital terrestre**
DVB-H	Transmisión de televisión digital terrestre para receptores portátiles.
DVB-SI	Información de servicio
DVB-TXT	Formato de teletexto
DVB-CI	Interfaz común para el acceso condicional
DVB-PTC	Canal de retorno en sistemas por cable
DVB-RCC	Canal de retorno por la RTPC
DVB-NIP	Protocolo de servicios interactivos
DVB-PDH	Interfaces de redes PDH
DVB-SDH	Interfaces de redes SDH
DVB-M	Medidas para sistemas DVB
DVB-PI	Interfaces de cabecera en redes CATV y SMATV

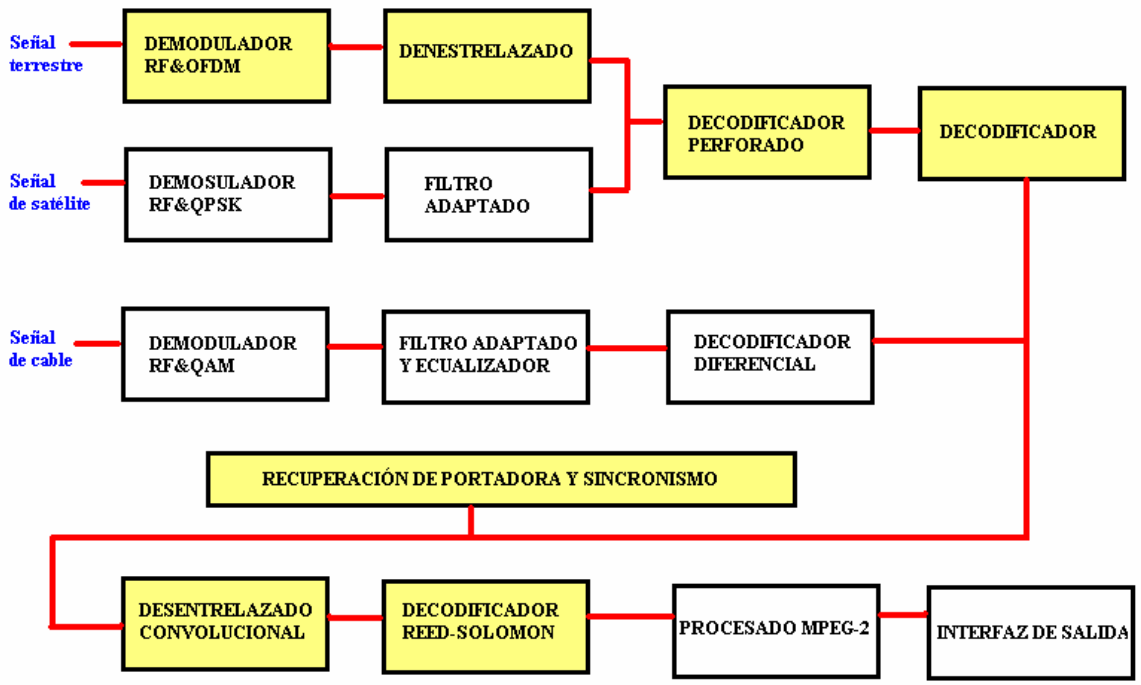


Fig. 2.3 Procesamiento de señales DVB

2.4 EL ESTÁNDAR DVB-T

DVB-S es el estándar que especifica cómo debe llevarse a cabo la transmisión digital vía satélite multiplexando varios canales de audio y video digital sobre una misma portadora. DVB-C es la variante para televisión por cable, y describe la codificación de

canal y la modulación empleados en redes por cable. Como puede apreciarse en la Fig. 2.3, el procesamiento es muy similar al de DVB-S, con salvedades importantes como su uso de modulación QAM y una codificación más sencilla. Sin embargo, DVB-T es la modalidad que mayor revolución ha sufrido [3] ya que presenta múltiples ventajas frente a los demás estándares de transmisión, como:

- Asegura la cobertura sin coste alguno para el usuario.
- Permite recepción portátil con ausencia total de doble imagen.
- Implica una reducción de la potencia transmitida respecto a la de los sistemas analógicos.
- Permite la implantación de nuevos servicios.
- Se obtiene un ahorro espectral.
- Permite mayor número de canales.

Al encontrarse la señal en formato digital, es posible dotarla de mecanismos de protección frente a errores antes de enviarla por el medio radioeléctrico. Por ello, previamente a su modulación se emplea una codificación Reed-Solomon con entrelazado convolucional. El objetivo de esta primera codificación es la dispersión de los paquetes para proteger la transmisión frente a errores de ráfaga. Además, se emplea otra codificación interior mediante un código perforado convolucional. El proyecto que se presenta en este documento se basa en la implementación de un sistema transmisor basado en el estándar DVB-T, cuya motivación principal es la implementación de algoritmos de estimación y sincronización aplicados directamente en su esquema de modulación OFDM, por lo que, en el Capítulo 4, se realiza una descripción detallada del sistema de transmisión dictado por dicho estándar.

Como se ha comentado anteriormente, la modulación utilizada es COFDM (*Coded Frequency Division Multiplexing*) en el que los datos modulan un elevado número de portadoras ortogonales empleando técnicas OFDM. El principio de la modulación ortogonal se basa en que los máximos de cada portadora se hacen coincidir con los ceros de las otras. Este esquema de modulación resulta especialmente apropiado para la televisión digital terrestre, entre otras, por las siguientes razones:

- Robustez frente a la propagación multicamino: este problema se manifiesta sobre todo en grandes centros urbanos con multitud de edificios en los que la señal sufre sucesivas reflexiones.
- Soporta interferencias cocanal de banda estrecha: la transición a la televisión digital terrestre se hace de manera gradual. Durante este periodo de convivencia con la televisión tradicional las señales analógicas aparecen como interferencias en los sistemas digitales.
- Una de las consecuencias de la televisión digital es que permite el despliegue de redes isofrecuenciales en las que todos los transmisores emiten simultáneamente y en la misma frecuencia. Por tanto, es posible aprovechar mejor el espectro aumentando el número de programas ofrecidos a los usuarios, pagando el precio de tener unos requisitos de sincronización muy estrictos.

En el Capítulo 3 se realiza una descripción mucho más detallada referente a la modulación OFDM y a su esquema de transmisión. Tal y como se ha comentado anteriormente, el mayor interés en trabajar con el sistema DVB-T es debido a este esquema de modulación, puesto que permite implementar y estudiar un conjunto de técnicas de estimación y sincronismo específicos que aprovechan las propiedades del mismo.

Capítulo 3 MODULACIÓN OFDM

3.1 INTRODUCCIÓN A LAS MODULACIONES MULTIPORTADORAS

Recientemente las modulaciones multiportadora, MCM (*Multicarrier Modulation*) están obteniendo mayor interés y están siendo usadas para una gran cantidad de aplicaciones debido a las múltiples ventajas que tiene frente a la modulación por portadora única. Una ventaja obvia es que transmitiendo simultáneamente N símbolos en N portadoras reduce la tasa de símbolo a un N-ésima parte de la tasa de símbolo de los datos dispuestos en serie, o bien, incrementa la duración del símbolo N veces. De este modo, el efecto de la Interferencia Intersimbólica, ISI (*Inter Simbol Interference*), debida a la dispersión temporal del canal se reduce y la ecualización en el receptor se vuelve sencilla, o incluso innecesaria.

Las portadoras múltiples en MCM se denominan subportadoras. La banda frecuencial ocupada por la señal contenida en una subportadora se denomina sub-banda. Para separar las señales de las sub-bandas en el receptor el método más sencillo es el denominado Acceso múltiple por División de Frecuencias, FDM (*Frequency Division Multiplexing*) y consiste en espaciar el centro frecuencial de las subportadoras de manera que queden bien separadas, haciendo que el espectro de las N sub-bandas virtualmente no estén solapadas. Éste proceso requiere que en recepción se utilicen N filtros paso banda muy selectivos para separar dichas sub-bandas, hecho que complica y encarece el sistema.

Otro método para separar las señales de las sub-bandas es permitir el solapamiento de las sub-bandas adyacentes haciendo que la obtención de cada sub-banda en recepción se consiga mediante un espaciado de los centros frecuenciales sea de $1/T$ entre dos subportadoras, siendo T el periodo de símbolo. Mediante este espaciado se consigue que todas las subportadoras sean ortogonales entre ellas y que puedan ser separadas en recepción mediante bloques de correlación y no mediante el uso de filtros selectivos. Este método, ya nombrado en apartados anteriores de este documento, es denominado OFDM.

Comparando el método OFDM con el método FDM, el primero utiliza menos ancho de banda y no requiere el uso de filtros paso banda para cada sub-banda. No obstante, es interesante destacar que mientras ambos métodos son utilizados para señales digitales, OFDM también puede ser utilizado para transmitir señales analógicas [2].

OFDM se presentó durante los años 60 pero hasta los años 70 y 80 no fue utilizado. A pesar de sus numerosas ventajas (Véase apartado 3.3.1), su complejidad de implementación ha perjudicado a sus aplicaciones. No obstante, recientemente este método de modulación está ganando popularidad gracias a los grandes avances en la velocidad y la potencia de procesamiento de los dispositivos de procesado digital de la señal.

OFDM se utiliza en servicios de ADSL y en redes WLAN (*Wireless Local Area Network*) así como en aplicaciones de radiodifusión, televisión digital (incluyendo de alta definición), comunicaciones ópticas y comunicaciones móviles.

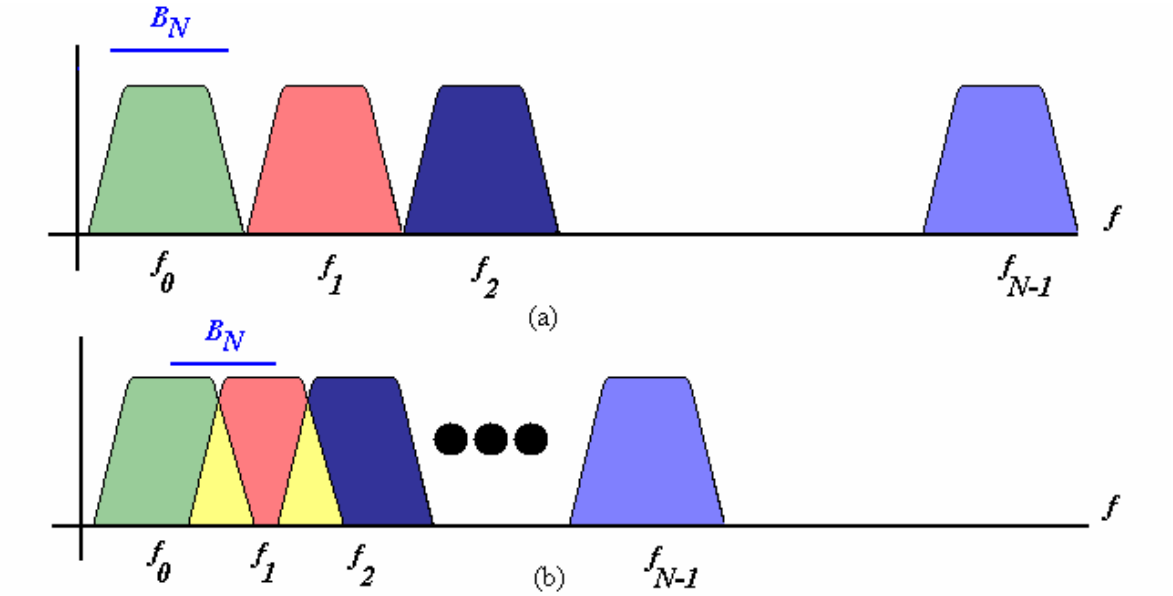


Fig. 3.1 Señal transmitida con múltiples portadoras
(a)Sub-canales separados (b) Sub-canales solapados

3.2 SEÑAL Y ESPECTRO OFDM

Tal y como se ha comentado en el apartado anterior, OFDM es un esquema de modulación en el cual los símbolos son transmitidos en paralelo empleando un número considerable de subportadoras ortogonales, tal y como ilustra la Fig. 3.2.

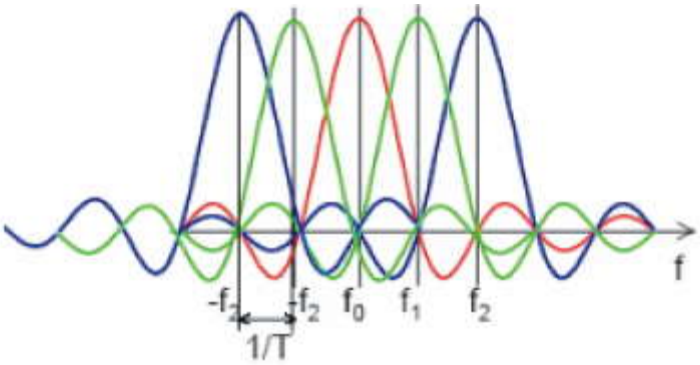


Fig. 3.2 Esquema de subportadoras OFDM en frecuencia

Un bloque de N símbolos que serían transmitidos de manera secuencial en T_s segundos cada uno se convierten en un bloque de N símbolos que se transmiten en paralelo en $T = N \times Ts$ segundos todos ellos. Los símbolos tienen entonces una duración N veces mayor permitiendo así reducir la interferencia intersimbólica, puesto que al durar más tiempo el porcentaje del símbolo afectado por otro símbolo adyacente es menor. Cada una de las subportadoras debe estar separada $1/T$ [Hz], condición que garantiza la ortogonalidad de las mismas [13].

Dependiendo del tipo de aplicaciones, la configuración final de una señal OFDM puede ser en forma de señal banda base o bien en forma de señal paso banda. Para sistemas de comunicaciones con hilos, debido al limitado ancho de banda de los cables utilizados, se transmite en banda base. Pero para sistemas de comunicaciones sin hilos, como es el especificado por el DVB-T, las bandas frecuenciales de la señal OFDM se localizan en l a banda frecuencial RF. Para este último caso, las señales OFDM se generan en banda base y entonces se sitúan en la banda RF mediante el uso de mezcladores.

Para la implementación del simulador DVB-T en Matlab realizado en este proyecto el modelo de señal OFDM que se ha seguido es el modelo banda base, del cual se extiende la explicación en el siguiente apartado.

3.2.1 Señal OFDM banda base. Condición de ortogonalidad.

La señal general OFDM en banda base en el primer periodo de símbolo se define de la siguiente manera:

$$s(t) = \sum_{i=0}^{N-1} s_i(t) = \sum_{i=0}^{N-1} A_i \cdot \cos\left(2\cdot\pi\cdot f_i \cdot t + \varphi_i\right), \qquad 0 \le t \le T \tag{3.1}$$

donde A_i, f_i, y φ_i son la amplitud, la frecuencia y la fase respectivamente de la i-ésima subportadora, N es el número de portadoras del símbolo OFDM y T el período de símbolo de datos OFDM. La Fig. 3.3 ilustra la forma de onda de la señal $s(t)$, expresada en (3.1), que contiene 1706 sub-portadoras moduladas en QPSK.

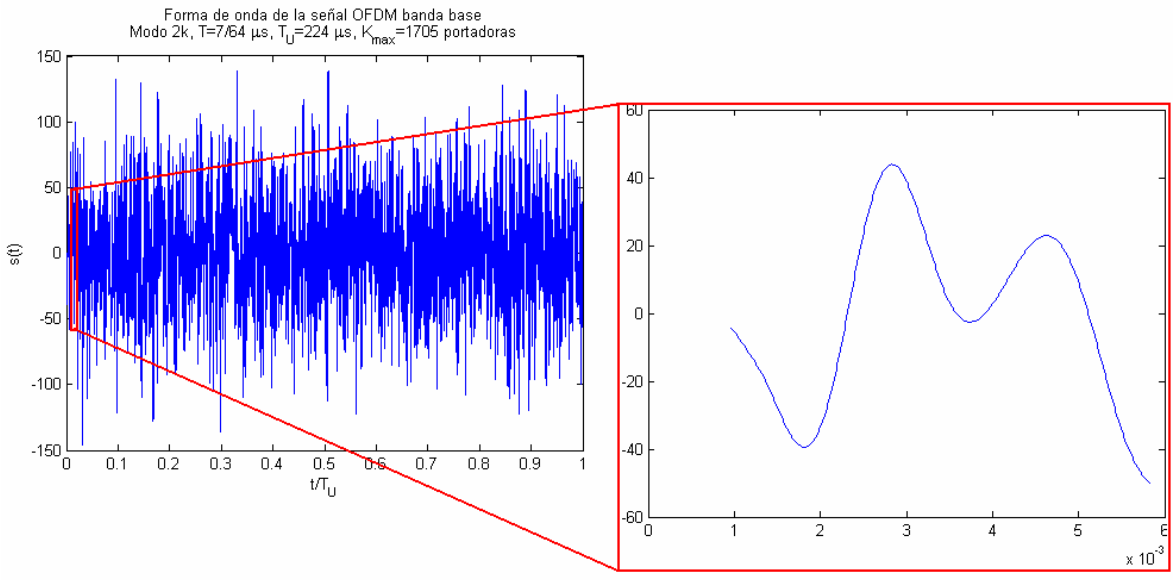


Fig. 3.3 Forma de onda de la señal OFDM en banda base.

Si cada portadora modula un símbolo ASK, A_i es una constante y φ_i representa una fase inicial cuyo valor es normalmente asumido como cero. Si cada portadora modula un símbolo PSK, A_i es una constante y φ_i está determinada por los datos. En cambio, si cada portadora modula un símbolo QAM tanto A_i como φ_i vienen determinados por los datos.

Tal y como se ha comentado anteriormente, para que las subportadoras sean ortogonales entre ellas, el valor de f_i tiene que ser un entero múltiple de la tasa de símbolo:

$$R_S = \frac{1}{T} \qquad \text{Condición de multiplicidad de } f_i \tag{3.2}$$

y la mínima separación frecuencial entre subporadoras debe ser de:

$$\frac{1}{T} \qquad \text{Condición de separación frecuencial entre subportadoras} \tag{3.3}$$

Normalmente f_i se escoge tal que resulta ser un múltiplo de la tasa de símbolo, pero si la modulación de los símbolos a la salida del mapeador son modulados en ASK, se demuestra que la condición de multiplicidad expresada en (3.2) se puede reducir a [2]:

$$\frac{1}{2 \cdot T} \qquad \text{Condición de multiplicidad de } f_i \tag{3.4}$$

En este documento, y durante la realización del proyecto, se asume que la condición de ortogonalidad para el valor de f_i es de $1/T$, asegurando que las portadoras sean ortogonales entre ellas independientemente de la modulación del símbolo que se transmite en cada subportadora.

Asumiendo que la frecuencia central de la primera subportadora es f_0, entonces las frecuencias centrales de las siguientes subportadoras de un símbolo OFDM son:

$$f_0, f_0 + R_S, f_0 + 2 \cdot R_S, ..., f_0 + (N-1) \cdot R_S \tag{3.5}$$

Para minimizar el ancho de banda requerido lo ideal sería destinar la frecuencia 0 a f_0 pero, en la práctica, para evitar problemas con el *offset* DC (componente continua) del conversor D/A, f_0 nunca es 0 Hz. A pesar de esto, por simplicidad se utiliza el valor de 0 en f_0 cuando se aplican estudios teóricos, o bien se puede dejar inutilizada la subportadora de la frecuencia f_0. Entonces, asumimos que las frecuencias centrales de las subportadoras son las siguientes:

$$f_i = i \cdot R_S = \frac{i}{T} \qquad \text{i} = 0,1,...., N\text{-}1 \tag{3.6}$$

Con las frecuencias expresadas en (3.6), fácilmente se puede verificar que se obtiene ortogonalidad entre las subportadoras:

$$\int_0^T \underbrace{A_i \cdot \cos(2 \cdot \pi \cdot f_i \cdot t + \varphi_i)}_{s_i(t)} \cdot A_j \cdot \underbrace{\cos(2 \cdot \pi \cdot f_j \cdot t + \varphi_j)}_{s_j(t)} \, dt = \begin{cases} {A_0}^2 \cdot T \cdot \cos^2(\varphi_0) & \text{si } i = j = 0 \\ \frac{1}{2} \cdot {A_i}^2 \cdot T & \text{si } i = j \neq 0 \\ 0 & \text{si } i \neq j \end{cases} \tag{3.7}$$

Nótese que la ortogonalidad se mantiene para cualquier valor de A_i, A_j, φ_i y φ_j siempre que se cumpla (3.6).

El espectro frecuencial de la señal OFDM se caracteriza por su densidad espectral de potencia, PSD (*Power Spectral Density*). Si se asume que los datos de cada subportadora son independientes del resto, entonces el PSD total es justamente la superposición de las PSD de todas las señales de las sub-bandas, siendo:

$$S(f) = \sum_{i=0}^{N-1} S_i(f) \tag{3.8}$$

donde $S_i(f)$ representa la PSD de la señal de la sub-banda i-ésima, $s_i(t)$, cuya expresión es:

$$\begin{aligned} (a)\quad & S_i(f) = \frac{1}{2}\cdot A^2{}_{avg}\cdot T\left[\left(\frac{sen(\pi(f-f_i)\cdot T)}{\pi(f-f_i)\cdot T}\right)^2 + \left(\frac{sen(\pi(-f-f_i)\cdot T)}{\pi(-f-f_i)\cdot T}\right)^2\right], \text{ si } i \neq 0 \\ (b)\quad & S_i(f) = \frac{1}{2}\cdot A^2{}_{avg}\cdot T\left[\left(\frac{sen(\pi\cdot f\cdot T)}{\pi\cdot f\cdot T}\right)^2\right], \qquad \text{si } i = 0 \end{aligned} \tag{3.9}$$

Combinando las expresiones (a) y (b) de (3.9) con (3.8), normalizando la PSD al nivel de su máximo y mostrando únicamente la parte frecuencial positiva, se obtiene la siguiente expresión de $S_i(t)$:

$$S(f) = \sum_{i=0}^{N}\left(\frac{sen(\pi(f-f_i)\cdot T)}{\pi(f-f_i)\cdot T}\right)^2 \quad f \geq 0 \tag{3.10}$$

Cada miembro de la PSD de la expresión en (3.10) tiene la envolvente de una función *sinc* al cuadrado. El primer nulo de la primera PSD se encuentra en $(f-f_i)=1/T$. Ya que la separación frecuencial de la subportadora es $1/T$, el primer punto nulo coincide con el pico de la PSD de la señal de la sub-banda adyacente.

3.3 VENTAJAS Y DESVENTAJAS DE LA MODULACIÓN OFDM

3.3.1 Ventajas

La modulación OFDM ofrece muchas ventajas frente a las modulaciones de portadora única, de las cuales las más notables son [2]:

- El periodo de símbolo T se alarga provocando que la señal sea más robusta contra la ISI causada por la dispersión temporal de canal (desvanecimientos selectivos en frecuencia) y las interferencias multicamino (ecos).
- Divide la anda frecuencial total en sub-bandas cercanas por lo que es menos sensible a un pulso ruidoso de banda ancha o bien a los desvanecimientos rápidos de canal.

- La ecualización de canal se puede realizar fácilmente mediante un único ecualizador si el desvanecimiento frecuencial selectivo es lento, ya que su atenuación afecta por igual a cada portadora.
- Se puede utilizar diferentes esquemas de modulación y tasas de código para las diferentes subportadoras dependiendo del nivel de ruido de cada sub-banda individual, aún teniendo que mantener el periodo de símbolo de cada portadora). El sistema OFDM se adapta al canal ruidoso reduciendo la tasa de datos o bien dejando de transmitir, según el nivel de ruido presente.
- OFDM permite la implementación digital gracias al uso de la transformación DFT/IDFT, que además permite obtener un algoritmo eficiente y sencillo mediante el uso de la FFT/IFFT, reduciendo así la complejidad del sistema. Para ello, el número de portadoras a transmitir debe ser un número potencia de 2.

3.3.2 Desventajas

Sin embargo, la modulación OFDM presenta algunos puntos problemáticos que se deben solventar mediante el uso de técnicas específicas [2]:

- Las desviaciones en frecuencia de las subportadoras degradan de manera considerable la calidad del enlace, ya que se pierde la ortogonalidad y las subportadoras se interfieren entre sí, fenómeno que se conoce como ICI (*Inter Carrier Interference)*. Es por ello que es necesario que se realice una sincronización perfecta tanto en tiempo como en frecuencia.
- El alto valor en la relación de PAPR (*Peak to Average Power Ratio*) de las señales OFDM a menudo fuerzan las capacidades de los amplificadores de potencia, haciéndolos trabajar en su zona de compresión, debido al gran margen dinámico. Por ello, es necesario utilizar algún método para reducir este nivel de señal.

3.4 MODULACIÓN Y DEMODULACIÓN OFDM

Históricamente, dos tipos de implementación del esquema modulador y demodulador OFDM (módem) se han usado. Uno utiliza osciladores y multiplicadores para el sistema modulador y correladores para el sistema demodulador, siguiendo el mismo esquema que el de portadora única, pero para gran número de portadoras dicho esquema resulta prohibitivamente complejo y muy caro. El segundo sistema utiliza la Transformación Discreta de Fourier, DFT (*Discrete Fourier Transform*), basándose en la envolvente compleja de la señal OFDM, ya descrita en el apartado anterior, resultando ser un método muy eficiente aún en presencia de un gran número de portadoras. A continuación se detallan estos métodos.

3.4.1 Módem OFDM analógico

En la Fig. 3.4 se presenta el esquema de modulación OFDM analógico cuya entrada son cadenas de bits (en el caso que se abarca en este proyecto serán cadenas de bits MPEG2 con codificación de canal) dispuestas en serie y cuya salida es la señal OFDM, *s(t),* expresada en (3.1). Después de obtener la señal en banda base se traslada la señal a frecuencias de RF (primero se puede trasladar a la banda IF y luego a la RF), para obtener la señal OFDM paso banda (RF) centrada en su f_c . Este último paso se omite en las explicaciones teóricas del documento puesto que el modelo de señal utilizado en las simulaciones es de banda base.

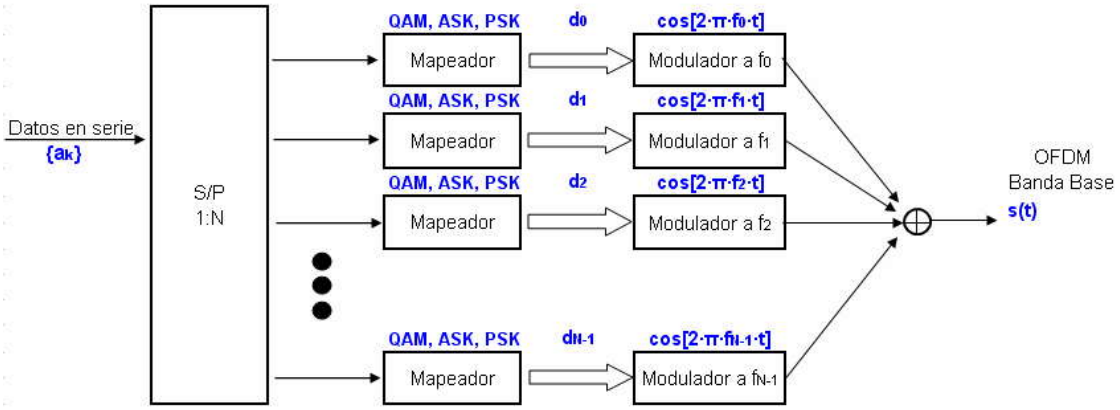


Fig. 3.4 Modulador OFDM analógico.

Los datos $\{a_k\}$ dispuestos en serie se convierten en N cadenas de datos dispuestos en paralelo al pasar por el bloque S/P. En el bloque de mapeo, los bits de cada cadena de datos independiente se agrupan en k-agrupaciones de bits dependiendo de su esquema de modulación (ASK, PSK, QAM), y cada k-agrupación se mapea a un símbolo que se denota en general por un número complejo:

$$d_i = A_i \cdot e^{j \cdot \phi_i} = I_i + j \cdot Q_i = A_i \cdot (\cos\phi_i + \operatorname{sen}\phi_i) \qquad (3.11)$$

donde I_i y Q_i son las componentes en fase y cuadratura del símbolo. Cada d_i es modulada en la sub-portadora i-ésima en el modulador. En total hay N moduladores para N frecuencias de sub-portadora: $f_0, f_1, \ldots, f_{N-1}$. La construcción de cada modulador depende del tipo de modulación empleada (MPSK, MASK, M-QAM), pero no se entrará en detalles debido a que se escapa del objetivo del proyecto. Una vez moduladas todas las sub-portadoras, éstas pasan por un bloque sumador en cuya salida se obtiene la señal OFDM banda base descrita en (3.1).

En la Fig. 3.5 se presenta el esquema de demodulación OFDM analógico que parte de la señal recibida ya convertida a banda base. El demodulador OFDM está formado básicamente por un banco de N demoduladores a N frecuencias portadoras. La estructura de cada uno de ellos, tal y como sucede en la modulación, depende del tipo de modulación que se aplica (MASK, MPSK, MQAM). Típicamente consisten en osciladores locales, multiplicadores, integradores y detectores de umbral.

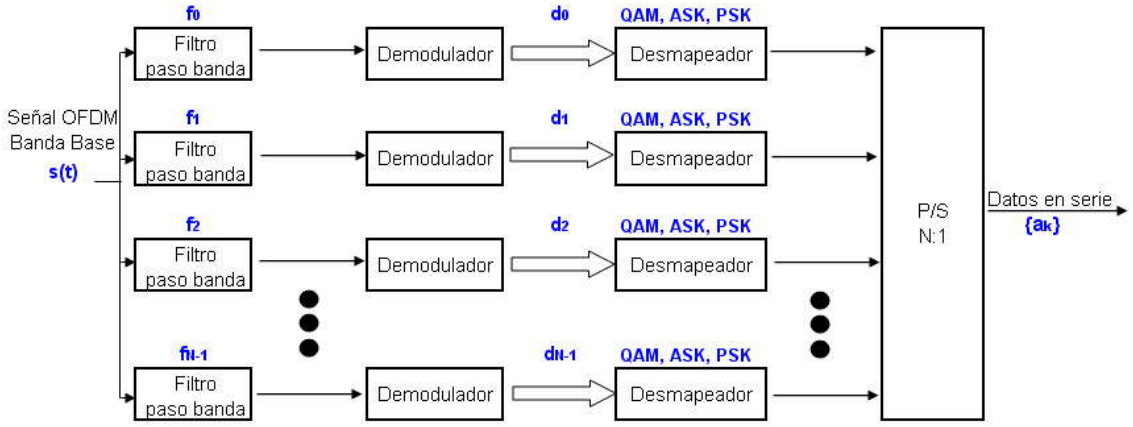


Fig. 3.5 Demodulador OFDM analógico.

Cuando el número de canales, N, es grande el número de componentes para el modulador y demodulador es también grande, lo cual puede ser nada práctico. Debido a esta complejidad, la implementación básica del módem OFDM utilizando osciladores, multiplicadores y correladores se ha visto obsoleta. Además, en demodulación se utilizan filtros paso banda que deben ser muy selectivos por lo que el precio y la complejidad aumentan mucho más.

3.4.2 Módem OFDM digital DFT. Implementación discreta

La señal OFDM banda base expresada en (3.1) se puede rescribir como la parte real de la envolvente compleja de la señal OFDM paso banda RF [2]:

$$s(t) = \mathrm{Re}\left[\sum_{i=0}^{N-1} \tilde{s}_i(t)\right] = \mathrm{Re}\left[\sum_{i=0}^{N-1} d_i \cdot e^{j\cdot 2\cdot\pi\cdot\frac{i}{T}t}\right], \qquad 0 \le \mathrm{t} \le T \tag{3.12}$$

Si se muestrea la envolvente compleja con un periodo de muestreo de $\Delta t=T/N$, y normalizando por el factor $1/N$, se obtiene la siguiente expresión:

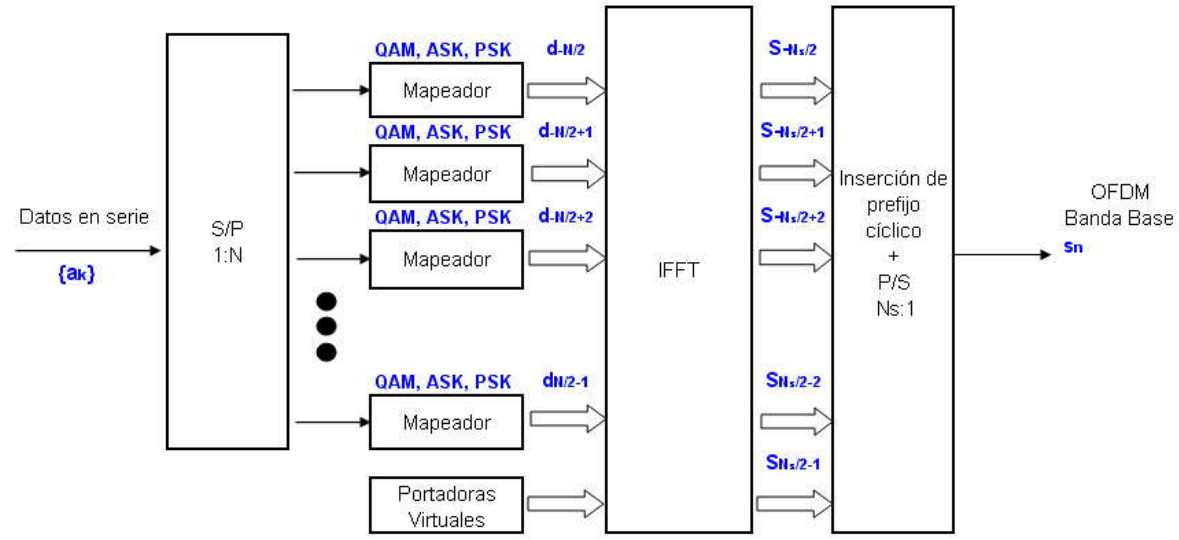
$$s_n = \frac{1}{N}\cdot\sum_{i=0}^{N-1} d_i \cdot e^{j\cdot 2\cdot\pi\cdot\frac{n}{N}t}, \qquad n = 0,1,\ldots, N-1 \tag{3.13}$$

La expresión (3.13) es la expresión de la Transformación Discreta Inversa de Fourier, IDFT (*Inverse Discrete Fourier Transform*). Este hecho implica que las muestras de la envolvente compleja de una señal OFDM pueden ser generadas a partir de la IDFT. Los datos complejos de entrada de la IDFT están en el dominio frecuencial y las muestras de salida de la IDFT, en general también complejas, están en el dominio temporal. Si el receptor recibe estas muestras sin ninguna distorsión los símbolos de datos d_i pueden ser recuperados perfectamente mediante la DFT, cuya expresión es:

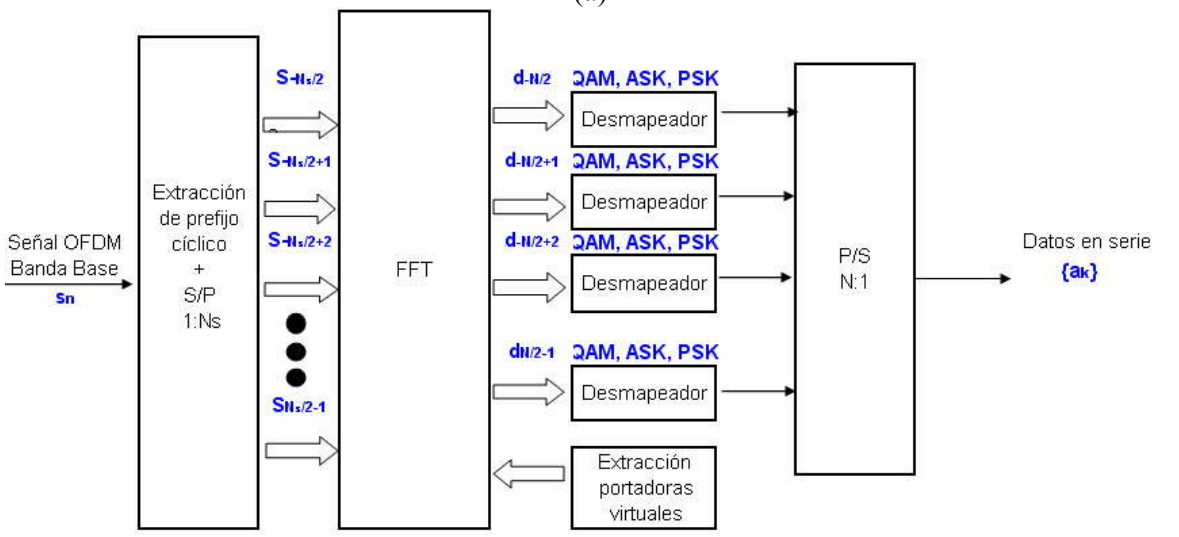
$$d_i = \sum_{n=0}^{N-1} s_n \cdot e^{-j\cdot 2\cdot\pi\cdot\frac{n}{N}t}, \qquad \mathrm{i} = 0,1,\ldots, N-1 \tag{3.14}$$

Esta relación implica que la modulación y la demodulación OFDM puede implementarse mediante el conjunto IDFT/DFT, simplificando gratamente el proceso. Además, el conjunto IDFT/DFT puede ser calculado computacionalmente mediante el conjunto IFFT/FFT (*Inverse Fast Fourier Transform*), siempre que el número de portadoras a modular sea una potencia de 2. Para ello, se suele aumentar el número de portadoras a un número N_S potencia de 2, normalmente mediante portadoras con valor 0. Mediante la adición de estas portadoras extra (virtuales) se realiza un sobremuestreo de la señal de datos con lo que, además, se consigue reducir el aliasing. Si se asume que se deben añadir L ceros, entonces se añaden $L/2$ ceros al inicio de la secuencia de portadoras y $L/2$ al final, quedando la siguiente señal de entrada del bloque IFFT:

$$\underbrace{0,0,....,0}_{L/2}, d_{-N/2}, d_{-N/2+1}, ..., d_{N/2-1}, \underbrace{0,0,....,0}_{L/2} \tag{3.15}$$



(a)



(b)

Fig. 3.6 Diagrama de bloques básico del módem OFDM implementado con IFFT/FFT (a) Modulador (b) Demodulador

En la Fig. 3.6 se ilustra el diagrama de bloques del módem OFDM basado en el conjunto IDFT/DFT. En el modulador, Fig. 3.6 (a), los datos dispuestos en serie, $\{a_k\}$, son convertidos a símbolos $\{d_i\}_{i=-N/2}^{N/2-1}$ gracias al bloque convertidor de serie a paralelo (S/P) y al bloque mapeador de bits a símbolos. Entonces se añaden el conjunto L de ceros necesarios para obtener (3.15) a la entrada del bloque IFFT, de longitud N_S. El bloque IFFT transforma los símbolos $\{d_i\}_{i=-N/2}^{N/2-1}$ a una señal de muestras complejas en el dominio del tiempo, $\{s_n\}_{n=-N_S/2}^{N_S/2-1}$. A continuación, se añade una extensión cíclica a la señal en forma de prefijo para hacerla más robusta frente al ISI y se prepara la señal s_n para ser transmitida en serie mediante el bloque P/S.

El demodulador es básicamente un sistema que realiza el proceso inverso al modulador (Fig. 3.6 (b)). Partiendo de la señal en banda base s_n muestreada, dicha señal pasa por el bloque de extracción del prefijo cíclico y convierte el resto de las muestras dispuestas en serie a muestras dispuestas en paralelo, $\{s_n\}_{n=-N_S/2}^{N_S/2-1}$. A continuación se realiza la FFT de la señal de N_s muestras (N_s puntos) y se eliminan las L portadoras virtuales (dispuestas en los extremos de la señal resultante, $\{d_i\}_{i=-N/2}^{N/2-1}$). Éste conjunto de datos resultante pasa por el bloque desmapeador para obtener, junto con el bloque P/S, la ristra de datos binarios $\{a_k\}$.

3.5 INTERVALO DE GUARDA. PREFIJO CÍCLICO

Tal y como se ha comentado en el apartado 3.3.1, una de las ventajas de utilizar OFDM en lugar de una modulación mediante portadora única es que en una señal OFDM se tiene mucha más duración de símbolo por lo que la longitud relativa de ISI causada por un canal de respuesta impulsional de longitud finita es mucho menor, haciendo que el efecto del ISI también sea menor. Un canal ideal debería representarse como un canal de respuesta impulsional de longitud nula, es decir, una función delta. En la práctica, los canales siempre tienen una respuesta impulsional de longitud finita. Los canales multicamino so uno de ellos, cuya respuesta impulsional es una suma de varias funciones delta (Véase Capítulo 5). Cada impulso o delta tiene distinta potencia, diferente fase y diferente retardo y, además, son variantes en el tiempo. La longitud de la respuesta impulsional del canal multicamino es la longitud de la dispersión del canal (*delay spread*).

Para eliminar el efecto de ISI se debe insertar un tiempo de guarda T_g entre los símbolos OFDM. El objetivo del intervalo de guarda es el de dejar tiempo suficiente para que las señales producidas por la dispersión multicamino se desvanezcan antes de que la información del símbolo actual sea recibida. Éste intervalo de guarda debe ser mayor que la longitud esperada de la respuesta impulsional del canal, es decir, si la longitud de *h(t)* o los retardos de las componentes multicamino son menores que T_g no aparece ISI.

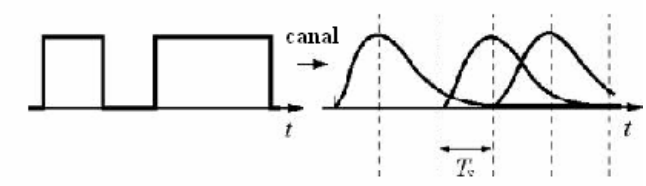


Fig. 3.7 Ejemplo de ISI

La implementación del simulador DVB-T se ha realizado en tiempo discreto, es decir, con señales *x[n],* donde *n* es un número entero que indexa las muestras de la señal muestreada *x*.

3.5.1 Convolución discreta y convolución circular. DFT/IDFT.

La convolución es un valor que se extiende a todos los sistemas que son invariantes linear del tiempo, LTI (*Linear Time Invariant*). La idea de convolución discreta es la misma que la de convolución continua. La convolución es un instrumento poderoso ya que se utiliza para determinar el resultado de un sistema, *y[n]*, conociendo la entrada, *x[n]*, y la respuesta impulsional del sistema, *h[n]*, del canal de transmisión en el caso que se abarca en este capítulo.

Al igual que sucede en tiempo continuo, la convolución discreta se representa por el símbolo * y puede ser expresada como:

$$y[n] = x[n] * h[n] = h[n] * x[n] \tag{3.16}$$

en donde podemos observar que se trata de un operador que posee la propiedad conmutativa.

Como ya ha sido mencionado, la suma de convolución provee una manera matemáticamente concisa para expresar el resultado de un sistema LTI, basado en una entrada arbitraria para una señal discreta y también el conocimiento de la respuesta del sistema. La suma de convolución se expresa como:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k] \tag{3.17}$$

Para entender mejor el funcionamiento, la Fig. 3.8 ilustra un ejemplo de convolución discreta mostrando sus propiedades. La convolución se puede realizar de esta manera ya que una señal discreta no es más que una suma de pulsos discretos escalados y desplazados.

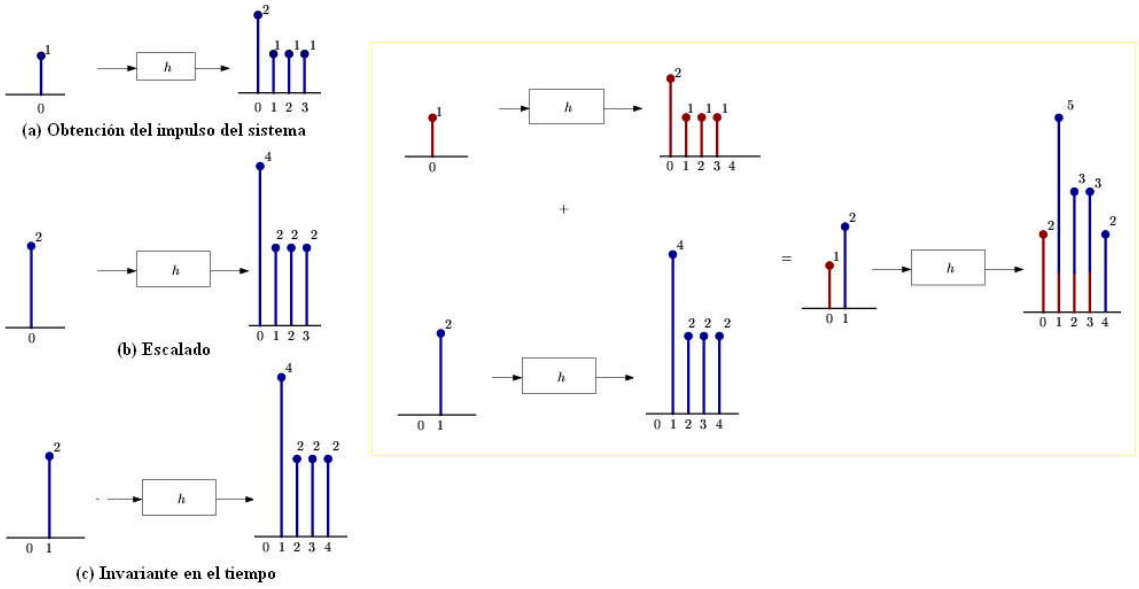


Fig. 3.8 Propiedades y ejemplo gráfico de convolución discreta

La interpretación de la convolución discreta más común es la que demuestra que la convolución construye el resultado a través del eje del tiempo. La Fig. 3.9 ilustra este proceso, en el que primero se debe reflejar la respuesta del sistema para, a continuación, "barrer" la señal de entrada, tal y como se expresa en (3.17).

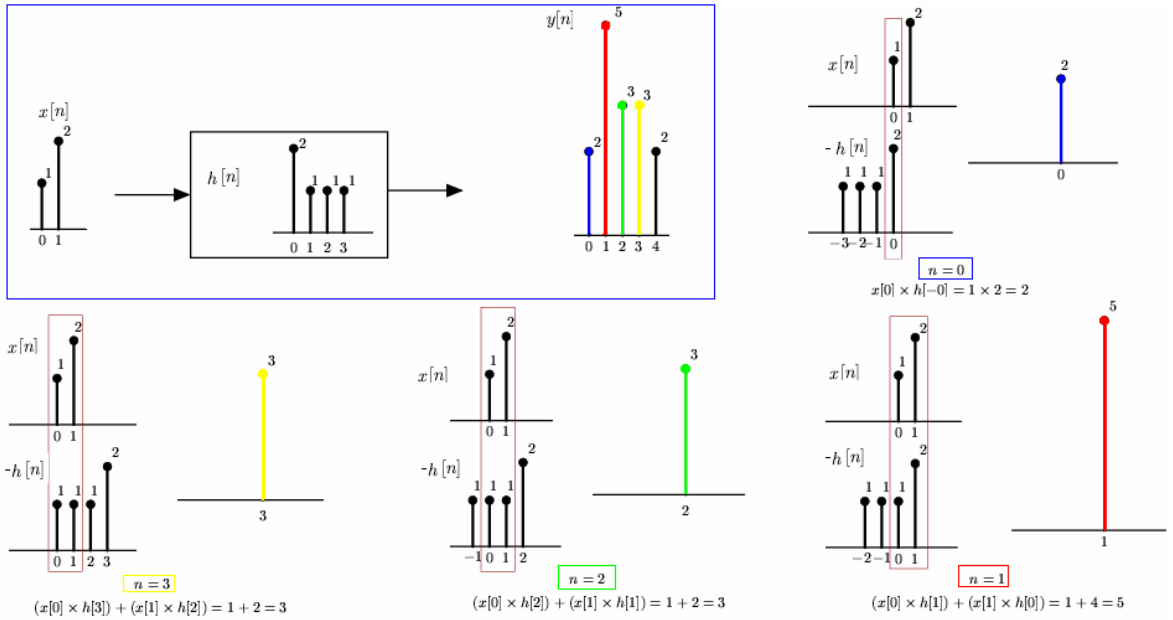


Fig. 3.9 Interpretación de la convolución discreta a través del eje del tiempo

En tiempo continuo, para facilitar el proceso de cálculo, se puede desarrollar la convolución lineal de dos señales continuas a partir de la multiplicación de dichas señales en el dominio frecuencial gracias a la Transformación de Fourier, FT (*Fourier Transform*):

$$x(t) * h(t) \overset{TF}{\leftrightarrow} X(f) \cdot Y(f) \tag{3.18}$$

Nótese que la flecha de la expresión (3.18) es bidireccional. Esto es debido a que la transformada frecuencial de dos señales continuas aperiódicas en el dominio del tiempo resulta en dos señales continuas y aperiódicas en el dominio frecuencial y viceversa. Entonces, si multiplicamos las dos señales continuas y aperiódicas en el dominio frecuencial obtenemos una señal continua y aperiódica en el dominio frecuencial y, por tanto, una señal aperiódica y continua en el dominio temporal tras realizar la IFT. De esta manera se puede calcular la convolución lineal de dos señales temporales continuas como resultado de la transformación a dominio frecuencial de la señal resultante de la multiplicación de ambas señales en el dominio frecuencial, gracias a la pareja FT/IFT.

La Fig. 3.10 ilustra el proceso de transformación de distintos tipos de señales a distintos dominios. En el caso de trabajar en el dominio temporal discreto aperiódico, la bidireccionalidad que se consigue con el caso anterior no se da. La transformación frecuencial de una señal temporal discreta y aperiódica resulta en una señal frecuencial continua y periódica. Si se entiende la señal discreta aperiódica en tiempo como una señal continua aperiódica en tiempo que ha sido discretizada a partir de la multiplicación de la segunda con un tren de deltas:

$$x[n] = \sum_{n=-\infty}^{\infty} \delta(t - nT) \cdot x(t) \tag{3.19}$$

se entiende que su transformada frecuencial se puede obtener mediante la convolución lineal de la transformada frecuencial del tren temporal de deltas con la transformada frecuencial de la señal continua temporal:

$$X(e^{j\omega}) = \frac{1}{T} \sum_{n=-\infty}^{\infty} \delta(\omega - \omega_S) * X(f) \quad \text{con } \omega_S = 2 \cdot \pi \cdot F_S = \frac{2 \cdot \pi}{T} \tag{3.20}$$

Nótese que el término *X(f)* en (3.20) es una señal frecuencial continua aperiódica y se ve convolucionada por un tren de deltas, por lo que se obtiene una señal frecuencial continua y periódica cada 2π. Éste proceso se puede realizar directamente con la Transformación de Fourier en Tiempo Discreto, DTFT (*Discrete Time Fourier Transform*), siendo:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} \quad \text{con } \omega_S = 2 \cdot \pi \cdot F_S = \frac{2 \cdot \pi}{T} \tag{3.21}$$

y cumpliéndose:

$$x[n] * h[n] \overset{DTFT}{\leftrightarrow} X(e^{j \cdot \varpi}) \cdot Y(e^{j \cdot \varpi}) \tag{3.22}$$

Al trabajar en el dominio frecuencial discreto para simplificar el cálculo de la convolución de la señal con la respuesta impulsional del sistema, o canal en el caso que este proyecto ocupa, se recurre a la Transformada Discreta de Fourier, DFT, cuya expresión se encuentra en (3.14) y se puede obtener fácilmente a partir de la discretización de la expresión de la DTFT en (3.21), a partir de la relación

$$\omega = \frac{2\pi}{N} k \tag{3.23}$$

Es decir, se puede realizar la convolución lineal de la señal temporal discreta aperiódica con el canal temporal discreto aperiódico mediante el producto de sus transformadas frecuenciales discretas y periódicas con el uso de la DFT:

$$x[n] * h[n] \overset{DFT}{\rightarrow} X(k) \cdot Y(k) \tag{3.24}$$

Nótese que en la relación expresada en (3.24) la implicación es unidireccional. Esto es debido a que la convolución de dos señales discretas y aperiódicas en el tiempo da como resultado una señal discreta y aperiódica en el tiempo. Si se realiza la transformada IDFT de la relación (3.22) se obtiene una señal temporal discreta periódica en donde aparecen réplicas cada *N* muestras, es decir, como si se tratara de una señal obtenida a partir de la convolución circular de dos señales temporales discretas aperiódicas.

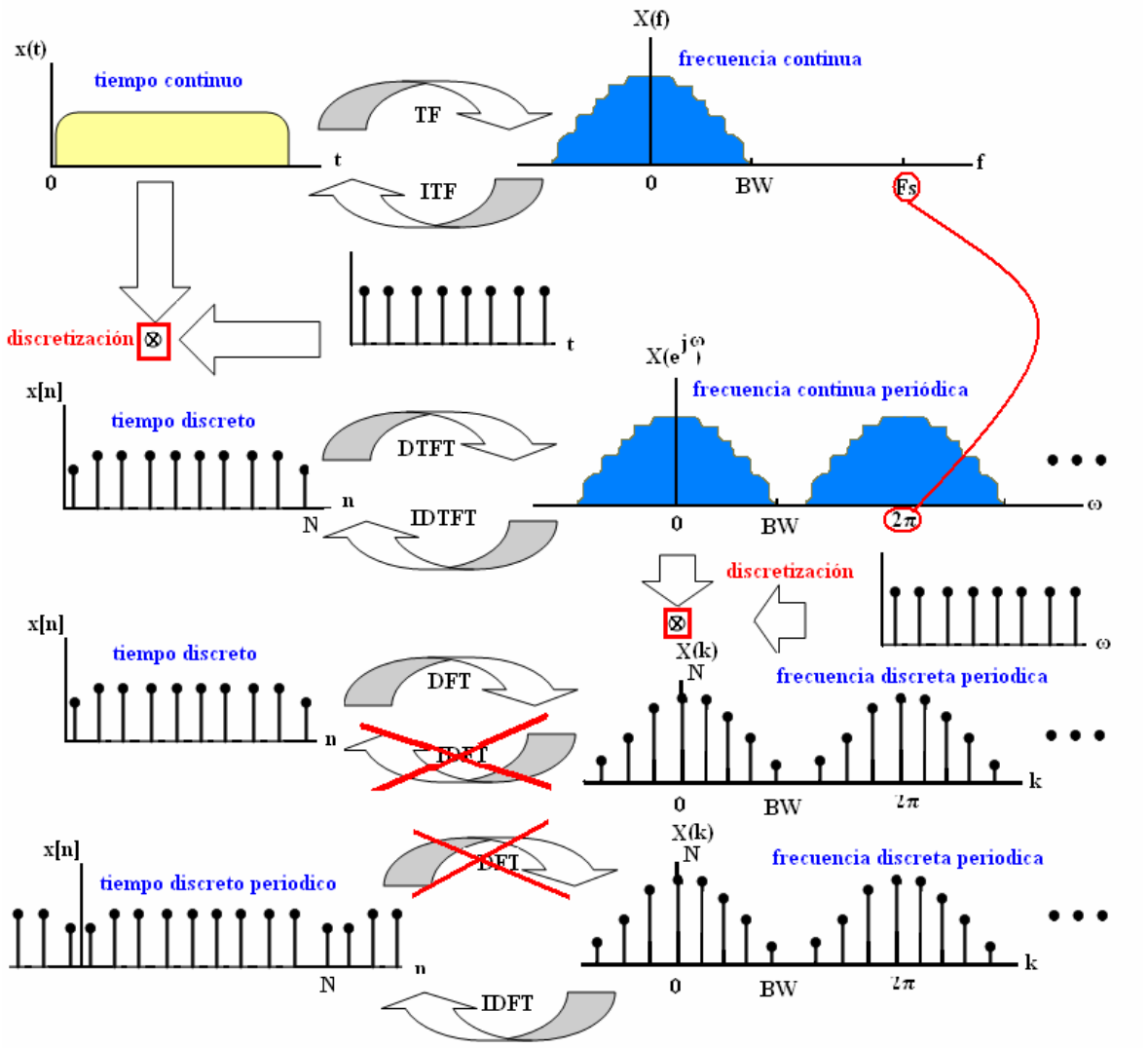


Fig. 3.10 Transformaciones de diferentes tipos de señales a distintos dominios

La convolución circular de dos señales, simbolizada con el operador $\otimes$, sigue la expresión:

$$y[n]_N = x[n] \otimes h[n] = \sum_{l=0}^{N-1} \left(h[l] \cdot \sum_{k=-\infty}^{\infty} x[n - k \cdot N - l] \right) = \sum_{k=-\infty}^{\infty} h[l] \cdot x[n-l]_{\text{mod } N} \qquad (3.25)$$

donde el subíndice N indica que la periodicidad se produce cada N muestras, es decir, que se trata de una versión periódica de la convolución lineal de periodo N.

Al utilizar la convolución circular, la expresión (3.24) adquiere implicación bidireccional en las transformaciones entre dominios, quedando:

$$x[n] \otimes h[n] \overset{DFT}{\leftrightarrow} X(k) \cdot Y(k) \qquad (3.26)$$

La estrategia que se sigue para poder utilizar la IDFT/DFT para realizar la convolución lineal de los datos de entrada de *N* muestras con la respuesta impulsional del sistema es hacer que la convolución lineal sea igual a la convolución circular a partir de la inserción de un prefijo en la señal de entrada, formando una nueva señal $\tilde{x}[n]$. Para que este prefijo sea cíclico debe estar formado por las últimas muestras de la señal temporal de entrada al sistema, tal y como ilustra la Fig. 3.11.

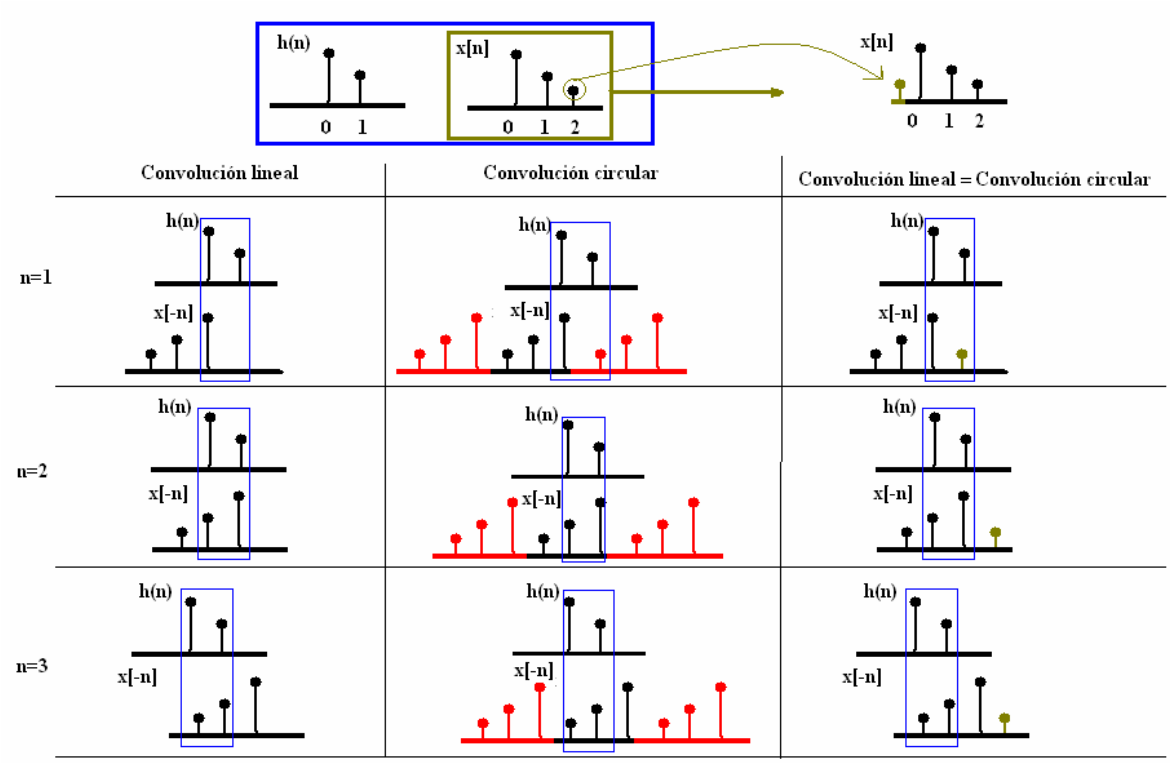


Fig. 3.11 Convolución lineal frente a convolución circular

pudiéndose expresar la convolución lineal como:

$$x[n] \otimes h[n] = \tilde{x}[n] * h[n] \overset{DFT}{\leftrightarrow} \tilde{X}(k) \cdot Y(k) \quad \text{si} \quad 0<n<N-1 \tag{3.27}$$

Obsérvese la restricción impuesta en (3.27). Dicha restricción indica que con el uso del prefijo cíclico la convolución lineal equivale a la convolución circular siempre y cuando la longitud de la respuesta impulsional del sistema (o canal), *L,* sea menor que la longitud de la señal de entrada al mismo, *N*.

3.5.2 Prefijo cíclico

Si se asume que el modelo de señal, vector de símbolos modulados por amplitud de pulsos PAM (*Pulse Amplitude Modulation*) a la salida del bloque mapeador, se puede representar como:

$$\underline{d} = (d_0, d_1, d_2, ..., , d_{N-1})^T \tag{3.28}$$

y que la matriz DFT, $[\underline{\underline{F}}]_{i,j} = e^{-j\cdot\frac{2\pi}{N}\cdot i\cdot j}$ con i=0, 1, ..., N-1 y j=0, 1, ..., N-1, es la siguiente:

$$\underline{\underline{F}}_{NxN} = \begin{pmatrix} 1 & 1 & \dots & \dots & 1 \\ 1 & e^{-j\cdot\frac{2\pi}{N}} & e^{-j\cdot\frac{2\pi}{N}\cdot 2} & \dots & e^{-j\cdot\frac{2\pi}{N}\cdot(N-1)} \\ 1 & e^{-j\cdot\frac{2\pi}{N}\cdot 2} & e^{-j\cdot\frac{2\pi}{N}\cdot 2\cdot 2} & \dots & e^{-j\cdot\frac{2\pi}{N}\cdot 2\cdot(N-1)\cdot} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & e^{-j\cdot\frac{2\pi}{N}(N-1)} & e^{-j\cdot\frac{2\pi}{N}(N-1)\cdot 2} & \dots & e^{-j\cdot\frac{2\pi}{N}(N-1)\cdot(N-1)} \end{pmatrix} \tag{3.29}$$

entonces el símbolo OFDM, $\underline{s}_{OFDM}$, a la salida del bloque IDFT se obtiene con la siguiente relación:

$$\underline{s}_{OFDM} = \underline{\underline{F}}^H \cdot \underline{d} \tag{3.30}$$

siendo el operador H el operador que define que la matriz $\underline{\underline{F}}$ es, en este caso, hermítica (traspuesta y conjugada). Es decir, $\underline{\underline{F}}^H$ representa la matriz IDFT.

Tras la obtención del vector columna $\underline{s}_{OFDM}$, dicho símbolo es procesado por el bloque P/S para llegar a obtener un vector serie que contiene todas las muestras de la señal representada en (3.30), siendo:

$$\underline{s}'_{OFDM} = \{s[0], s[1], s[2], ..., s[N-1]\} = \underline{s}_{OFDM}{}^T \tag{3.31}$$

La expresión de la señal (3.31) se puede rescribir como:

$$s'_{OFDM}[k] = \sum_{n=0}^{N-1} d[n]\cdot e^{j\cdot\frac{2\cdot\pi}{N}\cdot k\cdot n} \tag{3.32}$$

Al realizarse la transmisión, los datos de salida del transmisor atraviesan el canal, también discreto y en general complejo, cuya respuesta impulsional $h[n]$ se puede representar como:

$$h = \{h_0, h_1, h_2, ..., , h_{M-1}\} \tag{3.33}$$

siendo $M \leq N$.

En el receptor se obtiene la señal modificada por los efectos del canal, *y[n]*, que se obtiene al realizar la convolución lineal de los datos de salida del transmisor con la respuesta impulsional del canal:

$$y[n] = s'_{OFDM}[n] * h[n] = \sum_{k=-\infty}^{\infty} s[k] \cdot h[n-k] = \sum_{k=-\infty}^{\infty} h[k] \cdot s[n-k] \tag{3.34}$$

notándose que la longitud del vector $\underline{y}$ es de *N+M-1* muestras. La expresión descrita en (3.34) se puede rescribir en forma matricial como:

$$\begin{pmatrix} y[0] \\ y[1] \\ y[2] \\ ... \\ y[N+M-1] \end{pmatrix} = \begin{pmatrix} h[0] & 0 & 0 & ... & 0 \\ h[1] & h[0] & 0 & ... & 0 \\ h[2] & h[1] & h[0] & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & ... & 0 & 0 & h[0] \end{pmatrix}_{Nx(N+M-1)} \cdot \begin{pmatrix} s[0] \\ s[1] \\ s[2] \\ \\ s[N-1] \end{pmatrix} \tag{3.35}$$

Al añadir el prefijo cíclico de $N_{PC}<N$ muestras a la señal de salida del modulador OFDM, $\underline{s}'_{OFDM}$, se obtiene una señal $\underline{\tilde{s}}_{OFDM}$ de longitud $Ns=N+N_{PC,}$ que se puede expresar como:

$$\underline{\tilde{s}}_{OFDM} = (\underbrace{s_{N-N_{PC}}, s_{N-N_{PC}+1}, ..., s_{N-1}}_{\text{Prefijo cíclico}}, s_0, s_1, ..., s_{N-1})^T \tag{3.36}$$

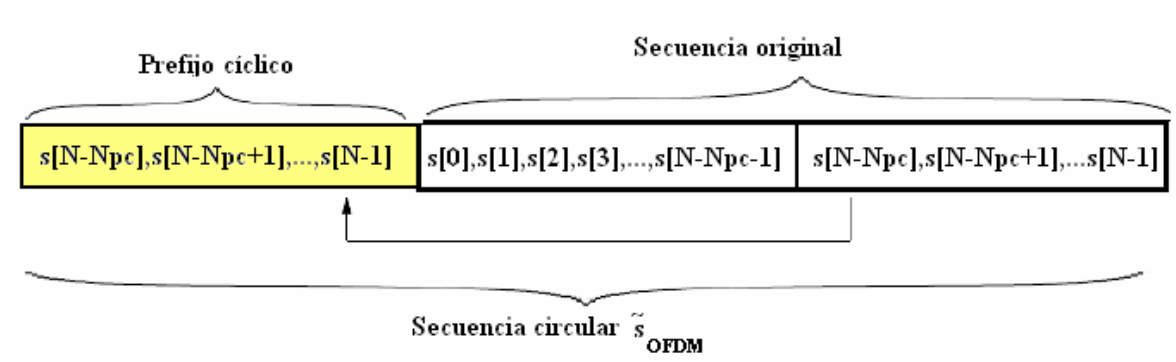


Fig. 3.12 Inserción de prefijo cíclico

por lo que la duración del símbolo aumenta pero de manera que no se pierde ortogonalidad entre portadoras ya que se copia al inicio una parte de señal de portadoras ortogonales entre ellas.

Suponemos ahora que $\tilde{s}_{OFDM}$ es la entrada a un canal discreto de longitud finita M, FIR (*Finite Impulse Response*). Entonces, la salida del canal $\tilde{y}$ [n] es:

$$\tilde{y}[n] = \tilde{s}_{OFDM} * h[n] \quad \text{para } 0 \le n \le N-1 \tag{3.37}$$

Si tenemos en cuenta la expresión en sumas de la convolución lineal, aplicando (3.17), y teniendo en cuenta (3.25), se puede rescribir (3.37) como:

$$\tilde{y}[n] = \sum_{k=-\infty}^{\infty} h[k] \cdot \tilde{s}_{OFDM}[n-k] = \sum_{k=-\infty}^{\infty} h[k] \cdot s_{OFDM}[n-k]_{\bmod N} = s_{OFDM}[n] \otimes h[n]$$
$$\tilde{x}[n-k] = x[n-k]_{\bmod N} \quad \text{si } 0 \le k \le N-1 \tag{3.38}$$

por lo que se puede afirmar que a la salida del canal, las N_s muestras corresponden a una convolución circular con el canal, tal y como ilustra la Fig. 3.13.

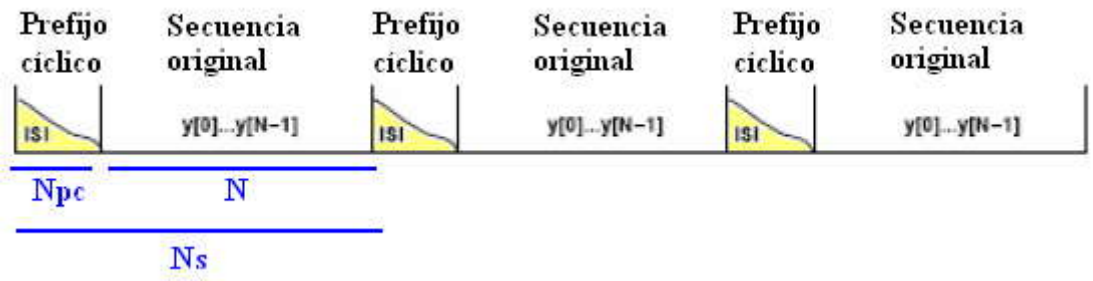


Fig. 3.13 Señal resultante a la salida del canal

Teniendo en cuenta (3.27), a la salida del canal la señal se puede expresar, en ausencia de ruido, como:

$$\begin{pmatrix} y[0] \\ y[1] \\ y[2] \\ \dots \\ y[N_S-1] \end{pmatrix} = \underbrace{\begin{pmatrix} h[0] & 0 & 0 & \dots & 0 \\ h[1] & h[0] & 0 & \dots & 0 \\ h[2] & h[1] & \dots & \dots & 0 \\ \dots & \dots & \dots & h[0] & \dots \\ 0 & h[M-1] & \dots & h[1] & h[0] \end{pmatrix}_{N_S x N_S}}_{\underline{\underline{H0}}} \begin{pmatrix} s[0] \\ s[1] \\ s[2] \\ \dots \\ s[N_S-1] \end{pmatrix} + \underbrace{\begin{pmatrix} 0 & 0 & h[M-2] & h[M-3] & h[M-4] \\ 0 & 0 & h[M-1] & h[M-2] & h[M-3] \\ 0 & 0 & 0 & h[M-1] & h[M-2] \\ \dots & 0 & 0 & .. & h[M-1] \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}_{N_S x N_S}}_{\underline{\underline{H1}}} \begin{pmatrix} s[0] \\ s[1] \\ s[2] \\ \dots \\ s[N_S-1] \end{pmatrix} \tag{3.39}$$

ya que tras la IDFT se obtiene una señal temporal discreta periódica. A modo de ejemplo, si se supone una longitud de canal M=3 y una longitud de símbolo de entrada N_s=4, la señal de salida se puede expresar como:

$$\underline{y} = (\underline{\underline{H}}_0 + \underline{\underline{H}}_1) \cdot \underline{\tilde{s}}_{OFDM} = \underline{\underline{\tilde{H}}} \cdot \underline{\tilde{s}}_{OFDM} = \begin{pmatrix} h[0] & 0 & h[2] & h[1] \\ h[1] & h[0] & 0 & h[2] \\ h[2] & h[1] & h[0] & 0 \\ 0 & h[2] & h[1] & h[0] \end{pmatrix} \begin{pmatrix} s[0] \\ s[1] \\ s[2] \\ s[3] \end{pmatrix} \tag{3.40}$$

siendo $\underline{\underline{\tilde{H}}}$ una matriz circulante (denominada matriz de Toeplitz).

Si la salida del modulador es la representada por la expresión (3.30) y la salida del canal es la representada por (3.40), a la salida del demodulador, en la que se ha aplicado la DFT, se tiene un vector $\underline{z}$ cuya expresión es:

$$\underline{z} = \underline{\underline{F}} \cdot \underline{y} = \underline{\underline{F}}(\underline{\underline{H}} \cdot \underline{s}_{OFDM}) = \underline{\underline{F}}\,\underline{\underline{\tilde{H}}}\,\underline{\underline{F}}^{H} \cdot \underline{d} \tag{3.41}$$

Dada una matriz circulante $\underline{\underline{M}}$ (Toeplitz) de dimensión *NxN*, se cumple que dicha matriz se puede descomponer en un producto de matrices de la forma:

$$\underline{\underline{M}} = \underline{\underline{F}} \cdot \underline{\underline{\Lambda}} \cdot \underline{\underline{F}}^{H} = \underline{\underline{F}} \begin{pmatrix} \lambda_0 & 0 & 0 & 0 \\ 0 & \lambda_1 & 0 & 0 \\ 0 & 0 & ... & 0 \\ 0 & 0 & 0 & \lambda_{N-1} \end{pmatrix} \underline{\underline{F}}^{H} \tag{3.42}$$

siendo $\underline{\underline{\Lambda}}$ una matriz que contiene los autovalores λ_i de $\underline{\underline{M}}$ es su diagonal y $\underline{\underline{F}}$ la matriz DFT definida en (3.29). Además, operando sobre la ecuación (3.42) y teniendo en cuenta que la matriz $\underline{\underline{M}}$ queda definida por un conjunto de coeficientes (columnas) $\underline{h}$, en el caso práctico que nos ocupa, se obtiene que:

$$\underline{\underline{\Lambda}} = \underline{\underline{F}}^{H}\,\underline{\underline{M}} \cdot \underline{\underline{F}} = diag\left\{ \underline{\underline{F}} \cdot \begin{bmatrix} \underline{h} \\ \underline{0} \end{bmatrix} \right\} \tag{3.43}$$

Por tanto, teniendo en cuenta la propiedad (3.42), la expresión que relaciona la salida del demodulador OFDM con la matriz circulante, (3.41), se puede rescribir de la siguiente manera:

$$\underline{z} = \underline{\underline{F}}\,\underline{\underline{\tilde{H}}}\,\underline{\underline{F}}^{H} \cdot \underline{d} = \underbrace{\underline{\underline{F}}(\underline{\underline{F}}^{H} \cdot \underline{\underline{\Lambda}} \cdot \underline{\underline{F}})\underline{\underline{F}}^{H}}_{Id}\,\underline{d} = \underline{\underline{\Lambda}} \cdot \underline{d} \tag{3.44}$$

Si, finalmente, se tiene en cuenta la relación obtenida en (3.43), se desprende que se recupera perfectamente la señal $\underline{d}$ de la entrada al modulador OFDM (transmisión) si se tiene conocimiento perfecto el canal.

$$\underline{d} = \underline{\underline{\Lambda}}^{-1} \cdot \underline{z} = diag\left\{ \underline{\underline{F}} \cdot \begin{bmatrix} \underline{h} \\ \underline{0} \end{bmatrix} \right\}^{-1} \cdot \underline{z} \tag{3.45}$$

Nótese que los elementos de la diagonal $\underline{\underline{\Lambda}}$ son la DFT de la respuesta impulsional del canal. Por este motivo, cuando no hay conocimiento de la respuesta impulsional del canal se recurre a técnicas de estimación sencillas en el dominio de la frecuencia. En este proyecto se estudia el caso la estimación de canal aplicada directamente teniendo en cuenta la estructura y las especificaciones del modelo de transmisión marcadas por el estándar DVB-T (Véase apartado 6.2.2).

Capítulo 4 ARQUITECTURA DEL SISTEMA TRANSMISOR DVB-T

En presente capítulo se realiza una introducción del sistema de transmisión basado en el estándar DVB-T [6] presentando el diagrama de bloques del transmisor cuyos parámetros de diseño vienen definidos por el estándar. A continuación se entra en profundidad en cada uno de estos bloques, dejando claro cuáles son sus principios de funcionamiento, objetivos y reglas de diseño, además de presentar el código Matlab implementado para cada uno de éstos en forma de función. También se incluyen gráficos y/o explicaciones de los métodos de validación de cada algoritmo. Para ofrecer información relativa a los datos de entrada y salida de cada bloque se presenta un diagrama conceptual del sistema implementado en el anexo 0, al cual se hace referencia en cada apartado.

La estructura de capítulo invita a conocer los bloques de transmisión en el mismo orden que el flujo de datos se los va encontrando, de manera que facilita la comprensión de la funcionalidad y los objetivos de los mismos.

4.1 INTRODUCCIÓN

El sistema de transmisión de TV digital denominado DVB-T (*Terrestrial Digital Video Broadcasting*), y recogido en el estándar ETSI (*European Telecommunications Standards Institute*) EN 300 744 [6] , especifica los procesos de codificación de canal y de modulación para un adecuado funcionamiento cuando se usan los canales de transmisión terrestre. Como el resto de estándares DVB, la señal de entrada normalizada es la denominada MPEG-2 TS (MPEG-2 *Transport Stream*), cuya traducción al idioma español es "*Flujo de transporte MPEG-2"*.

Dicho flujo de transporte, obtenido mediante el proceso denominado "Codificación de fuente" es una adaptación del estándar MPEG-2 (*Motion Picture Experts Group*) según ISO/IEC 13818-1 [7] que se estructura multiplexando varios programas y añadiendo la denominada SI (*Service Information*) correspondiente, según ETSI 300 468.

En este capítulo se describe la secuencia de operaciones denominada "Codificación de Canal", mediante la cual se añade suficiente redundancia y protección a la señal para hacerla más robusta con vistas a poder corregir los errores FEC (*Foward Error Correction*) después de pasar por el canal de transmisión.

También se describe el "Esquema de Modulación" usado en la transmisión, que es del tipo modulación multiportadora OFDM (*Orthogonal Frequency Division Multiplexing*).

El resultado, combinando el potente método de codificación para corrección de errores y la modulación multiportadora es una transmisión COFDM (*Coded Orthogonal Frequency Division Multiplexing*)

El sistema DVB-T es muy flexible, disponiéndose de una serie de opciones:

- 2 modos de transmisión: 2k (1705 portadoras de datos) y 8k (6817 portadoras de datos)
- 3 esquemas de modulación: QPSK, 16-QAM y 64-QAM.
- 5 relaciones de codificación para protección interna de errores:1/2, 2/3, 3/4, 5/6, 7/8
- 4 longitudes para el intervalo de guarda: 1/4, 1/8, 1/16, 1/32
- Modulación jerárquica o no jerárquica con diferentes valores del parámetro α (uniforme y no uniforme): 1, 2 o 4

Un aspecto a destacar de la técnica OFDM es que permite la operación, tanto en áreas pequeñas como en grandes, de redes de frecuencia única SFN (*Single Frequency Network*). Esto significa que mediante este sistema es posible la recepción cuando se radian idénticos programas desde diferentes transmisores que operan en la misma frecuencia. En estas condiciones se obtiene la máxima eficiencia del espectro, lo cual adquiere especial relevancia cuando se usa en las bandas de UHF (*Ultra-High Frequency*) asignadas para TV.

En la Fig. 4.1 se muestra de manera esquemática el diagrama de bloques funcional del sistema DVB-T según su estándar [6]. Los bloques funcionales usados para transmisiones jerárquicas están marcados con puntos. Los bloques implementados en este proyecto son los que están contenidos en el área de color amarillo.

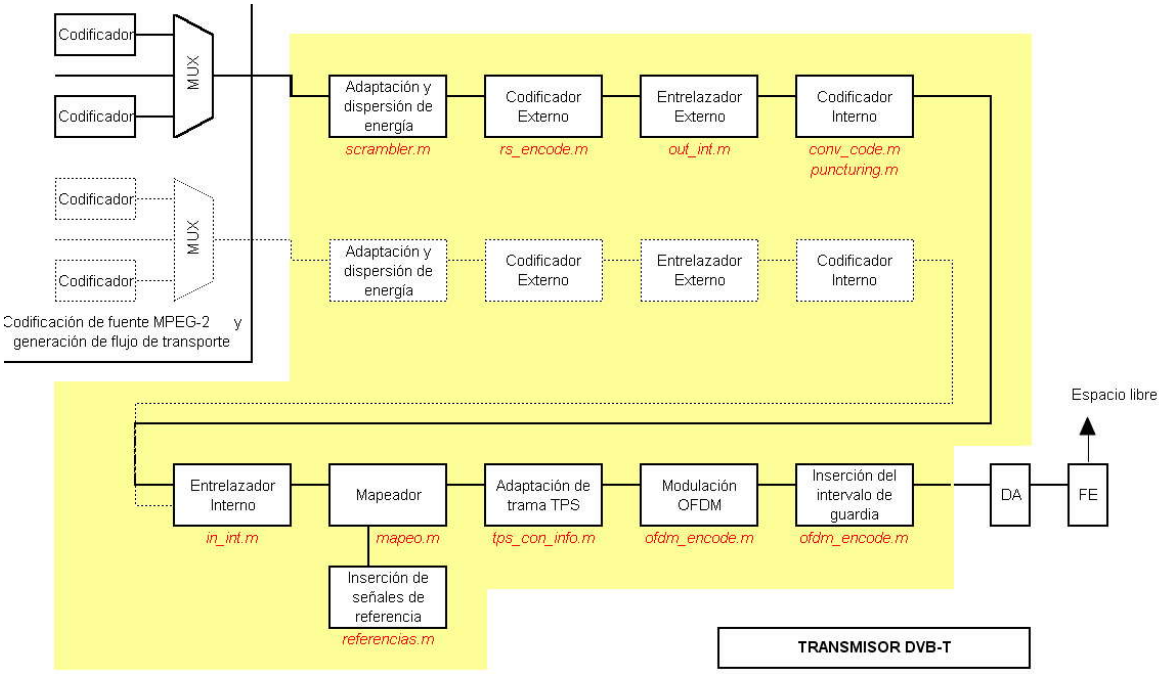


Fig. 4.1 Diagrama de bloques del sistema transmisor DVB-T.

El sistema DVB-T permite transmisiones jerárquicas (según el parámetro α, explicado en apartados posteriores). En este caso hay 2 flujos de transporte:

- Flujo de transporte de alta prioridad, HP (*High Priority bit stream)*: tiene baja velocidad y, por tanto, menor calidad de imagen, y que modula las portadoras con un esquema de modulación muy robusto frente al ruido QPSK

- Flujo de transporte de baja prioridad, LP (*Low Priority bit stream*): complementa al anterior en cuanto a velocidad y calidad de imagen y combina su información con el anterior de forma que las portadoras son moduladas finalmente con un esquema más exigente en cuanto a relación señal a ruido, SNR. En el caso de que este último utilice 4 bits por cada 2 bits de alta prioridad se alcanzaría una constelación total para la señal emitida de 64QAM.

En la zona del área de cobertura donde se reciba la señal con buena SNR, la imagen recuperada de alta calidad corresponderá a la combinación de los dos flujos (alta y baja prioridad), mientras que en caso contrario la calidad de imagen recibida será peor, correspondiendo sólo al flujo de alta prioridad [poner alguna referencia...]. En el apartado 0 se muestra un ejemplo del tratamiento especial de los datos según la jerarquía utilizada.

El documento que recoge las especificaciones del estándar DVB-T [6] dicta las pautas de diseño de un sistema de transmisión compatible con dicho estándar a partir de la definición de los parámetros de los bloques correspondientes a la codificación de canal y la modulación OFDM, es decir, de los bloques que forman el transmisor. De este modo, el sistema receptor queda abierto para así promover competencia entre los fabricantes de receptores [5].

Tal y como se ha comentado en el inicio del capítulo, la estructura de éste está enfocada a ir conociendo los bloques del sistema uno a uno teniendo en cuenta sus funcionalidades y sus especificaciones de diseño, tal y como dicta el estándar. Aprovechando esta estructura se incorpora un subapartado, en cada apartado de bloque, que incluye el código Matlab implementado para cada algoritmo en forma de función. En el anexo A.1 se presenta un diagrama conceptual del programa principal implementado que llama a estos bloques a partir de funciones Matlab, detallando su estructura general (flujo de datos) y ofreciendo información sobre el tipo de señal con la que trabaja cada bloque (entradas y salidas) además de detallar el dimensionado de los datos y los objetivos de cada bloque.

Para no extender el contenido de cada apartado por incluir un subapartado de entradas y salidas, se hace referencia en cada apartado a dicho anexoA.1, localización donde se encuentra este tipo de información de manera esquemática. Esta estructura de capítulo invita al lector a consultar el diagrama conceptual de la implementación del sistema por lo que, a medida que avanza en la lectura del presente documento, se va familiarizando con su funcionamiento.

Todas las especificaciones, condiciones y reglas de diseño comentadas en este capítulo están referenciadas al estándar DVB-T, documento [6]. Por este motivo no se harán referencias a éste cuando se presenten tablas, expresiones y estructuras de diseño de cada bloque. Si éste no es el caso, se hace referencia al documento de consulta en cuestión.

4.2 CODIFICACIÓN DE CANAL

Cada paquete de transporte TS del múltiplex de entrada en formato MPEG-2 tiene una longitud fija de 188 bytes, siendo el primer byte el de sincronización, cuyo valor es siempre:

$$0x47_{\text{HEXADECIMAL}} \equiv 01000111_{\text{BINARIO}} \equiv 74_{\text{DECIMAL}} \tag{4.1}$$

La estructura de paquete de transporte TS es la que muestra la Fig. 4.2

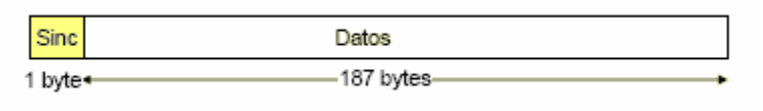


Fig. 4.2 Estructura del paquete de transporte MPEG-2

A continuación se presenta el código utilizado para crear los paquetes MPEG-2 (partir de una serie de bits aleatorios) que serán codificados, modulados y transmitidos.

```
function [mpeg2,ST,N,P,T,fft_mode] = genera_bitsMPEG2(ST, modo, n,supertrama,C,time_error,M )
%-------------------------------------------------------------------------
%    genera_bitsMPEG2.m
%-------------------------------------------------------------------------
fprintf('.');
switch modo
   case 2    N=1512;  P=193;  T=N+P;    fft_mode=2048;
   case 8    N=6048;  P=769;    T=N+P;    fft_mode=8192;
end
ST1=ST;
long_tramaMPEG2=188*8;                          En bits
long_tramaMPEG2_rs=204*8;                       En bits
long_superMPEG2=long_tramaMPEG2*8;              En bits
long_superMPEG2_rs=long_tramaMPEG2_rs*8;        Número de bits mpeg2 minimo a generar para formar
mpeg2_min=log2(n)*N*C;                          simbolo OFDM
N_trama=ceil(mpeg2_min/long_tramaMPEG2_rs);     Número de tramas a enviar para formar 1 simbolo
%-------------------------------------------------------------------------
%Aseguramos el envío de suficientes tramas/supertramas para garantizar la transmisión de 1 simbolo
%-------------------------------------------------------------------------
if supertrama==1
  if (ST<N_supertrama) ST=N_supertrama; end
elseif supertrama==0
  if (ST<N_trama) ST=N_trama; end
end
%-------------------------------------------------------------------------
%Creando ristra de datos MPEG2 (correspondiente a ST tramas/supertramas/simbolos)
%-------------------------------------------------------------------------
%Aseguramos que se transmite +1 simbolo piloto de estimación de error de tiempo, si esta activado!!)
if supertrama==1                                Adecuamos a tramas ya que el for del programa
ST=8*ST;                                        principal cuenta en tramas
   if time_error~=0    ST=ST+N_trama*8;         Le añadimos tramas para crear el simbolo piloto para
   end                                          estimación de tiempo
mpeg2=zeros(1,ST/8*(long_superMPEG2));
mpeg2=randint(1,ST/8*(long_superMPEG2),2);
elseif supertrama==0
if time_error~=0   ST=ST+N_trama;               Le añadimos tramas para crear el simbolo piloto para
 end                                            estimación de tiempo
mpeg2=zeros(1,ST*(long_tramaMPEG2));
mpeg2=randint(1,ST*(long_tramaMPEG2),2);
elseif supertrama==2  ST=ST*N_trama;            Adecuamos el numero de simbolos a enviar a tramas a
      if time_error~=0 ST=ST+N_trama;           crear
      end                                       Le añadimos tramas para crear el simbolo piloto para
mpeg2=zeros(1,ST*(long_tramaMPEG2));            estimación de tiempo
mpeg2=randint(1,ST*(long_tramaMPEG2),2);
```

Código 1 genera_bitsMPEG2.m

```
function [matriz_trama] = genera_paquetesMPEG2(data_in,T)
%-------------------------------------------------------------------------------------------
%           genera_paquetesMPEG2.m
%-------------------------------------------------------------------------------------------
long_tramaMPEG2=188*8;                          En bits
fprintf('·Tx: Creando trama %g MPEG-2',T);
fprintf('.');
SYNC_t=[0 1 0 0 0 1 1 1];
matriz_trama=[];
for i=1:188
   if (i==1)
      matriz_trama(i,:)=SYNC_t;
   else
   matriz_trama(i,:)=data_in((1+8*(i-1)+(long_tramaMPEG2*(T-1))):(8+8*(i-1)+(long_tramaMPEG2*(T-1))));
   end
end
fprintf('.');
fprintf('.\tOK\n');
```

Código 2 genera_paquetesMPEG2.m

4.2.1 Adaptación y dispersión de energía

4.2.1.1 Localización del bloque en el sistema

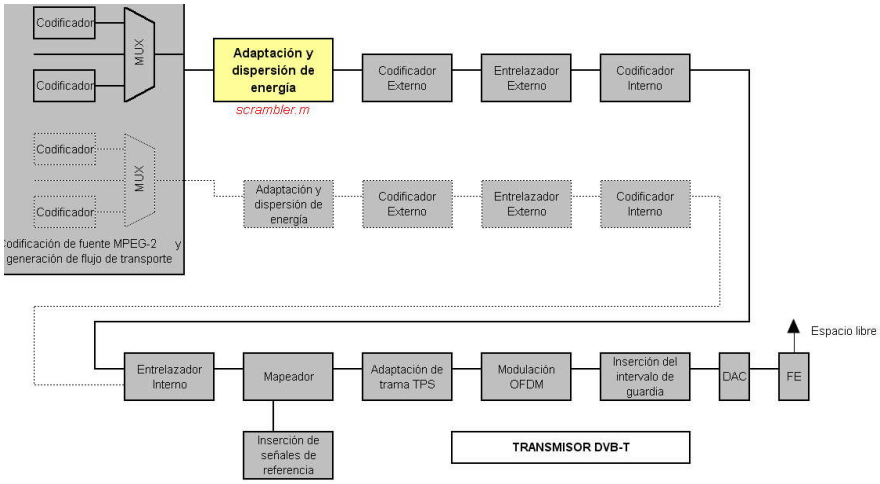


Fig. 4.3 Localización del bloque de Adaptación y dispersión de energía

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *scrambler.m.*

4.2.1.2 Objetivo y relación con el estándar

Para asegurar la dispersión de energía del espectro radiado, la señal de entrada debe hacerse casi-aleatoria, de forma que se eviten largas series de ceros o de unos. El razonamiento es el siguiente, la densidad espectral de potencia depende de la correlación entre los símbolos y la forma del pulso conformador que se transmite. En cuanto al pulso conformador, en caso de OFDM es un pulso rectangular, que es precisamente la ventana rectangular que multiplica a las exponenciales complejas de la FFT. Respecto a la correlación de símbolos, si los símbolos están completamente incorrelados el espectro depende únicamente de la transformada de Fourier del pulso conformador. Ahora bien, si los símbolos están correlados (por ejemplo, hay ráfagas de muchos 1's seguidos) en el espectro de la señal aparecen rayas espectrales (picos) que saturan a los amplificadores de potencia y que, por otro lado desperdician potencia porque son picos que no aportan información. Por ejemplo, si se envía una ráfaga de muchos 1's se malgasta potencia, por lo que es mejor enviar un único 1 e informar de la cantidad de unos que le siguen.

Por esta razón, el múltiplex de entrada debe ser tratado para convertirlo en aleatorio (*randomized*), lo cual se realiza mediante un proceso cuyo esquema se muestra en la Fig. 4.4 . En este proceso se trata de obtener una secuencia binaria pseudoaleatoria, o PRBS (*Pseudo Random Binary Sequence*), para el cual se emplea un generador que usa el siguiente polinomio:

$$1 + x^{14} + x^{15} \qquad (4.2)$$

El polinomio generador presentado en (4.2) se obtiene de las especificaciones de valores de referencia que se describen en [8].

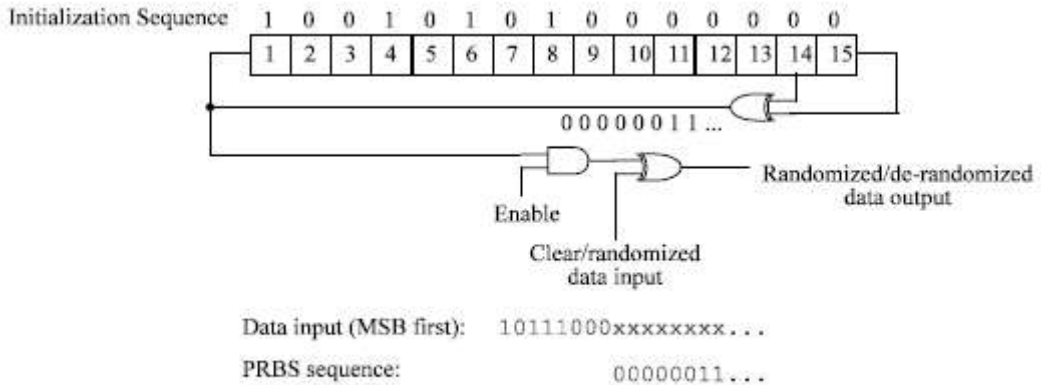


Fig. 4.4 Generador PRBS para la dispersión de energía del Flujo de Transporte

Los registros del generador PRBS, que sirve tanto para desordenar (Transmisión) como para ordenar (Recepción), tienen cargada inicialmente la siguiente secuencia:

$$100101010000000 \tag{4.3}$$

la cual debe iniciarse al comienzo de un conjunto de 8 paquetes de transporte. Es importante tener en cuenta que los bits de sincronización no se deben ver afectados por el proceso de desorden/orden.

Para proporcionar una señal de inicialización en el decodificador del receptor que permita identificar cada conjunto de 8 paquetes de transporte afectados por este proceso de desorden, el byte de sincronización del primer paquete de cada conjunto de 8 está invertido, pasando de (4.1) a:

$$0xB8_{\text{HEXADECIMAL}} \equiv 10111000_{\text{BINARIO}} \equiv 184_{\text{DECIMAL}} \tag{4.4}$$

Este último proceso se conoce como *Adaptación de flujo de transporte* y, juntamente con el proceso de dispersión, da como resultado la siguiente estructura de trama TS:

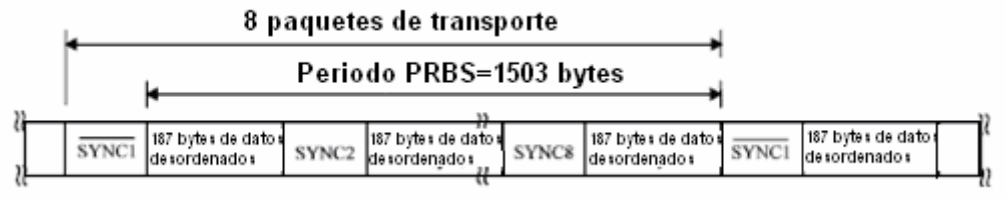


Fig. 4.5 Estructura del TS MPEG-2 resultado de la adaptación y dispersión de energía

4.2.1.3 Código implementado

```
1  function [trama_out, estado_r_prbs_out,T_index] = scrambler( trama_in, T_index,T, estado_r_prbs_in)
2  %-------------------------------------------------------------------------------
3  %    scrambler.m
4  %    Ver ETSI 300.744 v1.51 pag 11: Transport multiplex
5  %    adaptation and randomization for energy dispersal
6  %-------------------------------------------------------------------------------
7  if (mod(T_index-1,8)==0)                          Si se trata de la primera trama de supertrama...
8     SYNC_t=[1 0 1 1 1 0 0 0];                      Byte sincro trama pasa a invertido
9     estado_r_prbs_in=[1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0];   Se inicializa registro prbs (184dec)
10    fprintf('.');
11 else                                              Si se trata de las otras 7...
12    SYNC_t=[0 1 0 0 0 1 1 1];                      Byte sincro trama se queda igual (71dec)
13                                                   Avanzamos en 1byte el registro prbs, asi no afecta al de
      for i=1:8                                      sync pero si al de datos
14
bit_nuevo=xor(estado_r_prbs_in(14),estado_r_prbs_in(1
5));
15       estado_r_prbs_in=[bit_nuevo,
estado_r_prbs_in(1:14)];
16    end
17 fprintf('.');
18 end
19 %-------------------------------------------------------------------------
20 %Aplicamos scrambler al resto de tramas de datos
21 %-------------------------------------------------------------------------
22 secuencia_prbs=zeros(1,8*187);                   Vector fila en bits
23 for i=1:(8*187)
24
bit_nuevo=xor(estado_r_prbs_in(14),estado_r_prbs_in(1
5));
25       estado_r_prbs_in=[bit_nuevo,
estado_r_prbs_in(1:14)];
26       secuencia_prbs(i)=bit_nuevo;
27 end
28 secuencia_prbs_bytes=reshape(secuencia_prbs,8,187)'
;
29 estado_r_prbs_out=estado_r_prbs_in;
30 %-------------------------------------------------------------------------
31 % Formación de la trama (SYNC+prbs)
32 %-------------------------------------------------------------------------
33 trama_out=[SYNC_t;
bitxor(trama_in(2:188,:),secuencia_prbs_bytes)];
34 fprintf('.\tOK\n');
```

Código 3 scrambler.m

4.2.2 Codificación Externa (Reed-Solomon)

4.2.2.1 Localización del bloque en el sistema

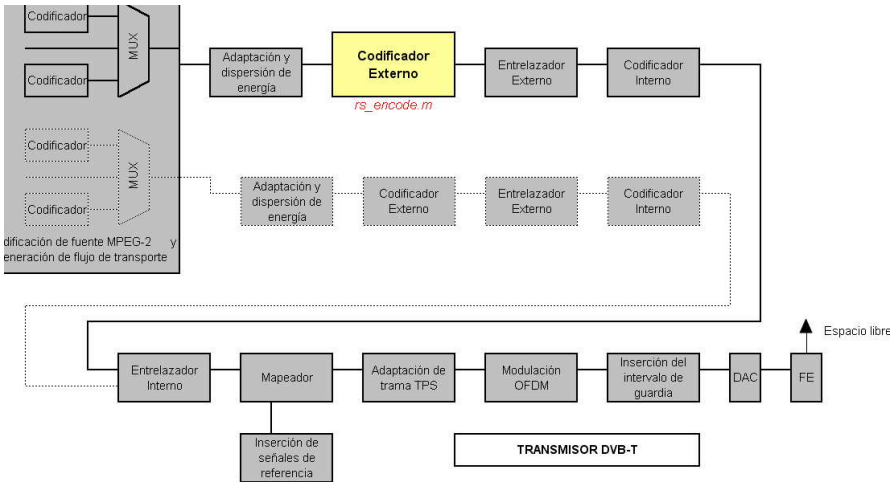


Fig. 4.6 Localización del bloque de codificación externa

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *rs_encode.m.*

4.2.2.2 Objetivo y relación con el estándar

Para permitir la corrección de errores, FEC, en la recepción, se introduce una cierta redundancia en la estructura de los paquetes de transporte, procedimiento que se conoce como codificación.

La codificación llamada "externa" se emplea en todos los estándares DVB y se complementa con otra llamada "interna" en el caso de los estándares de transmisión vía terrestre.

La "codificación externa" usada es de tipo Reed-Solomon, RS(204,188,t=8), que es una versión acortada de la codificación original RS(255,239,t=8), mediante la cual se añaden 16 bytes de paridad a los iniciales 188 bytes de cada paquete de transporte, dando como resultado un paquete protegido contra errores con la estructura que se muestra en la Fig. 4.7.

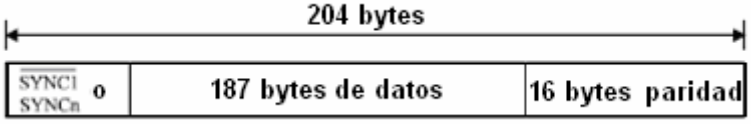


Fig. 4.7 Estructura del paquete de transporte protegido contra errores

En estas condiciones pueden corregirse hasta un total de 8 bytes erróneos, de ahí la denominación de código: RS(20,188,t=8).

La forma acortada se realiza añadiendo primeramente 51 bytes nulos delante de los 188 bytes de información, con lo que se obtienen 239 bytes con los 188 bytes iniciales del paquete TS. Al pasar por el codificador RS(255,239,t=8) se añaden los 16 bytes de paridad, por lo que se obtienen finalmente 255 bytes por cada paquete TS. Finalmente se eliminan los 51 bytes nulos insertados inicialmente con lo que resultan los 204 bytes para cada paquete de transporte afectado por la codificación externa (*Outer Interleaver*).

Para conocer más detalles referentes a los fundamentos de la codificación para la corrección de errores y los campos de Galois consultar el Anexo C.

4.2.2.3 Código implementado

```
1  function trama_out = rs_encode(trama_in,n,k,t,T)
2  %-------------------------------------------------------------------------------------------
3  %    rs_encode.m
4  %    Ver ETSI 300.744 v1.51 4.3.2 pag 11-12: OuterCoding
5  %-------------------------------------------------------------------------------------------
6  fprintf('·Tx: Paso de la trama %d por etapa de Outer Coding (Reed Solomon) para formar bloque %g',T,T);
7  fprintf('.');
8  %-------------------------------------------------------------------------------------------
9  % Creando estructura del polinomio generador de campo          8   4   3   2
10 %                                        p(x) = x  + x  + x  + 1
11 %-------------------------------------------------------------------------------------------
12 [q,all,nz,fgen,alfa,gflog,gfexp,x]=gf_init (2, 8, [8 4 3 2 0]);
13 %-------------------------------------------------------------------------------------------
14 %  Inicializacion del Codificador Reed Solomon
15 %-------------------------------------------------------------------------------------------
16 [SYrsn,SYrsk,padding,lamda,r,g,gfexp,gflog,q]=rs_init (n, k, t,q,alfa,gfexp,gflog);
17 fprintf('.');
18 %-------------------------------------------------------------------------------------------
19 %           Codificación Reed Solomon
20 %-------------------------------------------------------------------------------------------
21 parity=zeros(2*t,1);
22 parity(:)=0;
23 for index = 1:k
24 feedback = gf_add (trama_in(index), parity(1));
25 parity = gf_add ([parity(2:2*t) ; 0],gf_mul (feedback,g(2*t:-1:1),gflog,gfexp,q));
26 end
27 trama_out = [trama_in ; parity ];
28 fprintf('.\tOK\n');
```

Código 4 rs_encode.m

La inicialización del bloque rs_encode (línea 16) se realiza con el siguiente código implementado. Es el mismo para la inicialización del bloque rs_decode.

```
1  function [SYrsn,SYrsk,padding,lamda,r,g,gfexp,gflog,q]=rs_init (n, k, t,q,alfa,gfexp,gflog)
2  %-------------------------------------------------------------------------
3  %% rs_init.m
4  %%
5  %-------------------------------------------------------------------------
6  % RS(255,239,t=8) code (p. 11)
7  SYrsn =q-1; % length
8  SYrsk = SYrsn - 2*t;
9  % RS(204,188,t=8) code (p. 12)
10 padding =SYrsn -n;
11 % Polinomio generador codificado
12 lamda =alfa;
13 % Obtención de raíces del polinomio generador
14 r = zeros (2*t,1);
15 raised_lamda = 1;
16 for u = 1:2*t
17 r(u) = raised_lamda;
18    raised_lamda = gf_mul (raised_lamda, lamda,gflog,gfexp,q);
19 end
20 %------------------------------------------- u-1------
21 %Creando polinomio generador  g(x) := (x+lamda) g(x)
22 %-----------------------------------------------------
23 g = zeros (2*t+1,1);
24 g(1)=1;
25 for u = 1:2*t
26 g(u+1) =1;
27    for s = u:-1:2
28      g(s) = gf_add (g(s-1), gf_mul (r(u),g(s),gflog,gfexp,q));
29    end
30    g(1) = gf_mul(r(u),g(1),gflog,gfexp,q);
31  end
```

Código 5 rs_init.m

Para crear la estructura de datos según la aritmética de campos de Galois, se recurre al siguiente código implementado, que se llama desde la función principal rs_encode.m en la línea 12). Éste código es el mismo que se utiliza en la función rs_decode.m.

```
1  function [q,all,nz,fgen,alfa,gflog,gfexp,x]=gf_init (p, m, field_generator)
2  %-------------------------------------------------------------------------------------------
3  %  gf_init.m
4  %-------------------------------------------------------------------------------------------
5  %                                                    m
6  % Número de elementos del campo de Galois, GF(q) = GF(p )
7  q =p ^ m;
8  % Todos los elementos
9  all = 0:q-1;
10 nz = 1:q-1;
11 % Expandiendo la representación del polinomio generador de campo
12 fgen = zeros(m+1,1);
13 for k = 1:length(field_generator)
14   fgen(1+field_generator(k)) = 1;
15 end
16 alfa = p;
17 % Tabla logarítmica y exponencial para multiplicación y división
18 gflog = zeros(q-1,1);
19 gfexp = zeros(q-1,1);
20 x = zeros (m+1,1);
21 x(1) = 1;
22 for k = 1:q-1
23   index = 0;
24   for pos = 0:m-1
25     index = index + x(1+pos) * p^(pos);
26   end
27   gfexp(k) = index;
28   gflog(index) = k;
29   x = [ 0 ; x(1:m) ];
30   if x(m+1) == 1
31     x = xor (x, fgen);
32   end
33 end
```

Código 6 gf_init.m

Las operaciones de aritmética de Galois auxiliares utilizadas en la función gf_init.m se presentan a continuación:

```
1  %-------------------------------------------------------------------------------------------
2  % gf_add.m
3  %-------------------------------------------------------------------------------------------
4  %Suma dos elementos de un campo de Galois
5  function result = gf_add (a, b)
6    result = bitxor (a,b);
7
8  %-------------------------------------------------------------------------------------------
9  % gf_eval.m
10 %-------------------------------------------------------------------------------------------
11 %Evalua un polinomio con aritmética de Galois
12 function result = gf_eval (polynome, x,gflog,gfexp,q)
13 result = 0;
14 for ii = length(polynome):-1:1
15 result = gf_add (gf_mul (result, x,gflog,gfexp,q),
   polynome(ii));
16 end
17
18 %-------------------------------------------------------------------------------------------
19 % gf_inv.m
20 %-------------------------------------------------------------------------------------------
21 %Devuelve el inverso del campo de Galois
22 function result = gf_inv (a,q,gfexp,gflog)
23   result = gf_exp((q-1) - gf_log(a,gflog),gfexp,q);
24
25 %-------------------------------------------------------------------------------------------
26 % gf_inv.m
27 %-------------------------------------------------------------------------------------------
28 %Multiplica dos elementos de Galois
29 function result = gf_mul (a,b,gflog,gfexp,q)
30   result = gf_exp(gf_log(a,gflog) +
   gf_log(b,gflog),gfexp,q);
31
32
33
```

```
%-------------------------------------------------------------------------------------------
% gf_exp.m
%-------------------------------------------------------------------------------------------
%Eleva a un cierto exponente el elemento de Galois
function result = gf_exp (a,gfexp,q)
  [m, n] = size (a);
  result = zeros (m, n);
  for i = 1:m
    for j = 1:n
      if a(i,j) == -Inf
result(i,j) = 0;
      else
result(i,j) = gfexp(1+rem(a(i,j),q-1));
      end

    end
  end
```

```
%-------------------------------------------------------------------------------------------
% gf_log.m
%-------------------------------------------------------------------------------------------
%Devuelve un entero logarítmico de Galois
function result = gf_log (a,gflog)
  [m, n] = size (a);
  result = zeros (m, n);
  for i = 1:m
    for j = 1:n
      if a(i,j) == 0
result(i,j) = -Inf;
      else

result(i,j) = gflog(a(i,j))-1;
      end
    end
  end
```

Código 7 gf_add.m, gf_eval.m, gf_inv.m, gf_exp.m, gf_log.m, gf_exp.m

4.2.3 Entrelazado externo (Forney)

4.2.3.1 Localización del bloque en el sistema

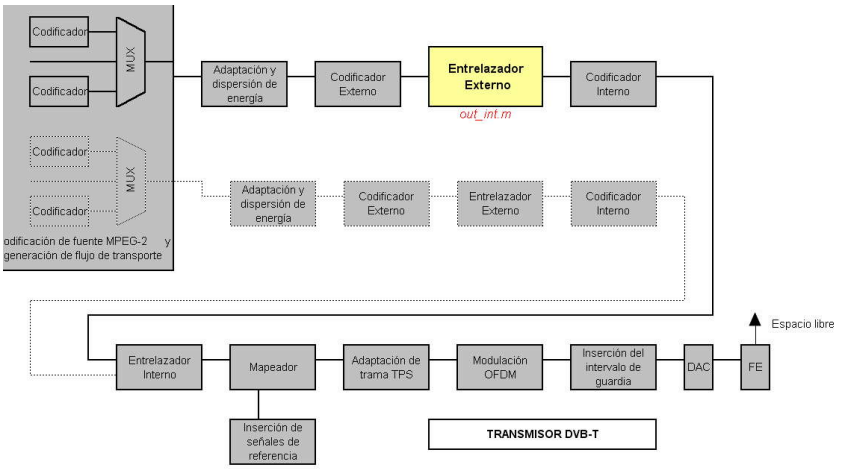


Fig. 4.8 Localización del bloque de entrelazado externo

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *out_int.m*.

4.2.3.2 Objetivo y relación con el estándar

La Fig. 4.9 muestra el esquema de concepto del "Entrelazado Convolucional Externo" (*Outer Interleaver*) relativo al byte y con profundidad I=12, al que se someten los datos que han sido previamente protegidos mediante la codificación externa RS.

El proceso de entrelazado convolucional se basa en la aproximación de Forney, compatible con la aproximación de Ramsey tipo III, con I=12. Los bytes de datos entrelazados, pertenecientes a los paquetes de transporte protegidos, están delimitados por los bytes de sincronización MPEG-2 (invertidos y no invertidos) que no sufren alteración alguna. Por tanto, el entrelazado preserva la periodicidad de 204 bytes de los paquetes de transporte.

El bloque de entrelazado se compone de 12 ramas, cíclicamente conectadas al flujo de datos de entrada mediante el conmutador de entrada. Cada rama *"j"* constituye un registro de desplazamiento FIFO (*Fist-In, Fist-Out*) con profundidad *"jxM"* células, donde:

$$M = N / I \tag{4.5}$$

por lo que M=17 celdas, siendo N=204 bytes e I=12 ramas.

Los conmutadores de entrada y de salida, que avanzan un paso por cada byte de datos, deben estar sincronizados. Los bytes de sincronización SYNC, invertidos y no invertidos, deben pasar siempre por la rama "0" del bloque de entrelazado, que corresponde a la de retardo nulo, para poder ser localizados.

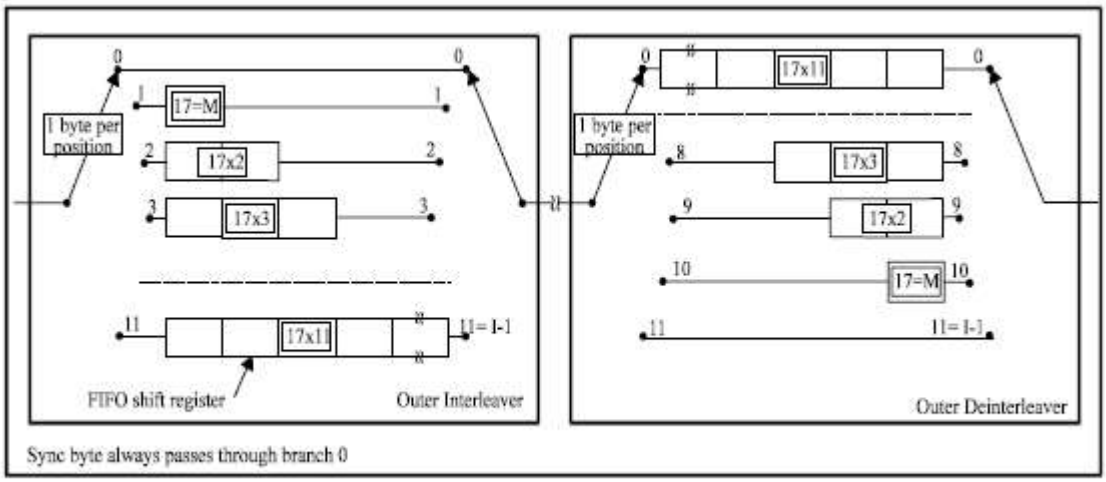


Fig. 4.9 Diagrama conceptual del Entrelazado y Desentrelazado Convolucional Externo

El resultado del proceso de entrelazado es que cada byte de los paquetes de transporte se encuentra desplazado en el tiempo un número de posiciones igual a "*jx17*", con lo que los bytes originales de un paquete de transporte quedarán repartidos entre dos paquetes consecutivos.

Tras el bloque de entrelazado externo, la estructura del paquete de datos MPEG-2 queda como se muestra en la Fig. 4.10.

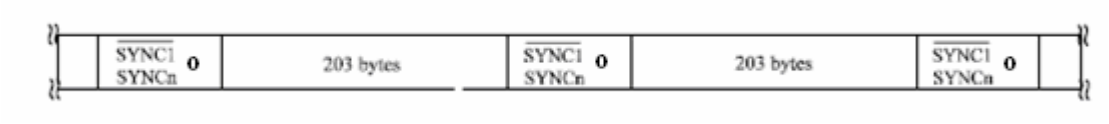


Fig. 4.10 Estructura de datos tras el bloque de Entrelazado Convolucional Externo

El bloque de desentrelazado tiene un principio similar al bloque de entrelazado con la salvedad de que los índices de las ramas están invertidos, es decir, "j=0" se corresponde al máximo retardo. La sincronización se obtiene haciendo pasar el primer byte SYNC reconocido, invertido o no, hacia la rama "0".

En estas condiciones, en el bloque de desentrelazado cada byte se retarda un total de posiciones igual a

$$(11 - j)\cdot 17 \qquad (4.6)$$

por lo que el retardo total entre emisión y recepción es de (j+11-j)=187, valor que permite recuperar en la recepción el orden original al ser un retardo idéntico para todos los bytes .

Todo este proceso reduce los errores por ráfagas introducidos por el canal de transmisión (errores que afectan a varios bytes consecutivos), ya que, después de la reordenación de los datos en el receptor, estos errores se habrán distribuido entre paquetes sucesivos. Este hecho favorecerá que no se excedan los límites en los que la codificación RS puede recuperar la información original.

4.2.3.3 Código implementado

```
1  function [trama_out,cola_out_int] = out_int(trama_in,I,M,N,cola_out_int,T)
2  %--------------------------------------------------------------------------------
3  %    out_int.m
4  %    Ver ETSI 300.744 v1.51 4.3.2 pag 11-12:
5  %    Outer Interleaving
6  %--------------------------------------------------------------------------------
7  fprintf('·Tx: Paso del bloque %d por etapa de Outer Interleaving (Entrelazado externo)',T);
8  fprintf('.');
9  for u = 1:M
10    for i = 1:I
11      cola_out_int((i-1)*N + (u-1)*I + i) =trama_in((u-1)*I + i); %Aquí trama_in es una columna con el valor de
                                                                    cada byte en decimal.
12    end
13 end
14 fprintf('.');
15 trama_out = cola_out_int(1:N);
16 cola_out_int = [cola_out_int(N+1:(I^2*M)); zeros(N,1)];
17 fprintf('.\tOK\n');
```

Código 8 out_int.m

4.2.4 Codificación Interna (Convolucional)

4.2.4.1 Localización del bloque en el sistema

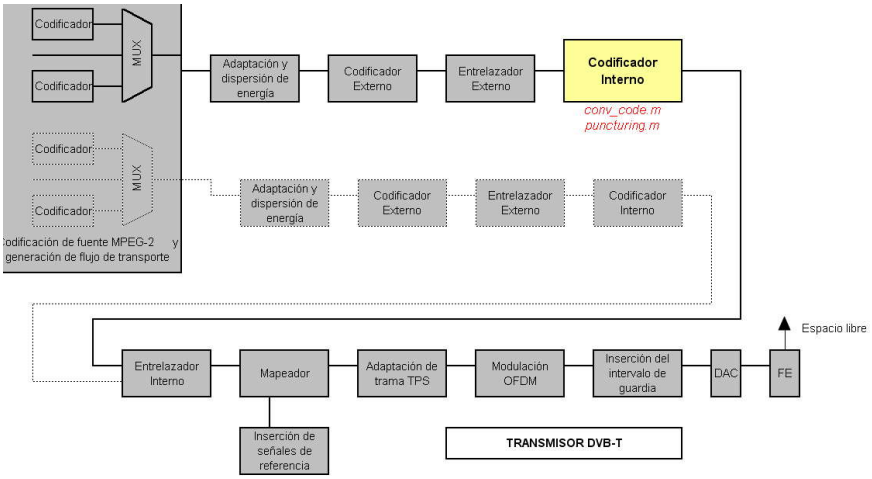


Fig. 4.11 Localización del b*loque de codificación interna*

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase apartado A.1), es *conv_code.m.*

4.2.4.2 Objetivo y relación con el estándar

Después de la codificación y entrelazado externos, los datos se someten a un nuevo proceso de codificación y entrelazado denominados "internos", realzado por el conjunto de bloques que se muestra en la Fig. 4.12.

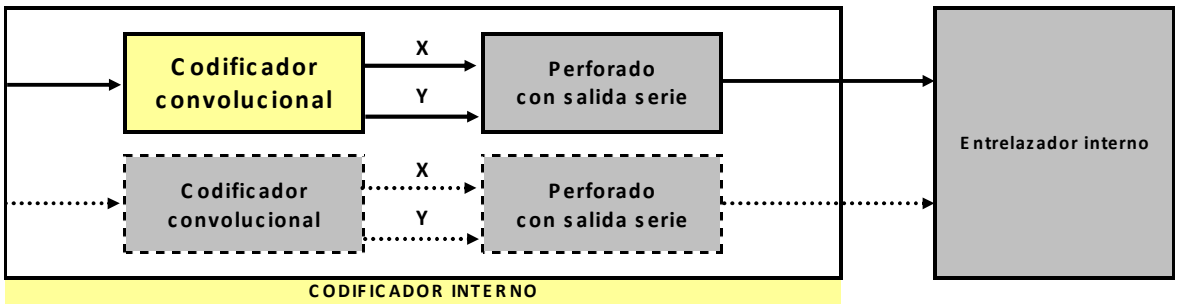


Fig. 4.12 Bloques de procesos internos

El bloque de codificación interna, que con frecuencia es denotado erróneamente como "codificador de Viterbi" debido al algoritmo usado en recepción para la decodificación, está orientado al bit y distribuye en dos salidas (X e Y) el flujo de datos original a base de combinar, con sumas de módulo 2, los datos de entrada con los obtenidos en las tomas situadas detrás de una serie de registros de desplazamiento. La Fig. 4.13 muestra el esquema de principio del "codificador convolucional", de relación 1/2 con 64 estados (K=7 tomas), en el que se basa el mecanismo.

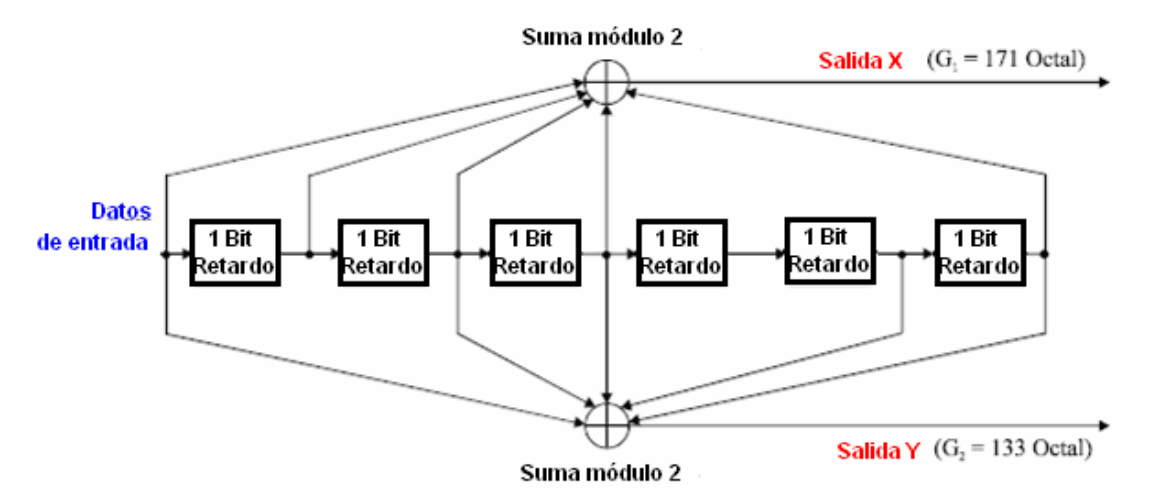


Fig. 4.13 Esquema de codificación convolucional de relación 1/2

Los polinomios generadores para las salidas X e Y se denotan como G_1 y G_2, y sus valores y descripciones son las siguientes:

$$X \rightarrow G_1 = 171_{OCTAL} \equiv 1111001_{BINARIO} \rightarrow G_1(D) = 1 + D + D^2 + D^3 + D^6 \qquad (4.7)$$

$$Y \rightarrow G_2 = 133_{OCTAL} \equiv 1011011_{BINARIO} \rightarrow G_2(D) = 1 + D^2 + D^3 + D^5 + D^6 \qquad (4.8)$$

Esto significa que se suma a los datos de entrada el dato de la toma correspondiente cuando el valor binario del polinomio generador expresado en (4.6), para el caso de X, y el valor expresado en (4.7), para el caso de Y, en la toma en cuestión corresponde a "1" y no se usa el dato de la toma cuando el valor es "0".

La gran redundancia obtenida mediante el codificador de relación 1/2 descrito hace muy potente la corrección de errores cuando éstos son aleatorios, generalmente asociados a una baja relación señal a ruido del canal de transmisión utilizado, pero reduce a la mitad la capacidad del canal (por cada bit de entrada hay dos de salida).

4.2.4.3 Código implementado

```
function [X,Y,estado] = conv_code(simbol_in,S,estado)
%--------------------------------------------------------------------------
%    conv_code.m
%    Ver ETSI 300.744 v1.51 4.3.3 pag 14: Inner Coding
%    Codificación convolucional
%--------------------------------------------------------------------------
fprintf('·Tx: Paso del símbolo %d por etapa de Inner Coding (Codificación convolucional)',S);
fprintf('.');
registros=6;
n=length(simbol_in);
conv_code_X=[1 2 3 6];
conv_code_Y=[2 3 5 6];
X=zeros(n,1);
Y=zeros(n,1);
fprintf('.');
for k = 1:n
  simbol=simbol_in(k);
  X(k) = simbol;
  for v = conv_code_X;
  X(k) = xor(X(k), estado(v));
  end
  Y(k) = simbol;
  for v = conv_code_Y
  Y(k) = xor(Y(k), estado(v));
  end
  estado = [simbol, estado(1:registros-1)];
end
fprintf('.\tOK\n');
```

Código 9 conv_code.m

4.2.5 Perforado

4.2.5.1 Localización del bloque en el sistema

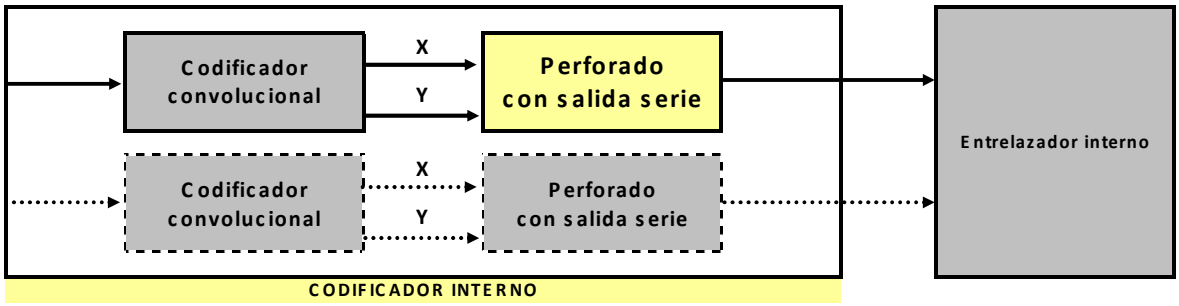


Fig. 4.14 Localización del bloque de perforado

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *puncturing.m.*

4.2.5.2 Objetivo y relación con el estándar

Tal y como se comenta en el apartado 4.2.4, la gran redundancia obtenida mediante el codificador de relación 1/2, denotado como codificador convolucional, hace muy potente la corrección de errores cuando éstos son aleatorios, asociados generalmente a una transmisión a baja SNR, pero reduce a la mitad la capacidad del canal.

Para no limitar tanto la capacidad del canal, el sistema permite seleccionar para la transmisión sólo algunos de los datos obtenidos en las salidas X e Y, los cuales son posteriormente convertidos a secuencia en serie. A éste proceso, recogido en la Fig. 4.15, se le conoce como "perforado" o "picado" (*puncturing*).

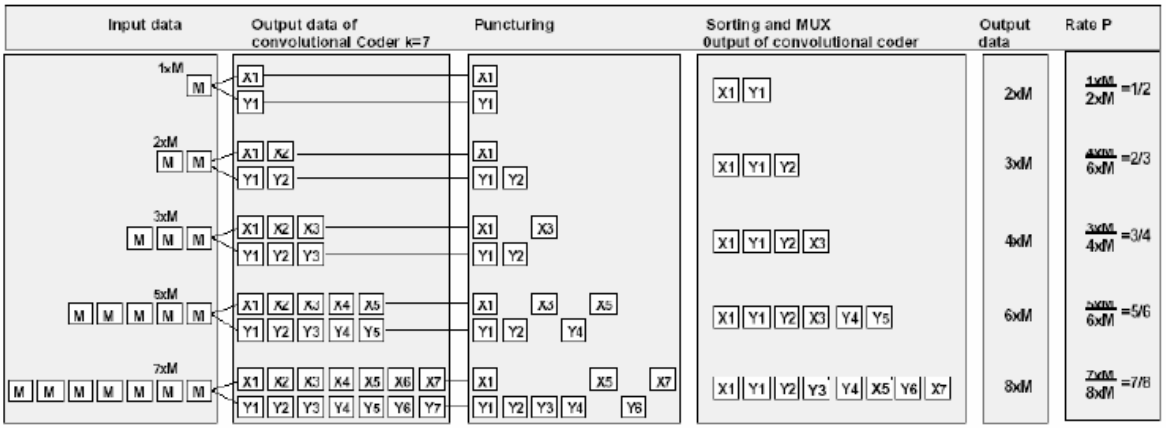


Fig. 4.15 Esquema de perforado y secuencia transmitida en la salida serie

Siguiendo el esquema de la Fig. 4.15, se puede definir un patrón de perforado para obtener una secuencia transmitida después de la conversión paralelo-serie según la tasa de código requerida, tal y como se muestra en la Tabla 4..

Tabla 4.1 Patrón de perforado y secuencia transmitida después de la conversión paralelo-serie según tasa de código requerida

Tasa de código *r*	Patrón de perforado	Secuencia transmitida (Tras la conversión paralelo-serie)
1/2	X:1 Y:1	$X_1\, Y_1$
2/3	X:1 0 Y:1 1	$X_1\, Y_1\, Y_2$
3/4	X:1 0 1 Y:1 1 0	$X_1\, Y_1\, Y_2\, X_3$
5/6	X:1 0 1 0 1 Y:1 1 0 1 0	$X_1\, Y_1\, Y_2\, X_3\, Y_4\, X_5$
7/8	X:1 0 0 0 1 0 1 Y:1 1 1 1 0 1 0	$X_1\, Y_1\, Y_2\, Y_3\, Y_4\, X_5\, Y_6\, X_7$

De esta manera, dependiendo del esquema de perforado escogido, siendo *M* el número de celdas, el funcionamiento es el siguiente:

- 1xM datos de entrada se convierten en 2xM datos de salida en el caso de tasa de código r=1/2.
- 2xM datos de entrada se convierten en 3xM datos de salida en el caso de tasa de código r=2/3
- En general, ixM datos de entrada se convierten en (i+1)xM datos de salida para el caso de tasa de código r=i/(i+1), con $i \in \{1,2,3,5,7\}$.

La protección contra errores es mayor para valores más altos de la relación de codificación pero también lo será la capacidad del canal. Existe un compromiso entre la protección contra errores requerida y la capacidad del canal por lo que la decisión de utilizar una determinada tasa de código en un caso práctico dependerá del área de cobertura deseada para una potencia dada de emisión.

4.2.5.3 Código implementado

```
function simbol_out = puncturing(X,Y,C,S)
%-------------------------------------------------------------------------------
%    puncturing.m
%    Ver ETSI 300.744 v1.51 4.3.3 pag 14: Inner Coding
%    Codificación convolucional
%-------------------------------------------------------------------------------
fprintf('·Tx: Paso del símbolo %g por etapa de Puncturing para adaptación a code rate de %d',S,C);
fprintf('.');
n=length(X);
 switch C
    case 1/2
    data=zeros(2*n,1);
    for i = 1:n
      data(2*(i-1)+1) = X(i);
      data(2*(i-1)+2) = Y(i);
    end
   case 2/3
    data=zeros(3*n/2,1);
    for i = 1:2:n
      data(3*(i-1)/2+1) = X(i);
      data(3*(i-1)/2+2) = Y(i);
      data(3*(i-1)/2+3) = Y(i+1);
    end
   case 3/4
   data=zeros(4*n/3,1);
    for i = 1:3:n
      data(4*(i-1)/3+1) = X(i);
      data(4*(i-1)/3+2) = X(i+2);
      data(4*(i-1)/3+3) = Y(i);
      data(4*(i-1)/3+4) = Y(i+1);
    end
    case 5/6
   data=zeros(6*n/5,1);
    for i = 1:5:n
      data(6*(i-1)/5+1) = X(i);
      data(6*(i-1)/5+2) = X(i+2);
      data(6*(i-1)/5+3) = X(i+4);
      data(6*(i-1)/5+4) = Y(i);
      data(6*(i-1)/5+5) = Y(i+1);
      data(6*(i-1)/5+6) = Y(i+3);
    end
    case 7/8
   data=zeros(8*n/7,1);
    for i = 1:7:n
      data(8*(i-1)/7+1) = X(i);
      data(8*(i-1)/7+2) = X(i+4);
      data(8*(i-1)/7+3) = X(i+6);
      data(8*(i-1)/7+4) = Y(i);
      data(8*(i-1)/7+5) = Y(i+1);
      data(8*(i-1)/7+6) = Y(i+2);
      data(8*(i-1)/7+7) = Y(i+3);
      data(8*(i-1)/7+8) = Y(i+5);
    end
fprintf('.');
end
simbol_out=data;
fprintf('.\tOK\n');
```

Código 10 puncturing.m

4.2.6 Entrelazado interno

4.2.6.1 Localización del bloque en el sistema

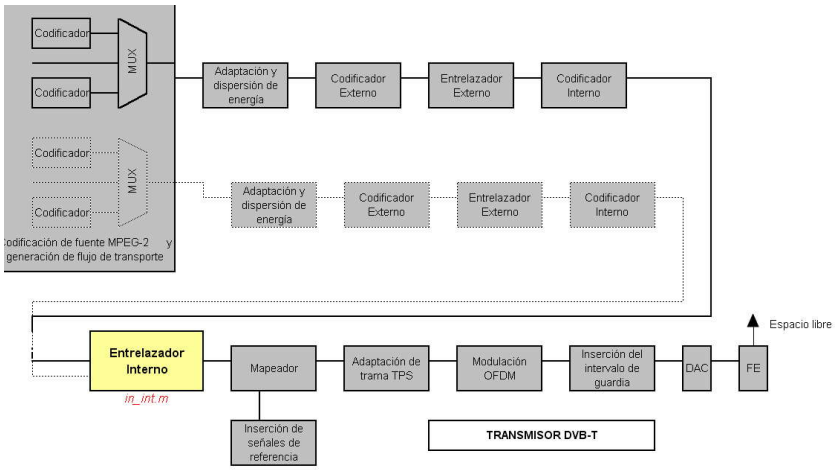


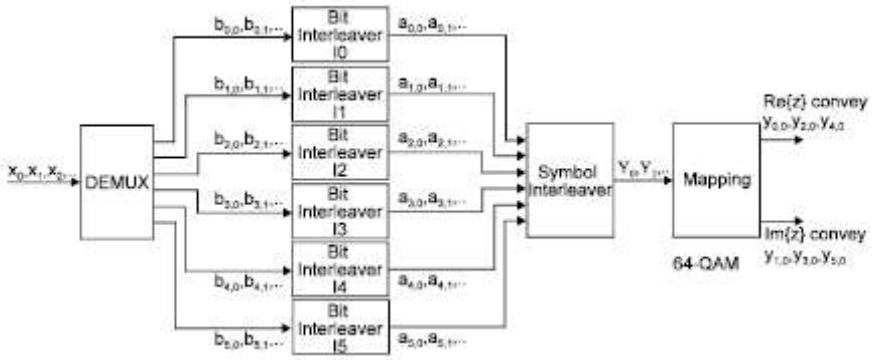
Fig. 4.16 Localización del bloque de entrelazado interno

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *in_int.m*.

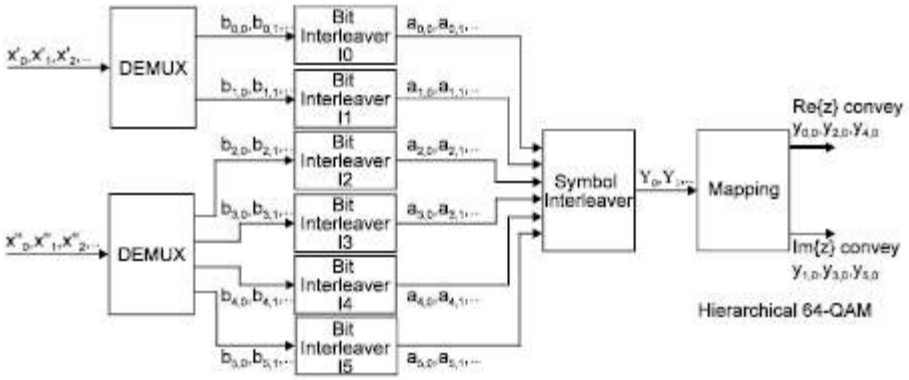
4.2.6.2 Objetivo y relación con el estándar

El "entrelazado interno" se compone de un primer proceso de entrelazado relativo al bit que va seguido de un entrelazado de símbolos. Ambos, el relativo al bit y el relativo al símbolo, están basados en bloques.

Desde el principio, el proceso de entrelazado se orienta al tamaño de los símbolos, que es función de la constelación. Así, los símbolos tienen 2 bits en QPSK, tienen 4 bits en 16-QAM y 6 bits en 64-QAM (Véase Fig. 4.21). Dos esquemas de principio del bloque de entrelazado interno y de modulación para el caso particular de uso de la constelación 64-QAM se muestra en la Fig. 4.17.



(a) *64-QAM, transmisión no jerárquica*



(b) *64-QAM, transmisión jerárquica*

Fig. 4.17 Esquemas de entrelazado interno y mapeado

En el caso de modos de transmisión no jerárquicos el único flujo de transporte de entrada, compuesto por bits *xi*, se demultiplexa en *"v"* sub_flujos, siendo *v* el número de bits por símbolo, resultando en cada vía los bits *bij* (Ver Tabla 4.2). En cambio, en el caso de modos de transmisión jerárquicos el flujo de transporte de entrada de alta prioridad (HP), compuesto por los bits *x'i*, se demultiplexa en 2 sub-flujos, a la vez que el de baja prioridad, bits *x''i*, lo hace en (*v*-2) sub-flujos, siendo *v* el número de bits por símbolo, resultando en cada vía los bits *bij* (Ver Tabla 4.2). En los esquemas presentados en Fig. 4.17 la constelación utilizada es 64-QAM, por lo que el valor de *v* es de 6 bits.

Después del demultiplexado, cada sub-flujo se procesa por separado en un bloque de entrelazado de bits. Hay, por tanto, *"v"* bloques de entrelazado de bits que se numeran desde I0 hasta I5 (ya que en el sistema de transmisión del estándar DBV-T *v* es como mucho 6). QPSK utiliza los dos primeros, 16-QAM utiliza los 4 primeros y 64-QAM los utiliza todos.

Tabla 4.2 Esquemas de demultiplexado de los bits x_i (x'_i y x''_i, jerárquico) en bits b_{ij}

	Transmisión no jerárquica			Transmisión jerárquica		
	xi		***bij***	***xi***		***bij***
QPSK	x_0	→	b_{00}			
	x_1	→	b_{10}			
16-QAM	x_0	→	b_{00}	x'_0	→	b_{00}
	x_1	→	b_{20}	x'_1	→	b_{10}
	x_2	→	b_{10}	x''_2	→	b_{20}
	x_3	→	b_{30}	x''_3	→	b_{30}
64-QAM	x_0	→	b_{00}	x'_0	→	b_{00}
	x_1	→	b_{20}	x'_1	→	b_{10}
	x_2	→	b_{40}	x''_2	→	b_{20}
	x_3	→	b_{10}	x''_3	→	b_{40}
	x_4	→	b_{30}	x''_4	→	b_{30}
	x_5	→	b_{50}	x''_5	→	b_{50}

El "entrelazador de bits" se realiza por bloques y sólo actúa sobre datos útiles. El tamaño de estos bloques es de 126 bits para todos los bloques de entrelazado, aunque la secuencia de entrelazado es diferente de unos a otros. Éste tamaño de 126 bits implica que este proceso se repita un número exacto de veces por cada símbolo OFDM tanto si se usa el estándar 2k (1512 portadoras de datos) o el estándar 8k (6048 portadoras de datos):

Tabla 4.3 Número de repeticiones de entrelazado por símbolo OFDM

	Estándar 2k	Estándar 8k
Portadoras activas de datos	1512	6048
Bits totales por símbolo OFDM	vx1512	vx6048
Entrelazadores de bit	v	v
Repeticiones por símbolo OFDM	(vx1512)x(vx126)=**12**	(vx6048)x(vx126)=**48**

Así, los bloques de bits de entrada $B(e)$ y de salida $A(e)$ a cada entrelazador están compuestos por:

$$B(e) = (b_{e,0} \;\; b_{e,1} \;\; b_{e,2} \;\; b_{e,3} \;\ldots\; b_{e,125}) \quad \text{donde e} = 0...\text{v} - 1 \tag{4.9}$$

$$A(e) = (a_{e,0} \;\; a_{e,1} \;\; a_{e,2} \;\; a_{e,3} \;\ldots\; a_{e,125}) \quad \text{donde e} = 0...\text{v} - 1 \tag{4.10}$$

siendo la relación entre los bits de los bloques de (4.9) y (4.10) la siguiente:

$$a_{e,w} = b_{e,He(w)} \quad \text{donde w} = 0...125 \tag{4.11}$$

En (4.11) aparece la función de permutación $H_e(w)$, que es diferente de unos entrelazadores a otros, siendo sus diferentes expresiones definidas en el estándar:

Tabla 4.4 Funciones de permutación, $H_i(w)$, para los distintos entrelazadores, I_i

Entrelazador	Función de permutación
I0	$H_0(w)=w$
I1	$H_1(w)=(w+63) \bmod 126$
I2	$H_2(w)=(w+105) \bmod 126$
I3	$H_3(w)=(w+42) \bmod 126$
I4	$H_4(w)=(w+21) \bmod 126$
I5	$H_5(w)=(w+84) \bmod 126$

Las salidas de los *v* entrelazadores se agrupan para formar palabras (y'_w) de *v* bits de manera que se toma cada vez un único bit de la salida de cada entrelazador, correspondiendo el bit más significativo a la salida del entrelazador I0, es decir:

$$y'_w = (a_{0,w} \;\; a_{1,w} \;\; a_{2,w} \ldots a_{v-1,w}) \tag{4.12}$$

El "entrelazado de símbolos" se realiza sobre las anteriores palabras y'_w conteniendo v bits cada una de ellas, de forma que a la salida del entrelazador queden agrupadas en bloques cuyo tamaño está calculado para que los datos se puedan distribuir directamente entre las portadoras activas de datos del símbolo OFDM. Para ello, las palabras y'_w se agrupan para formar unos vectores Y' de la siguiente forma, dependiendo del estándar (2k o 8k) de transmisión:

- Estándar 2k: 12 conjuntos de 126 palabras

$$Y' = (y'_0 \;\; y'_1 \;\; y'_{2,} \ldots \; y'_{1511}) \tag{4.13}$$

- Estándar 8k: 48 conjuntos de 126 palabras

$$Y' = (y'_0 \;\; y'_1 \;\; y'_{2,} \ldots \; y'_{6047}) \tag{4.14}$$

Los vectores Y', expresados en (4.13) y (4.14), a la entrada del entrelazador de símbolos se convierten en los vectores entrelazados, Y, a su salida:

$$Y = (y_0 \;\; y_1 \;\; y_{2,} \ldots \; y_{N\max -1}) \tag{4.15}$$

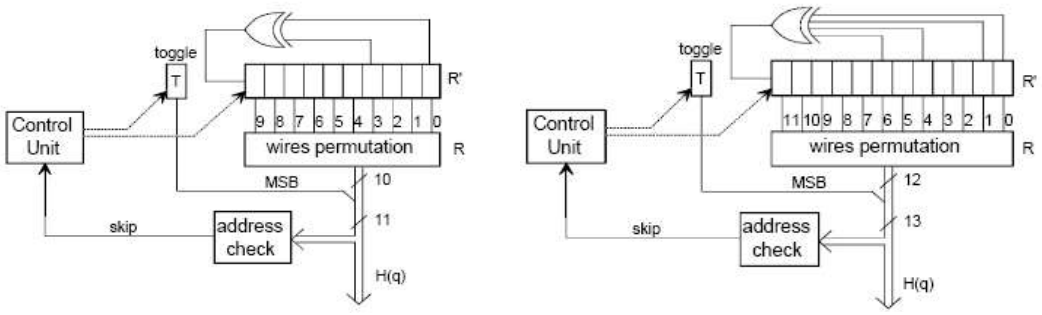
siendo N_{max}= 1512 en modo 2k y N_{max}=6048 en modo 2k.

El contenido del vector Y, expresado en (4.15), se define según el símbolo OFDM que se va a generar, de la siguiente manera:

$$y_q = y'_{H(q)} \text{ donde q = 0..N}_{\max\text{-}1} \text{ para símbolos OFDM impares} \tag{4.16}$$

$$y_{H(q)} = y'_q \text{ donde q = 0..N}_{\max\text{-}1} \text{ para símbolos OFDM pares} \tag{4.17}$$

donde $H(q)$ es una función de permutación que se puede generar mediante el algoritmo cuyo diagrama de bloques esquemático se muestra en la Fig. 4.18.



(a) Modo 2k (b)Modo 8k

Fig. 4.18 Esquemas del algoritmo generador de la función de permutación H(q)

En los diagramas de bloques presentados en la Fig. 4.18, *R'i* son palabras binarias compuestas por (N_r-1) bits siendo

$$N_r = \log_2(M_{\max}) \tag{4.18}$$

con M_{max}=2048 portadoras en el modo 2k y M_{max}=8192 portadoras en el modo 8k. De (4.18) se desprende que *R'i* es una palabra de 10 bits en modo 2k y una palabra de 12 bits en modo 8k, cuyos valores de bits vienen definidos por el estándar DBV-T de la siguiente manera:

Tabla 4.5 Valores de R'$_i$ de la función de permutación, H(q)

i=0,1	$R'_i[N_r\text{-}2, N_r\text{-}3....1,0]=0,0,....0,0$
i=2	$R'_i[N_r\text{-}2, N_r\text{-}3....1,0]=0,0,....0,1$
2<i<M$_{max}$	$R'_i[N_r\text{-}2, N_r\text{-}3....1,0]= X, R'_{i-1}[N_r\text{-}2, N_r\text{-}3....1]$ siendo $X= R'_i[N_r\text{-}2]= R'_{i-1}[0] \oplus R'_{i-1}[3]$ para modo 2k siendo $X= R'_i[N_r\text{-}2]= R'_{i-1}[0] \oplus R'_{i-1}[1] \oplus R'_{i-1}[4] \oplus R'_{i-1}[6]$ para modo 8k

Los valores presentados en la Tabla 4.5 se pueden expresar de la siguiente manera, según el modo en el que se realiza la transmisión:

Tabla 4.6 Valores de R'$_i$ para modo 2k

i=0,1	$R'_0[9,8,7,6,5,4,3,2,1,0]=0,0,0,0,0,0,0,0,0,0$ $R'_1[9,8,7,6,5,4,3,2,1,0]=0,0,0,0,0,0,0,0,0,0$
i=2	$R'_2[9,8,7,6,5,4,3,2,1,0]=0,0,0,0,0,0,0,0,0,1$
2<i<M$_{max}$	$R'_i[8,7,6,5,4,3,2,1,0]=R'_{i-1}[9,8,7,6,5,4,3,2,1]$ siendo $R'_i[9]= R'_{i-1}[0] \oplus R'_{i-1}[3]$ para modo 2k

Tabla 4.7 Valores de R'$_i$ para modo 8k

i=0,1	$R'_0[11,10,9,8,7,6,5,4,3,2,1,0]=0,0,0,0,0,0,0,0,0,0,0,0$ $R'_1[11,10,9,8,7,6,5,4,3,2,1,0]=0,0,0,0,0,0,0,0,0,0,0,0$
i=2	$R'_2[11,10,9,8,7,6,5,4,3,2,1,0]=0,0,0,0,0,0,0,0,0,0,0,1$
2<i<M$_{max}$	$R'_i[10,9,8,7,6,5,4,3,2,1,0]=R'_{i-1}[11,10,9,8,7,6,5,4,3,2,1]$ siendo $R'_i[11]= R'_{i-1}[0] \oplus R'_{i-1}[1] \oplus R'_{i-1}[4] \oplus R'_{i-1}[6]$ para modo 8k

En la Tabla 4.5, Tabla 4.6 y Tabla 4.7 se expresa la función binaria OR exclusiva, *xor,* con el símbolo $\oplus$.

En los diagramas de bloques presentados en la Fig. 4.18, *Ri* son vectores derivados de las palabras *R'i* mediante las permutaciones de bits que se recogen en la siguiente tabla:

Tabla 4.8 Valores de R$_i$ derivados de R'$_i$

Modo 2k	Posiciones de bits de R'_i			9	8	7	6	5	4	3	2	1	0
	Posiciones de bits de R_i			0	7	5	1	8	2	6	9	3	4
Modo 8k	Posiciones de bits de R'_i	11	10	9	8	7	6	5	4	3	2	1	0
	Posiciones de bits de R_i	5	11	3	0	10	8	6	9	2	4	1	7

4.2.6.3 Código implementado

```
function simbol_out = in_int(simbol_in,N,n,S,fft_mode,S_index)
%-------------------------------------------------------------------------------------
%    in_int.m
%    Ver ETSI 300.744 v1.51 4.3.4 pag 14: Inner Interleaving
%    VER ETSI 300.744 v1.51 pag 14 Tabla 2
%-------------------------------------------------------------------------------------
fprintf('·Tx: Paso del símbolo %g por etapa de Entrelazado interno (Inner Interleaving)',S);
fprintf('.');
m=N*log2(n);
interliv_bits=126;
tam_bloque=interliv_bits*log2(n);
long_simb=length (simbol_in);
num_blocs=long_simb/tam_bloque;
%-------------------------------------------------------------
%Primero calculamos Heq (Ver ETSI 300.744 v1.51 pag 20 Tabla 3)
%-------------------------------------------------------------
Nmax = N;
Mmax = fft_mode;
Nr = log2(Mmax);
H = zeros(Nmax,1);
 qq = 0;
 switch fft_mode
   case 2048
     perm_table=[4 3 9 6 2 8 1 5 7 0];
   case 8192
     perm_table=[7 1 4 2 9 6 8 10 0 3 11 5];
 end
 for ii = 0:Mmax-1
   % calcula Ri
   switch ii
     case 0
       r1 = zeros (1, Nr-1);
     case 1
       r1 = zeros (1, Nr-1);
     case 2
       r1 = zeros (1, Nr-1);
       r1(1) = 1;
       otherwise
           if fft_mode ==2048
             r1 = [r1(2:Nr-1) , xor(r1(1+0), r1(1+3))];
           else
             xor1= xor(r1(1+0), r1(1+1));
             xor2= xor(r1(1+4), r1(1+6));
             r1 = [r1(2:Nr-1) , xor(xor1,xor2)];
           end
      end
   r = zeros(1, Nr-1);
   for k = 0:Nr-2
     r(1+perm_table(1+k)) = r1(1+k);
   end
   % calcula H(q)
   H(1+qq) = rem(ii,2) * 2^(Nr-1);
   for jj = 0:Nr-2
     H(1+qq) = H(1+qq) + r(1+jj) * 2^jj;
   end
   if H(1+qq) < Nmax
     qq = qq + 1;
   end
 end
```

```
%-----------------------------------------------
%            Demultiplexador
%-----------------------------------------------
y = zeros (m/log2(n), log2(n));
  for i = 1:num_blocs
  x=simbol_in(1+(i-1)*tam_bloque:i*tam_bloque);
a=reshape (x, log2(n), interliv_bits)';
  b=zeros (interliv_bits, log2(n));
  switch n
    case 4
      mapeo=[0 1];
    case 16
      mapeo=[0 2 1 3];
    case 64
      mapeo=[0 2 4 1 3 5];
  end
  b(:,1+mapeo) = a;
%-----------------------------------------------
%            Bit Interleaver
%-----------------------------------------------
a=zeros (interliv_bits, log2(n));
h_param=[0 63 105 42 21 84];
for e = 0:log2(n)-1
   for w = 0:interliv_bits-1
      % compute H(e,w)
      h=rem(w+h_param(1+e), interliv_bits);
      a(1+w,1+e)=b(1+h,1+e);
   end
end
y(1+(i-1)*interliv_bits:i*interliv_bits,:) = a;

  end
fprintf('.');
%-----------------------------------------------
%            Simbol Interleaver
%-----------------------------------------------
simbol_out = zeros (m/log2(n), log2(n));
  for i = 1:m/log2(n)
    if rem((S_index-1), 2) == 0
      simbol_out(1+H(i),:) = y(i,:);
    else
      simbol_out(i,:) = y(1+H(i),:);
    end
  end
fprintf('.\tOK\n');
```

Código 11 in_int.m

4.2.7 Mapeado de símbolos

4.2.7.1 Localización del bloque en el sistema

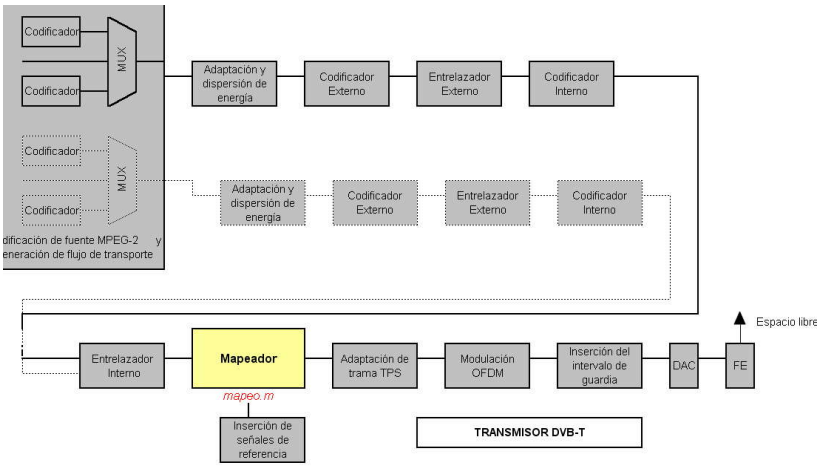


Fig. 4.19 Localización del bloque de mapeo

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *mapeo.m*.

4.2.7.2 Objetivo y relación con el estándar

Cada palabra $y_{q'}$ perteneciente al vector Y de salida del entrelazador de símbolos (Véase Fig. 4.17), tiene v bits:

$$y_{q'} = (y_{0,q'} \;\; y_{1,q'} \;\; y_{2,q'} \;\; \ldots \;\; y_{v-1,q'}) \quad \begin{matrix} \text{con } q'=0,1,...,1511 \text{ en modo 2k} \\ \text{con } \;\; q'=0,1,...,6047 \text{ en modo 2k} \end{matrix} \qquad (4.19)$$

Estos v bits de la palabra número q' se modulan con un esquema de modulación correspondiente al valor de v, la portadora número q' de las 1512 (modo 2k) o 6048 (modo 8k) portadoras activas para datos que tiene cada símbolo OFDM.

Para dar una forma física a estas agrupaciones de bits de manera que se puedan representar energéticamente, es necesario distribuir ("mapear") en dos señales I y Q el flujo de serie de datos presente a la salida del entrelazador de símbolos. La Fig. 4.20 ilustra la situación en que se convierten bytes (8 bits), pasando por símbolos de v=6 bits, en las 2 señales I y Q necesarias para obtener una constelación del tipo 64-QAM. Nótese que la duración inicial de cada bit se duplica en las salidas I y Q.

El resultado de la modulación de cada portadora según v puede verse en la Fig. 4.21, para el cual la distribución de los bits entre las señales I y Q se hace con este orden:

Tabla 4.9 Distribución de bits de las señales I y Q para cada modulación

Constelación	Bits del flujo de datos serie	Bits en la salida I	Bits en la salida Q
QPSK	$y_{0,q'}\; y_{1,q'}$	$y_{0,q'}$	$y_{1,q'}$
16-QAM	$y_{0,q'}y_{1,q'}y_{2,q'}y_{3,q'}$	$y_{0,q'}y_{2,q'}$	$y_{1,q'}y_{4,q'}$
64-QAM	$y_{0,q'}y_{1,q'}y_{2,q'}y_{3,q'}\,y_{4,q'}\,y_{5,q'}$	$y_{0,q'}\,y_{2,q'}\,y_{4,q'}$	$y_{1,q'}\,y_{3,q'}\,y_{5,q'}$

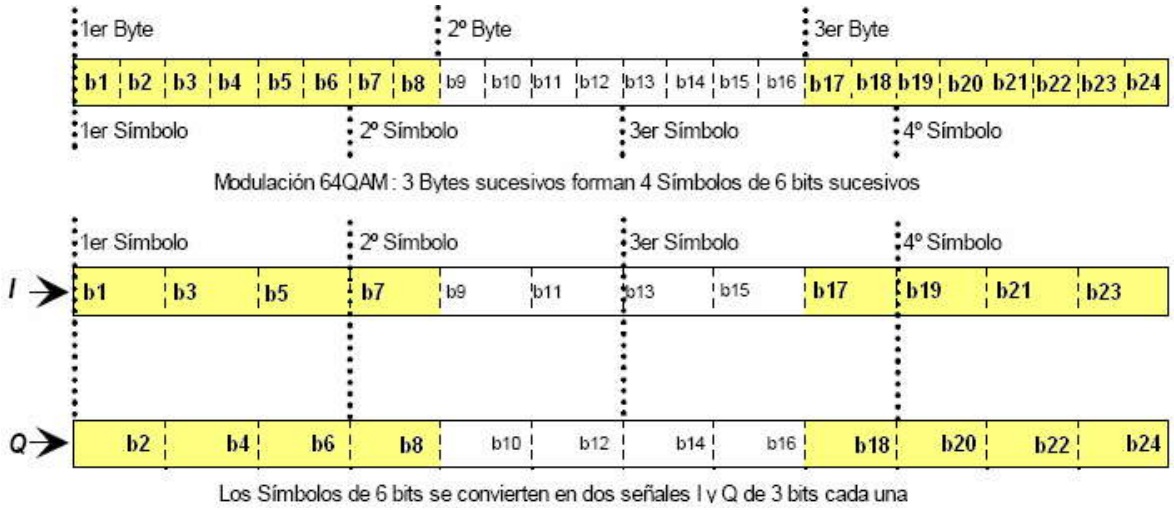


Fig. 4.20 Mapeado de bits a símbolos para la modulación 64-QAM

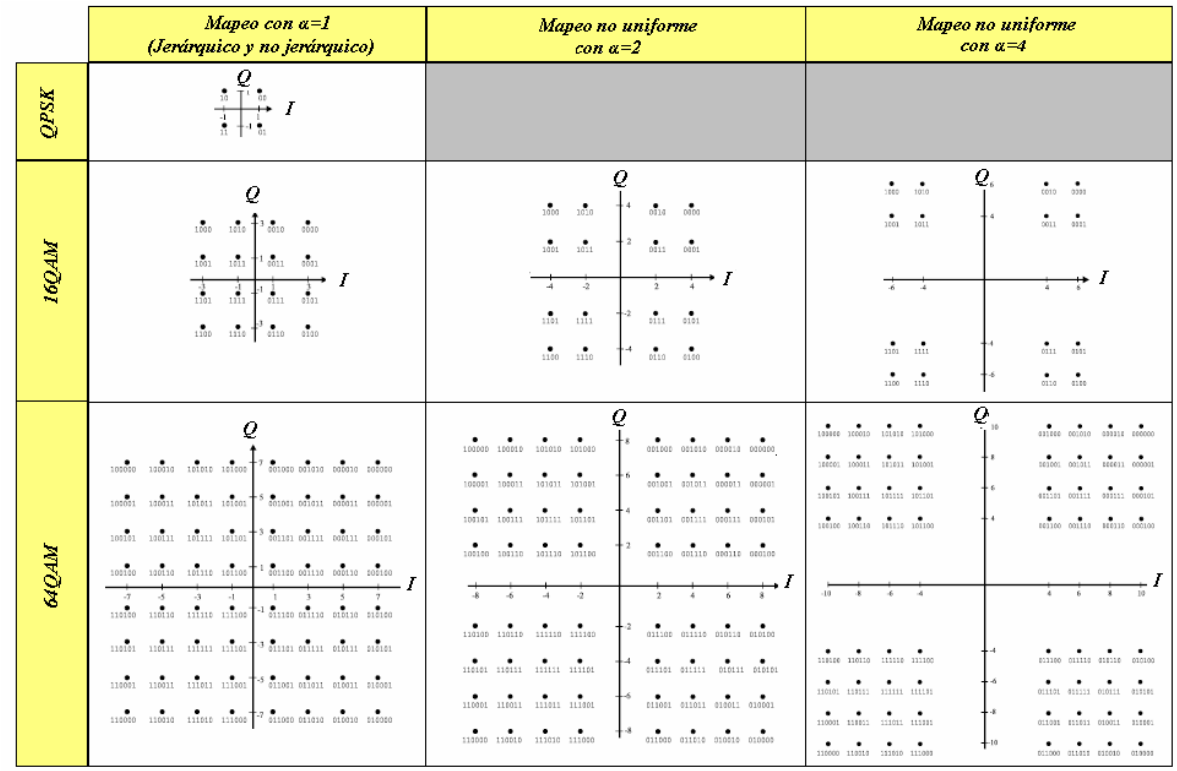


Fig. 4.21Constelaciones resultado de las modulaciones de las portadoras con mapeado Gray

El mapeado de los símbolos PAM se realiza en base a la codificación Gray. El código Gray es un tipo especial de código binario que no es ponderado, ya que los dígitos que componen el código no tienen un peso asignado. Su característica es que es un código progresivo, es decir, entre una combinación de dígitos y la siguiente, sea ésta anterior o posterior, sólo hay una diferencia de un dígito. Esta progresión sucede también entre la última y la primera combinación, por lo que se le llama también código cíclico. La característica de pasar de un código al siguiente cambiando sólo un dígito asegura menos posibilidades de error.

Los modos jerárquicos se pueden "mapear" según todas las constelaciones presentadas en la Fig. 4.21, pero los modos no jerárquicos lo hacen únicamente de la misma manera que en el caso uniforme (α=1). En el caso de modo jerárquico los bits de alta prioridad son los $y_{0,q'}y_{1,q'}$ siendo los restantes los de baja prioridad (Véase (4.19)). Con la distribución de bits mostrada en la Tabla 4.9, en caso de que la constelación se decodifique como QPSK (un sólo punto de la constelación identificado en cada cuadrante) solamente se recuperarán los bits de alta prioridad $y_{0,q'}y_{1,q'}$ mientras que la recuperación de todos los bits exigirá la correcta identificación de la constelación completa.

La distancia entre los puntos de la constelación está determinada por el parámetro de modulación α, que se define como la relación de la distancie entre dos puntos de la constelación pertenecientes a cuadrantes adyacentes y la distancia entre puntos del mismo cuadrante. El estándar DVB-T especifica 3 valores para este parámetro, distinguiendo entre modulaciones uniformes, los que tienen valor de α=1, y modulaciones no uniformes, con α=2 y α=4).

Los valores exactos de los puntos de la constelación son $z \in \{n+j\cdot m\}$ con valores de n, m para las distintas constelaciones tal y como se expresa a continuación:

Tabla 4.10 Valores de n, m siendo los puntos de la constelación del tipo z ∈ {n+j·m}

	n	***m***
QPSK	{-1,1}	{-1,1}
16-QAM jerárquica y no jerárquica con α=1	{-3,-1,1,3}	{-3,-1,1,3}
16-QAM no uniforme con α=2	{-4, -2, 2, 4}	{-4, -2, 2, 4}
16-QAM no uniforme con α=4	{-6, -4, 4, 6}	{-6, -4, 4, 6}
64-QAM jerárquica y no jerárquica con α=1	{-7, -5, -3, -1, 1, 3, 5, 7}	{-7, -5, -3, -1, 1, 3, 5, 7}
64-QAM no uniforme con α=2	{-8, -6 ,-4, -2, 2, 4, 6, 8}	{-8, -6 ,-4, -2, 2, 4, 6, 8}
64-QAM no uniforme con α=4	{-10,-8, -6, -4, 4, 6, 8, 10}	{-10,-8,-6, -4, 4, 6, 8, 10}

Las portadoras que contienen datos se transmiten con nivel de potencia normalizado de manera que cumplen:

$$E[c\, x\, c^*] = 1 \qquad (4.20)$$

correspondiendo "c" a los distintos puntos "$c_{f,l,k}$" de la constelación de la portadora k (0 a 6816 en modo 8k) del símbolo OFDM número l (0 a 67) de la trama OFDM número f. Por ello, los factores de normalización que cumplen la condición de normalización (4.20) vienen dados en la siguiente tabla:

Tabla 4.11 Factores de normalización para los símbolos OFDM de datos

QPSK	α=1	$c = z/\sqrt{2}$
16-QAM	α=1	$c = z/\sqrt{10}$
	α=2	$c = z/\sqrt{20}$
	α=4	$c = z/\sqrt{52}$
64-QAM	α=1	$c = z/\sqrt{42}$
	α=2	$c = z/\sqrt{60}$
	α=4	$c = z/\sqrt{108}$

Los factores de normalización expresados en la Tabla 4.11 se obtienen a partir del cálculo de la energía media de cada uno de los puntos de la constelación (raíz cuadrada del módulo de *z*, al expresarse el módulo de *z* en unidades de potencia) de un único cuadrante. La siguiente figura recoge las constelaciones unitarias de las distintas constelaciones según el parámetro de modulación *α*.

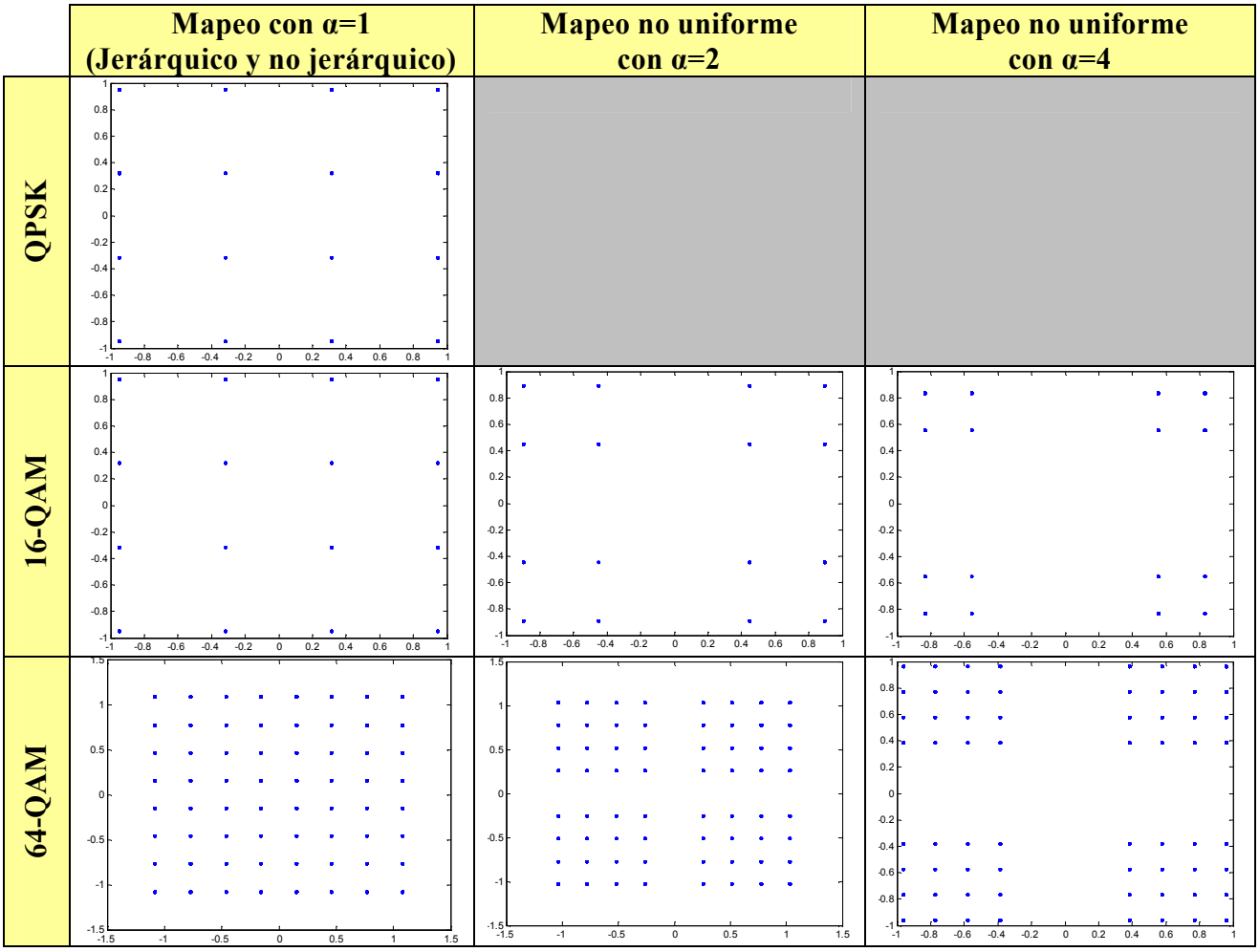


Fig. 4.22Constelaciones unitarias resultado de las modulaciones de las portadoras con mapeado Gray

4.2.7.3 Código implementado

```
function map_out = mapeo(grupos,n,alpha,S,N,const)
%-------------------------------------------------------------------------------------
%    mapeo.m
%     Ver ETSI 300.744 v1.51 4.3.5 pag 21: Signal constellations and mapping
%-------------------------------------------------------------------------------------
if (n==4)
fprintf('·Tx: Mapeo del símbolo OFDM %g con constelación unitaria QPSK con alpha=%g',S,alpha);
else
fprintf('·Tx: Mapeo del símbolo OFDM %g con constelación unitaria %gQAM con alpha=%g',S,n,alpha);
end
fprintf('.');
%-------------------------------------------------------------
%      Mapeado de bits  n-QAM con Gray
%-------------------------------------------------------------
s=[];
%QPSK
if (n==4)
for i=1:N
    sPAM=grupos(i,:);
    if (sPAM==[0 0]) s(i)=1+j; end
    if (sPAM==[0 1]) s(i)=1-j; end
    if (sPAM==[1 0]) s(i)=-1+j; end
    if (sPAM==[1 1]) s(i)=-1-j; end
end
fprintf('.');
end
%16QAM
if(n==16)
for i=1:N
    sPAM=grupos(i,:);
    if (sPAM==[0 0 0 0]) s(i)=(alpha+2)+j*(alpha+2); end
    if (sPAM==[0 0 0 1]) s(i)=(alpha+2)+j*(alpha); end
    if (sPAM==[0 0 1 0]) s(i)=(alpha)+j*(alpha+2); end
    if (sPAM==[0 0 1 1]) s(i)=(alpha)+j*(alpha); end
    if (sPAM==[0 1 0 0]) s(i)=(alpha+2)-j*(alpha+2); end
    if (sPAM==[0 1 0 1]) s(i)=(alpha+2)-j*(alpha); end
    if (sPAM==[0 1 1 0]) s(i)=(alpha)-j*(alpha+2); end
    if (sPAM==[0 1 1 1]) s(i)=(alpha)-j*(alpha); end
    if (sPAM==[1 0 0 0]) s(i)=-(alpha+2)+j*(alpha+2); end
    if (sPAM==[1 0 0 1]) s(i)=-(alpha+2)+j*(alpha); end
    if (sPAM==[1 0 1 0]) s(i)=-(alpha)+j*(alpha+2); end
    if (sPAM==[1 0 1 1]) s(i)=-(alpha)+j*(alpha); end
    if (sPAM==[1 1 0 0]) s(i)=-(alpha+2)-j*(alpha+2); end
    if (sPAM==[1 1 0 1]) s(i)=-(alpha+2)-j*(alpha); end
    if (sPAM==[1 1 1 0]) s(i)=-(alpha)-j*(alpha+2); end
    if (sPAM==[1 1 1 1]) s(i)=-(alpha)-j*(alpha); end
end
fprintf('.');
end
%64QAM
if(n==64)
for i=1:N
    sPAM=grupos(i,:);
    if (sPAM==[0 0 0 0 0 0]) s(i)=(alpha+6)+j*(alpha+6); end
    if (sPAM==[0 0 0 0 0 1]) s(i)=(alpha+6)+j*(alpha+4); end
    if (sPAM==[0 0 0 0 1 0]) s(i)=(alpha+4)+j*(alpha+6); end
    if (sPAM==[0 0 0 0 1 1]) s(i)=(alpha+4)+j*(alpha+4); end
    if (sPAM==[0 0 0 1 0 0]) s(i)=(alpha+6)+j*(alpha); end
    if (sPAM==[0 0 0 1 0 1]) s(i)=(alpha+6)+j*(alpha+2); end
    if (sPAM==[0 0 0 1 1 0]) s(i)=(alpha+4)+j*(alpha); end
```

```
61      if (sPAM==[0 0 0 1 1 1]) s(i)=(alpha+4)+j*(alpha+2); end
62      if (sPAM==[0 0 1 0 0 0]) s(i)=(alpha)+j*(alpha+6); end
63      if (sPAM==[0 0 1 0 0 1]) s(i)=(alpha)+j*(alpha+4); end
64      if (sPAM==[0 0 1 0 1 0]) s(i)=(alpha+2)+j*(alpha+6); end
65      if (sPAM==[0 0 1 0 1 1]) s(i)=(alpha+2)+j*(alpha+4); end
66      if (sPAM==[0 0 1 1 0 0]) s(i)=(alpha)+j*(alpha); end
67      if (sPAM==[0 0 1 1 0 1]) s(i)=(alpha)+j*(alpha+2); end
68      if (sPAM==[0 0 1 1 1 0]) s(i)=(alpha+2)+j*(alpha); end
69      if (sPAM==[0 0 1 1 1 1]) s(i)=(alpha+2)+j*(alpha+2); end
70      if (sPAM==[0 1 0 0 0 0]) s(i)=(alpha+6)-j*(alpha+6); end
71      if (sPAM==[0 1 0 0 0 1]) s(i)=(alpha+6)-j*(alpha+4); end
72      if (sPAM==[0 1 0 0 1 0]) s(i)=(alpha+4)-j*(alpha+6); end
73      if (sPAM==[0 1 0 0 1 1]) s(i)=(alpha+4)-j*(alpha+4); end
74      if (sPAM==[0 1 0 1 0 0]) s(i)=(alpha+6)-j*(alpha); end
75      if (sPAM==[0 1 0 1 0 1]) s(i)=(alpha+6)-j*(alpha+2); end
76      if (sPAM==[0 1 0 1 1 0]) s(i)=(alpha+4)-j*(alpha); end
77      if (sPAM==[0 1 0 1 1 1]) s(i)=(alpha+4)-j*(alpha+2); end
78      if (sPAM==[0 1 1 0 0 0]) s(i)=(alpha)-j*(alpha+6); end
79      if (sPAM==[0 1 1 0 0 1]) s(i)=(alpha)-j*(alpha+4); end
80      if (sPAM==[0 1 1 0 1 0]) s(i)=(alpha+2)-j*(alpha+6); end
81      if (sPAM==[0 1 1 0 1 1]) s(i)=(alpha+2)-j*(alpha+4); end
82      if (sPAM==[0 1 1 1 0 0]) s(i)=(alpha)-j*(alpha); end
83      if (sPAM==[0 1 1 1 0 1]) s(i)=(alpha)-j*(alpha+2); end
84      if (sPAM==[0 1 1 1 1 0]) s(i)=(alpha+2)-j*(alpha); end
85      if (sPAM==[0 1 1 1 1 1]) s(i)=(alpha+2)-j*(alpha+2); end
86      if (sPAM==[1 0 0 0 0 0]) s(i)=-(alpha+6)+j*(alpha+6); end
87      if (sPAM==[1 0 0 0 0 1]) s(i)=-(alpha+6)+j*(alpha+4); end
88      if (sPAM==[1 0 0 0 1 0]) s(i)=-(alpha+4)+j*(alpha+6); end
89      if (sPAM==[1 0 0 0 1 1]) s(i)=-(alpha+4)+j*(alpha+4); end
90      if (sPAM==[1 0 0 1 0 0]) s(i)=-(alpha+6)+j*(alpha); end
91      if (sPAM==[1 0 0 1 0 1]) s(i)=-(alpha+6)+j*(alpha+2); end
92      if (sPAM==[1 0 0 1 1 0]) s(i)=-(alpha+4)+j*(alpha); end
93      if (sPAM==[1 0 0 1 1 1]) s(i)=-(alpha+4)+j*(alpha+2); end
94      if (sPAM==[1 0 1 0 0 0]) s(i)=-(alpha)+j*(alpha+6); end
95      if (sPAM==[1 0 1 0 0 1]) s(i)=-(alpha)+j*(alpha+4); end
96      if (sPAM==[1 0 1 0 1 0]) s(i)=-(alpha+2)+j*(alpha+6); end
97      if (sPAM==[1 0 1 0 1 1]) s(i)=-(alpha+2)+j*(alpha+4); end
98      if (sPAM==[1 0 1 1 0 0]) s(i)=-(alpha)+j*(alpha); end
99      if (sPAM==[1 0 1 1 0 1]) s(i)=-(alpha)+j*(alpha+2); end
100     if (sPAM==[1 0 1 1 1 0]) s(i)=-(alpha+2)+j*(alpha); end
101     if (sPAM==[1 0 1 1 1 1]) s(i)=-(alpha+2)+j*(alpha+2); end
102     if (sPAM==[1 1 0 0 0 0]) s(i)=-(alpha+6)-j*(alpha+6); end
103     if (sPAM==[1 1 0 0 0 1]) s(i)=-(alpha+6)-j*(alpha+4); end
104     if (sPAM==[1 1 0 0 1 0]) s(i)=-(alpha+4)-j*(alpha+6); end
105     if (sPAM==[1 1 0 0 1 1]) s(i)=-(alpha+4)-j*(alpha+4); end
106     if (sPAM==[1 1 0 1 0 0]) s(i)=-(alpha+6)-j*(alpha); end
107     if (sPAM==[1 1 0 1 0 1]) s(i)=-(alpha+6)-j*(alpha+2); end
108     if (sPAM==[1 1 0 1 1 0]) s(i)=-(alpha+4)-j*(alpha); end
109     if (sPAM==[1 1 0 1 1 1]) s(i)=-(alpha+4)-j*(alpha+2); end
110     if (sPAM==[1 1 1 0 0 0]) s(i)=-(alpha)-j*(alpha+6); end
111     if (sPAM==[1 1 1 0 0 1]) s(i)=-(alpha)-j*(alpha+4); end
112     if (sPAM==[1 1 1 0 1 0]) s(i)=-(alpha+2)-j*(alpha+6); end
113     if (sPAM==[1 1 1 0 1 1]) s(i)=-(alpha+2)-j*(alpha+4); end
114     if (sPAM==[1 1 1 1 0 0]) s(i)=-(alpha)-j*(alpha); end
115     if (sPAM==[1 1 1 1 0 1]) s(i)=-(alpha)-j*(alpha+2); end
116     if (sPAM==[1 1 1 1 1 0]) s(i)=-(alpha+2)-j*(alpha); end
117     if (sPAM==[1 1 1 1 1 1]) s(i)=-(alpha+2)-j*(alpha+2); end
118   end
119   fprintf('.');
120   end
```

```
%-------------------------------------------------------------------------
%Cálculo de energía media de la constelación QAM
%-------------------------------------------------------------------------
Es=0;
Estot=0;
for i=alpha:2:(ceil(sqrt(n)))-1+alpha
  for u=alpha:2:(ceil(sqrt(n)))-1+alpha
      Es=Es+(i^2+u^2);
   end
    Estot=Estot+Es;
    Es=0;
end
Estotmedia=Estot/(n/4);%Media entre el número de símbolos de un cuadrante
s=s/sqrt(Estotmedia);
map_out=s;
fprintf('.\tOK\n');
if const==1
figure(1)
plot(s+i*eps, 'bo','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','w',...
            'MarkerSize',10);
hold on
plot(s+i*eps, '.b');
hold off
if n==4
title(sprintf('Constelación QPSK unitaria del simbolo %g OFDM a transmitir',S))
else
title(sprintf('Constelación %gQAM unitaria del simbolo %g OFDM a transmitir',n,S))
end
end
```

Código 12 mapeo.m

4.3 MODULACIÓN OFDM

El principio de la modulación OFDM consiste en distribuir el flujo binario de información entre un gran número de portadoras de forma que cada una posea una velocidad de datos reducida con respecto a la del flujo total. En consecuencia, la duración T_U de los símbolos aumenta respecto al caso de modular una sola portadora, haciendo a la señal muy robusta frente a interferencias por trayectos múltiples (ecos) ya que el retardo de éstos resulta ser muy pequeño comparado con la duración citada.

Por otra parte, la separación en frecuencia entre las portadoras se hace igual al inverso de la duración T_U de los símbolos, con lo que la posición de las portadoras en el espectro de frecuencias, mostrado en la Fig. 4.23, coincide con los nulos del espectro de las portadoras adyacentes. Éste último hecho se conoce como condición de ortogonalidad de portadoras. En estas condiciones se consigue mínima interferencia intersimbólica, ISI (*Inter Simbolic Iterference*). Véase Capítulo 3.

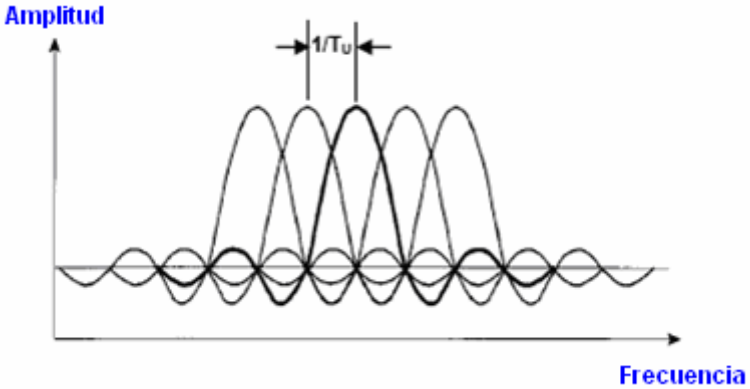


Fig. 4.23 Espectro de portadoras adyacentes en la modulación OFDM

Para fortalecer todavía más a la señal transmitida frente a los ecos se amplía la duración de los símbolos añadiendo un tiempo Δ denominado "intervalo de guarda" a la duración útil con lo que la duración total del símbolo pasa a ser:

$$T_S = \Delta + T_U \tag{4.21}$$

El intervalo de guarda es una continuación cíclica de la parte útil del símbolo, normalmente la parte final del símbolo de datos, el cual se inserta delante de él. En estas condiciones, si la señal se recibe por dos caminos distintos con un retardo relativo entre ellas, siempre que este retardo no supere el intervalo de guarda, coincidirá en las dos la información contenida dentro del tiempo útil de la señal principal. La Fig. 4.24 ilustra esta situación. Como los receptores ignoran la señal recibida durante el intervalo de guarda de la señal principal, el resultado es que no habrá ISI, Sin embargo, la inserción de este intervalo de guarda supone una pérdida de capacidad de transmisión del canal. El tiempo Δ del intervalo de guarda se mide en fracciones de la duración útil T_U del símbolo, disponiéndose en el presente estándar 4 posibles valores:

$$\Delta / T_U = 1/4 \quad 1/8 \quad 1/16 \quad 1/32 \tag{4.22}$$

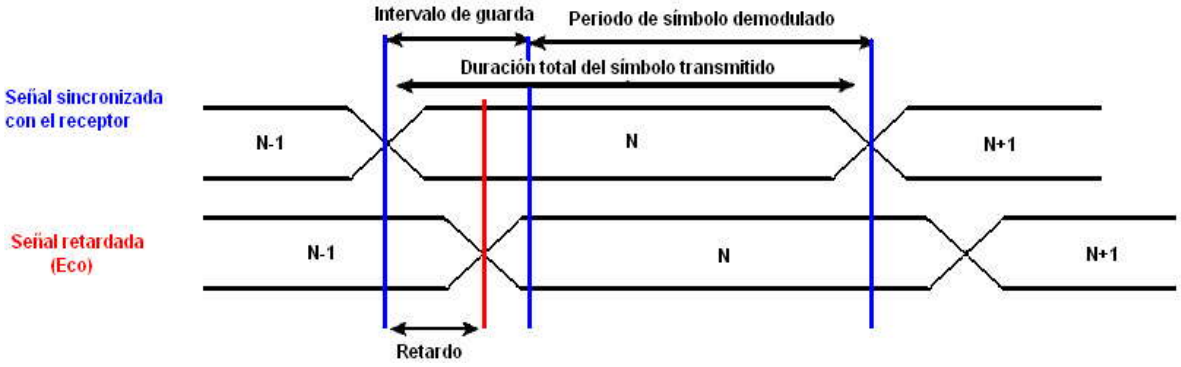


Fig. 4.24 Multitrayecto con retardo inferior al intervalo de guarda, Δ

La Tabla 4.12 muestra los valores numéricos que toman los parámetros descritos para los dos modos de transmisión (2k y 8k) en cada uno de los anchos de banda de canal (6MHz, 7MHz y 8MHz) que permite el estándar DBV-T:

Tabla 4.12 Valores numéricos de los parámetros OFDM según modo y ancho de banda de canal

	8 MHz		7MHz		6MHz	
Parámetro	**Modo 2k**	**Modo 8k**	**Modo 2k**	**Modo 8k**	**Modo 2k**	**Modo 8k**
Número de portadoras de datos, L	1512	6048	1512	6048	1512	6048
Número de portadoras pilotos, P	193	769	193	769	193	769
Número de portadoras, K=D+P	1705	6817	1705	6817	1705	6817
Portadora K_{min}	0	0	0	0	0	0
Portadora K_{max}	1706	6816	1706	6816	1706	6816
Duración de símbolo, T_U	224 µs	896 µs	256 µs	1024 µs	298.6 µs	1195 µs
Espaciado entre portadoras, 1/ T_U	0.447 kHz	1.12 kHz	3.91kHz	0.97 kHz	3.35 kHz	0.8370 kHz
Espaciado entre portadoras K_{min} y K_{max}	7.61 MHz	7.61 MHz	6.66 MHz	6.66 MHz	5.71 MHz	5.71 MHz
Periodo elemental, T	7/64 µs		1/8 µs		7/48 µs	

(a)Número de portadoras y espaciados frecuenciales

	8 MHz							
Parámetro	**Modo 2k**				**Modo 8k**			
Número puntos de la FFT, N_{FFT}	2048				8192			
Duración de símbolo de datos, T_U	N_{FFT} x T=224µs				N_{FFT} xT=896µs			
Intervalo de guarda. Δ/T_U	1/4	1/8	1/16	1/32	1/4	1/8	1/16	1/32
Duración intervalo de guarda, Δ	512xT 56 µs	256xT 28 µs	128xT 14 µs	64xT 7 µs	2048xT 224 µs	1024xT 112 µs	512xT 56µs	256xT 28 µs
Duración de símbolo OFDM, $T_S=\Delta+T_U$	2560xT 280 µs	2304xT 252 µs	2176xT 238 µs	2112xT 231 µs	10240xT 1120 µs	9216xT 1008µs	8704xT 952µs	8448xT 924 µs

(b)Duraciones temporales de los símbolos OFDM para canales radioeléctricos de 8MHz

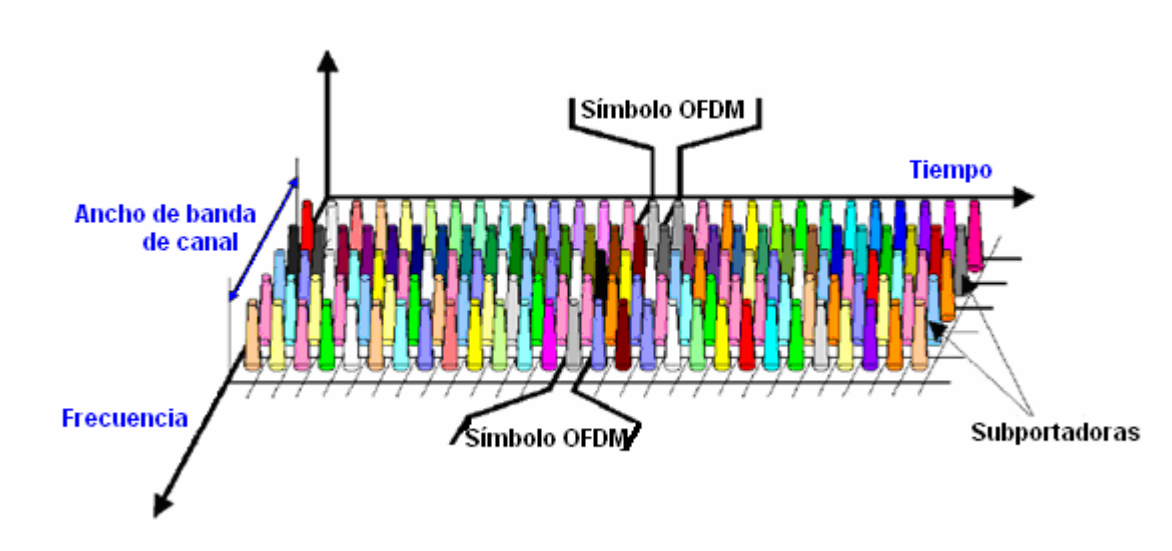
	7 MHz							
Parámetro	**Modo 2k**				**Modo 8k**			
Número puntos de la FFT, N_{FFT}	2048				8192			
Duración de símbolo de datos, T_U	N_{FFT} x T=256µs				N_{FFT} xT=1024µs			
Intervalo de guarda. Δ/T_U	1/4	1/8	1/16	1/32	1/4	1/8	1/16	1/32
Duración intervalo de guarda, Δ	512xT 64 µs	256xT 32 µs	128xT 16 µs	64xT 8 µs	2048xT 256 µs	1024xT 128 µs	512xT 64µs	256xT 32 µs
Duración de símbolo OFDM, $T_S=\Delta+T_U$	2560xT 320 µs	2304xT 288 µs	2176xT 272 µs	2112xT 264 µs	10240xT 1280 µs	9216xT 1152µs	8704xT 1088µs	8448xT 1056µs

(c)Duraciones temporales de los símbolos OFDM para canales radioeléctricos de 7MHz

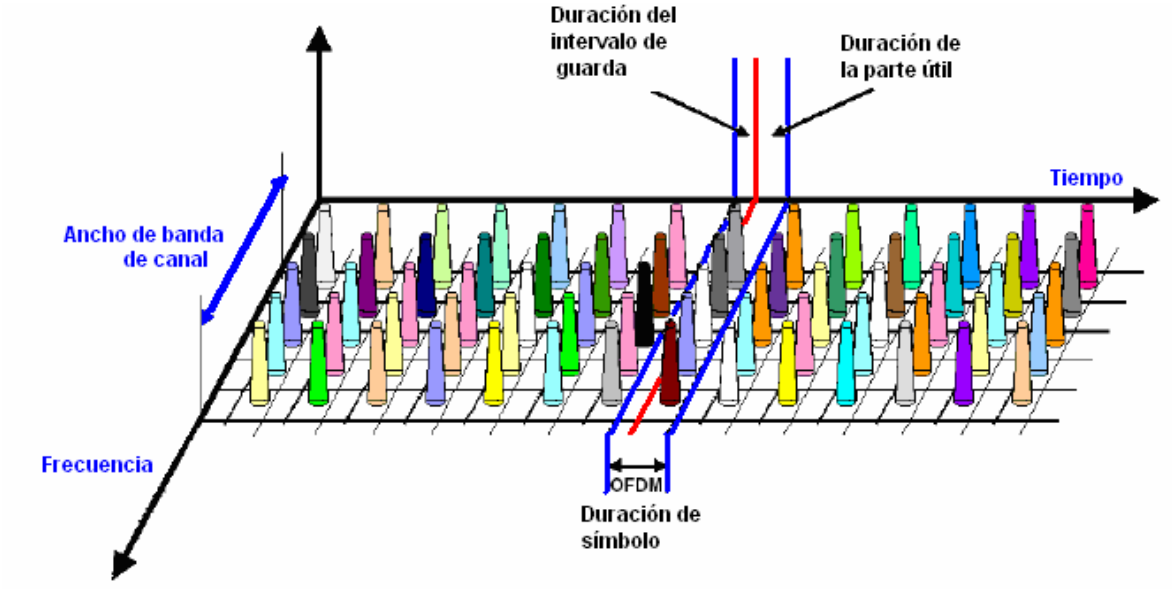
	6 MHz							
Parámetro	**Modo 2k**				**Modo 8k**			
Número puntos de la FFT, N_{FFT}	2048				8192			
Duración símbolo de datos, T_U	N_{FFT} x T=298.667µs				N_{FFT} xT=1194.667µs			
Intervalo de guarda. Δ/T_U	1/4	1/8	1/16	1/32	1/4	1/8	1/16	1/32
Duración intervalo de guarda, Δ	512xT 74.6 µs	256xT 37.3 µs	128xT 28.6 µs	64xT 9.3 µs	2048xT 298.6 µs	1024xT 149.3µs	512xT 74.6µs	256xT 37.3µs
Duración de símbolo OFDM, $T_S=\Delta+ T_U$	2560xT 373.3µs	2304xT 336µs	2176xT 317.3µs	2112xT 308µs	10240xT 1493.3µs	9216xT 1344µs	8704xT 1269.3µs	8448xT 2132µs

(d)Duraciones temporales de los símbolos OFDM para canales radioeléctricos de 6MHz

La figura Fig. 4.25 ilustra la disposición de las portadoras en el tiempo y en la frecuencia antes y después de la inserción del intervalo de guarda.



(a)Antes de la inserción del intervalo de guarda



(b)Después de la inserción del intervalo de guarda

Fig. 4.25 Distribución de las portadoras del símbolo OFDM en tiempo y frecuencia.

4.3.1 Capacidad del canal

El estándar 2k especifica, como se ha indicado en el apartado anterior, 1705 portadoras totales en cada símbolo OFDM mientras que el estándar 8k tiene 6817 portadoras. Ahora bien, no todas las portadoras están moduladas por los datos procedentes de la Codificación de Canal (Apartado 4.2). Como se ha indicado al describir el mapeado de los símbolos (Apartado 4.2.7), sólo 1512 portadoras en modo 2k y 6048 portadoras en modo 8k son portadoras útiles para datos.

Denotando F_T al flujo binario total transportado por las portadoras útiles para datos, este parámetro viene definido por:

$$F_T = f_S \cdot v \cdot L \quad (bits/s) \tag{4.23}$$

siendo f_S la frecuencia de los símbolos (símbolos/s), $T_S=1/f_S$ la duración del símbolo OFDM, v el número de bits por portadora (en función del esquema de modulación) y L el número de portadoras activas para datos.

La capacidad del canal o flujo binario útil, F_U, resultará de descontar del flujo binario total, F_T, las redundancias incluidas en la codificación interna (Véase apartado 4.2.4) y en la codificación Reed-Solomon (Véase apartado 4.2.2), es decir:

$$F_U = F_T \cdot r \cdot 188/204 \quad (bits/s) \tag{4.24}$$

siendo r la relación de codificación interna.

A modo de ejemplo, en el caso de transmisión en modo 8k (L=6048), relación de codificación r=2/3, intervalo de guarda Δ=1/4 y constelación 64QAM (v=6), para canales de 8MHz (T_S=1120μs) se tendrá un flujo binario total, según (4.23), de F_T=32.4Mbps y una capacidad de canal, según (4.24), de F_U=19.9Mbps.

Se debe comentar que el flujo binario útil, F_U, o capacidad de canal es independiente del modo de transmisión empleado (2k o 8k) puesto que, a pesar de emplear diferente número de portadoras, el espaciado frecuencial entre la primera portadora de datos, K_{min}, y la última, K_{maz}, es el mismo (Véase Tabla 4.12 (a)).

4.3.2 Portadoras Piloto. Estructura de Tramas de la señal OFDM

Un símbolo OFDM está compuesto por un conjunto de K elementos o celdas correspondiendo cada uno de ellos a una portadora (Véase Fig. 4.25 (a)), donde K corresponde a 1705 elementos (portadoras totales) en el modo 2k y a 6048 elementos en el modo 8k. Sin embargo los datos procedentes de la Codificación de canal modulan solamente 1512 portadoras en modo 2k y 6048 portadoras en modo 8k, que son las portadoras activas para datos contenidas en cada símbolo OFDM.

Esto significa que, además de las portadoras para datos, la señal transmitida incluye otras portadoras o celdas cuya utilidad es la siguiente.

- Portadoras Piloto Continuas (*Continued Pilots*): usadas para sincronización del receptor en frecuencia, fase e inicio de símbolo (Véase apartado 6.2.1.2)
- Portadoras Piloto Dispersas (*Scattered Pilots*): usadas para regeneración del canal en amplitud y fase en el receptor. (Véase apartado 6.2.2.2)
- Portadoras TPS (*Transmission Parameter Signalling*): transmiten información referente a parámetros de transmisión empleados. (Véase apartados 6.3.3 y 6.3.4)

En el siguiente apartado se estudiará con más profundidad la estructura y utilidad de las portadoras piloto. La incorporación de estas portadoras piloto en número y distribución adecuadas exige organizar la señal transmitida en unidades de "Trama", de la siguiente manera:

- Cada trama OFDM de duración T_F consiste en 68 símbolos OFDM, que se numeran de 0 a 67, tanto en modo 2k como en modo 8k. En consecuencia, $T_F=68\cdot T_S$.
- Una super-trama OFDM está formada por 4 tramas OFDM, numeradas de 0 a 3, tanto en modo 2k como en modo 8k.
- En cambio, una mega-trama OFDM está formada por 32 tramas en el estándar 2k y por 8 tramas en el estándar 8k.

Ésta estructura de trama OFDM permite transmitir un número entero de paquetes de 204 bytes (paquetes a la salida del bloque de codificación Reed-Solomon, véase apartado 4.2.2) por supertrama OFDM, por lo que se evita el uso de cualquier tipo de relleno de datos ya sea cual sea la constelación utilizada, el intervalo de guarda, el ancho de banda radioeléctrico o la tasa de código convolucional que se utilice. El único requisito reside en que el primer byte transmitido en una supertrama OFDM debe ser uno de los bytes de sincronización, *SYNC* (o su negado, $\overline{SYNC}$), de los datos MPEG (véase 4.2.2).

Tabla 4.13 Número de paquetes RS transmitidos por supertrama OFDM según combinaciones de modulaciones y tasas de código

	QPSK		16-QAM		64-QAM	
Tasa de código	**Modo 2k**	**Modo 8k**	**Modo 2k**	**Modo 8k**	**Modo 2k**	**Modo 8k**
1/2	252	1008	504	2016	756	3024
2/3	336	1344	672	2688	1008	4032
3/4	387	1512	756	3024	1134	4536
5/6	420	1680	840	3360	1260	5040
7/8	441	1764	882	3528	1323	5292

La Fig. 4.26 ilustra la estructura en tramas de la señal OFDM, donde puede verse la distribución de las portadoras piloto dispersas, continuas y TPS, cuyo número es el siguiente:

Tabla 4.14 Tipo y número de portadoras por símbolo OFDM

Tipo de portadora	**Modo 2k**	**Modo 8k**
Continuas	45	177
Dispersas	131	524
TPS	17	68
Datos	1512	6048
TOTAL portadoras:	1705	6817

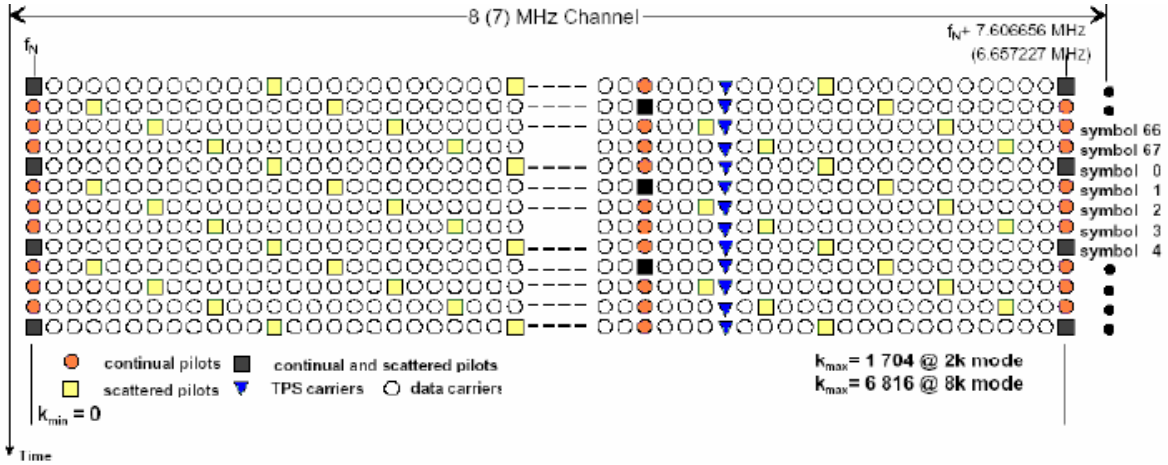


Fig. 4.26 Estructura de las portadoras OFDM en tiempo y frecuencia.

Tal y como se comenta en el apartado 0, las portadoras que contienen datos se transmiten con nivel de potencia normalizado, de manera que cumplen la condición expresada en (4.20). En cambio, las celdas que contienen las portadoras piloto Continuas y portadoras piloto Dispersas se transmiten con el nivel de potencia reforzado (*boosted*), cumpliendo:

$$E[c\,x\,c^*] = 16/9 \tag{4.25}$$

Finalmente, las portadoras TPS se transmiten con el nivel de potencia normalizado, es decir, cumpliendo la condición expresada en (4.20).

La Fig. 4.27 muestra las posiciones ocupadas por las Portadoras Piloto Continuas, Dispersas y TPS en una constelación tipo 64-QAM uniforme (α=1).

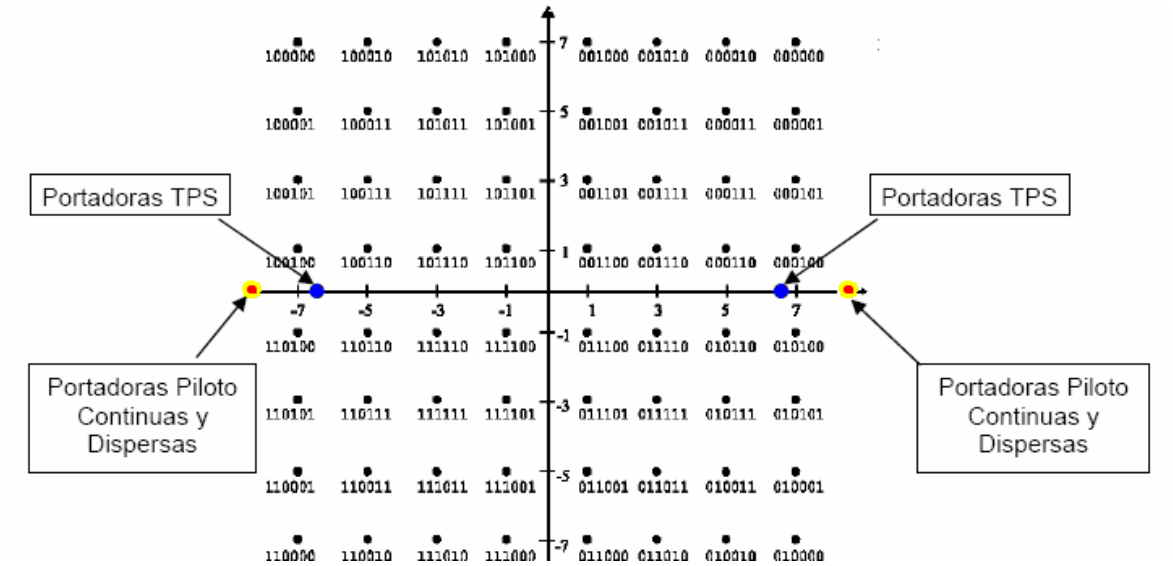
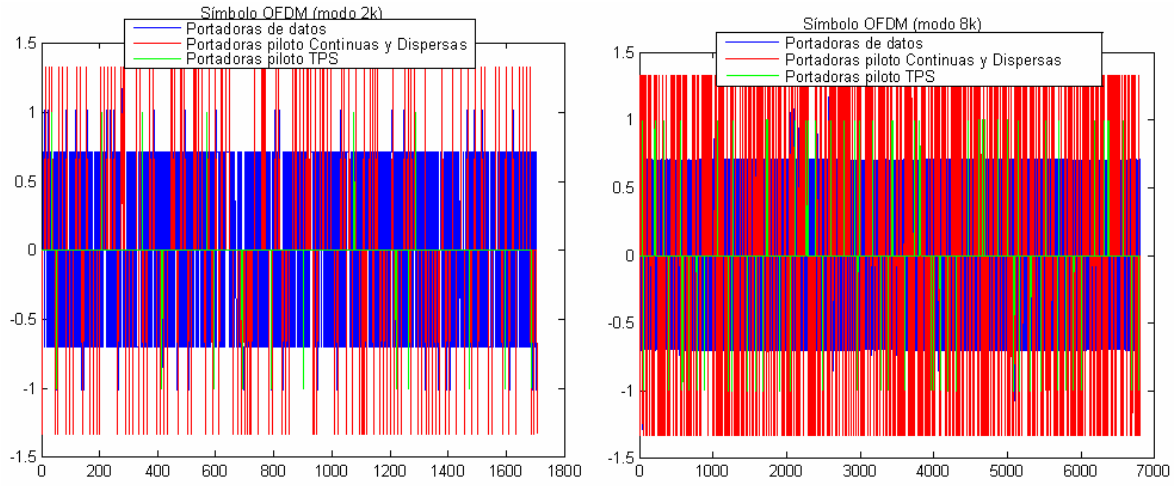


Fig. 4.27 Constelación 64QAM uniforme (α=1), con portadoras Piloto Continuas, Dispersas y TPS

En la Fig. 4.28 se muestra la estructura de pilotos del símbolo OFDM en el dominio frecuencial, una vez insertadas las portadoras piloto. En (a) se presenta la estructura de un símbolo de modo 2k y en (b) se presenta la estructura de otro símbolo OFDM de modo 8k.



(a) Modo 2k (b) Modo 8k

Fig. 4.28 Estructura de las portadoras de un símbolo OFDM

Obsérvese que el número de portadoras piloto en el símbolo OFDM del modo 8k tiene muchas más portadoras piloto que las del símbolo OFDM de modo 2k. Obsérvese también la diferencia de potencia que tienen las portadoras continuas y dispersas con frente a las portadoras TPS o las portadoras de datos. Esta estructura de portadoras piloto está diseñada en base a poder utilizar su posición y valor conocidos para transportar información de la transmisión así como para ser utilizadas para emplear técnicas de estimación de canal y otros parámetros de sincronización. A continuación se explica detalladamente la generación, función y objetivos de cada tipo de portadoras piloto.

4.3.3 Señales de referencia

4.3.3.1 Localización del bloque en el sistema

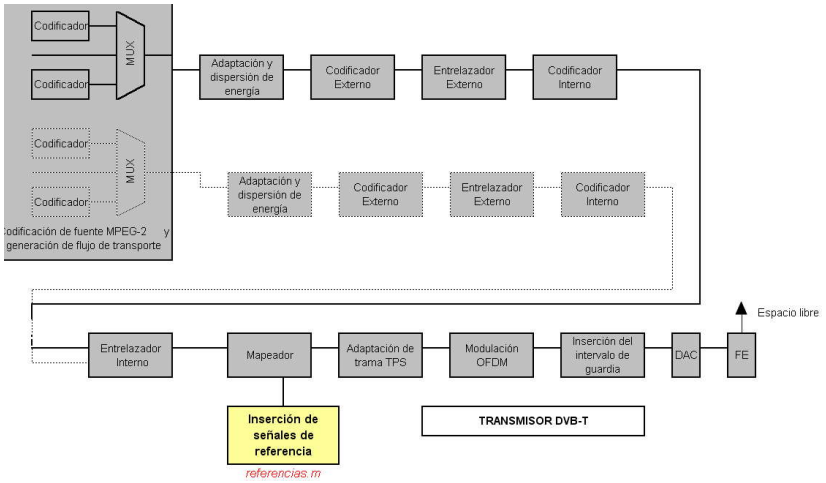


Fig. 4.29 Localización del bloque de inserción de señales de referencia

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *referencias.m.*

4.3.3.2 Objetivo y relación con el estándar

4.3.3.2.1 Modulación de las Portadoras Piloto Continuas y Dispersas

Las portadoras Piloto Continuas y las portadoras Piloto Dispersas se modulan con la denominada "Información de Referencia".

Cada portadora Piloto Continua coincide con una portadora Piloto Dispersa cada 4 símbolos OFDM (véase Fig. 4.26) y la información transmitida por ambos tipos de portadora piloto se deriva de una Secuencia PRBS, que se genera de acuerdo con el esquema mostrado en la siguiente figura:

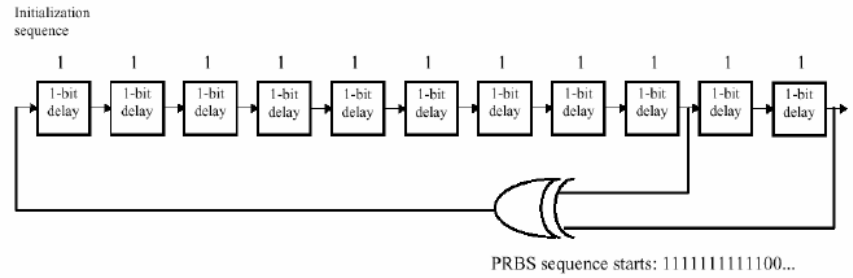


Fig. 4.30 Generador PRBS para las portadoras piloto Continuas y Dispersas

y que tiene como polinomio generador de secuencia PRBS:

$$x^{11} + x^2 + 1 \tag{4.26}$$

El PRBS se inicializa de forma que el primer bit de salida coincide con la primera portadora activa. Con cada portadora, ya sea piloto o no, se genera un nuevo valor.

El resultado del generador PRBS, denominado "w_k", aplicable a la portadora número k, modula las portadoras tanto Continuas como Dispersas, con el siguiente esquema:

$$\begin{aligned} \mathrm{Re}(c_{m,l,k}) &= 4/3x2(1/2 - w_k) \\ \mathrm{Im}(c_{m,l,k}) &= 0 \end{aligned} \tag{4.27}$$

el cual corresponde, como se ha indicado anteriormente, a un nivel de potencia reforzado que cumple (4.25).

4.3.3.2.2 Localización de las Portadoras Piloto Continuas

A diferencia de lo que ocurre con las portadoras Piloto Dispersas, las posiciones que ocupan dichas portadoras Piloto Continuas en cada símbolo OFDM son siempre las mismas (véase Fig. 4.26) y se recogen en la siguiente tabla:

Tabla 4.15 Posiciones de las portadoras piloto continuas (índices k) del símbolo OFDM

Modo 2k	Modo 8k
0 48 54 87 141 156 192 201 255 279 282 333 432 450 483 525 531 618 636 714 759 765 780 804 873 888 918 939 942 969 984 1050 1101 1107 1110 1137 1140 1146 1206 1269 1323 1377 1491 1683 1704	**0 48 54 87 141 156 192 201 255 279 282 333 432 450 483 525 531 618 636 714 759 765 780 804 873 888 918 939 942 969 984 1050 1101 1107 1110 1137 1140 1146 1206 1269 1323 1377 1491 1683 1704** 1752 1758 1791 1845 1860 1896 1905 1959 1983 1986 2037 2136 2154 2187 229 2235 2322 2340 2418 2463 2469 2484 2508 2577 2592 2622 2643 2646 2673 2688 2754 2805 2811 2814 2841 2844 2850 2910 2973 3027 3081 3195 3387 3408 3456 3462 3495 3549 3564 3600 3609 3663 3687 3690 3741 3840 3858 3891 3933 3939 4026 4044 4122 4167 4173 4188 4212 4281 4296 4326 4347 4350 4377 4392 4458 4509 4515 4518 4545 4548 4554 4614 4677 4731 4785 4899 5091 5112 5160 5166 5199 5253 5268 5304 5313 5367 5391 5394 5445 5544 5562 5595 5637 5643 5730 5748 5826 5871 5877 5892 5916 5985 6000 6030 6051 6054 6081 6096 6162 6213 6219 6222 6249 6252 6258 6318 6381 6435 6489 6603 6795 6816

Obsérvese que todas las portadoras piloto continuas del estándar 2k coinciden en posición con las primeras 45 del estándar 8k.

4.3.3.2.3 Localización de las Portadoras Piloto Dispersas

Se define m el número de trama OFDM, l el número de símbolo OFDM dentro de cada trama OFDM (con l=0,1,...67) y k el índice de la portadora dentro de cada símbolo OFDM (con k= 0,1,...k_{max} siendo k_{max}=1704 en modo 2k y k_{max}=6816 en modo 8k). Entonces, para el símbolo número l de una determinada trama OFDM, los índices k que corresponden al subconjunto de Portadoras Dispersas vienen dados por:

$$k = 3x(l \bmod 4) + 12\mathrm{p} \qquad (4.28)$$

siendo p un número entero mayor que 0 que debe cumplir la condición de que el valor resultante para k no exceda el valor k_{max} correspondiente al estándar de que se trate.

De acuerdo con la expresión de (4.28), es evidente que las portadoras dispersas están separadas entre sí 12 posiciones y que ocupan la posición 0 (compartida con portadoras continuas) en todos los símbolos numerados l=0, 4, 8, 12..., es decir, las que cumplen la condición de "$l$mod4=0". En este caso, hay 569 portadoras dispersas por símbolo OFDM en modo 8k, de las cuales 45 coinciden con pilotos continuas, y 143 portadoras dispersas por símbolo en modo 2k, de las que 12 coinciden en posición con las continuas.

El resto de símbolos OFDM contienen 568 portadoras piloto dispersas en modo 8k, de las cuales 44 coinciden con las continuas, mientras que en modo 2k hay 142 portadoras dispersas por símbolo OFDM de las que 11 coinciden con las continuas.

4.3.3.2.4 Localización de las portadoras TPS

Las portadoras TPS sirven para señalizar los parámetros correspondientes al esquema de transmisión empleado, es decir, para informar de todo lo relativo a la Codificación de Canal y a la Modulación usadas en la transmisión.

Estas portadoras TPS se transmiten en paralelo, ocupando 17 posiciones de cada símbolo OFDM en el estándar 2k y 68 posiciones en el estándar 8k (Véase Fig. 4.26). La Tabla 4.16 recoge los índices de las posiciones que ocupan las portadoras TPS en cada símbolo OFDM para ambos estándares.

Tabla 4.16 Posiciones de las portadoras piloto TPS (índices k) del símbolo OFDM

Modo 2k	Modo 8k
34 50 209 346 413 569 595 688 790 901 1073 1219 1262 1286 1469 1594 1687	**34 50 209 346 413 569 595 688 790 901 1073 1219 1262 1286 1469 1594 1687** 1738 1754 1913 2050 2117 2273 2299 2392 2494 2605 2777 2923 2966 2990 3173 3298 3391 3442 3458 3617 3754 3821 3977 4003 4096 4198 4309 4481 4627 4670 4694 4877 5002 5095 5146 5162 5321 5458 5525 5681 5707 5800 5902 6013 6185 6331 6374 6398 6581 6706 6799

Obsérvese que todas las portadoras piloto TPS del estándar 2k coinciden en posición con las primeras 17 del estándar 8k.

4.3.3.3 Código implementado

```
1  function [data_ref,tps,continual,w,indices_pilotos,sintetic] = referencias(data_in,modo, N, P, T, S, tps, continual,
   S_index, w)
2  %-------------------------------------------------------------------------------------------
3  %   referencias.m
4  %   Ver ETSI 300.744 v1.51 4.5 pag 27 : Reference Signals
5  %-------------------------------------------------------------------------------------------
6  fprintf('·Tx: Adición de %g señales de referencia al simbolo %g OFDM de %g portadoras de datos',P,S,N);
7  fprintf('.');
8  %-------------------------------------------------------------------------
9  % Apuntamos a las posiciones de los piloto (índices)
   definidas por el estandar
10 %-------------------------------------------------------------------------
   ----
11    if S==1&&modo==2                                  Cargamos la primera vez que se ejecuta la función. Son
                                                        siempre los mismos indices.
12       %--------------------TPS-------------------------
13       tps=[34 50 209 346 413 569 595 688 790 901
14       1073 1219 1262 1286 1469 1594 1687];
15       %Posición fija para cada símbolo OFDM
16        %--------------------CONTINUAL-------------------
17       continual=[0 48 54 87 141 156 192 201 255 279
   282 333 432 450  483 525 531 618 636 714 759 765
   780 804 873 888 918 939 942 969 984 1050 1101 1107
   1110 1137 1140 1146 1206 1269 1323 1377 1491 1683
   1704];
18       %posicion fija para cada simbolo OFDM
19    elseif(S==1&&modo==8)
20       %--------------------TPS-------------------------
21
         tps=[34 50 209 346 413 569 595 688 790 9011073
   1219 1262 1286 1469 1594 1687 1738 1754 1913 2050
   2117 2273 2299 2392 2494 2605 27772923 2966 2990
   3173 3298 3391 3442 3458 3617 3754 3821 3977 4003
   4096 4198 4309 4481 4627 4670 4694 4877 5002 5095
   5146 5162 5321 5458 5525 5681 5707 5800 5902 6013
   6185 6331 6374  6398 6581 6706 6799];

22       %posicion fija para cada simbolo OFDM
23       %--------------------CONTINUAL-------------------
24
         continual=[0 48 54 87 141 156 192 201 255 279
   282 333 432 450  483 525 531 618 636 714 759 765
   780 804 873 888 918 939 942 969 984 1050 1101 1107
   1110 1137 1140 1146 1206 1269 1323 1377 1491 1683
   1704 1752 1758 1791 1845 1860 1896 1905 1959 1983
   1986 2037 2136 2154 21872229 2235 2322 2340 2418
   2463 2469 2484 2508 2577 2592 2622 2643 2646 2673
   2688 2754 2805 2811 2814 2841 2844 2850 2910 2973
   3027 3081 3195 3387  3408 3456 3462 3495 3549 3564
   3600 3609 3663 3687 3690  3741 3840 3858 3891 3933
   3939 4026 4044 4122 4167 4173 4188 4212 4281 4296
   4326 4347 4350 4377 4392 4458 4509 4515 4518 4545
   4548 4554 4614 4677 4731 4785 4899 5091 5112 5160
   5166 5199 5253 5268 5304 5313 5367 5391 5394 5445
   5544 5562 5595 5637 5643 5730 5748 5826 5871 5877
   5892 5916 5985 6000 6030 6051 6054 6081 6096 6162
   6213 6219 6222 6249 6252 6258 6318 6381 6435 6489
   6603 6795 6816 ];

25       %posicion fija para cada simbolo OFDM
26    end
```

```
27    %-----------------SCATTERED)-------------------------
28    %Estas posiciones son diferentes en cada simbolo .
      VER ETSI 300.744 v1.51 pag 28 4.5.3
29    Kmin=0;                                  Índice de la primera portadora dentro del símbolo
                                               completo(indexado como en el estandar)
30    Kmax=T-1;                                Índice de la ultima portadora de un simbolo(indexado
                                               como estandar)
31    if modo==2
32    p=(0:ceil(T-9)/12);                      El -9 es para restarle el valor max de un 3*mod(x,4) que
                                               tiene los valores 3*(0,1,2,3).
33    else
34    p=(0:ceil(T-9)/12);                      Número escalar (que empieza en 0) que asegura que k
                                               no superará Kmax
35    end
36    k =(Kmin +3*rem((S_index-1),4)+ 12*p);   Esta posicion se repite cada 4 simbolos.Ponemos el -1
                                               para indexar bien!!
37                                             Ahora juntamos las posiciones de los pilotos continuos y
      indices_pilotos= union (continual,k);    los de scattered debido a que coinciden en algunas
                                               posiciones. Con union podemos combinar dos vectores
                                               eliminando los elementos repetidos.
38    %-----------------------------------------------------------------
      ----
39    % Secuencia PRBS para modulacion de pilotos. VER
      ETSI 300.744 v1.51 4.5.2 pag 28
40    %-----------------------------------------------------------------
      ----
41                                             Calculamos la secuencia la primera vez que se realiza
      if S_index==1                            la función. Luego se mantiene fija para todos los
                                               simbolos
42    w= zeros(1,T);                           Almacenaremos la secuencia PRBS
43    num_reg=11;                              Numero de  registros
44    reg = ones(1,num_reg);
45    for m = 1:T
46      w(m) = reg(11);
47      bit_nuevo = xor (reg(9), reg(11));
48      reg = [ bit_nuevo reg(1:10) ];
49    end
50    end
51    fprintf('.');
52    %-----------------------------------------------------------------------------------------------------------------------
53    %Preparación  vector de pilotos continuos "sinteticos" para sincronizacion en tiempo y estimación de canal
54    %-----------------------------------------------------------------------------------------------------------------------
55    sintetic=zeros(T,1);
56    %-----------------------------------------------------------------------------------------------------------------------
57    %Creación del vector paralelo (columna) de portadoras de datos y pilotos del simbolo OFDM
58    %-----------------------------------------------------------------------------------------------------------------------
59    data_ref = zeros(T, 1);
60    d=1;                                     Aquí irán metidos las portadoras de datos y piloto en
                                               comlumna (paralelo)
61    ps=1;                                    apuntador a la primera posición de la portadora de
                                               datos
62    pt=1;                                    apuntador a la primera posición de la portadora piloto
                                               scattered+continual
63    for u = 1:T                              apuntador a la primera posición de la portadora piloto
                                               tps
64      p = indices_pilotos(ps)+1;
65      if pt <= length(tps)                   Sumamos 1 para referenciarlo a un vector que empieza
                                               en la pos 1 y no en la 0
66        t = 1+tps(pt);
67      else
68        t = 0;
69      end
```

```
70  %------------------Modulacion de pilotos------------------------
71   if u == p
72     data_ref(u) =( 4/3 * 2 * (0.5 - w(p)));
73     sintetic(u)=data_ref(u);
74     ps = ps + 1;
75   elseif u == t; % Señal tps
76     data_ref(u) = 2 * (0.5 - w(t));
77      pt = pt + 1;
78   else
79     data_ref(u) =  data_in(d);
80     d = d + 1;
81   end
82  end
83 fprintf('.\tOK\n');
```

Estamos en la posicion que corresponde a la piloto scatered+continua
Modulacion scattered y/o continua.
Se escala la energía de estos pilotos a 16/9 (4/3^2), VER ETSI EN 300 744 v.1.5.1 pag 29 apartado 4.5.5
Creamos el vector columna de pilotos continuos para la sincronización en tiempo
VER ETSI 300.744 v1.51 4.5.2 pag 28 4.5.3 pag 29

Modulacion tps (BPSK) para todas las portadoras de todos los simbolos
VER ETSI 300.744 v1.51 4.6.3 pag 34
señal de datos
Se requiere unitaria y ya lo está del bloque anterior de mapeo.

Código 13 referencias.m

4.3.4 Señales de información TPS

4.3.4.1 Localización del bloque en el sistema

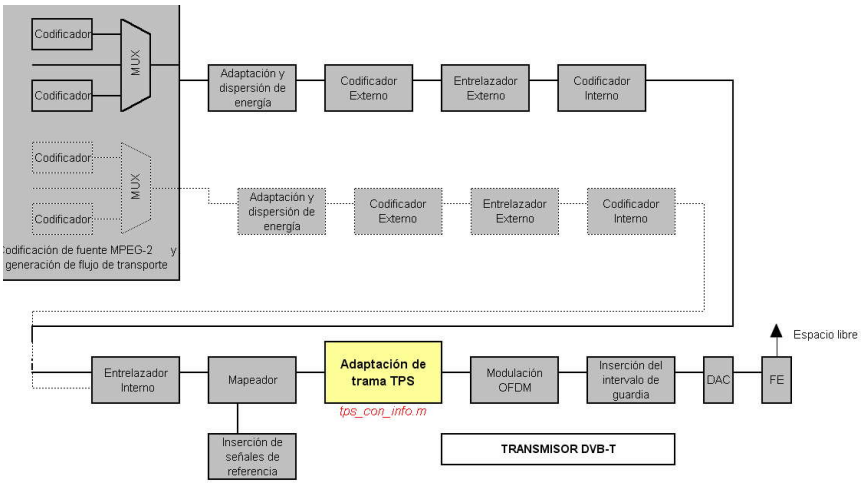


Fig. 4.31 Localización del bloque de adaptación de trama TPS de información

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *tps_con_info.m*. Utiliza una función auxiliar que se ocupa de la creación del bloque de información (usando además extensión de código BCH) que es *tps_info.m*.

4.3.4.2 Objetivo y relación con el estándar

Las portadoras TPS sirven para señalizar los parámetros correspondientes al esquema de transmisión empleado, es decir, para informar de todo lo relativo a la Codificación de Canal y a la Modulación usadas en la transmisión. La localización de dichas portadoras en cada uno de los estándares de transmisión DBV-T (2k y 8k) se encuentra recogida en el apartado 4.3.3.2.4.

Todas las portadoras TPS de un mismo símbolo OFDM transportan el mismo bit de información, y se agrupan en bloques de 68 símbolos consecutivos coincidentes con una trama OFDM. Así pues, cada bloque de portadoras TPS contiene 68 bits, los cuales se distribuyen de la siguiente forma:

- 1 bit de inicialización
- 16 bits de sincronización
- 37 bits de información
- 14 bits redundantes para protección contra errores

De los 37 bits de información, por el momento únicamente se utilizan 23, por lo que los restantes 14 bits están reservados para usos futuros y deben estar puestos a 0.
Las portadoras TPS transportan la siguiente información:

a) Tipo de constelación QAM, incluyendo el valor del parámetro α (modulaciones uniformes y no uniformes).
b) Información sobre el tipo de transmisión: jerárquica o no jerárquica.
c) Intervalo de guarda.
d) Relación de codificación interna.
e) Modo de transmisión: 2k u 8k.
f) Número de la trama OFDM dentro de la supertrama OFDM correspondiente (una supertrama OFDM contiene 4 tramas OFDM).

4.3.4.2.1 *Formato de transmisión de las portadoras TPS*

La siguiente tabla indica la manera en que la información relativa a la Codificación de Canal y a la Modulación se distribuye entre los 68 bits correspondientes a cada bloque de portadoras TPS, coincidentes con una trama OFDM (68 símbolos OFDM):

Tabla 4.17 Formato de transmisión de las portadoras TPS

Número de bit	Formato	Función
s_0		Bit de inicialización
s_1 - s_{16}	0011010111101110 Tramas 1º y 3º 1100101000010001 Tramas 2º y 4º	Palabra de sincronización
s_{17} - s_{22}	010111 (número 23 en binario)	Indicador de longitud de los bits de información usados
s_{23} - s_{24}	00 Trama 1º en la supertrama 01 Trama 2º en la supertrama 10 Trama 3º en la supertrama 11 Trama 4º en la supertrama	Número de trama
s_{25} - s_{26}	00 QPSK 01 16-QAM 10 64-QAM	Constelación
s_{27} - s_{29}	000 Transmisión no Jerárquica 001 α=1 010 α=2 011 α=4	Información Jerárquica
s_{30} - s_{32}	000 r=1/2 001 r=2/3 010 r=3/4 011 r=5/6 100 r=7/8	Relación de Codificación HP
s_{33} - s_{35}	000 r=1/2 001 r=2/3 010 r=3/4 011 r=5/6 100 r=7/8	Relación de Codificación LP
s_{36} - s_{37}	00 Δ/T_U=1/32 01 Δ/T_U=1/16 10 Δ/T_U=1/8 11 Δ/T_U=1/4	Intervalo de Guarda
s_{38} - s_{39}	00 Modo 2k 01 Modo 8k	Estándar de Transmisión
s_{40} - s_{53}	Todos a 0	Reservados para uso futuro
s_{54} - s_{67}	Extensión BCH	Protección de errores

4.3.4.2.2 Protección contra errores de las portadoras TPS. Codificación BCH

Para permitir la corrección de errores en la recepción se introduce una cierta redundancia en la estructura del bloque de información TPS de 54 bits iniciales, añadiendo 14 bits de paridad para formar un total de 68 bits TPS por bloque a transmitir. A este procedimiento que le conoce como codificación y es del tipo BCH(n,k,t) (*Bose-Chaudlhuri-Hocquenghem code*), donde n es el número de bits a la salida del bloque, k es el número de bits a la entrada del bloque y t el número de bits erróneos capaz de corregir.

La codificación externa usada es de tipo BCH(67,53,2), que es una versión acortada de la codificación original BCH(127,113,2), cuyo polinomio generador es:

$$p(x) = x^{14} + x^9 + x^8 + x^6 + x^5 + x^4 + x^2 + x + 1 \qquad (4.29)$$

La forma acortada se realiza añadiendo primeramente 60 bits nulos delante de los 53 bits de información (ya que no contamos el de inicialización), con lo que se obtienen 113 bits con los bytes iniciales del paquete TPS. Al pasar por el codificador BCH(127,113,2) se añaden los 14 bits de paridad, por lo que se obtienen finalmente 127 bits por cada paquete TS. Finalmente se eliminan los 60 bits nulos insertados inicialmente con lo que resultan los 67 bits para cada paquete de información TPS. Teniendo en cuenta el bit de inicialización, no afectado por la codificación BCH, se tiene un total de 68 bits por bloque TPS de información protegido contra errores. La Fig. 4.32 muestra un mapa conceptual en el que se ilustra el procedimiento para la obtención de un bloque TPS protegido contra errores (con codificación BCH) a partir de un bloque TPS de información básico y que se va a transmitir en una trama OFDM (68 símbolos OFDM).

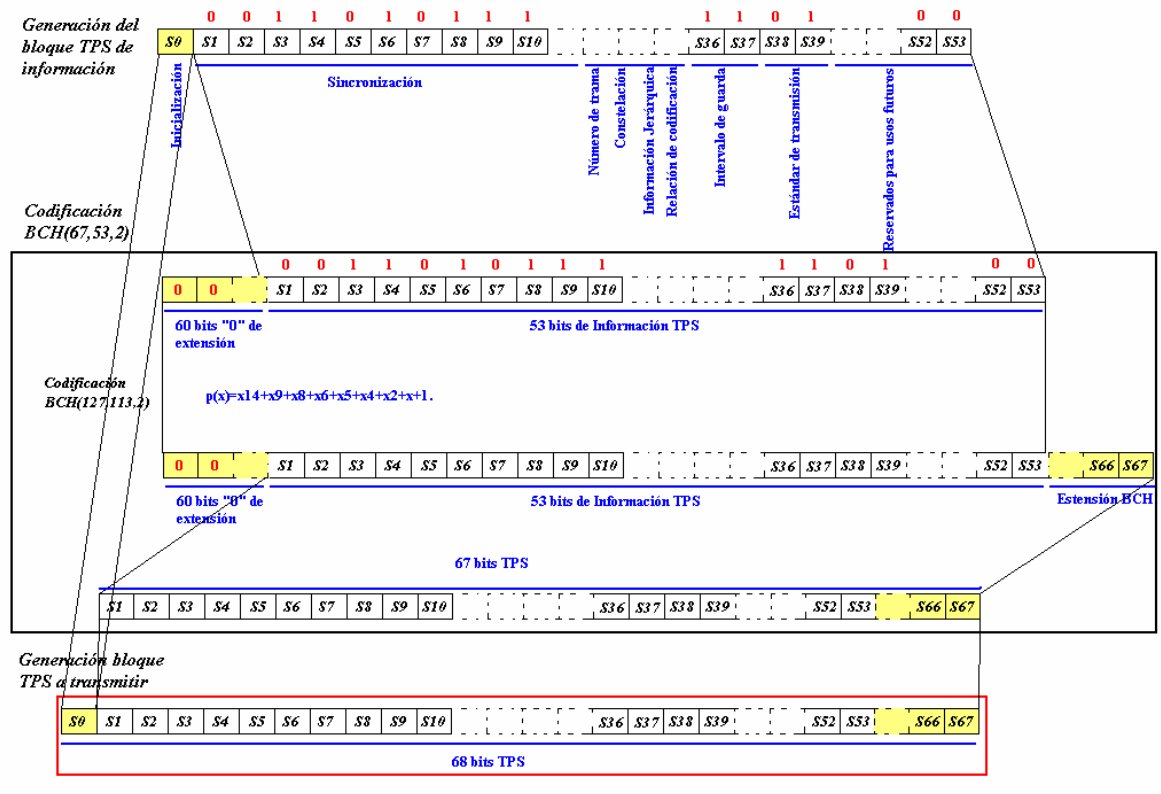


Fig. 4.32 Algoritmo de obtención del bloque TPS protegido contra errores (BCH)

La función implementada en Matlab que genera el bloque TPS y aplica la codificación BCH es *tps_info.m*. Esta función es llamada desde la función *tps_con_info.m*, que es la que se ocupa de modular el bloque TPS. Ambos códigos se presentan en el apartado 4.3.4.3.

4.3.4.2.3 Modulación de las portadoras TPS (DBPSK)

Cada portadora TPS está modulada con un esquema del tipo 2-PSK (*Phase Shift Keying)* diferencial, es decir, DBPSK. Esta modulación se inicializa al principio de cada bloque TPS, es decir, al principio de cada trama, para lo cual las portadoras TPS pertenecientes al primer símbolo OFDM (símbolo l=0) de una trama llevan el bit s_0 de inicialización.

La modulación absoluta de este bit s_0 de inicialización se deriva de la secuencia de referencia w_k obtenida del generador PRBS usado para suministrar la Información de Referencia a las portadoras piloto Continuas y Dispersas (Véase 4.3.3.2.1), y es como sigue:

$$\begin{aligned} \mathrm{Re}(c_{m,l,k}) &= 2x(1/2 - w_k) \\ \mathrm{Im}(c_{m,l,k}) &= 0 \end{aligned} \tag{4.30}$$

Para la modulación diferencial del resto de portadoras TPS, se aplica la siguiente regla a la portadora de índice k del símbolo l en la trama m:

$$\begin{aligned} &\text{Si el bit } s_l = 0: && \mathrm{Re}(c_{m,l,k}) = \mathrm{Re}(c_{m,l-1,k}) \\ &\text{Si el bit } s_l = 1: && \mathrm{Re}(c_{m,l,k}) = -\mathrm{Re}(c_{m,l-1,k}) \\ &\text{En cualquier caso:} && \mathrm{Im}(c_{m,l,k}) = 0 \end{aligned} \tag{4.31}$$

Como puede verse en (4.30), las celdas TPS se transmiten un nivel de potencia normalizado, es decir, con una energía igual al de la media de todas las celdas de datos, cumpliéndose nuevamente la condición expresada en (4.20) (Véase Fig. 4.27).

En la codificación diferencial, los datos son representados mediante cambios de signo en lugar de amplitudes o niveles. Este hecho hace que la detección sea más fiable ya que se detectan transiciones de cambio de signo y no determinados niveles o amplitudes que pueden ser alterados fácilmente por el canal de transmisión. Además, en sistemas de transmisión complejos es fácil perder la polaridad, es decir, se cambian los 0's por 1's, y si la detección está diseñada de manera que se detecten niveles o amplitudes dicha detección será poco robusta. Con la codificación diferencial esta dificultad no existe ya que las transiciones de cambio de signo siguen siendo las mismas bajo estas condiciones.

La Fig. 4.33 ilustra el proceso de modulación de portadoras TPS. Cada trama OFDM llevará información sobre la transmisión al receptor mediante el uso de los símbolos OFDM. Hay 68 bits de información TPS en un bloque TPS protegido contra errores por lo que se transmitirá un bit en cada símbolo OFDM. El primer símbolo OFDM es el transmite el bit de inicialización, y es en el que se modulan las portadoras TPS mediante una modulación 2-PSK, tal y como se expresa en (4.30). Dicho símbolo OFDM sirve de referencia para el resto de símbolos OFDM modulados en DBPSK, tal y como se expresa en (4.31), según el valor del bit TPS a transmitir por todas las portadoras de dicho símbolo.

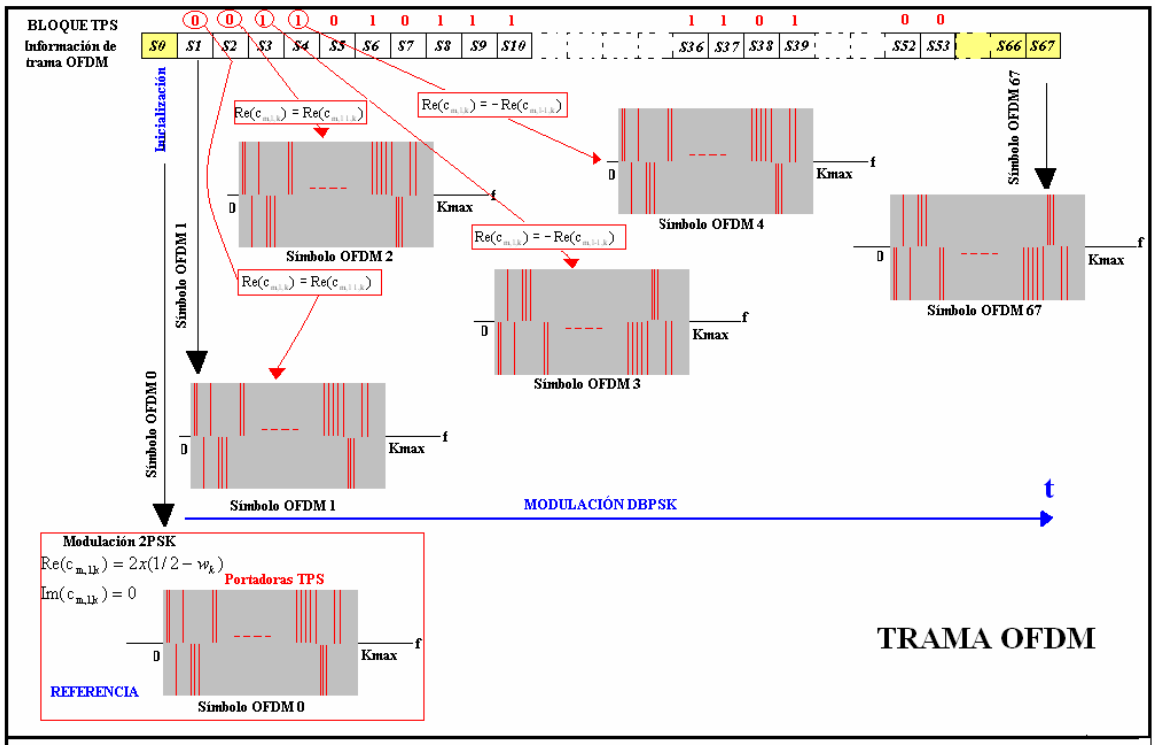


Fig. 4.33 Algoritmo de modulación DBPSK del bloque TPS para la transmisión

La función de Matlab implementada que realiza la modulación de las portadoras TPS, a partir de un bloque de información TPS protegido contra errores, es la denominada *tps_con_info.m*.

En el apartado 6.3.3 se explica el proceso de demodulación DBPSK y decodificación BCH para la obtención de la información de los parámetros de transmisión.

4.3.4.3 Código implementado

```
function [data_ref,data_ref_ant] =  tps_con_info(data_ref,data_ref_ant, n,alpha,D,C,modo,O_index,
S,tps,S_index,SO,T,const);
%-------------------------------------------------------------------------------------------
%   tps_con_info..m
%     Ver ETSI 300.744 v1.51 4.6.2
%-------------------------------------------------------------------------------------------
fprintf('·Tx: Adición y modulacion de información a las portadoras TPS del simbolo OFDM %g de la trama OFDM
%g de la supertrama OFDM %g',S_index,O_index,SO);
fprintf('.');
%-------------------------------------------------------------------------
% Generacion del bloque tps                    Solo hace falta una vez por trama pero bueno....
%-------------------------------------------------------------------------
bloque_tps=tps_info(n,alpha,D,C,modo,O_index);
%-------------------------------------------------------------------------
%Obtencion del bit de datos tps para las portadoras
%-------------------------------------------------------------------------
diferencial=bloque_tps(S_index);             Ésta es la info a modelar para las portadoras tps (
                                             misma info )
fprintf('.');
%-------------------------------------------------------------------------
%      Modulacion tps (DBPSK)
%-------------------------------------------------------------------------
if S_index~=1%Las portadoras TPS del símbolo ya   Las portadoras tps ya vienen moduladas en BPSK. Solo
vienen moduladas en 2_PSK del bloque de           falta aplicarles el diferencial (menos a las del primer
referencias.m                                     simbolo OFDM, que llevan inicializacion)
   pt=1;                                          apuntador a la primera posición de la portadora piloto
                                                  tps
 for u = 1:T
  if pt <= length(tps)
    t = 1+tps(pt);
  else
    t = 0;
  end
  if u == t;                                      Señal tps
    if diferencial==0                             Si se va a transmitir un 0...
      data_ref(u)=data_ref_ant(u);                Dejamos el mismo signo de cada portadora tps que la
                                                  anterior
      tps_actual(u)=data_ref(u);
      tps_anterior(u)=data_ref_ant(u);
    else%Si se va a transmitir un 1...
      data_ref(u)=-data_ref_ant(u);               Cambiamos el signo a cada portadora tps respecto al
                                                  anterior
     tps_actual(u)=data_ref(u);
     tps_anterior(u)=data_ref_ant(u);
    end
  pt = pt + 1;                                    VER ETSI 300.744 v1.51 4.6.3 pag 34
  end
 end
```

```
41 if const==1
42 figure
43 stem(tps_actual);
44 hold on;
45 stem(tps_anterior,'r');
46 title(sprintf('Modulacion DBPSK de las portadoras TPS
   del simbolo ODFM %g\nTransportan un bit %g de
   informacion',S_index,diferencial))
47 xlabel('Portadoras
   tps'),ylabel('Amplitud'),legend('Simbolo actual','Símbolo
   anterior');
48 hold off
49 end
50 data_ref_ant=data_ref;                    Nos guardamos el anterior simbolo para conocer su
                                             signo de modulacion
51 else
52 data_ref_ant=data_ref;                    Nos guardamos el anterior simbolo para conocer su
                                             signo de modulacion
53 end
54 fprintf('.\tOK\n');
```

Código 14 tps_con_info.m

```
1  function [data_ref,data_ref_ant] =  tps_con_info(data_ref,data_ref_ant, n,alpha,D,C,modo,O_index,
   S,tps,S_index,SO,T,const);
2  %-------------------------------------------------------------------------------------------
3  %   tps_con_info..m
4  %   Ver ETSI 300.744 v1.51 4.6.2
5  %-------------------------------------------------------------------------------------------
6  %-----------------------------------------------------------------
7   % Generación del bloque de info TPS (68 simbolos tps            Ver ETSI EN 300744 v1.5.1 apartado 4.6.2: TPS
   = 1 bloque TPS)                                                 transmission format, pag 31
8   %-----------------------------------------------------------------
9   %Se tranmsite 1 simbolo TPS en cada simbolo OFDM
   (se repite en sus  distintas portadoras TPS)
10  bloque_tps=zeros(1,68);                                        preparamos vector que contiene la info codificada del
                                                                   bloque TPS+señalizacion
11  s=zeros(1,54);                                                 preparamos vector que contiene la info del bloque
12  %INICIALIZACIÓN
13  s(1)=1;
14  %PALABRA DE SINCRONIZACIÓN                                     Aquí irá el código PRBS..
15  %Sabemos que cada supertrama OFDM = 4 tramas OFDM = 4 bloques TPS
16  if O==1||O==3
17  s(2:17)=[0 0 1 1 0 1 0 1 1 1 1 0 1 1 1 0];
18  else                                                           Sincro para la 1ª y 3ª trama OFDM dentro de una
                                                                   supertama
19  s(2:17)=[1 1 0 0 1 0 1 0 0 0 0 1 0 0 0 1];
20  end                                                            Sincro para la 2ª y 4ª trama OFDM dentro de una
                                                                   supertama
21  %INDICADOR LONGITUD DE BITS DE
   INFORMACIÓN USADOS
22   s(18:23)=[0 1 0 1 1 1];                                       Número 23 en binario. Indica que no se transmire info
                                                                   sobre el tipo de ID
```

```
%NÚMERO DE TRAMA OFDM DENTRO DE LA SUPERTRAMA OFDM
switch O
    case 1
        s(24:25)=[0 0];
    case 2
        s(24:25)=[0 1];
    case 3
        s(24:25)=[1 0];
    case 4
        s(24:25)=[1 1];
end
%CONSTELACIÓN
switch n
    case 4
        s(26:27)=[0 0];
    case 16
        s(26:27)=[0 1];
    case 64
        s(26:27)=[1 0];
end
%INFORMACIÓN JERÁRQUICA
switch alpha
    case 1
        s(28:30)=[0 0 1];
    case 2
        s(28:30)=[0 1 0];
    case 4
        s(28:30)=[0 1 1];
end
%CODIFICACIÓN FLUJO HP (para transmisión jerarquica). Nos da igual
switch C
    case 1/2
        s(31:33)=[0 0 0];
    case 2/3
        s(31:33)=[0 0 1];
    case 3/4
        s(31:33)=[0 1 0];
    case 5/6
        s(31:33)=[0 1 1];
    case 7/8
        s(31:33)=[1 0 0];
end
%RELACIÓN DE CODIFICACIÓN FLUJO LP (lo consideramos igual que HP)
switch C
    case 1/2
        s(34:36)=[0 0 0];
    case 2/3
        s(34:36)=[0 0 1];
    case 3/4
        s(34:36)=[0 1 0];
    case 5/6
        s(34:36)=[0 1 1];
    case 7/8
        s(34:36)=[1 0 0];
end
%INTERVALO DE GUARDA
switch D
    case 1/32
        s(37:38)=[0 0];
    case 1/16
        s(37:38)=[0 1];
    case 1/8
        s(37:38)=[1 0];
    case 1/4
        s(37:38)=[1 1];
end
%ESTANDAR DE TRANSMISIÓN
switch modo
    case 2
        s(39:40)=[0 0];
    case 8
        s(39:40)=[0 1];
end
%RESERVADOS
```

Si se transmite info de CELL ID ponemos [0 1 1 1 1 1] (31 en binario). Lo consideramos indiferente...

Dejamos a 0 los últimos bits del vector s -->(s(41:54)). Lo estan de la inicializacion

```
97  %CODIFICACIÓN BCH.PROTECCION DE ERRORES
98  N=127;                                          Longitud de salida
99  K=113;                                          Longitud de entrada
100 s_ext=zeros(1,K);
101 s_galois=zeros(1,K);
102 s_ext(61:K)=s(2:54);
103 s_galois=gf(s_ext);                             Añadimos los zeros delante del bloque de info tps (sin
                                                    señalizacion)
104 [polinomio_BCH,T] = bchgenpoly(N,K);
105 code_tps = bchenc(s_galois,N,K);                 Demuestra que los errores corregibles son 2 y que el
                                                    polinomio generador es x14+x9+x8+x6+x5+x4+x2+x+1.
                                                    Solo curiosidad
106 %FORMACIÓN DEL BLOQUE TPS                       Codificación BCH(127,113,2)
107 bloque_tps_galois=[s(1),code_tps(61:N)];
108 %el vector recibido esta en formato de Galois (cada bit                Formamos el bloque tps con el codigo BCH (sin los 60
    es un array...).Para operar normalmente reescribimos el              0's del principio) y el bit de inicializacion
    vector en formato "normal"
109 for u=1:68
110     if(bloque_tps_galois(u)==1) bloque_tps(u)=1;
111     else bloque_tps(u)=0;
112     end
113 end
```

Código 15 tps_info.m

4.3.5 Modulación OFDM e inserción del intervalo de guarda

4.3.5.1 Localización del bloque en el sistema

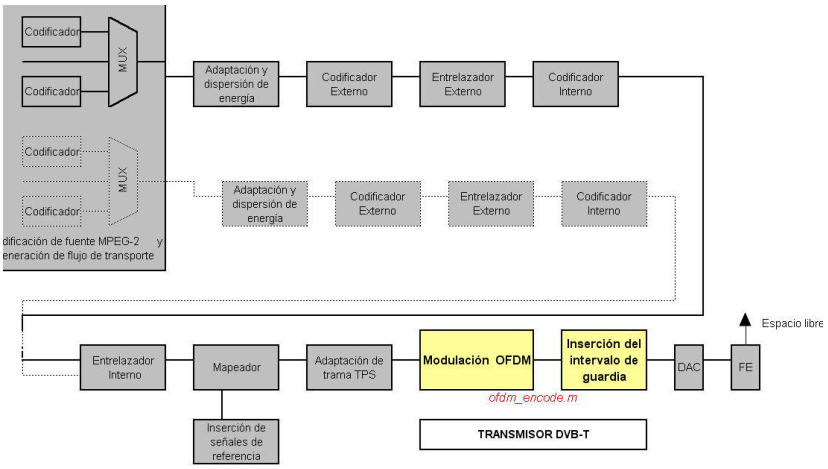


Fig. 4.34 Localización del bloque de modulación OFDM

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *ofdm_encode.m*. Realiza el proceso de modulación OFDM descrito en el apartado 3.4.2, con algunas modificaciones en su estructura impuestas por el estándar.

4.3.5.2 Objetivo y relación con el estándar

A continuación se muestra el diagrama de bloques del modulador OFDM determinado por la estructura del DVB-T.

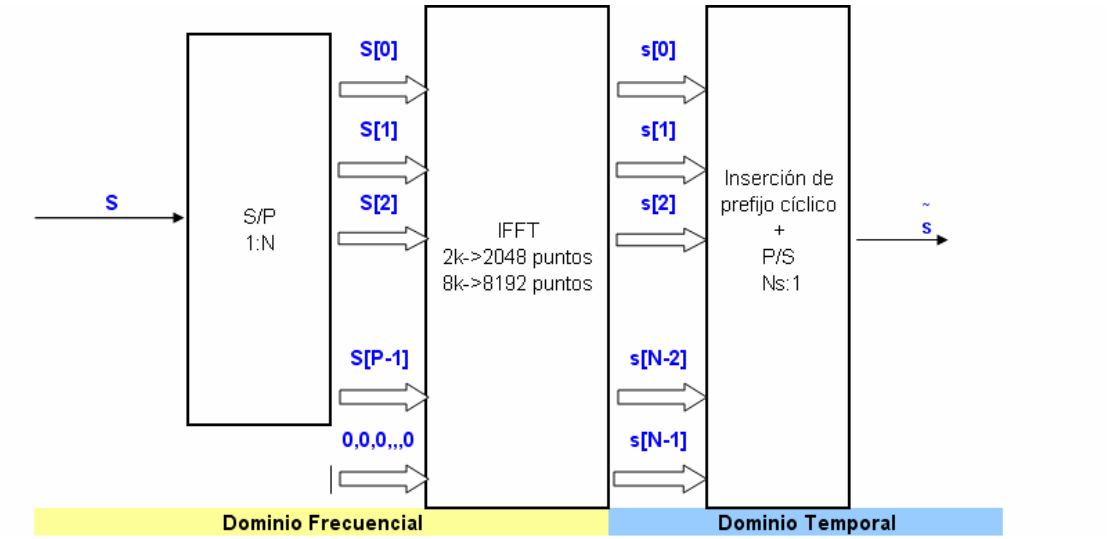


Fig. 4.35 Diagrama de bloques del sistema modulador OFDM empleado

A la entrada del bloque S/P se encuentran los datos en serie modulados según el esquema de modulación utilizado (QPSK, 16-QAM o 64-QAM). La función de este bloque

es la de separar los datos puestos en serie, como si de una señal de portadora única se tratara, para crear las portadoras del símbolo OFDM. El número de datos de entrada al bloque S/P depende del número de portadoras que transmite el símbolo OFDM en el modo de transmisión utilizado (N=2048 en modo 2k y N=8192 en modo 8k).

La definición de un símbolo OFDM de datos indica que el número de portadoras útiles es de P=1705 en modo 2k y P=6817 en modo 8k. Por este motivo, se sitúa el vector columna de datos S en el centro de un vector de N muestras y se rellenan los extremos con ceros. De esta manera la IDFT de los datos en paralelo se puede implementar de manera eficiente con la IFFT, ya que 2048 y 8192 son enteros múltiplos de 2. Mediante este bloque IFFT se transforman los datos discretos del dominio frecuencial al dominio temporal discreto, según la expresión:

$$IDFT[S] = s[n] = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} S[n] e^{j \cdot \frac{2\pi i n}{N}} \quad \text{con } 0 \le n \le N-1 \tag{4.32}$$

Cada símbolo queda modulado por la frecuencia portadora $e^{j \cdot \frac{2\pi n}{N}}$ con n=0,...,N-1, de manera que a la salida del bloque IFFT cada portadora tiene repartida la información de cada símbolo entrante, ya que cada símbolo temporal s[n] se forma a partir de todos los símbolos entrantes S[n].

A continuación las muestras temporales resultantes son tratadas de manera que su duración temporal aumenta un cierto intervalo de tiempo relativo al símbolo, llamado Intervalo de guarda (Véase Tabla 4.12), dando paso a la señal temporal discreta $\tilde{s}[n]$ de longitud $Ns=N_s+N_{PC}$. Este intervalo de guarda estará formado por un prefijo cíclico, obtenido a partir de copiar un determinado número de muestras del final del símbolo temporal, acorde con la longitud del intervalo de guarda, al inicio del mismo. Este prefijo cíclico sirve para hacer que la señal sea más robusta frente a ISI en las muestras con contenido de información útil, siendo las muestras del prefijo cíclico las que se ven afectadas por ISI (por dispersión del canal y ecos recibidos).Véase apartado 3.5.2 para más información.

El hecho de utilizar prefijo cíclico tiene un inconveniente. Para cada bloque de N portadoras se añaden N_{PC} portadoras copia, hecho que se traduce en la reducción de la tasa de transmisión (4.23), siendo ésta de:

$$\tilde{F}_T = F_T \cdot \frac{N}{N + N_{PC}} \tag{4.33}$$

El último bloque del modulador OFDM, el bloque P/S, es el que pone en serie todas las muestras de $\tilde{s}[n]$ que le llegan en paralelo para crear el símbolo OFDM que se transmitirá por el canal.

4.3.5.3 Código implementado

```
1  function [data_ofdm] =ofdm_encode(data_in,D, fft_mode,modo, T,S)
2  if modo~=0
3  fprintf('·Tx: Realización de IFFT de %g puntos para el símbolo %g de modo %gK (%g portadoras) con intervalo
   de guarda de %g',fft_mode,S,modo,T,D);
4  fprintf('.');
5  end
6  long_guard=fft_mode*D;                          Longitud del intervalo de guarda una vez se haya
                                                   realizado la ifft
7  %-------------------------------------------------------------------------
8  % Realizamos IFFT de fft_mode puntos (2048 o 8192)
9  %-------------------------------------------------------------------------
10  data_freq = zeros(fft_mode,1);
11  data_freq(1+(fft_mode-T+1)/2:(fft_mode+T+1)/2) =
   data_in;                                        Preparamos el vector donde irá la ifft de nuestra señal
12  data_time =ifft(fftshift (data_freq));          El simbolo OFDM en el centro del vector al que aplicar
                                                   la ifft
13  fprintf('.');                                   Realiza la ifft con origen en el centro del vector
14  %-------------------------------------------------  Hemos multiplicado por fft_mode ya que la función no
                                                   tiene en cuenta la energia de la modulacion!
15  % Aplicamos el intervalo de guarda
16  %-------------------------------------------------
17   data_ofdm = [ data_time(1+fft_mode-
   long_guard:fft_mode) ; data_time ];
18  if modo~=0                                      Copiamos la ultima parte del simbolo al inicio
19  fprintf('.\tOK\n');
20  end
```

Código 16 ofdm_encode.m

Capítulo 5 MODELADO DE CANAL

Este capítulo presenta las características del canal empleado para la transmisión de datos DBV-T, que es un canal particular al tratarse de un enlace sin hilos. Tras presentar las características se procede a detallar las que genera a la señal transmitida y los problemas con los que se encontrará el receptor a la hora de tratar los datos. Paralelamente se presentan explicaciones referentes al modelado de canal realizado a la hora de implementar el simulador DBV-T.

Se debe comentar que el estudio y el modelado de canales inalámbricos no ha sido uno de los objetivos del proyecto. Por este motivo se añaden explicaciones referentes a los diversos fenómenos físicos del entorno de transmisión que afectan al canal desde un punto de vista introductorio. Para poder estudiar e implementar un bloque de la estimación de canal se ha modelado distintos entornos intentando caracterizar el número máximo de fenómenos. Aún así, el grado de exactitud del canal modelado no es una prioridad del proyecto puesto que, realmente, el interés reside en obtener distintos modelos de canal para observar el comportamiento del bloque estimador en diferentes áreas de transmisión. Es decir, el interés ha residido en obtener diferentes tipos de canales sin dar relevancia a qué entornos pertenecen.

5.1 INTRODUCCIÓN A LOS CANALES SIN HILOS

Desde la transmisión hasta la recepción, la señal transmitida experimenta un conjunto de cambios debido a diversos fenómenos que tienen lugar durante la propagación debidos a las particularidades del canal utilizado. Por este motivo, para poder una buena caracterización del enlace DBV-T se debe analizar y cuantificar los cambios producidos en la señal debidos a diferentes fenómenos físicos mediante un modelado de canal lo más realista posible. En el caso de canal sin hilos, la señal viaja por un medio de propagación libre no guiado, hecho que comporta que este estudio sea más complicado. De todos modos, es imposible calcular detalladamente los efectos que experimentan estas señales para un camino de propagación entre dos puntos cualesquiera ya que la mayoría de casos los parámetros de entorno no están disponibles o los cálculos son demasiado complejos. Normalmente, para modelar el canal se parte de la estadística y de los modelos empíricos. En casos donde la complejidad y la variabilidad del canal inalámbrico dificulta el hecho de obtener un modelo determinista del canal y son más adecuados para modelos estadísticos. La atenuación causada por los objetos que se encuentra la señal, tales como edificios y árboles se caracterizan estadísticamente. A la atenuación se le deben sumar efectos de reflexiones, difracciones y dispersiones.

Los entornos *indoor* o de interior tienden a ser más irregulares que los entornos *outdoor*, o de exterior ya que sus características dieléctricas cambian drásticamente según

cual sea el entorno interior (fábrica abierta, oficina, habitación con elementos metálicos...). En estos casos la mejor solución de modelado está basada en modelados computacionales.

A continuación se detallan las características principales que se debe tener en cuenta para modelar un canal inalámbrico.

5.1.1 Enlaces LOS, NLOS y Propagación Multicamino

Un canal de radio puede ser clasificado como canal de visión directa, LOS (*Line of Sight*), o como canal sin visón directa, NLOS (*Non Line of Sight*) [13] .

El canal LOS se puede definir como el canal radio con línea de visión directa entre una estación emisora y una estación receptora. En este tipo de canales la señal viaja a través de un camino directo, sin ningún obstáculo que dificulte la propagación de la señal. Sin embargo, los canales NLOS son los canales radio que no tienen visión directa entre ambas estaciones. En estos enlaces la señal transmitida viaja hasta la estación receptora a través de reflexiones y difracciones siguiendo diferentes caminos de propagación (propagación multicamino), mientras que la componente directa no llega al receptor o llega demasiado atenuada. En una oficina, por ejemplo, las reflexiones se producen en las paredes y el mobiliario).

En la propagación multicamino las señales reflejadas, al haber recorrido caminos diferentes a la señal directa, llegarán al receptor con un cierto desfase temporal y a su vez con un nivel de potencia y fase diferente a la señal original. Como consecuencia de la propagación multicamino, en un escenario inalámbrico aparecen dos fenómenos que limitan la calidad y capacidad del enlace: el desvanecimiento de la señal y distorsión de la señal.

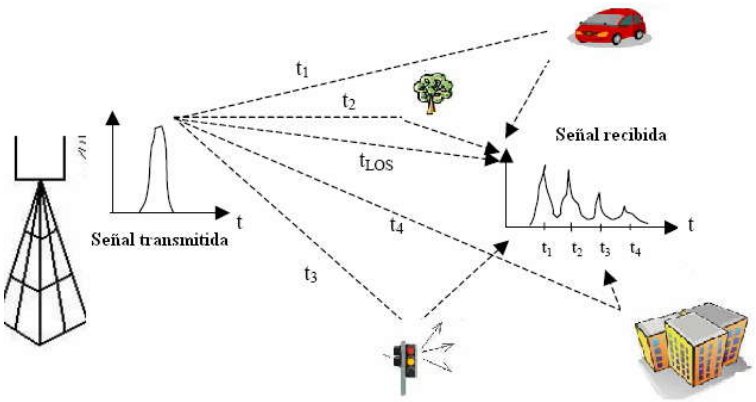


Fig. 5.1 Propagación multicamino

Si el transmisor, el receptor y los objetos u obstáculos reflectores están inmóviles, entonces el impacto de las señales multicamino recibidas y sus respectivos retardos en el tiempo serán fijos. Pero si el transmisor y/o el receptor no son móviles las características de las componentes multicamino variarán temporalmente. En ambos casos es imprescindible el uso de modelos estadísticos para caracterizar la señal recibida ya que el número de réplicas de la señal transmitida no es cuantificable.

5.1.2 Dispersión de canal

La dispersión de canal o *delay spread* se produce debido a que los caminos NLOS son más largos que el camino directo, LOS. Como consecuencia, éstas señales alcanzan al receptor mucho más tarde que la señal LOS, es decir, las señales viajan con distinto retardo. Entonces, la dispersión de canal se define como la máxima diferencia de tiempo en alcanzar el receptor entre la señal directa y la última señal multicamino y es función del entorno de transmisión [15].

Con el objetivo de poder comparar diferentes canales con propagación multicamino se utilizan parámetros que permiten cuantificar la propagación multicamino de dichos canales. El *rms delay spread* es uno de ellos, y se calcula como el promedio espacial o temporal de las respuestas impulsionales consecutivas obtenida en un escenario concreto.

Análogamente al *rms delay spread* del dominio temporal, el ancho de banda de coherencia *Bc* es utilizado para caracterizar el canal en el dominio de la frecuencia. El *delay spread* y el ancho de banda de coherencia son inversamente proporcionales entre ellos. En el caso de que el ancho de banda de nuestra señal sea mas grande que el ancho de banda de coherencia, $Bc << BW$, se obtiene un canal en banda ancha, en el caso contrario, $Bc >> BW$, se obtiene un canal en banda estrecha.

5.1.3 Variabilidad temporal del canal

El ancho de banda de coherencia es un parámetro que describe la dispersión temporal del canal en un escenario concreto. Sin embargo, este parámetro no nos ofrece ningún tipo de información sobre la variabilidad temporal del canal debido al movimiento tanto del receptor móvil como de la antena transmisora. Tanto la frecuencia Doppler como el tiempo de coherencia son parámetros que describen la variabilidad temporal del canal para un área determinada.

A partir de la frecuencia Doppler, f_D, se puede obtener una medida del ensanchamiento espectral causado por la variación del canal y se define como el rango de frecuencias sobre el cual el espectro Doppler es de valor diferente a 0. El cálculo de la frecuencia Doppler se puede realizar de la siguiente manera:

$$f_D = \frac{v}{\lambda}\cos\sigma \tag{5.1}$$

siendo σ el ángulo de incidencia.

El tiempo de coherencia, T_C, es la medida temporal que nos indica durante cuánto tiempo la respuesta impulsional del canal es invariante. La frecuencia Doppler y el tiempo de coherencia son inversamente proporcionales:

$$T_C \approx \frac{1}{f_D} \tag{5.2}$$

Siendo *Ts* el periodo de símbolo OFDM, en el caso en que $Tc << Ts$ los símbolos no se verán distorsionados debido a la variación temporal del canal. En el caso contrario, $Tc >> Ts$, el tiempo de coherencia del canal será inferior a la duración del símbolo, el canal variará durante la transmisión de la información y causará distorsión en el receptor. Este hecho hace que la BER (*Bit Error Rate*) aumente debido a la aparición de ISI e ICI, disminuyendo la calidad del enlace.

5.2 MODELADO DE CANAL

Cuando una fuente de información intenta transmitir hacia un destino, los datos son convertidos a una señal habilitada para ser enviada de transmisor a receptor a través del canal. El propio canal modifica la señal en diferentes aspectos que son impredecibles para el receptor. Por este motivo el receptor debe ser diseñado de manera que pueda regenerar la señal para ser capaz de extraer los datos con el menor número de errores o distorsiones posible. Este razonamiento se extrapola a todo tipo de canales, con o sin hilos.

El modelado de canal es una buena herramienta para poder estudiar los efectos de determinados canales sobre la señal y poder evaluar y modificar métodos para su regeneración a partir de la estimación de estos efectos.

5.2.1 Modelado de canal de distorsión multicamino

Se supone un canal discreto con un número finito de componentes multicamino donde la señal multicamino está formada por un número relativamente pequeño de componentes reflejadas en obstáculos de entornos abiertos. Este canal se puede modelar como un sistema lineal variante en el tiempo. Si hay M componentes multicamino discretas, la salida del canal consistirá en la suma de las M versiones de la señal transmitidas y atenuadas [15]:

$$y(t) = \sum_{k=1}^{N(t)} \alpha_k(t) \cdot e^{-j\phi_n(t)} \cdot s(t - \tau_k(t)) \tag{5.3}$$

donde α_k, φ_k y τ_k(t) representan los coeficientes de atenuación complejos, la fase y el retardo de la k-ésima multiportadora en el tiempo t, $s(t)$ representa a la señal transmitida y $N(t)$ representa el número de componentes multicamino que llegan al receptor en el instante t. La respuesta impulsional del canal en el instante de tiempo t para un impulso dado en el instante de tiempo t-τ, es:

$$h(\tau, t) = \sum_{k=1}^{N} \alpha_k(t) \cdot e^{-j\phi_n(t)} \cdot \delta(t - \tau_k(t)) \tag{5.4}$$

5.2.1.1 Modelado de canal mediante línea de retardos TDL

El modelado de canales radio utilizando estructuras TDL (*Tapped Delay Line*) con coeficientes variantes en el tiempo, tal y como muestra la Fig. 5.2, ayuda a comprender las distorsiones provocadas por componentes reflejadas con diferentes retardos de propagación que son réplicas atenuadas y desfasadas de la señal transmitida.

Para modelar el canal mediante TDL se parte de unos valores de atenuación, fase y retardo medidos de cada réplica de la señal transmitida, también llamada *tap*. La Tabla 5.1 recoge los valores de estos parámetros (no incluye fase, la cual ha sido agregada de forma aleatoria en las realizaciones presentadas en el documento) obtenidos a partir del estudio del resultado de la realización de medidas en diferentes entornos de tres ciudades de Finlandia [17]. El sistema de medida estaba formado por un receptor móvil OFDM y hasta tres antenas transmisoras. La tabla únicamente contempla réplicas de hasta -30 dB respecto de la señal LOS, ésta última normalizada a 0 dBW de potencia y con un retardo de 0 μs.

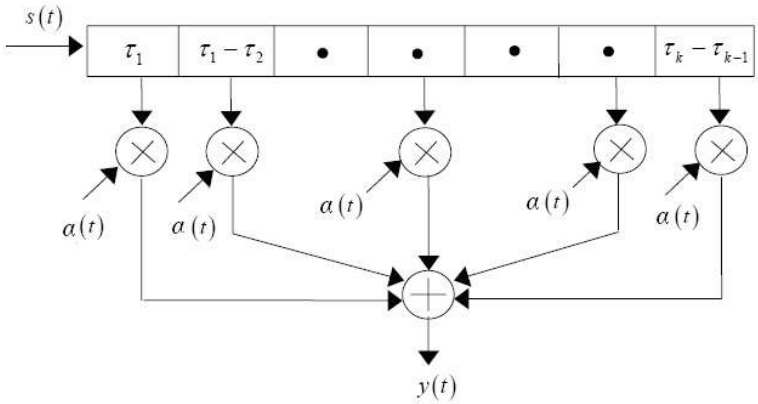


Fig. 5.2 Modelo TDL para canales discretos multicamino con retardos variables

Tabla 5.1 Coeficientes de modelos TDL de diferentes entornos

	Autopista 1 transmisor		Indoor 2 transmisores		Zona urbana 3 transmisores	
TAP	Retardo (µs)	Potencia (dB)	Retardo (µs)	Potencia (dB)	Retardo (µs)	Potencia (dB)
1	0	0	0	0	0	0
2	0,1125	-5,56	0,1125	-0,47	0,1125	-5,56
3	0,225	-8,78	0,3375	-2,11	0,225	-7,94
4	0,325	-10,01	0,55	-4,02	0,325	-8,71
5	0,4375	-11,57	0,6625	-4,2	0,55	-10,53
6	0,55	-12,66	0,9875	-6,58	0,7625	-11,3
7	0,6625	-14,12	1,2125	-8,48	0,875	-11,82
8	0,7625	-15,17	1,425	-9,26	1,1	-13,43
9	0,875	-16,34	1,975	-13,09	1,2	-14,799
10	0,9875	-17,41	2,3	-15,49	1,75	-20,42
11	1,1	-18,86	2,525	-17,45	2,3	-19,83
12	1,2	-19,32	2,7375	-18,41	3,5	-24,24
13	1,3125	-20,01	3,175	-22,22	9,74	-27,06
14	1,425	-21,46	4,7125	-25,25	10,175	-21,31
15	1,5375	-22,46	7,9875	-6,75	10,3875	-19,27
16	1,6375	-23,12	8,425	-4,88	10,725	-19,32
17	1,75	-23,73	8,5375	-4,55	11,1625	-21,04
18	1,8625	-23,41	8,65	-4,53	11,8125	-21,71
19	1,975	-23,82	8,975	-5,64	12,25	-20,97
20	2,075	-23,04	9,0875	-5,57	12,6875	-19,84
21	2,1875	-23,76	9,4125	-8,25	13,125	-21,27
22	2,3	-26,42	9,85	-11,91	13,45	-23,58
23	2,4125	-28,86	10,175	-14,84	18,4875	-25,51
24	2,5125	-29.71	10.5	-18.13	19,0375	-21.08

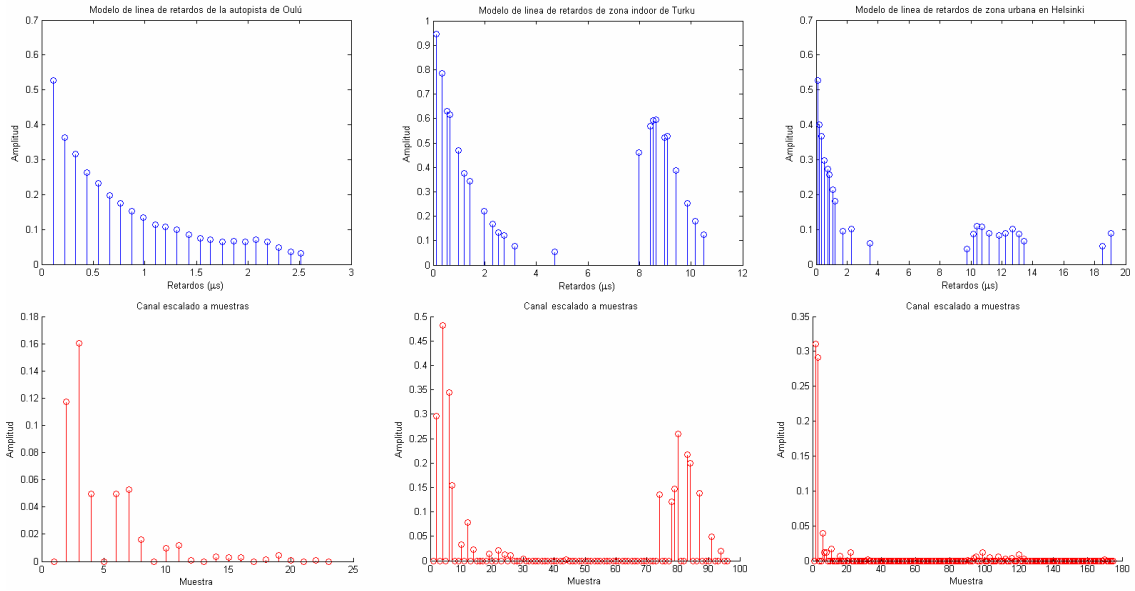


Fig. 5.3 Modelos TDL de diferentes entornos en tiempo y escalados a muestras OFDM

El proceso para obtener las realizaciones Rayleigh (Véase anexo D) de canal partiendo de los coeficientes presentados en la Tabla 5.1 ha sido el siguiente:

1. Determinación del periodo de cada muestra, T_m a partir del tiempo de símbolo T_s y el número de muestras del símbolo (Ver Tabla 4.12).
2. Determinación del retardo de los *taps* en muestras a partir del periodo de muestra obtenido en el punto 1. Esto es debido a que los *taps* tienen retardos no uniformes por lo que se puede dar el caso que más de un *tap* afecte a una misma muestra temporal de la señal transmitida. Se define la longitud del canal en muestras y el intervalo de participación de cada tap sobre las muestras OFDM.
3. Realización Rayleigh de cada *tap*. Se generan dos muestras gausianas (fase y cuadratura) de media 0 y desviación típica, σ, igual a la mitad de potencia lineal recibida en cada *tap* (D.4). A partir de aquí se obtiene el módulo de la envolvente Rayleigh, (D.3) y se le añade la información de fase (dicha información de fase no viene tabulada en la Tabla 5.1, sus valores se han obtenido aleatoriamente)
4. Una vez obtenidas las realizaciones Rayleigh de cada *tap* en módulo y fase, se determina la fase y amplitud total que afecta a cada muestra de la señal OFDM. Para ello se descompone cada realización Rayleigh en coordenadas cartesianas (a+j·b) y se procede a sumar la contribución (amplitud y fase) de dichas realizaciones de los *taps* según los intervalos de muestras obtenidos en el punto 2 del proceso.
5. Finalmente, se pasan los datos por el canal a partir de la multiplicación de ambas respuestas en el dominio frecuencial mediante el uso de la herramienta FFT.

La Fig. 5.3 ilustra el inicio y el final de este proceso de realización Rayleigh para los modelos TDL de la Tabla 5.1. Los gráficos en azul representan las amplitudes (en módulo) y los retardos de los respectivos taps. Los gráficos en rojo representan las amplitudes (módulo) de realizaciones Rayleigh de los *taps* de los modelos anteriormente citados escalados a muestras de la señal OFDM.

Se ha implementado la función *modelo_canal.m* que se ocupa de realizar todos los procesos comentados en este apartado: realización Rayleigh del canal y paso de los datos por el mismo. Para simular el efecto de variación de canal en el tiempo se añade un parámetro de simulación que permite definir su longitud temporal en relación al número de símbolos OFDM que se procesan (Véase anexo A). Es decir, cada cierto número de símbolos OFDM, especificado por el usuario, procesados a través del bloque *modelo_canal* se obtiene una nueva realización del canal.

Tal y como se ha comentado al inicio del capítulo, el objetivo principal del proyecto ha sido implementar un sistema de transmisión que utiliza un esquema de modulación OFDM, en este caso el DVB-T, para poder estudiar e implementar un conjunto de herramientas que permitan regenerar la señal, entre otras un estimador de canal para regenerar la señal contaminada por éste. En el caso que ocupa el proyecto el canal es inalámbrico y, por ello, tiene sus particularidades. En este capítulo se ha querido recoger brevemente el conjunto de particularidades, siendo la de más interés la del efecto multicamino. Para poder evaluar el estimador de canal implementado se ha creído conveniente hacerlo con un canal que tuviera esta propiedad, a pesar de que cualquier forma de canal habría servido. Se ha visto que hay técnicas de modelado de canal que parten de medidas empíricas y realizaciones estadísticas. De hecho, hay muchos estudios realizados referentes al modelado de canales, con hilos y sin hilos, de comunicaciones móviles o fijas, por lo que hay mucha teoría implicada. Al final del documento, en el apartado de Referencias, se recoge la bibliografía consultada, [15], [17], [18] y [19], a la que se puede recurrir para ampliar los conocimientos referentes al modelado de canal.

5.2.2 Código implementado

```
function [data_ofdm] =[data_out,canal] =modelo_canal(data_in, C_c, S_c, M, fft_mode, long_canal, canal, Tsimbolo, D, L, v, fc, T, const);
long_guard=fft_mode*D;
if M==0

fprintf('·C: Paso de los datos por canal ideal h(t)=1 ');
fprintf('.');
data_out=data_in;

canal=[];
retardo=[];
fprintf('.');
fprintf('.\tOK\n');
%-------------------------------------------------------------
% Modelo de canal con  TDL no uiforme
%-------------------------------------------------------------
elseif M~=1&&M~=0                                   Rayleigh Oulú Motorway (PAPER "Novel Radio Channel Models For Evaluation Of DVB-H Broadcast Systems" de H.Parvianen)
 if M==2
   fprintf('·C: Paso de los datos por realización de canal en autopista .');
   fprintf('.');
   retardo=[0.1125 0.225 0.325 0.4375 0.55 0.6625 0.7625 0.875 0.9875 1.1 1.2 1.3125 1.425 1.5375 1.6375 1.75 1.8625 1.975 2.075 2.1875 2.3 2.4125 2.5125];   24, en microseg
   potencia=[-5.56 -8.78 -10.01 -11.57 -12.66 -14.12 -15.17 -16.34 -17.41 -18.86 -19.32  -20.01 -21.46 -22.46 -23.12 -23.73 -23.41 -23.82 -23.04 -23.76 -26.42 -28.86 -29.71];   En DBW
   fase =randint(length(retardo),1,361).*rand(length(retardo),1);   No especificado, (en grados)

   K_factor=14.3;                                   Para Rice(en dB)
   if(mod(S_c,long_canal)==0)&& const==1
   figure
   stem(retardo,10.^(potencia/20)','o-');
   title('Modelo de linea de retardos de la autopista de Oulú');
   xlabel('Retardos (\mus)'); ylabel('Amplitud');
   save Modelos_de_Linea_de_Retardos\autopista.mat retardo potencia fase K_factor

   end
 elseif M==3                                        Rayleigh Turku Indoor (PAPER "Novel Radio Channel Models For Evaluation Of DVB-H Broadcast Systems" de H.Parvianen)
   fprintf('·C: Paso de los datos por realización de canal de interior .');
   fprintf('.');
```

```
%-------------------------------------------------------------------
%            Realización de canal tipo Rayleigh
%-------------------------------------------------------------------

Suponemos una frecuencia de muestreo tal que a cada muestra de la señal OFDM e corresponde un solo valor de amplitud de la realización Es decir, la frecuencia de muestreo es Fs=1/Tmuestra.Tsimbolo calculado en carga_parametros.m-->Tiene en cuenta el intervalo de guarda

Tmuestra=Tsimbolo/(fft_mode*(1+D));                 Duración de la muestra en segundos
muestra_afect=ceil(retardo*10^-6./Tmuestra);        Indica a partir de que muestra la señal se verá afectada por la amplitud y la fase de cada tap.Hasta que se inicie otro tap.
dim_canal=max(muestra_afect);                       Miramos la longitud del canal en muestras
if(mod(S_c,long_canal)==0)%%Modelado de canal cada "long_canal" simbolos
canal=zeros(1,dim_canal);                           Vector fila que representa el canal realizado
muestra=1;                                          llevará la cuenta de las muestras
tap=1;                                              llevará la cuenta de los taps que tendra el canal
 while muestra<dim_canal;                           Sumamos las amplitudes complejas que afectan a cada muestra
 pot_lin=10^(potencia(tap)/10);                     potencia del tap en lineal
 sigma=pot_lin/sqrt(2);                             Valor de desviación tipica en función del valor cuadratico medio (potencia en lineal)
 muestra_gaus_real=randn(1,1)*sigma;                Muestra gausiana fase (media 0 y desviación típica sigma)
 muestra_gaus_imag=randn(1,1)*sigma;%Muestra gausiana cuadratura
                                                    Realizacion Rayleigh (módulo) normalizado
Rayl_mod=sqrt(muestra_gaus_real.^2+muestra_gaus_imag.^2)/sqrt(2);
                                                    Añadimos la fase del tap
Rayl(muestra)=Rayl_mod*exp(j*2*pi*(fase(tap)*pi/180));

 amplitud=real(Rayl(muestra))+j*imag(Rayl(muestra));  Valor de la amplitud del canal en complejo

   if muestra_afect(tap)==muestra
   canal(muestra)=canal(muestra)+amplitud;          Modelamos el canal muestra a muestra
   tap=tap+1;                                       Apuntamos al siguiente tap
   else
   muestra=muestra+1;
   end
 end
end
   if(mod(S_c,long_canal)==0)&& const==1
   figure
   hold on
   stem((abs(canal)),'ro-');
   title(sprintf('Canal %g escalado a muestras',C_c));
   xlabel('Muestra'); ylabel('Amplitud');
   end
   fprintf('.\tOK\n');
%-------------------------------------------------------------------
Paso por canal en frecuencia. Pasamos los datos a dominio frecuencial
%-------------------------------------------------------------------
if M~=0
   data_canal=(zeros(fft_mode,1));
   data_canal((1+(fft_mode-T+1)/2:(fft_mode+T+1)/2))=ofdm_decode(data_in,D, fft_mode,0, T,0);   Paso por canal en frecuencia. Pasamos los datos a dominio frecuencial
   C_freq=fft(canal,fft_mode);                      Obtenemos el canal en el dominio frecuencial
   C=(zeros(1,fft_mode));
   C(1+(fft_mode-T+1)/2:(fft_mode+T+1)/2)=C_freq(1:T);
   data_modelo_canal_out=data_canal.*(C.');
                                                    Volvemos al dominio temporal
data_out=ofdm_encode(data_modelo_canal_out((1+(fft_mode-T+1)/2:(fft_mode+T+1)/2)),D, fft_mode,0, T,0);

end
```

Código 17 modelo_canal.m

5.3 RUIDO ADITIVO GAUSSIANO BLANCO

Además de los efectos que sufre la señal debidas a las características del canal (vistas anteriormente), el canal también puede estar afectado por ruido aditivo blanco gaussiano, AWGN (*Additive White Gaussian Noise*), que puede considerarse uno de los factores más limitantes en el comportamiento de un sistema.

AWGN afecta a cada símbolo transmitido de manera independiente. El término aditivo se refiere a que el ruido es añadido a la señal y que no hay mecanismos multiplicativos involucrados.

Si el canal se considera ideal expresamos la respuesta impulsional del canal como $h_{ID}[n]=\delta[n]$. Si el canal es ideal no se introduce ISI por lo que la longitud del prefijo cíclico no tiene relevancia respecto a las prestaciones del sistema. Además, si se desprecia la contribución de ruido, el sistema tiene una SNR infinita. En la práctica, hasta el modelo más simple de canal radio tiene modelada la presencia de ruido AWGN, tal y como se muestra en la Fig. 5.4.

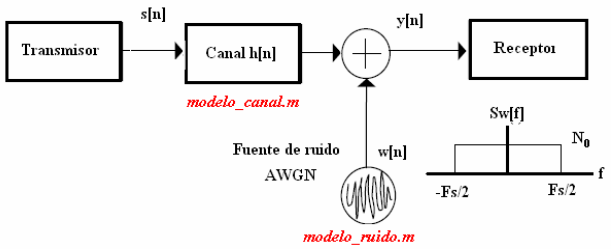


Fig. 5.4 Modelo de canal AWGN

La señal a la entrada del receptor, y[n], se puede expresar como:

$$y[n] = s[n] * h[n] + w[n] \tag{5.5}$$

donde *s[n]* es la señal transmitida, *h[n]* s la respuesta impulsional del sistema y *w[n]* representa al ruido AWGN.

Para generar este ruido aleatorio, gausiano y blanco de media 0 y varianza 1 se ha hecho servir la función *randn* de Matlab. Para controlar la potencia de ruido de la señal compleja *w[n]* generada se ha usado la relación de energía de bit frente a energía de ruido, E_b/N_0 (para estudiar el comportamiento del sistema en función de las modulaciones QPSK, 16-QAM y 64-QAM a nivel de bits), y la relación energía de símbolo frente a energía de ruido, E_s/N_0 (para estudiar el comportamiento del sistema en función de la energía del símbolo PAM con indiferencia del número de bits de su modulación).

La relación señal a ruido, SNR, se expresa como:

$$SNR = \frac{P_{OFDM}}{P_N} \tag{5.6}$$

La potencia de ruido se obtiene aplicando la integral en todas las frecuencias de la densidad de potencia de ruido, siendo ésta de:

$$P_N = N_0 \cdot F_S \tag{5.7}$$

con F_S denotando la frecuencia de muestreo. El periodo de muestreo T_S es inversamente proporcional a F_S y a la vez, en la implementación realizada, coincide con el periodo de cada muestra de la señal OFDM. Por ello, siendo T el período de símbolo OFDM, se cumple la relación:

$$\frac{T}{T_S} = N \tag{5.8}$$

siendo N el número de muestras de la señal OFDM (coincidente con el número de puntos de la IFFT del transmisor DVB-T, 2048 para modo 2k y 8192 para modo 8k).

Así, la SNR a la entrada del receptor y la potencia de ruido siguen la siguiente relación:

$$SNR = \frac{P_{OFDM}}{P_N} = \frac{E_{OFDM} \big/ T}{N_0 F_s} = \frac{E_{OFDM}}{N_0}\frac{T_s}{T} = \frac{E_{OFDM}}{N_0}\frac{1}{N} \tag{5.9}$$

Si se modela la potencia de ruido a partir de una potencia de señal OFDM de valor $P_{OFDM}=N$ (ya que el símbolo OFDM incluye N señales PAM unitarias) y se reserva la variable E_s/N_0 como parámetro de simulación, siguiendo la relación (5.9) la potencia de ruido que se debe aplicar a $w[n]$ sigue las siguientes relaciones:

$$P_N = \sigma^2{}_w = N\frac{N}{E_{OFDM}\big/N_0} = N\frac{N}{N\cdot E_s\big/N_0} = \frac{N}{E_s\big/N_0} = \frac{N}{n\cdot E_b\big/N_0} \tag{5.10}$$

siendo E_{OFDM} la energía del símbolo OFDM, E_S la energía de cada símbolo PAM, E_b la energía de cada bit del símbolo PAM y n el número de bits de cada símbolo PAM.

Se puede obtener el valor de potencia de ruido complejo, $w[n] = w_r + j\cdot w_i$, siguiendo el siguiente razonamiento:

$$\sigma^2{}_w = E\sigma^2{}_{wr} = \left[\left|w[n]\right|^2\right] = E\left[\left|wr[n]\right|^2\right] + E\left[\left|wi[n]\right|^2\right] = \sigma^2{}_{wr} + \sigma^2{}_{wi} \overset{\sigma^2{}_{wr}=\sigma^2{}_{wi}}{=} 2\cdot\sigma^2{}_{wr}.$$
$$\sigma^2{}_{wr} = \frac{\sigma^2{}_w}{2}; \qquad \sigma^2{}_{wr} = \frac{\sigma^2{}_w}{2} \tag{5.11}$$

Una vez obtenida la relación para el factor de escala que se aplicará a cada componente compleja del vector de ruido gausiano blanco de media 0 y varianza 1 (w_r y w_i), dicho vector de ruido se genera del siguiente modo:

$$w[n] = \sqrt{\frac{P_N}{2}}\cdot(w_r + j\cdot w_i) \tag{5.12}$$

5.3.1 Código implementado

```
function [data_out] =modelo_ruido(data_in,R,EBN0dB,n,fft_mode,ESN0dB);
%-------------------------------------------------------------------
%               CANAL SIN RUIDO
%-------------------------------------------------------------------
if (R==0)
fprintf('·C: Paso de los datos por canal sin ruido');
fprintf('.');
data_out=data_in;
fprintf('.');
fprintf('.\tOK\n');
end

%-------------------------------------------------------------------
%               CANAL RUIDOSO  Ruido gaussiano
%-------------------------------------------------------------------
if (R==1)
   if ESN0dB~=1                                  Si es 0 usamos energía de bit
   fprintf('·C: Paso de los datos por canal ruidoso
modelado con EBN0=%gdB .',EBN0dB)
   fprintf('.');
   EBN0lin=10^(EBN0dB/10);                       Definición de EB/N0 (SNR a la salida)
   Pn=(1/fft_mode)*(1/((EBN0lin*log2(n))));      Cálculo de la energia del ruido
   [a,b]=size(data_in);
   Pnnr=randn(a,b)*sqrt(Pn/2);
   Pnni=randn(a,b)*sqrt(Pn/2);
   Pnnt=Pnnr+j*Pnni;
   data_out=data_in+Pnnt;
   fprintf('.');
   else                                          Aquí los valores de EBN0 se refieren a valores de
                                                 energía de simbolo frente N0
   fprintf('·C: Paso de los datos por canal ruidoso
modelado con ESN0=%gdB .',EBN0dB)
   fprintf('.');
   EBN0lin=10^(EBN0dB/10);                       Definición de ES/N0 (SNR a la salida)
   Pn=(1/fft_mode)*(1/((EBN0lin)));              Cálculo de la energía del ruido
   [a,b]=size(data_in);
   Pnnr=randn(a,b)*sqrt(Pn/2);
   Pnni=randn(a,b)*sqrt(Pn/2);
   Pnnt=Pnnr+j*Pnni;
   data_out=data_in+Pnnt;
   fprintf('.');
   end
fprintf('.\tOK\n');
end
```

Código 18 modelo_ruido.m

Capítulo 6 ARQUITECTURA DEL RECEPTOR DVB-T IMPLEMENTADO

En el Capítulo 4 se ha presentado la arquitectura del transmisor DVB-T basado en las especificaciones marcadas por su estándar [6]. Dicho estándar no recoge la arquitectura del sistema receptor por lo que su diseño queda abierto a la competencia de mercado. De hecho, la gran mayoría de estándares son de transmisión. Por este motivo, los grandes avances e innovaciones están destinados a los sistemas receptores.

En este capítulo se pretende recoger la arquitectura del sistema receptor implementado de manera que permite la recepción óptima de los datos. Básicamente, el receptor está implementado por bloques que, en su mayoría, son complementarios a los del sistema transmisor. Sin embargo, el receptor debe hacer frente a diversas dificultades a la hora de reconstruir la información recibida ya sea debido a fenómenos físicos provocados por el canal inalámbrico como por desajustes de sincronización entre el transmisor y receptor.

Se continúa con la misma estructura utilizada en el capítulo 4 en la que se presenta un bloque del sistema receptor en cada apartado, identificando sus objetivos, su funcionamiento y el código implementado en cada caso. En los apartados de los bloques que son totalmente complementarios a los bloques del transmisor no se hará hincapié en una explicación de su funcionamiento. Al tratarse del sistema receptor, en este capítulo se recogerán las simulaciones utilizadas para valorar las prestaciones del bloque o sistema y evaluar el correcto funcionamiento del mismo (Véase Anexo B).

6.1 INTRODUCCIÓN

En el diagrama de bloques del sistema receptor que ilustra la Fig. 6.1 se destacan tres etapas principales: estimación y corrección de parámetros sincronización, demodulación OFDM y estimación de canal y, para finalizar, decodificación de canal. A continuación se presentan estas etapas como grandes bloques.

La zona coloreada en verde muestra los bloques implementados en el simulador, que para la banda del receptor parte de la señal que ha sido afectada por las características del canal inalámbrico (Véase Capítulo 5), que ya ha sido trasladada a banda base y muestreada, cuantificada. El tiempo de muestreo es tal que cada portadora o símbolo PAM es de muestra única, es decir, la longitud de los vectores OFDM es igual al número de portadoras que se envían por símbolo OFDM (según modo de transmisión y longitud de intervalo de guarda).

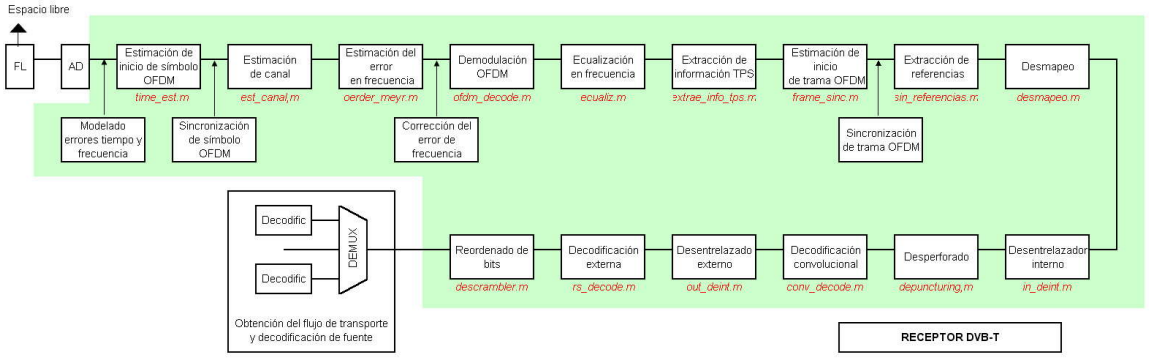


Fig. 6.1 Diagrama de bloques del sistema receptor DVB-T implementado

A partir de aquí se asume que cada bloque procesa una unidad de transporte entera de manera secuencial: para el sistema OFDM se tratará de un símbolo OFDM detrás de otro, para el sistema MPEG se tratará de un bloque codificado de 204 bytes o bien de un bloque MPEG de 188 bytes. Para conocer el tipo de entrada a cada bloque y su dimensión véase Fig A.1.

6.2 ESTIMACIÓN Y SINCRONIZACIÓN

COFDM es una técnica prometedora que tiene muchas ventajas que no tiene el sistema de modulación de portadora única como puede ser su gran robustez frente al efecto multicamino. En contra, tiene ciertas desventajas como puede ser su gran sensibilidad a los errores de sincronización.

La sincronización es un aspecto muy importante en los sistemas OFDM debido a que se trata de un sistema de modulación multiportadora. Cualquier offset o desajuste de frecuencia o de fase entre la portadora en transmisión y la portadora en recepción causa interferencias entre portadoras, ICI. Se destacan tres aspectos principales de sincronización en los sistemas OFDM [21]:

- La sincronización de tiempo de símbolo: determina la muestra correcta de inicio de símbolo antes de la demodulación mediante FFT.
- La sincronización de frecuencia de portadora: se utiliza para eliminar la frecuencia offset causada por el desajuste entre los osciladores locales del transmisor y el receptor. Se trata de un efecto típico de los canales inalámbricos como del desplazamiento Doppler (en caso de canales inalámbricos móviles).
- La sincronización del reloj de muestreo: se utiliza para mitigar los errores del reloj de muestreo debido al desajuste de los osciladores de cristal.

Todos estos errores de sincronización degradan significativamente el comportamiento del sistema.

6.2.1 *Estimación de inicio de símbolo OFDM* (estimación gruesa)

La sincronización de inicio de símbolo OFDM (también llamado sincronización de tiempo del símbolo del sistema) [20] tiene como objetivo encontrar el instante de tiempo en el que empieza el símbolo OFDM que significa, en términos de procesado digital, encontrar la posición en la que centrar la ventana de datos correspondiente a un símbolo OFDM. Cuando las señales son transmitidas a través de condiciones de canal multicamino severas donde además hay influencia de ruido AWGN y de desplazamiento Doppler, es importante resolver primero la sincronización de tiempo de símbolo.

Hoy en día existen muchas técnicas basadas en la estimación del inicio de símbolo OFDM y pueden ser clasificadas en dos categorías:

1. Técnicas basadas en las portadoras piloto de las señales OFDM y en símbolos piloto.
2. Técnicas basadas en la estructura inherente del símbolo OFDM: uso del prefijo cíclico (*Unique Word* [22])

Todos los algoritmos de estimación que se pueden englobar en las dos categorías presentadas anteriormente tienen sus ventajas e inconvenientes. Algunos de ellos tienen una gran complejidad y otros tienen poca eficiencia. Todo depende de la aplicación a la que esté destinada la modulación OFDM. Algunos de ellos son adecuados para modos de transmisión continuas, como el DAB (*Digital Audio Broadcasting*) o el DVB-T, otros son adecuados únicamente para modos de transmisión a ráfagas (*WLANs*).

Puesto que se ha implementado un algoritmo de transmisión basado fielmente en el estándar DVB-T, se ha decidido implementar un estimador de inicio de símbolo OFDM que utiliza una técnica basada en las portadoras piloto de las señales OFDM, de la primera categoría presentada anteriormente. De este modo se puede evaluar además la estructura de portadoras piloto y comprobar que a partir de ellas se puede estimar este parámetro.

6.2.1.1 Localización del bloque en el sistema

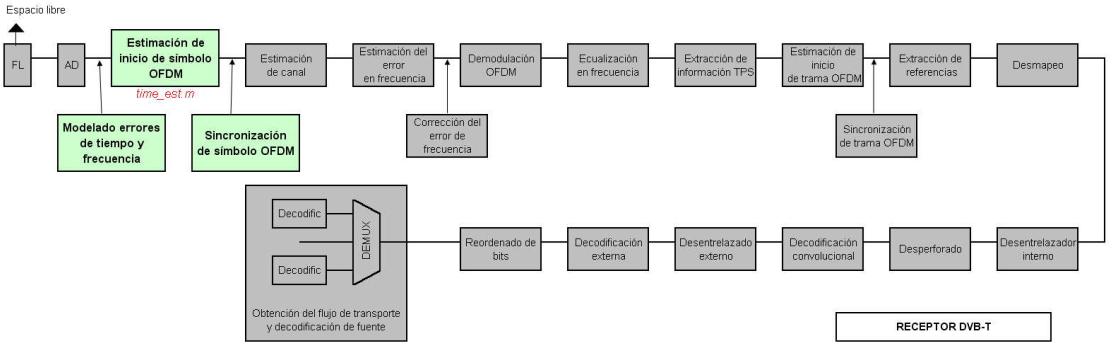


Fig. 6.2 Localización del bloque de estimación de inicio de símbolo.

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A.1), es *time_est.m*.

6.2.1.2 Algoritmo propuesto

El algoritmo implementado para estimar el inicio de símbolo OFDM es el que se presenta en el documento [20]. Únicamente las portadoras piloto son utilizadas para la estimación gruesa de inicio de símbolo. Con este algoritmo se pretende mantener un buen comportamiento del sistema aún cuando la SNR de la transmisión sea baja y las condiciones de la dispersión multicamino sean desfavorables.

El razonamiento parte del modelo de señal en el transmisor, cuya expresión de la *n*-ésima muestra de un símbolo tras el bloque IFFT, tal y como se presenta en (3.33), es:

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k] e^{j\cdot\frac{2\pi i n}{N}} \quad \text{con } 0 \le n \le N-1 \tag{6.13}$$

donde *X[k]* representa al símbolo de datos modulado en la *k*-ésima portadora. Para evitar la interferencia debida al multicamino, las últimas N_{PC} muestras de la salida del bloque IFFT son copiadas y añadidas al inicio de cada símbolo en forma de intervalo de guarda (prefijo cíclico). Consecuentemente, el símbolo OFDM transmitido se puede expresar, en concordancia con (3.36), como:

$$s[n] = \frac{1}{N}\left(\sum_{n=-N_{PC}}^{-1}\sum_{k=0}^{N-1} X[k] e^{j\cdot\frac{2\pi i (n+N)}{N}} + \sum_{n=0}^{N-1}\sum_{k=0}^{N-1} X[k] e^{j\cdot\frac{2\pi i n}{N}} \right) \tag{6.14}$$

La respuesta impulsional del canal multicamino puede ser denotada aproximadamente, tal y como expresa (5.4), como:

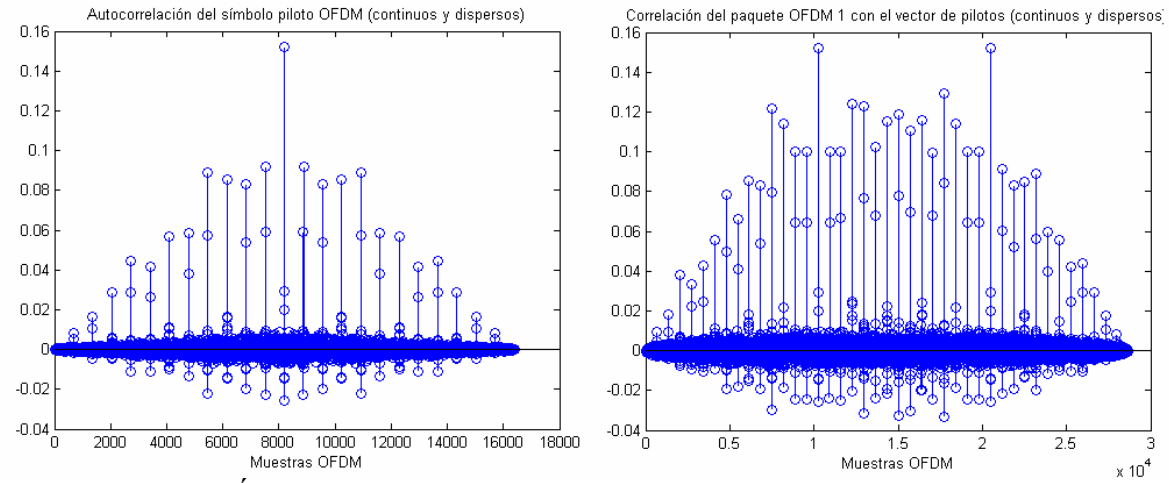
$$h_t = \sum_{i=1}^{N} \alpha_i \cdot \delta(t-\tau_i) \tag{6.15}$$

donde τ_i representa el retardo del *i*-ésimo *tap*, *M* es el número total de *taps* y α_i es el valor complejo discreto en el tiempo del canal. Entonces, la secuencia recibida se puede expresar como:

$$y_n = \frac{1}{N}\sum_{k=0}^{N-1} X[k] e^{j\cdot\frac{2\pi i n}{N}} \cdot H_k \cdot e^{j\cdot(\frac{2\pi i \xi}{T_U} + \sigma_0 + 2\pi\varepsilon T)} + n(nT_S) \tag{6.16}$$

donde H_k representa la función de transferencia de la *k*-ésima portadora, T_U, T_S, y *T* representan la duración temporal de los datos útiles, el tiempo nominal de muestreo t la duración total de un símbolo OFDM recibido respectivamente, *n(t)* representa el ruido AWGN y ξ, ε y θ_0 representan el offset de tiempo del símbolo (que es el parámetro a estimar), el offset de la frecuencia portadora y la rotación de fase respectivamente.

El método propuesto [20] utiliza todas las portadoras piloto integradas en los símbolos OFDM (TPS, dispersas y continuas), véase apartado 4.3.2, debido a que tienen buenas propiedades de autocorrelación. La Fig. 6.3 ilustra esta propiedad de autocorrelación. En (a) se muestra la correlación de un símbolo OFDM transmitido en modo 8k que contiene únicamente pilotos (portadoras de datos puestas a 0) con él mismo. En (b) se muestra la correlación de dos símbolos OFDM transmitidos en modo 8k, con portadoras de datos y pilotos, con el símbolo OFDM con únicamente portadoras piloto.



(a)Únicamente pilotos *(b) Datos y pilotos*

Fig. 6.3 Propiedades de correlación de las portadoras piloto en ausencia de ruido y efecto multicamino

La estimación del parámetro ξ, offset de tiempo de símbolo, se basa en la correlación del símbolo OFDM recibido, y_n, con una secuencia conocida por el receptor, la secuencia de portadoras piloto P_n, que viene definida por el estándar DVB-T. Entonces, la posición de inicio del símbolo OFDM se determina mediante:

$$\hat{\xi} = \max_{\xi} \mathrm{Re}[IFFT(P_n)] \otimes \mathrm{Re}[y_n] \qquad (6.17)$$

en el que se entiende que la posición de inicio de símbolo es la posición que hace que la convolución, denotada como $\otimes$, es máxima.

La sincronización que emplea técnicas de uso del prefijo cíclico del símbolo OFDM basadas en la correlación no tiene grandes prestaciones puesto que el prefijo cíclico se encuentra contaminado por el ISI introducido por el canal multicamino. Únicamente las últimas muestras del prefijo cíclico pueden mantenerse inalteradas por dicho efecto, reduciendo así sus propiedades de correlación (aún cumpliendo la condición de que el intervalo de guarda sea mayor que la respuesta impulsional del canal).

En el caso que ocupa este proyecto se ha supuesto que el error de inicio de símbolo OFDM es producido a un retardo en la recepción OFDM. Se ha supuesto que el receptor, al inicializarse, ha empezado a capturar y muestrear señal temporal OFDM que, como ocurre en la realidad, no tiene porque empezar en el inicio de un símbolo.

Para simular este efecto, se ha hecho uso de un *buffer* (líneas 316-322 del código principal situado en el apartado A.5) que almacena los datos OFDM que se reciben. El usuario de la simulación introduce un error ξ en el fichero de parámetros de simulación, que es directamente el número de muestras retardadas del símbolo OFDM. En este tipo de simulaciones en las que se encuentra presente un error de tiempo, ξ, se envía un símbolo inicial llamado "símbolo piloto" que es el primero que se almacena en el *buffer* de datos. Para simular el efecto de retardo éste último no es almacenado completamente en el *buffer*, sino que es almacenado a partir de la muestra número ξ. Una vez almacenada la parte del "símbolo piloto" se inicializa todo el sistema de transmisión y se empieza la transmisión de datos. Los símbolos, al llegar al *buffer* se almacenan detrás de la porción de símbolo piloto. En cuanto se ha almacenado el primer símbolo de datos empieza la estimación. La Fig. 6.4 ilustra este proceso de modelado de error de inicio de símbolo OFDM.

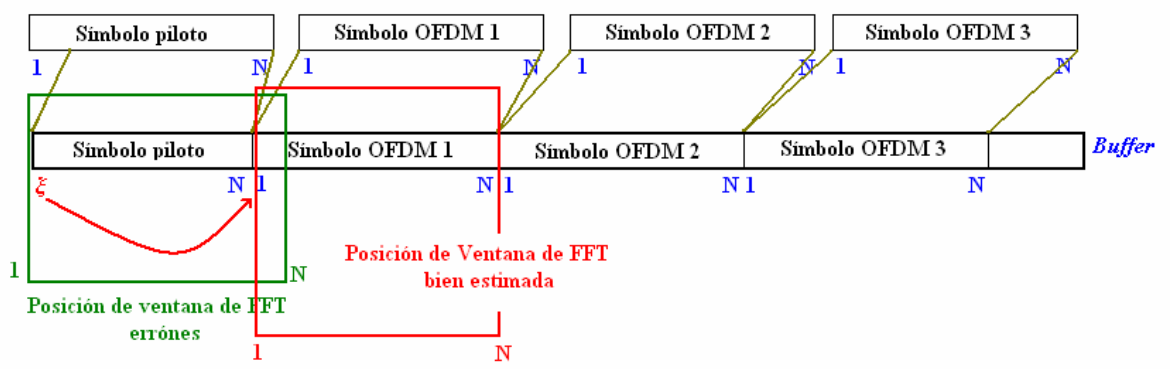
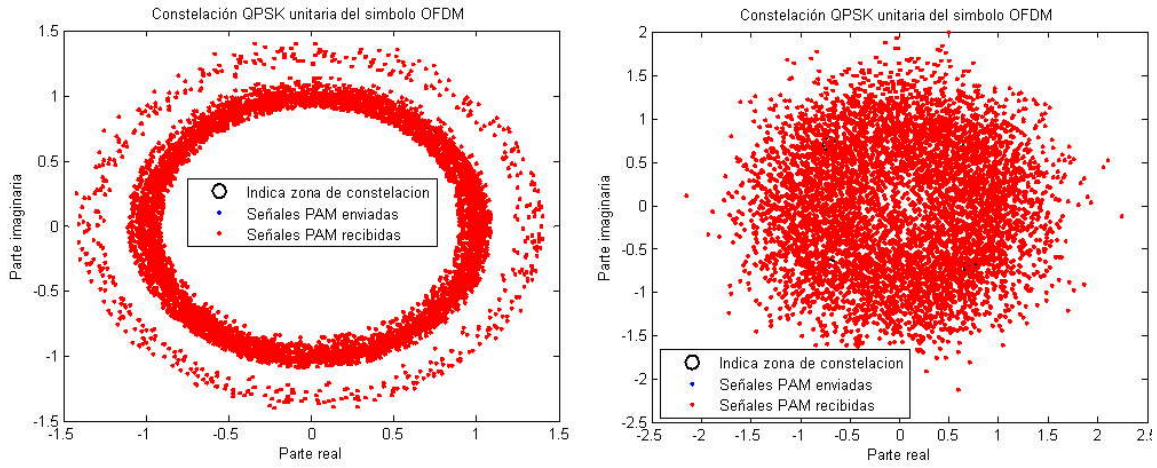


Fig. 6.4 Modelado del error de inicio de símbolo OFDM

El uso del símbolo piloto nos permite añadir unas muestras de offset en el *buffer* del cual el receptor recoge símbolos OFDM con una ventana de N muestras. Éste método facilita el hecho de poder comparar todos los datos transmitidos y recibidos y obtener así un resultado de BER (ya que el símbolo piloto no se incluye en esta comparación).

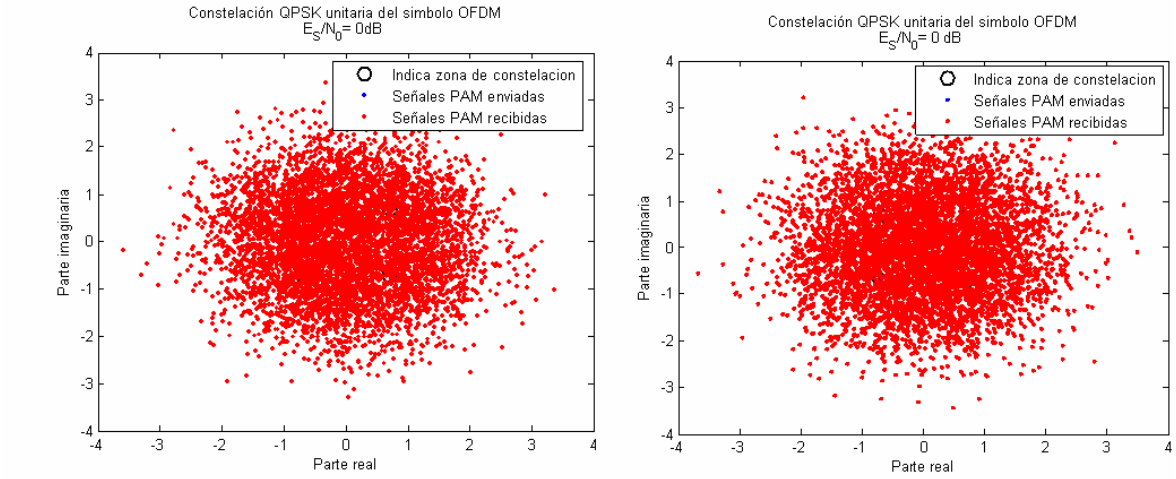
La Fig. 6.5 ilustra la variación de la señal debido a los errores de tiempo de símbolo que no han sido corregidos en el receptor. Por tanto, se ilustra el caso en la que no ha habido estimación de esté retardo ξ y sincronización o bien no ha habido una correcta estimación. En (a) se representa el caso en el que el inicio de la ventana de FFT en una muestra que se encuentra incluida dentro del intervalo de guarda (contaminada por ISI). En (b) se representa el caso en que el inicio de la ventana de FFT, es decir, el inicio de símbolo OFDM se toma en una muestra incluida en la zona libre de ISI. Se observa claramente cuán mala puede ser la constelación que surge tras este error de tiempo. También se refleja perfectamente el hecho de trabajar en el dominio de la frecuencia a la hora de desmapear los símbolos. La variación de la constelación presentada en la Fig. 6.5, sobretodo apreciable en (a), es la típica variación que se obtiene al tener un error de frecuencia en el dominio del tiempo. En este caso, debido a la dualidad entre dominios, esta variación ocurre debido a un error en tiempo.



(a) (b)

Fig. 6.5 Dispersión de la constelación debido al error de tiempo de símbolo OFDM en ausencia de ruido

En la Fig. 6.6 se puede observar el mismo efecto pero ya con presencia del ruido AWGN en la señal recibida.



(a) (b)

Fig. 6.6 Dispersión de la constelación debido al error de tiempo de símbolo OFDM con ESN0=0dB

Una vez estimado el parámetro $\hat{\xi}$ se procede a corregir este error de inicio de símbolo, es decir, se procede a sincronizar en tiempo. Para ello, se sitúa la ventana de FFT de N muestras (las muestras que se transmiten según el modo de transmisión y las muestras dentro del intervalo de guarda) cuya posición de inicio es la estimada $\hat{\xi}$, tal y como se ilustra en la Fig. 6.4. El código de este proceso de corrección se encuentra en la línea 350 del código presentado en el apartado A.3.

Para evaluar las prestaciones que tiene el estimador propuesto implementado se hace uso de la herramienta del cálculo de varianza de estimación (Véase apartado B.2) frente a E_s/N_0. La Para realizar la simulación se han generado una cantidad de bits tal que se han transmitido:

E_P/N_0 (dB)	0	5	10	15	20	25
# Símbolos OFDM	2000	2000	4000	8000	10000	10000

en modo 8 k y modulación QPSK. El error de tiempo introducido ha sido de ξ=100 muestras. La simulación se ha realizado sin presencia de un modelo de canal multicamino.

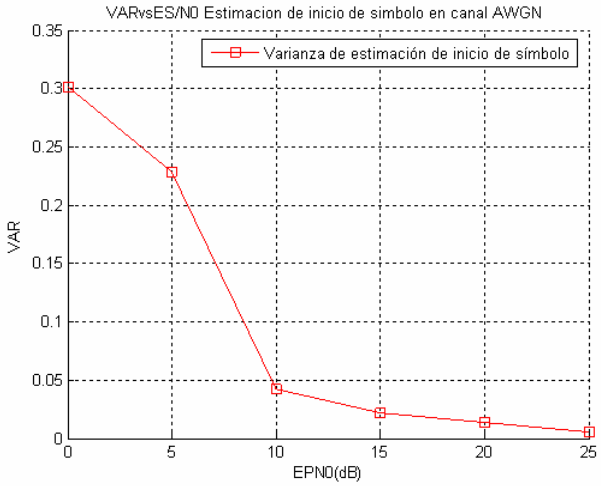


Fig. 6.7 Varianza de estimación de error de inicio de símbolo OFDM frente a E_p/N_0

El tiempo utilizado por el ordenador para realizar la simulación ha sido de 23h y 45 minutos aproximadamente.

Se debe comentar que el gráfico de varianza se ha hecho en relación a la energía de portadora E_p frente a N_0, y no frente a E_S/N_0 para obtener un resultado con igualdad de condiciones de energía. Esto es debido a que la convolución se realiza con un símbolo piloto donde las muestras correspondientes a portadoras de datos están a cero. Por ello, se aplica un factor de escala al valor de E_S/N_0 introducido por el usuario para obtener el vector de ruido AWGN complejo (Véase apartado 5.3) de:

$$\frac{E_P}{N_0} = \frac{E_S \cdot N_{pp}}{N_0} = \frac{n \cdot E_b \cdot N_{pp}}{N_0} \rightarrow \frac{E_S}{N_0}[dB] = \frac{E_P}{N_0}[dB] - 10\log_{10}(N_{PP}) \qquad (6.18)$$

donde N_{pp} representa el número de portadoras piloto utilizadas para la estimación. En sí se está penalizando la potencia del símbolo PAM transmitido debido a que se está aumentando, en relación, la potencia de ruido AWGN.

Para realizar una sincronización precisa y firme del inicio de símbolo OFDM se suele realizar una estimación fina de tiempo en el dominio frecuencial para corregir la rotación de fase de las portadoras tras el bloque FFT de recepción y que es causado también por el offset de tiempo de símbolo. No se contempla ni se ha estudiado este tipo de sincronización en este proyecto.

6.2.1.3 Código implementado

```
function time_error_est=time_est(fft_mode,D,T_portadoras,sintetic,data_canal_out_error, S_c,const);
if S_c==1
fprintf('·Rx: Paso del paquete OFDM piloto por el estimador de error de tiempo para determinar inicio del simbolo OFDM 1');
fprintf('.');
else
fprintf('·Rx: Paso del paquete OFDM %g por el estimador de error de tiempo para determinar inicio de simbolo OFDM %g',S_c-1,S_c);
fprintf('.');
end
%-------------------------------------------------
%   Estimación en tiempo.
%-------------------------------------------------
                                                    Preparamos el vector donde irá la ifft de nuestra señal
sintetic_time=zeros(fft_mode,1);                    sintética
sintetic_time(1+(fft_mode-T_portadoras+1)/2:
(fft_mode+T_portadoras+1)/2) = sintetic;            Se posiciona en el centro del vector al que aplicar la ifft
                                                    Pasamos a tiempo el vector de referencia de pilotos
sintetic_time=ifft(fftshift(sintetic_time));        continuos
long_guard=fft_mode*D;
signal_conv=conv(sintetic_time,data_canal_out_error);  convolucion.Mas información, mejor estimación
fprintf('.');
if const==1
figure
stem((1:length(signal_conv)),signal_conv);
title(sprintf('Correlación del paquete OFDM %g con el vector de pilotos (continuos y dispersos)',S_c));
xlabel('Muestras OFDM'); ylabel ('Valor de correlación');
end
[nada,time_error_est]=max(signal_conv(fft_mode*(1+D)
+1:end));                                           Enventanamos y buscamos la posicion del maximo
                                                    Nos referimos al simbolo anterior a este!! Por eso
                                                    enventanamos a partir de la longitud de un simbolo.
fprintf('.\tOK\n');
fprintf('\t·T_est: Se estima que el inicio del simbolo OFDM %g es en la muestra %g\n',S_c,time_error_est);
```

Código 19 time_est.m

6.2.2 Estimación de canal

En el apartado 5.1 se han visto los efectos negativos que provoca el canal inalámbrico multicamino sobre la señal transmitida. En general, para diseñar un sistema OFDM inalámbrico, se supone que el canal tiene una longitud de respuesta impulsional finita. Una extensión cíclica, de duración mayor que esta respuesta impulsional, es insertada entre símbolos consecutivos con el objetivo de prevenir el ISI y preservar la ortogonalidad de los tonos. En el apartado 3.5.2 se ha estudiado la funcionalidad del prefijo cíclico y se ha determinado que para recuperar la señal transmitida es necesario tener conocimiento del canal, tal y como se expresa en (3.45):

$$\underline{d} = \underline{\underline{\Lambda}}^{-1} \cdot \underline{z} = diag\left\{ \underline{\underline{F}} \cdot \begin{bmatrix} \underline{h} \\ \underline{0} \end{bmatrix} \right\}^{-1} \cdot \underline{z} \tag{6.19}$$

donde $\underline{d}$ representa el vector de símbolos PAM a la entrada del modulador OFDM (transmisión) en el dominio de la frecuencia y $\underline{z}$ representa el vector de símbolos PAM a la salida del demodulador OFDM (recepción), también en el dominio de la frecuencia. En dicho apartado se determina que la matriz $\underline{\underline{\Lambda}}$ es una matriz que contiene la DFT del canal. Por este motivo, una de las grandes ventajas de la modulación OFDM es que resulta muy sencillo ecualizar en el dominio de la frecuencia, siempre y cuando se tenga conocimiento del canal.

En este apartado se aborda el problema de determinar cuál es la respuesta frecuencial del canal a través del cual la señal transmitida ha viajado hasta llegar al receptor. Cabe destacar que hay múltiples métodos para estimar el canal: estimador LS y estimador MMSE, estimador PCMB, estimaciones con comunicación con el transmisor para la ecualización desde transmisión...[27].

El algoritmo estimador que se ha escogido implementar en este proyecto es el estimador LS y se ha decidido por él debido a su fácil implementación. De hecho es uno de los algoritmos de estimación más utilizados.

6.2.2.1 Localización del bloque en el sistema

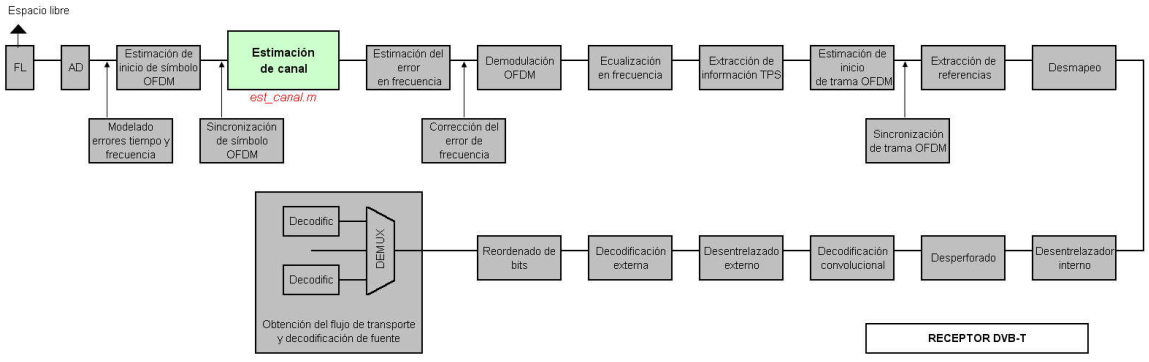


Fig. 6.8 Localización del bloque de estimación de canal.

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A.1), es *est_canal.m.*

6.2.2.2 Algoritmo propuesto. Least Squares.

La estimación de canal permite calcular con precisión las ganancias, véase apartado 5.2, siguientes:

$$H(f_i) = \alpha_i \text{, con } i\text{= 0,1,2,...}N\text{-1} \tag{6.20}$$

que han afectado a la señal recibida.

El algoritmo propuesto, [24] [25], denominado algoritmo de estimación LS (*Least Squares*), consiste en que el transmisor, una vez sincronizado en tiempo de símbolo con el receptor, envíe una señal OFDM piloto conocida por el receptor a través del canal y que el receptor determine las ganancias expresadas en (6.20) a partir del procesado de esta señal de prueba conocida. Se trata de minimizar la diferencia entre la señal esperada (la conocida) y la señal recibida (la conocida pero contaminada por el canal).

Se considera un símbolo piloto enviado y afectado por el canal, tal y como indica la siguiente expresión:

$$Y[i] = X[i]\cdot H[i] + N[i] \text{, con } i\text{= 0,1,2,...}N\text{-1} \tag{6.21}$$

donde *X[i]* es la DFT de la señal esperada *x[i]*y *N[i]* es la DFT del ruido AWGN. El algoritmo LS minimiza:

$$\sum (Y[i] - H[i]X[i])^2 = \left\| \underline{Y} - \underline{\underline{X}}\,\underline{H} \right\|^2 = \left\| \underline{Y} - \underline{\underline{H}}\,\underline{X} \right\|^2 \tag{6.22}$$

donde $\underline{\underline{X}}$ es una matriz son las muestras de la señal conocida en su diagonal:

$$\underline{\underline{X}} = \begin{pmatrix} X_0 & 0 & \dots & 0 \\ 0 & X_1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & X_{N-1} \end{pmatrix} \tag{6.23}$$

Por tanto, si la función de coste a minimizar es su error cuadrático medio, (6.22), el canal estimado $\hat{H}_{LS}$ a partir del algoritmo LS es:

$$\hat{H}_{LS} = \min \left\| \underline{Y} - \underline{\underline{X}}\,\underline{H} \right\|^2 = \min \left[(\underline{Y} - \underline{\underline{X}}\,\underline{H})^H \cdot (\underline{Y} - \underline{\underline{X}}\,\underline{H}) \right] = \min \left[(\underline{Y}^H - \underline{H}^H\,\underline{\underline{X}}^H) \cdot (\underline{Y} - \right. \tag{6.24}$$

donde, si se desarrolla la expresión (6.24), se obtiene:

$$\hat{h}_{LS} = \min \left[(\underline{Y}^H\,\underline{Y} - \underline{Y}^H\,\underline{\underline{X}}\,\underline{H} - \underline{H}^H\,\underline{\underline{X}}^H\,\underline{Y} + \underline{H}\,\underline{H}^H\,\underline{\underline{X}}\,\underline{\underline{X}}^H \right] \tag{6.25}$$

donde $\underline{Y}$ es la señal recibida (esperada pero contaminada por el canal) en el dominio frecuencial. Para minimizar la función de coste expresada en (6.25) se realiza la derivada de dicha función respecto de $\underline{H}^H$ y se iguala a 0 para encontrar el mínimo:

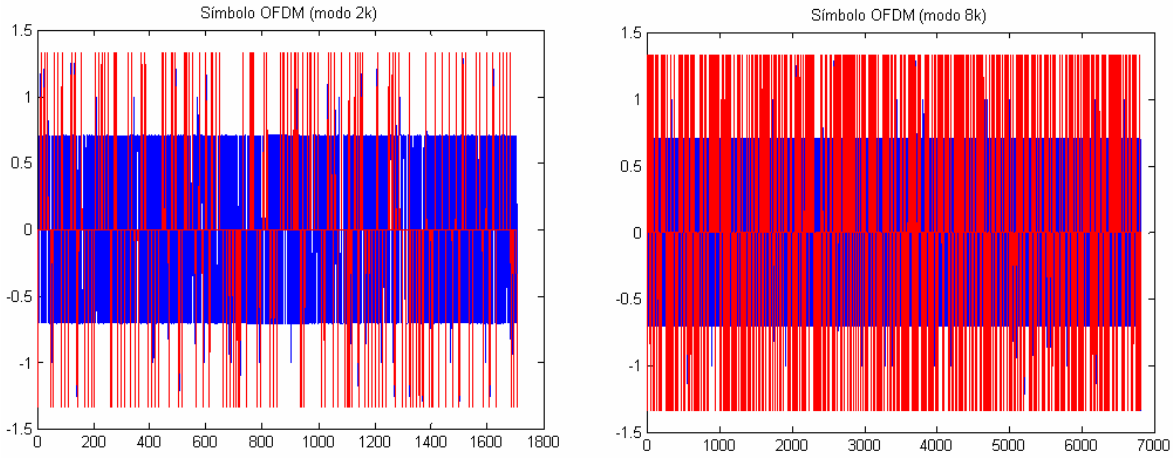
$$\frac{\partial \left\| \underline{Y} - \underline{\underline{X}}\,\underline{H} \right\|^2}{\partial \underline{h}^H} = -\underline{\underline{X}}^H \underline{Y} + \underline{\underline{X}}^H \underline{\underline{X}}\,\hat{\underline{H}}_{LS} = 0 \tag{6.26}$$

Aislando $\hat{H}_{LS}$ de (6.26), se obtiene que la respuesta frecuencial del canal estimado mediante LS cumple la siguiente expresión:

$$\hat{\underline{H}}_{LS} = \underline{\underline{X}}^{-1} \underline{Y} = \underline{\Lambda} \tag{6.27}$$

En el caso de utilizar DVB-T como capa física de transporte no hace falta enviar una señal piloto conocida para estimar el canal. Debido al posicionamiento de las portadoras piloto continuas y dispersas (Véase apartado 4.3.3) no es necesario utilizar un símbolo piloto entero. Lo que se suele hacer es estimar el canal en el dominio frecuencial en aquellas posiciones en las que se encuentran las portadoras continuas y dispersas y hacer uso de técnicas de interpolación para predecir el comportamiento frecuencial del canal en aquellas posiciones donde no ha habido estimación. Se utilizan dichas portadoras piloto debido a que sus posiciones y modulaciones son conocidas por el receptor (ya que son dictadas por el estándar DVB-T). Nótese que el bloque estimador de canal se encuentra en una posición del diagrama de bloques del sistema en el que se trata la señal en el dominio temporal. Es por ello que para realizar la estimación de canal se debe transformar la señal de pilotos esperada al dominio frecuencial y realizar el algoritmo con la señal de pilotos conocida (ya en dominio frecuencial).

En la Fig. 6.9 se presenta la estructura de las portadoras que contiene un símbolo OFDM. En (a) se presenta el caso para un símbolo OFDM en modo 2k y en (b) se presenta para el cado de modo 8k. Las portadoras en color azul son las portadoras de datos y las portadoras piloto TPS. Las portadoras en rojo son las portadoras utilizadas para estimar el canal, es decir, continuas y dispersas. Obsérvese la diferencia de número de portadoras entre las de un modo u otro.



(a) Modo 2k (b) Modo 8k

Fig. 6.9 Estructura de las portadoras piloto continuas y dispersas de un símbolo OFDM usadas para la estimación de canal.

En el caso que este proyecto ocupa, la señal conocida por el receptor es la parte del símbolo OFDM en color rojo (en ausencia de canal y ruido AWGN) que ilustra la Fig. 6.9.

En la Fig. 6.10 se presenta el proceso de estimación estudiado en este apartado. En el ejemplo se parte de una señal OFDM en modo 2k que es transmitida a través de un canal cuya respuesta frecuencial altera la magnitud y la fase de las portadoras de la señal transmitida. En recepción se procede a estimar la respuesta frecuencial del canal a partir de la información que transportan las portadoras continuas y dispersas de la señal recibida. A partir de la señal de referencia conocida por el receptor se procede a estimar el canal mediante (6.27), para obtener la estimación del canal en las muestras en las que se alojan las portadoras piloto utilizadas. Para finalizar, se realiza una aproximación en este resultado que consiste en interpolar la respuesta estimada para predecir su comportamiento frecuencial en las muestras en las que se transportan datos y/o portadoras piloto TPS. La interpolación puede ser lineal, por vecino más cercano o cúbica. Se ha estudiado que el tipo de interpolación utilizada que ofrece mejores resultados es la interpolación cúbica [26], por lo que todas las simulaciones presentadas en este documento se realizan con ese método de interpolación.

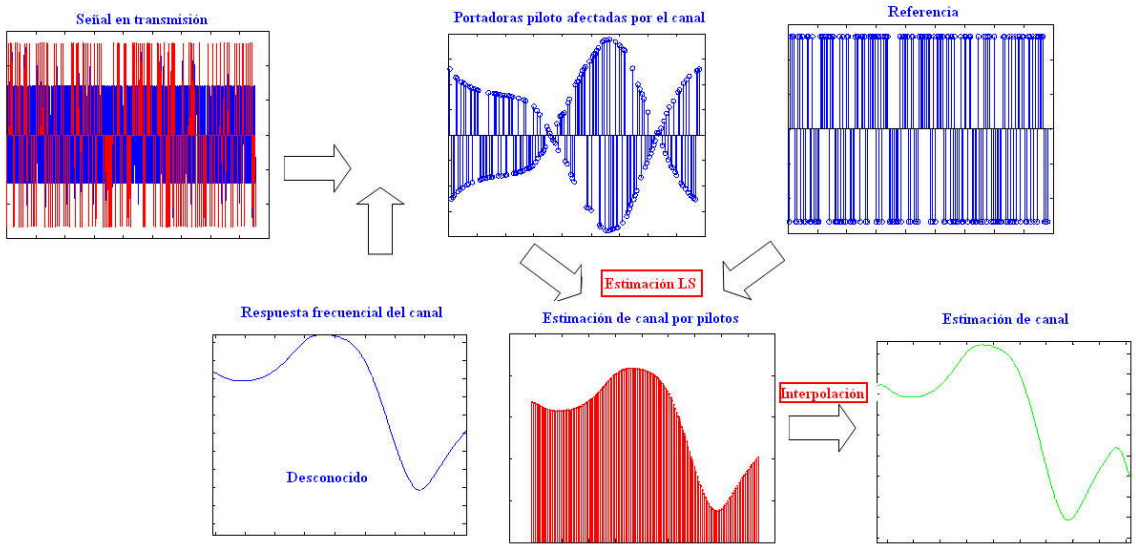


Fig. 6.10 Proceso de estimación de canal mediante LS basado en portadoras piloto en del dominio frecuencial en ausencia de ruido AWGN

En la Fig. 6.11 se ilustra la variación en las constelaciones que provoca el canal multicamino con la respuesta impulsional que se observa en la Fig. 6.10. Aún en ausencia de ruido, los símbolos PAM ((a) QPSK, (b) 16QAM, (c) 64-QAM) se ven modificados en fase y amplitud, hecho que impide la correcta detección y decisión de los símbolos. Por este motivo, resulta de vital importancia el proceso de estimación de canal de manera correcta y, posteriormente, realizar la ecualización de la señal recibida con la información que se tiene del canal.

La estimación LS tiene que resultar precisa tanto en módulo como en fase. Se han realizado simulaciones para determinar su funcionamiento en ausencia de ruido. En las Fig. 6.12 y Fig. 6.13 se ilustra el modelo de línea de retardos TDL utilizado y una de sus realizaciones Rayleigh, a partir del cual se ha estudiado el comportamiento del estimador sin ausencia de ruido. En las Fig. 6.14 y Fig. 6.15 se presenta el proceso y el resultado de la estimación LS, tanto en módulo como en fase, en condiciones de ausencia de ruido. Se puede observar como la estimación es perfecta.

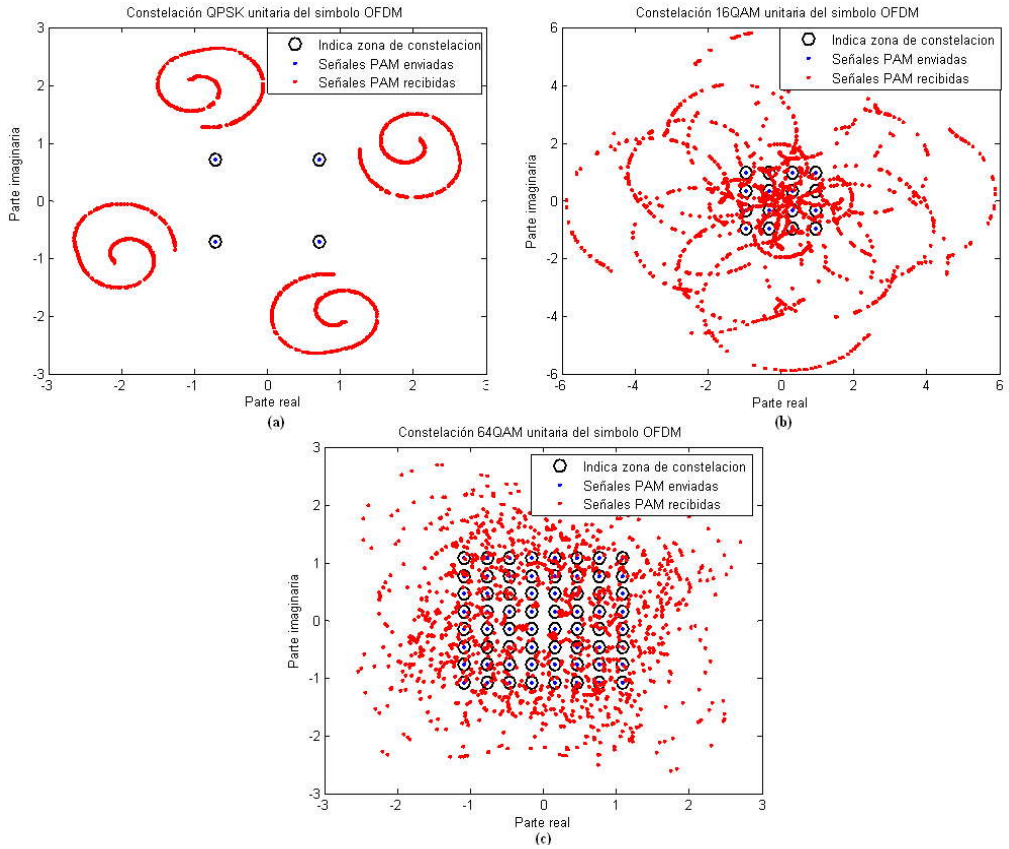


Fig. 6.11 Dispersión de las constelaciones debido a las variaciones de fase y amplitud del símbolo OFDM en ausencia de ruido

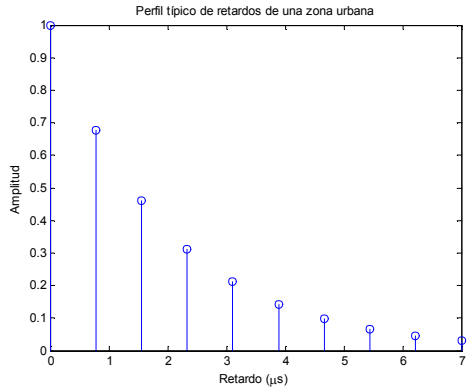


Fig. 6.12 Modelo TDL de retardos uniformes con perfil típico de zona urbana.

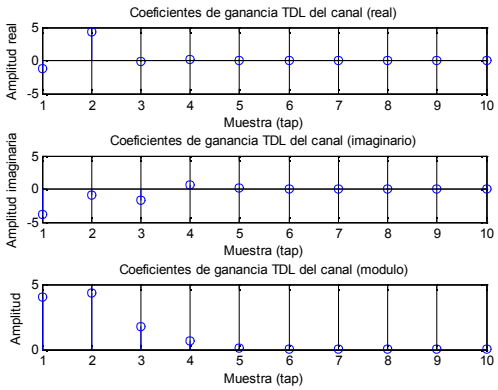


Fig. 6.13 Realización Rayleigh del canal modelado con TDL uniforme.

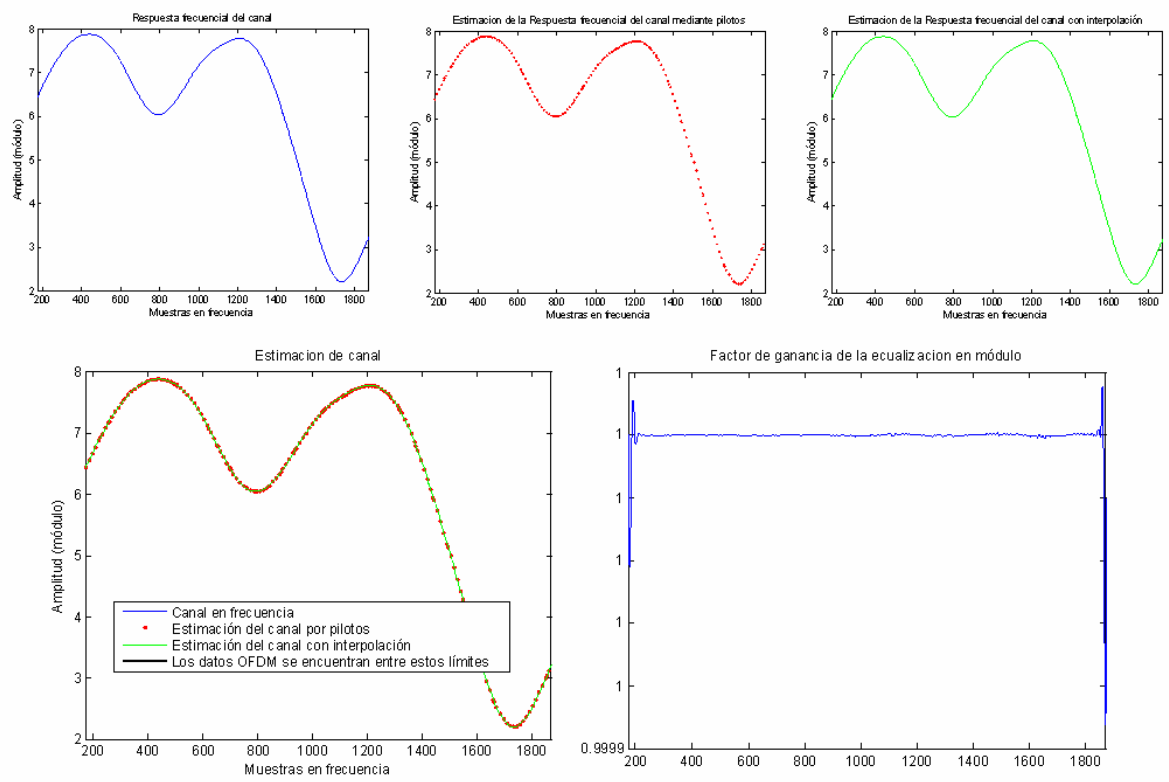


Fig. 6.14Proceso de estimación LS de canal (módulo) en ausencia de ruido

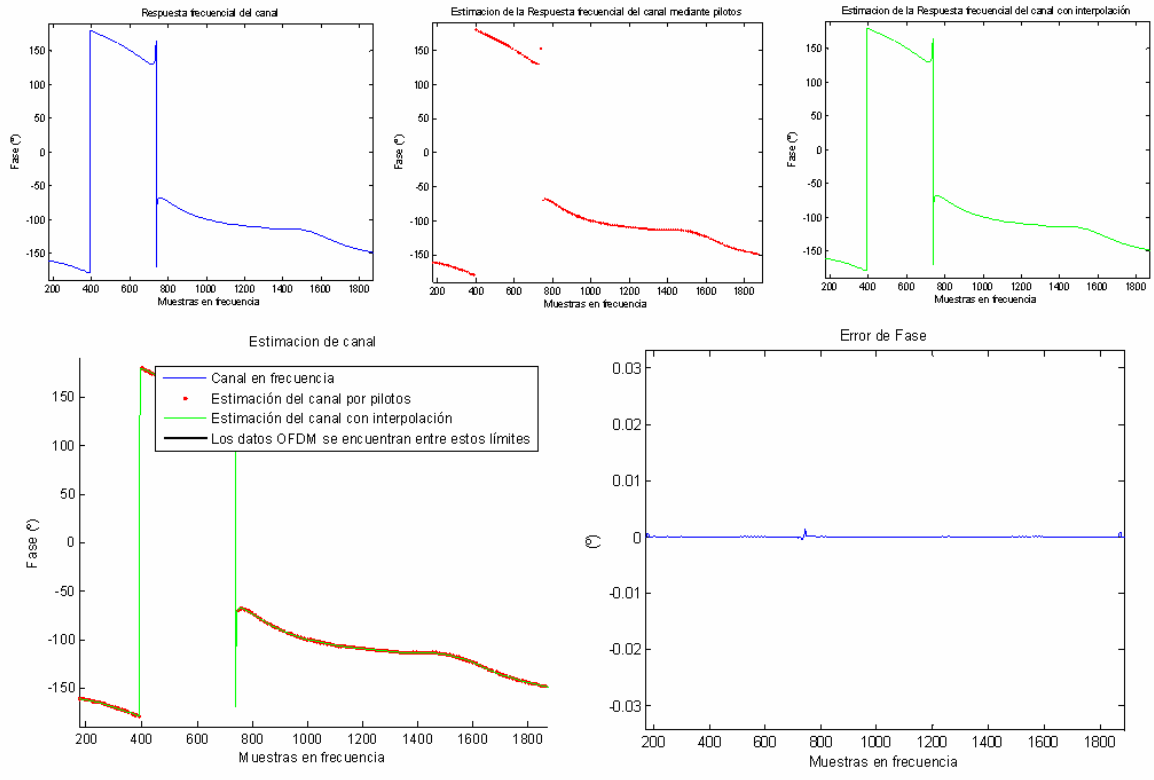


Fig. 6.15Proceso de estimación LS de canal (fase) en ausencia de ruido

En la Fig. 6.16 se ilustra el resultado de estimación de canal, (a) módulo y (b) fase, en condiciones de presencia de ruido de otra realización Rayleigh del mismo modelo de canal. Las simulaciones han sido tomadas transmitiendo un símbolo OFDM en modo 2k con una longitud de intervalo de guarda de 1/4. Se puede observar como la estimación no es tan precisa con la presencia del ruido AWGN.

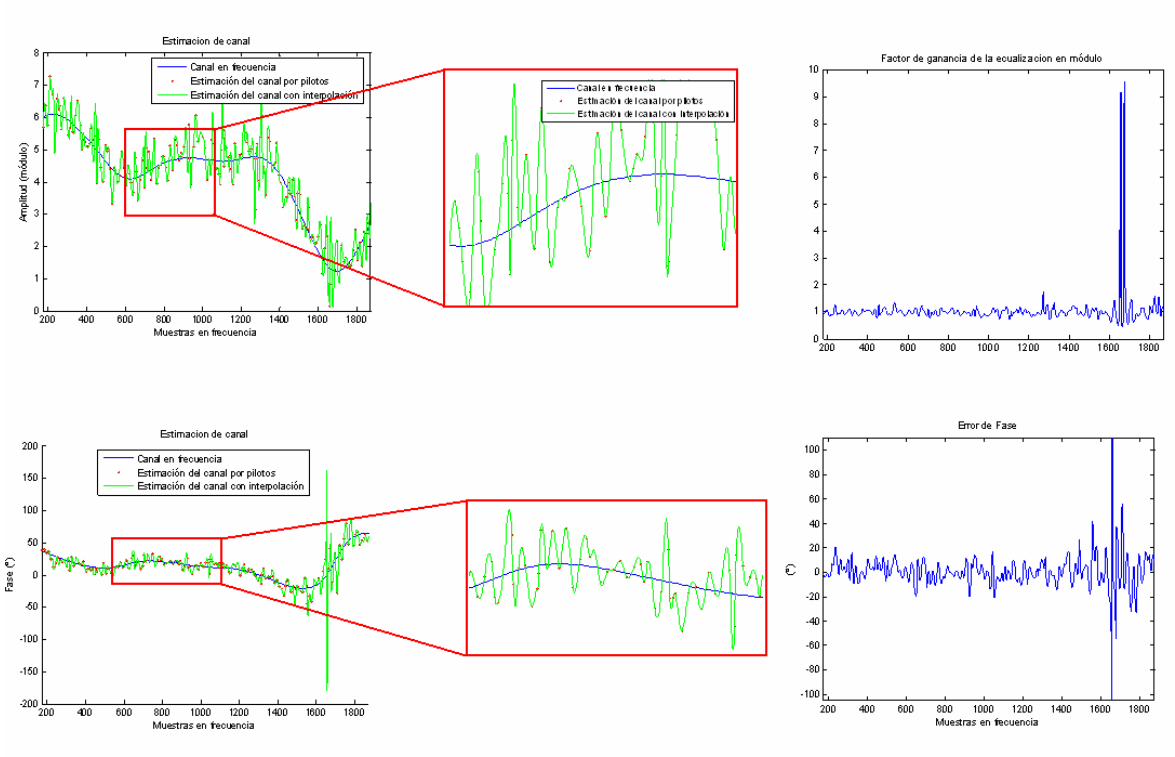


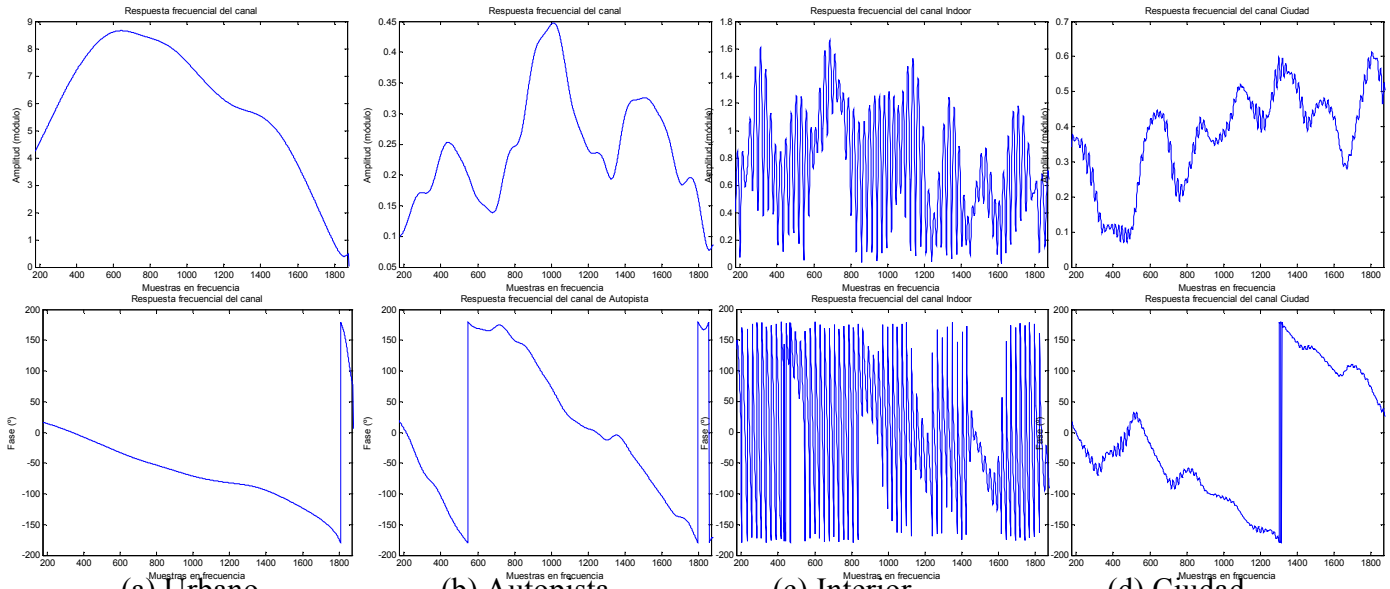
Fig. 6.16 Resultado de estimación LS de canal con presencia de ruido E_S/N_0= 0dB

Para evaluar las prestaciones que tiene el estimador implementado se hace uso de la herramienta del cálculo de varianza de estimación (Véase apartado B.2) frente a E_s/N_0. Para realizar la simulación se han generado una cantidad de bits tal que se han transmitido:

Es/N_0 (dB)	0	5	10	15	20	25
# Símbolos OFDM	1000	1000	1000	3000	3000	3000

en modo 8k y modulación QPSK (de hecho es independiente del tipo de modulación ya que la simulación se realiza en función de la energía de símbolo PAM), donde para cada símbolo transmitido se genera una nueva realización Rayleigh basada en el modelo TDL presentado anteriormente.

Se realiza la misma simulación sobre diferentes modelos TDL para observar el comportamiento del estimador sobre diferentes escenarios (el interés, según objetivos del proyecto, radica en la forma de la respuesta frecuencial en módulo y fase, no tanto en si pertenece a un entorno u otro). Los modelos TDL añadidos en la simulación son los que se han presentado en el apartado 5.2.1.1, cuyos parámetros de ganancia y retardo de cada *tap* aparecen en la Tabla 5.1 y representados en la Fig. 5.3. En la Fig. 6.17 se muestran diferentes ejemplos de realizaciones Rayleigh de cada modelo TDL de canal empleado en la simulación.



(a) Urbano (b) Autopista (c) Interior (d) Ciudad

Fig. 6.17 Ejemplos de realizaciones Rayleigh (módulo y fase) de los modelos TDL

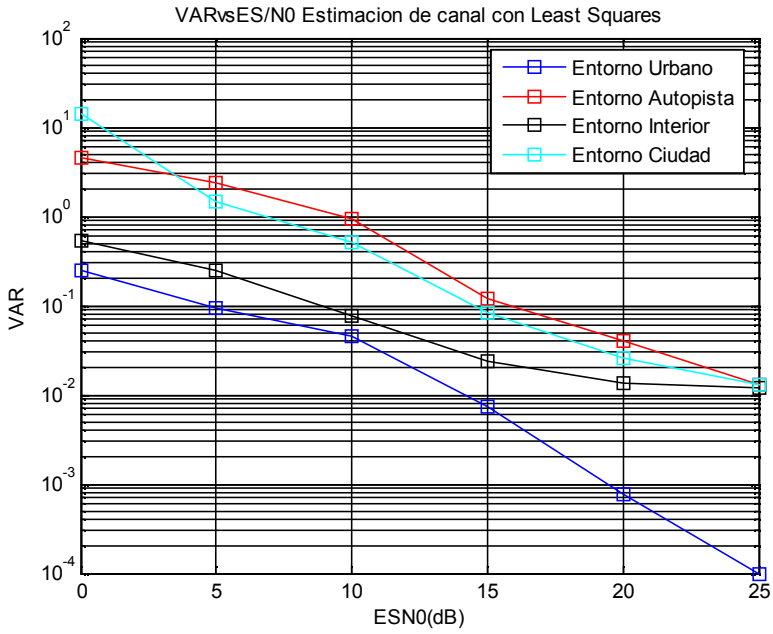


Fig. 6.18 Varianza de estimación LS de canal frente a Es/N_0

El estimador de canal LS estima correctamente todo tipo de canales en ausencia de ruido con el número de portadoras piloto continuas y dispersas de las que dispone en cada modo. Cuando estima el canal en presencia de ruido AWGN sus prestaciones disminuyen drásticamente. Además se desprende de la última simulación de varianza que la estimación depende el tipo de escenario que atraviesa la señal transmitida. Se desprende una conclusión directa, a mayor dispersión de canal peor estimación se obtiene. Se puede observar como la estimación de canal del entorno de autopista es el que peor resultado ha dado. Si se consulta la Tabla 5.1 se puede observar que es el escenario con mayor retardo. Se comprueba dicha conclusión observando el resto de retardos en los modelos TDL. Se observa una conclusión directa de la simulación realizada. Cuanto menores (en amplitud) son los coeficientes de la respuesta frecuencial del canal pero estimación se obtiene.

6.2.2.3 Código implementado

```
function [canal_est,canal_dif] =est_canal(M,data,sintetic,canal,fft_mode,T,D,C_c,const,indices_pilotos,
VARvsESN0);
%-----------------------------------------------
%       Estimación de canal
%-----------------------------------------------
fprintf('·Rx: Realizando estimación de canal %g. ',C_c);
%Extraemos pilotots de la señal (en freq), pasados por el canal y el ruido!
data_freq=ofdm_decode(data, D, fft_mode, 0, T,0);
sintetic_data=zeros(T,1);
for u=1:length(indices_pilotos)
sintetic_data(indices_pilotos(u)+1)=data_freq(indices_pilotos(u)+1);
end
sintetic_canal=zeros(fft_mode,1);                    Preparamos el vector donde irá la nuestra señal
                                                     sintética de fft_mode puntos (extraido de la señal
                                                     pasada por el canal y el ruido)
sintetic_canal(1+(fft_mode-T+1)/2:(fft_mode+T+1)/2) =  se posiciona en el centro del vector al que aplicar  ifft
sintetic_data;
%Preparamos el vector que contiene los datos sinteticos (pilotos) que conoce el receptor (sin pasar por el canal
ni el ruido)
sintetic_freq=zeros(fft_mode,1);
sintetic_freq(1+(fft_mode-T+1)/2:(fft_mode+T+1)/2) = sintetic;
pilot_pos=find(sintetic_freq~=0);                    Cogemos las posiciones de los pilotos(lo mismo que
                                                     indices_pilotos).
%Preparamos el canal en frecuencia (solo para plotearlo)
C_freq=fft(canal,fft_mode);                          Obtenemos el canal en el dominio frecuencial
C=(zeros(1,fft_mode));
C(1+(fft_mode-T+1)/2:(fft_mode+T+1)/2)=C_freq(1:T);
%La siguiente acción no la hacemos ya que hemos extraido los pilotos de la señal pasada por el canal y que
contiene también ruido. Los datos de sintetic_freq son los conocidos por el receptor (sin canal ni ruido) y nos
ayudan a estimar el canal.
fprintf('.');
if const==1
figure
plot((1:fft_mode),abs(C));
xlabel('Muestras en frecuencia'); ylabel('Amplitud');
end
Y=sintetic_canal(pilot_pos);                         Estimamos el canal
X=sintetic_freq(pilot_pos);
canal_pilotos=X.\(Y);                                Ojo, es división de matrices-->X^-1/Y. No podemos
                                                     hacer Y/X!! Por eso hacemos .\ y no /
if const==1                                          canal_pilotos=zeros(fft_mode,1);
hold on
plot(pilot_pos,abs(canal_pilotos),'r.');
end
%Aplicamos la interpolación para completar la estimación
pilot_poss=zeros(length(pilot_pos)+2,1);
pilot_poss=[1;pilot_pos;fft_mode];
canal_pilotoss=[mean(C);canal_pilotos;mean(C)];
canal_est=interp1(pilot_poss,(canal_pilotoss),1:fft_mod  Añadimos la posicion inicial de simbolo y el final...
e,'splines');
%-----------------------------------------------------   ...y el valor medio de estos (por poner algo)
%       Cálculo de varianza de canal                   Aplicamos interpolación para intentar estimar todo el
                                                       canal
%-----------------------------------------------------
if VARvsESN0==1
H=C(1+(fft_mode-T+1)/2:(fft_mode+T+1)/2);
Hls=canal_est(1+(fft_mode-T+1)/2:(fft_mode+T+1)/2);
ls_error=mean(((abs(H-Hls))/abs(H)).^2);
ls_error_pos=find(ls_error~=0);
canal_dif=ls_error(ls_error_pos);
else
canal_dif=0;
end
fprintf('.\tOK\n');
```

Código 20 est_canal.m

6.2.3 *Estimación del error de frecuencia* (estimación fina)

Los errores de frecuencia son introducidos, típicamente, por un pequeño desajuste frecuencial entre los osciladores locales del transmisor y el receptor, o bien por el desplazamiento Doppler. El impacto del error de frecuencia puede introducir ICI y destruir la ortogonalidad de las portadoras, resultando en elevados valores de BER. Con la inserción del intervalo de guarda, el símbolo OFDM se hace más robusto frente a ISI e ICI provocados por el canal multicamino, pero aún así, continúa siendo muy sensible al error de frecuencia de portadora como al error de tiempo de muestreo.

Se parte de la señal obtenida en el receptor tras su conversión a banda base y muestreada en los instantes de tiempo:

$$t_n = (n+\varepsilon)T_S \quad \text{con } n=-N_G,....,0,1,...,\ N-1 \tag{6.28}$$

donde N_G representa la primera muestra del símbolo OFDM perteneciente al intervalo de guarda, T_S representa el tiempo de muestreo y ε representa el error de tiempo normalizado.

Este error de tiempo de muestreo, ε, puede verse tras la demodulación OFDM como un error en frecuencia de la portadora, tal y como ilustra la Fig. 6.19. Se puede observar como el error de frecuencia hace peligrar la ortogonalidad entre portadoras.

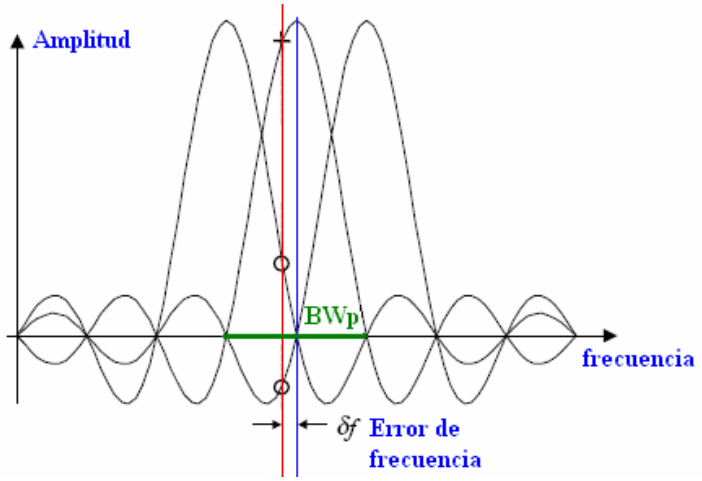


Fig. 6.19 Error de frecuencia de portadora

El error de frecuencia de portadora se puede modelar en el simulador (líneas 312-314 del código principal presentado en el anexo A.5) a partir de la inserción de un término de frecuencia, f_e, a la señal temporal recibida:

$$\tilde{s}_{OFDM}[n] = s_{OFDM}[n]\cdot e^{j\cdot 2\cdot\pi\cdot f_e\cdot n} \quad \text{con } n=-N_G,....,0,1,...,\ N-1 \tag{6.29}$$

estando el valor de f_e contenido dentro del intervalo:

$$|f_e| \le \frac{1}{2} BW_p \quad \text{con } n=-N_G,....,0,1,...,\ N-1 \tag{6.30}$$

con BW_p como ancho de banda de la portadora OFDM. Si no se cumple la condición (6.30) se puede dar el caso en que el estimador obtenga el error de frecuencia en relación a la portadora adyacente, llegando a provocar ICI. Si se produce la igualdad en la condición no se determinará error de frecuencia ya que la recepción empieza en una muestra que cumple la condición de ortogonalidad. A partir de ahora se expresará el término de error de frecuencia como:

$$\Delta = f_e \cdot T_{OFDM} = \frac{f_e}{BW_p} \tag{6.31}$$

siendo Δ el error de frecuencia normalizado a la separación frecuencial entre portadoras. El valor de este parámetro es el que se quiere estimar y es el que se introduce en el fichero de configuración de la simulación para modelar el error de frecuencia. Entonces, la condición (6.30) se reescribe como:

$$|\Delta| \leq \frac{1}{2} \quad \text{con } n=-N_G,....,0,1,...,\ N-1 \tag{6.32}$$

De hecho, el error de frecuencia f_e se puede descomponer como:

$$f_e = f'_e + \Delta f_e \tag{6.33}$$

donde f'_e es la suma de un múltiplo entero de espaciados frecuenciales de las portadoras, $(N \cdot T_{OFDM})^{-1}$, y Δf_e es un término fraccional del error de frecuencia cuyo valor absoluto cumple la condición (6.30). De hecho, el primer término de f_e se corrige mediante un proceso de estimación gruesa y posterior sincronización y, a continuación, el término fraccional se corrige a partir de un proceso de estimación fina y posterior sincronización. El algoritmo de estimación implementado es del tipo estimación fina, por lo que se asume que la estimación gruesa ya ha sido realizada y, por este motivo, se asumirá que $f_e = \Delta f_e$.

A continuación se presenta el algoritmo propuesto en este proyecto para realizar la estimación fina del error de frecuencia de portadora y su posterior sincronización.

6.2.3.1 Localización del bloque en el sistema

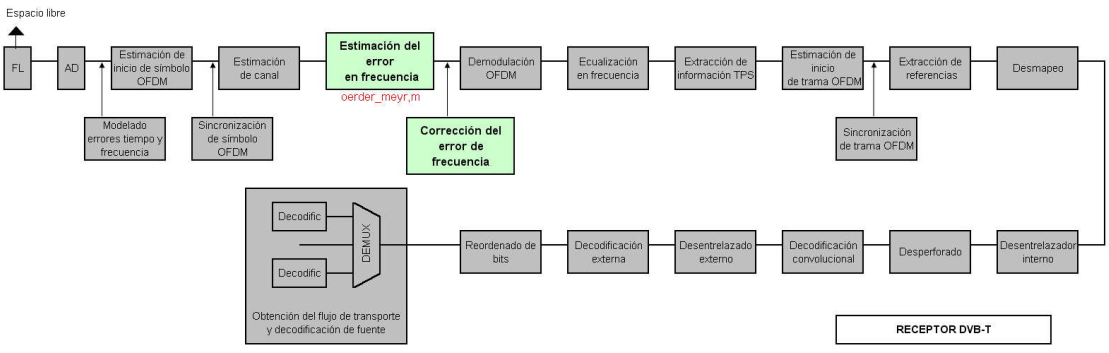


Fig. 6.20 Localización del bloque de estimación de error de frecuencia

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A.1), es *oerder_meyr.m.*

6.2.3.2 Algoritmo propuesto. Oerder&Meyr

El esquema de estimación de frecuencia que se propone para este proyecto es del tipo NDA (*NonData Aided*), robusto frente a canales AWGN y de baja complejidad que puede operar sobre canales selectivos en frecuencia.

Éste parte del reconocimiento de que el error de frecuencia de portadora afecta a la recuperación del símbolo OFDM de la misma manera que lo que afecta el error de tiempo en un sistema de modulación QAM convencional. Es decir, debido a la dualidad entre dominios, se entiende que el canal selectivo en frecuencia tiene sobre la señal demodulada OFDM un impacto similar al que tiene el canal temporal selectivo sobre la modulación QAM convencional.

El estimador propuesto en el proyecto es una adaptación de un algoritmo NDA de arquitectura *feedforward* (sin realimentación) para la sincronización de tiempo, llamado Oerder&Meyr [28], que se basa en la afirmación anterior.

Si la señal compleja muestreada a la entrada del demodulador QAM convencional se expresa como:

$$\psi[k] = \sum_{m} a[m] \cdot g[(k - m - \varepsilon)T] + n[k] \tag{6.34}$$

donde *a[m]* denota el m-ésimo símbolo QAM, *n[k]* representa el ruido complejo AWGN, *g(t)* es el pulso de la señal (normalmente pulsos de Nyquist), *T* es el espaciado temporal entre símbolos y *Tε* representa el error de tiempo, entonces se puede ver una estructura similar con la expresión de los datos de salida del bloque FFT del demodulador OFDM, que se puede expresar como:

$$u[k] = A \sum_{m=-N_G}^{N-1} c[m] \cdot H[m] \cdot IF(k - m - \Delta) + n[k] \tag{6.35}$$

donde *H[m]* representa la respuesta frecuencial del canal, *IF(n-Δ)* representa la función de ISI introducida en el canal y Δ denota el error de frecuencia normalizado al espaciado frecuencial entre portadoras. Si los términos *IF()* y *Δ* de (6.35) juegan respectivamente el papel de los términos *g(t)* y *ε* de (6.34), se puede determinar que el algoritmo de sincronización de tiempo original Oerder&Meyr para sistemas de modulaciones PAM puede ser adaptado para la sincronización en frecuencia para sistemas de modulaciones OFDM.

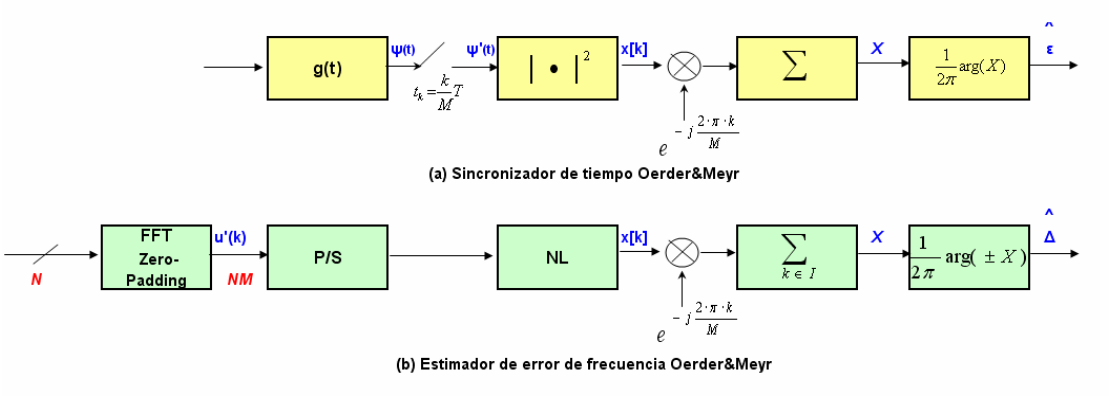


Fig. 6.21 Esquemas de estimación Oerder&Meyr

En la Fig. 6.21 se ilustran los esquemas de estimación y sincronización de Oerder&Meyr para error de tiempo (a) y error de frecuencia (b), donde se observan muchas similitudes. El sincronizador de tiempo (a) parte de la señal banda base obtenida después del filtro adaptado, $\psi(t)$. Dicha señal es sobremuestreada por un factor M y las muestras obtenidas se pasan por un bloque no lineal gobernado por una ley cuadrática. Cada una de las muestras desplazadas en frecuencia por el término multiplicativo es sumada, obteniendo así un escalar complejo, X. La estimación de error de tiempo, $\hat{\varepsilon}$, se obtiene a partir de la extracción del argumento de dicho valor que posteriormente es escalado .

Observando el esquema de estimación de tiempo, (a), se puede determinar que para adaptar el algoritmo a estimador de error de frecuencia para receptores OFDM se requiere sobremuestreo de factor M en el dominio frecuencial. Este sobremuestreo (interpolación) se puede realizar a partir de insertar $(M\text{-}1)\cdot N$ ceros (*zero-padding*) para obtener un vector de $M\cdot N$ muestras a la entrada del bloque FFT. La secuencia interpolada de salida del bloque FFT se puede expresar como:

$$u'[k] = \frac{A}{\sqrt{M}} \sum_{m=-N}^{N-1} c[m]\cdot H[m]\cdot IF\left(\frac{k}{M} - m - \Delta\right) + n'[k] \quad \text{con } k=0,....,M\cdot N\text{-}1 \qquad (6.36)$$

Se puede observar que la expresión (6.36) que las N muestras cuyo índice es un entero múltiple de M coinciden, sin tener en cuenta la constante $1/\sqrt{M}$, con aquellas obtenidas de (6.35), mientras que las otras muestras pertenecen a muestras interpoladas. A continuación, la secuencia *u'[k]* se procesa a través de un bloque no lineal (NL) con la única salvedad de que no puede ser gobernada por una ley cuadrática ya que se ha demostrado que dicha NL no ofrece buen rendimiento tanto en canales ruidosos como en canales selectivos en frecuencia. Esto es debido a que la función *sinc* que surge de la FFT puede verse como un pulso coseno raíz de Nyquist con factor de *roll-off* igual a 0 y este tipo de no linealidad no obtiene buenos resultados con este pulso. Por este motivo se debe emplear otro tipo de no linealidades de orden superior como puede ser directamente el valor absoluto, $|\cdot|$, y su tercera y cuarta potencia, $|\cdot|^3$ y $|\cdot|^4$. Dicha no linealidad produce una línea espectral que se desplaza a frecuencias DC mediante el uso del multiplicador y que es promediada en el acumulador. Es decir, el conjunto de muestras de *x[k]* desplazadas en frecuencia, X, con excepción de aquellas que pertenecen a la región de intervalo de guarda, se expresa como:

$$X = \sum_{k \in \chi} x[k] \cdot e^{-j \cdot 2 \cdot \pi \cdot \frac{k}{M}} \quad \text{con } k=0,....,M \cdot N-1 \tag{6.37}$$

donde χ recoge el conjunto de índices de las muestras que han sido procesadas por el acumulador. Finalmente, se estima $\hat{\Delta}$ a partir de:

$$\hat{\Delta} = \begin{cases} -\frac{1}{2\pi}\arg(X) \\ \\ -\frac{1}{2\pi}\arg(-X) \end{cases} \tag{6.38}$$

Una vez obtenido el parámetro estimado, se procede a corregir el error de fase introducido en la señal (6.29), (véase líneas 378-388 del código principal presentado en el anexo A.5), a partir de:

$$s'_{OFDM} = \tilde{s}_{OFDM}[n] \cdot e^{-j \cdot 2 \cdot \pi \cdot \hat{f}_e \cdot n} = s_{OFDM}[n] \cdot e^{j \cdot 2 \cdot \pi \cdot (f_e - \hat{f}_e) \cdot n} \tag{6.39}$$

siendo $\hat{f}_e$ el error en frecuencia desnormalizado:

$$\hat{f}_e = \hat{\Delta} \cdot BW_p \tag{6.40}$$

Una vez implementado el algoritmo, se procede a validar su correcto funcionamiento a partir de simulaciones de transmisión en ausencia de ruido AWGN. La Fig. 6.23 presenta los efectos del error de frecuencia sobre los símbolos mapeados en sus respectivas constelaciones al simular un error de frecuencia normalizada de Δ=0.2.

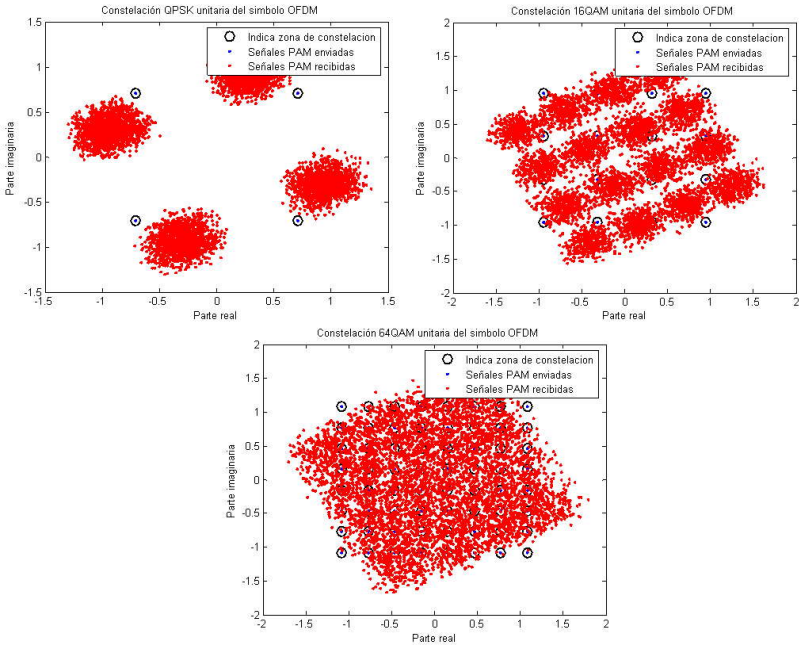


Fig. 6.22 Dispersiones en las constelaciones debido a un error de frecuencia de Δ=0.2

Nótese que la dispersión por error de frecuencia provocada en la señal OFDM es del mismo tipo que la provocada por un error de tiempo en una señal QAM convencional. Esta dualidad también se podía observar en la dispersión provocada por error de tiempo en una señal OFDM con la dispersión provocada por error de frecuencia en una señal QAM convencional, pero en sentido inverso (Véase Fig. 6.5)

La Fig. 6.23 muestra las diferentes constelaciones tras la estimación del error de frecuencia de $\Delta=0.2$ y su posterior corrección. Se puede observar cómo se consigue corregir el efecto, con cierto error de estimación debido a que el efecto de dispersión no ha desaparecido del todo, de manera que la detección de símbolos es óptima.

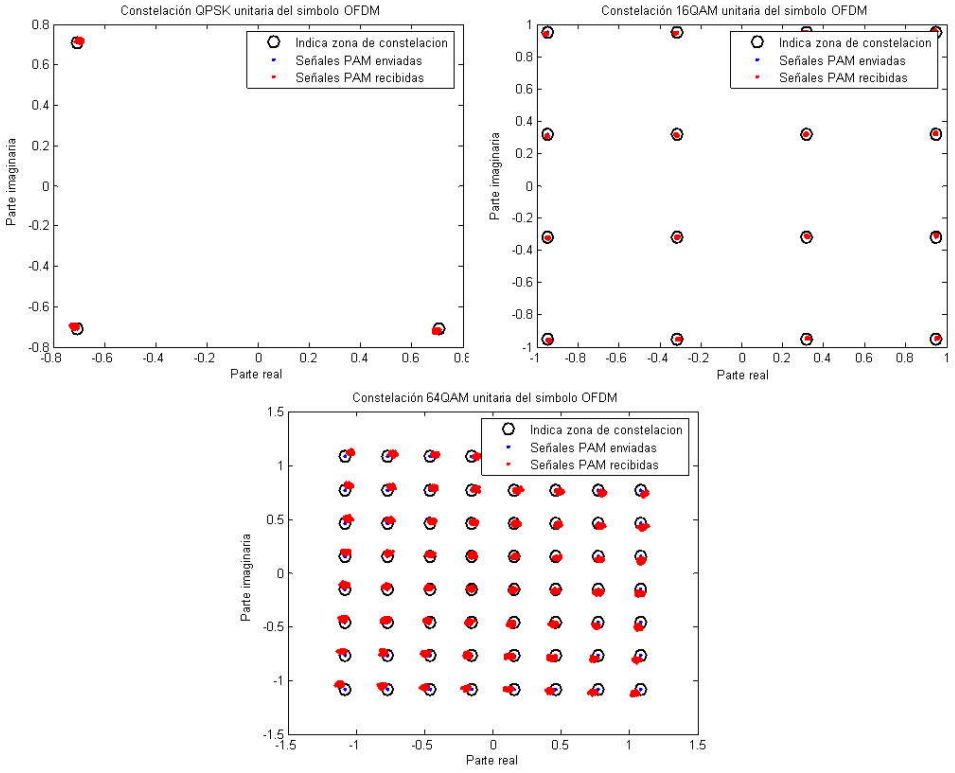


Fig. 6.23 Dispersiones en las constelaciones debido a un error de frecuencia de $\Delta=0.2$

El efecto de dispersión que continua vigente en las constelaciones tras la sincronización en frecuencia, presentadas en la Fig. 6.23, son debidas a que el término de la exponente de (6.39) no es 0, debido a una mínima diferencia entre el valor estimado y el valor real del error de frecuencia.

A continuación se ha evaluado el comportamiento del estimador O&M en condiciones de presencia de un canal AWGN (con $h(t)=\delta(t)$). Bajo esta condición todos los estimadores son insesgados. El comportamiento del estimador se ha realizado asumiendo la varianza del error de estimación como figura de mérito y su varianza ha sido comparada con la cota de Cramér Rao, MCRB. (Véase anexo B.2). La varianza del error de estimación, con diversos valores del exponente de la no linealidad, representada en función de la relación Es/N_0 se ilustra en la Fig. 6.24. Para realizar la simulación se ha generado un número tal de bits que permite enviar las siguientes cantidades de símbolos OFDM:

Es/N_0 (dB)	0	5	10	15	20	25
# Símbolos OFDM	1000	1000	1000	2000	2000	2000

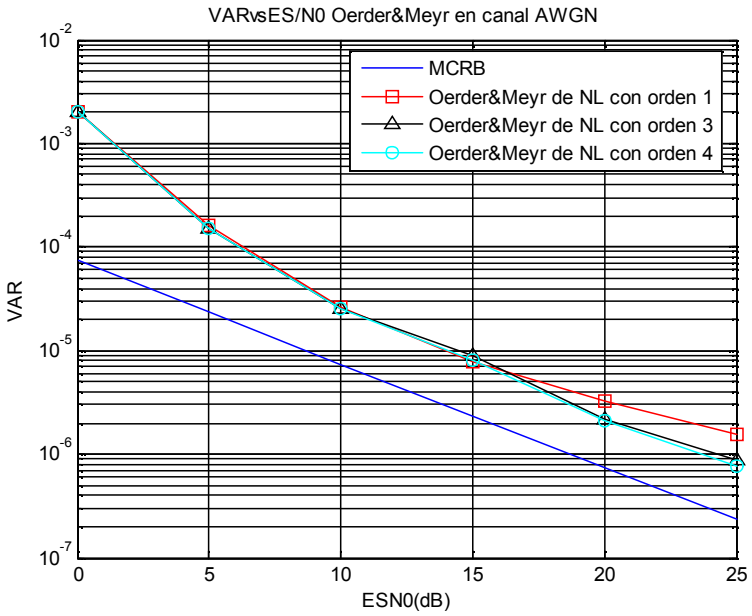


Fig. 6.24 Varianza del error de estimación de frecuencia en función de Es/N_0

El gráfico representa además los valores de la cota inferior de Cramér Rao [28], cuya expresión en el caso de estimadores de frecuencia (véase línea 60 del código presentado en el anexo A.5):

$$MCRB(\Delta) = \frac{3}{2 \cdot \pi^2 N} \cdot \frac{1}{E_S / N_0} \tag{6.41}$$

Se puede observar que los estimadores basados en el tercer y cuarto orden de NL, $|\cdot|^3$ y $|\cdot|^4$, son prácticamente iguales, siendo los más cercanos a la cota de Cramér Rao sobre un amplio rango de valores de Es/N_0. Tal y como se ha comentado anteriormente, la simulación se ha realizado con un valor de error de frecuencia normalizada de Δ=0.2. No se han encontrado diferencias significativas de comportamiento de los estimadores con otros valores de Δ que cumplen la condición (6.32).

6.2.3.3 Código implementado

```
1  function [fe_est] =oerder_meyr(data_in,M,P,S)
2  fprintf('·Rx: Paso del simbolo %g por estimador de error de frecuencia Oerder&Meyr de NL con orden %g',S,P);
3  fprintf('.');
4  %-------------------------------------------------
5  %   Estimación en frecuencia. Oerder&Meyr
6  %-------------------------------------------------
7  data_zp=(fft([zeros((M-1)*length(data_in),1);data_in]));
8  switch P
9     case 1
10       signo=1;
11    case 3
12       signo=-1;
13    case 4
14       signo=-1;
15 end
16 fprintf('.');
17 data_zp_nl=(abs(data_zp)).^P;
18 data_zp_nl_mult=data_zp_nl.*exp(-j*2*pi/M.*(0:length(data_zp_nl)-1)).';
19 data_zp_nl_mult_sum=sum(data_zp_nl_mult);
20 fe_est=(-1/(2*pi)).*angle(signo*data_zp_nl_mult_sum);
21 fprintf('.\tOK\n');
22 fprintf('\t·O&M: El error en frecuencia estimado con NL de orden %g es de %g\n',P,fe_est);
```

Código 21 oerder_meyr.m

6.3 DEMODULACIÓN OFDM

Tal y como se ha visto en el apartado 4.3, se puede englobar una serie de procesos de tratamiento digital de la señal dentro de un gran bloque denominado Demodulación OFDM. Éste comprende los bloques de demodulación OFDM y extracción del intervalo de guarda, ecualización en frecuencia, extracción de información TPS, estimación de inicio de trama OFDM, extracción de referencias y, finalmente, detección y desmapeo de símbolos PAM.

La mayoría de bloques implementados presentan funcionamientos complementarios a sus análogos situados en el transmisor. En los casos en que ocurre no se realizarán explicaciones extensas puesto que ya están presentes en el capítulo Capítulo 4. Para continuar con la estructura del documento se presentarán sus localizaciones en el sistema y el código implementado. En los casos estudiados se presentarán las simulaciones pertinentes.

6.3.1 Extracción del intervalo de guarda y demodulación OFDM

6.3.1.1 Localización del bloque en el sistema

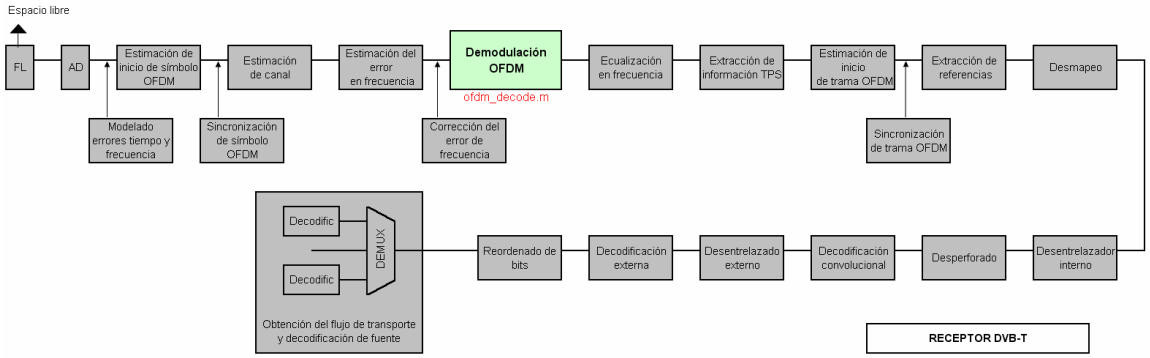


Fig. 6.25 Localización del bloque de demodulación OFDM

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *ofdm_decode.m*. Realiza el proceso de demodulación OFDM descrito en el apartado 3.4.2, con algunas modificaciones en su estructura impuestas por el estándar.

6.3.1.2 Objetivo y relación con el estándar

A continuación se muestra el diagrama de bloques del demodulador OFDM en recepción determinado por la estructura del transmisor DVB-T.

El receptor recibe la señal $\tilde{y}$ [n] correspondiente al símbolo OFDM convolucionado con la respuesta impulsional del canal, *h[n]*, y con influencia del ruido AWGN, *v[n]*:

$$\tilde{y}[n] = h[n] * \tilde{s}_{OFDM} + v[n] \qquad \text{con } n=-Ns,...,0,...,N-1 \tag{6.42}$$

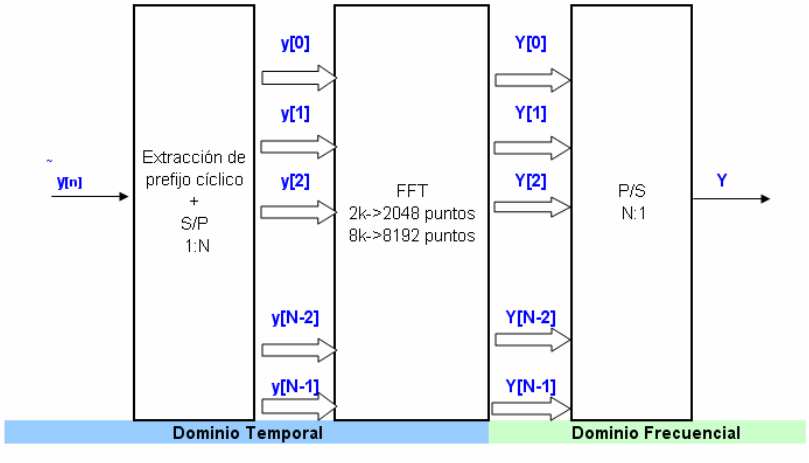


Fig. 6.26 Diagrama de bloques del sistema demodulador OFDM empleado

Las primeras *Ns* muestras (correspondientes al intervalo de guarda) están afectadas por la ISI del canal dichas muestras se eliminan, resultando en una señal *y[n]* de N muestras. A continuación, las *N* muestras de *y[n]* se disponen en paralelo gracias al bloque convertidor S/P y se le aplica la DFT (FFT al ser *N* un valor potencia de 2):

$$DFT[y[n]] = Y[n] = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} y[n] e^{-j \cdot \frac{2\pi i n}{N}} \quad \text{con } 0 \le n \le N-1 \tag{6.43}$$

Finalmente, las muestras de *Y[n]* son procesadas por el bloque conversor paralelo-serie, P/S.

6.3.1.3 Código implementado

```
function [data_ofdm] =ofdm_decode(data_in,D, fft_mode,modo, T,S)
long_guard=fft_mode*D;%Longitud del intervalo de guarda una vez se haya realizado la fft
%-------------------------------------------------
 % Qitamos el intervalo de guarda
 %-------------------------------------------------
data_time=data_in(1+long_guard:length(data_in));
%-------------------------------------------------------------------------------------
% Realizamos FFT de "fft_mode" puntos (2048 o 8192)
%-------------------------------------------------------------------
if modo~=0
fprintf('·Rx: Realización de FFT de %g puntos para el símbolo %g de modo %gK (de %g portadoras) una vez
fprintf('.');
end
data_freq=fftshift(fft(data_time));
data_ofdm=data_freq((1+(fft_mode-T+1)/2:(fft_mode+T+1)/2));
if modo~=0
fprintf('.\tOK\n');
end
```

Código 22 ofdm_decode.m

6.3.2 Ecualización en frecuencia

La ecualización es una técnica empleada para regenerar la señal que ha sufrido modificaciones en amplitud y fase debido a las características del canal de transmisión que se ejecuta una vez se ha estimado el canal y se ha sincronizado la transmisión en tiempo y frecuencia. Existen muchas técnicas de ecualización de señal, que tienen las siguientes características:

- Ecualización fraccional o no fraccional: según si el tiempo de muestreo T_S es mayor o igual al periodo de la muestra T_m
- Ecualización símbolo a símbolo, SBS, o bien de secuencia (óptimo para cuando el canal añade ISI en las muestras de datos y no sólo en el intervalo de guarda)
- Ecualización fija (cambian cada cierto tiempo) o adaptativa (tienen una fase de aprendizaje y una fase de seguimiento)
- Ecualización lineal (sumas y productos de coeficientes) o no lineales.

Las diferentes técnicas propuestas para su uso en sistemas OFDM combinan las distintas características mencionadas. De los ecualizadores existentes los más conocidos son:

- Ecualizador forzador de zeros, ZF (*Zero Forcing*): lineal, símbolo a símbolo y fijo.
- Ecualizador MMSE (filtro de Wienner): lineal, símbolo a símbolo, adaptativo.
- Ecualizador MLSE (Máxima Verosimilitud): óptimo, no lineal, de secuencia y fijo.
- Ecualizador DFE (*Decision Feedback Equalizer*): invierte el efecto del canal en el transmisor ya que el receptor le ha dado información de la estimación. Entre otras cosas no aumenta el nivel del ruido AWGN.

A continuación se presenta el estimador propuesto en el proyecto. Se trata de un estimador ZF que se ha escogido por su gran sencillez de implementación.

6.3.2.1 Localización del bloque en el sistema

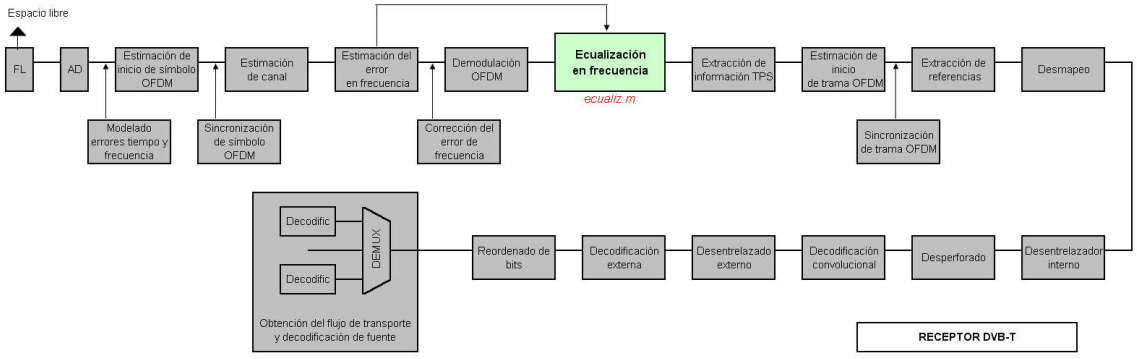


Fig. 6.27 Localización del bloque de ecualización en frecuencia

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *ecualiz.m*.

6.3.2.2 Objetivo y relación con el estándar

En el apartado 5.1 se han visto los efectos negativos que provoca el canal inalámbrico multicamino sobre la señal transmitida. En general, para diseñar un sistema OFDM inalámbrico, se supone que el canal tiene una longitud de respuesta impulsional finita. Una extensión cíclica, de duración mayor que esta respuesta impulsional, es insertada entre símbolos consecutivos con el objetivo de prevenir el ISI y preservar la ortogonalidad de los tonos. En el apartado 3.5.2 se ha estudiado la funcionalidad del prefijo cíclico y se ha determinado que para recuperar la señal transmitida es necesario tener conocimiento del canal, tal y como se expresa en (3.45):

$$\underline{d} = \underline{\underline{\Lambda}}^{-1} \cdot \underline{Y} = diag\left\{ \underline{\underline{F}} \cdot \begin{bmatrix} \underline{h} \\ \underline{0} \end{bmatrix} \right\}^{-1} \cdot \underline{Y} \tag{6.44}$$

donde $\underline{d}$ representa el vector de símbolos PAM a la entrada del modulador OFDM (transmisión) en el dominio de la frecuencia y $\underline{z}$ representa el vector de símbolos PAM a la salida del demodulador OFDM (recepción), también en el dominio de la frecuencia. En dicho apartado se determina que la matriz $\underline{\underline{\Lambda}}$ es una matriz que contiene la DFT del canal. Por este motivo, una de las grandes ventajas de la modulación OFDM es que resulta muy sencillo ecualizar en el dominio de la frecuencia, siempre y cuando se tenga conocimiento del canal.

La técnica que ejecuta una ecualización acorde con (6.44) es la conocida como ZF. Intuitivamente se puede entender su funcionamiento. La señal de entrada al ecualizador *Y[n]*, en el dominio frecuencial, se puede expresar como:

$$Y[i] = H[i] \cdot S_{OFDM}[i] + N[i] \tag{6.45}$$

donde *N[i]* representa la transformada frecuencial del vector de ruido AWGN, $S_{OFDM}[i]$ es la respuesta frecuencial de la señal OFDM a la salida del transmisor y *H[i]* es la respuesta frecuencial del canal. Entonces, para recuperar la señal OFDM se procede de la siguiente manera:

$$S_{OFDMeq}[i] = \frac{H[i] \cdot S_{OFDM}[i]}{H_{est}[i]} + \frac{N[i]}{H_{est}[i]} \tag{6.46}$$

donde $H_{est}[i]$ es la respuesta frecuencial del canal estimado. Por tanto, la calidad de la regeneración de la señal dependerá directamente de la calidad de la estimación del canal. Además, se puede observar como el ruido AWGN también es amplificado y puede introducir más ISI por lo que, dependiendo de las características del canal y de la potencia de ruido, puede que haya más ISI y ruido a la salida del ecualizador que a la entrada del mismo.

En el apartado 6.2.2.2 se ha introducido el concepto de estimación de canal mediante la explicación del estimador LS propuesto. La Fig. 6.18 muestra el gráfico de varianza de error de estimación, de cuatro canales modelados con diferentes TDL, como figura de mérito para evaluar el estimador. La Fig. 6.28 muestra la respuesta del sistema OFDM mediante el uso de curvas de BER en función de la relación Es/N_0. para el caso en que se ecualiza con el canal estimado del modelo de canal urbano. El BER ha sido calculado comparando los bits de la entrada del mapeador en transmisión con los bits obtenidos a la salida del desmapeador/detector en recepción.

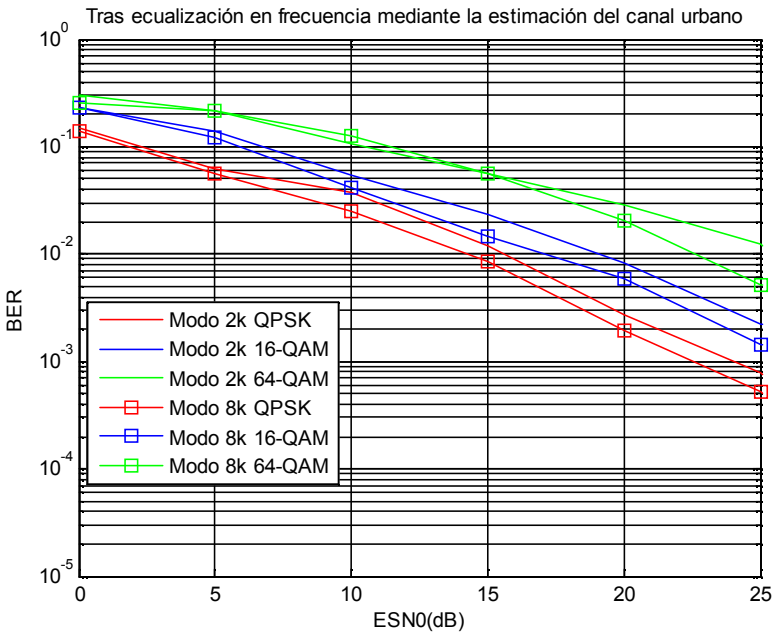


Fig. 6.28 Localización del bloque de ecualización en frecuencia.

Se representan las curvas de BER para diferentes condiciones de transmisión. Se hace una comparación entre todas las respuestas de BER obtenidas por modos de transmisión (2k y 8k) y por modulación PAM (QPSK, 16 QAM, 64 QAM). Fácilmente se puede apreciar que el BER del sistema disminuye a medida que la relación Es/N_0 aumenta en todas las curvas representadas. Este hecho realza el buen funcionamiento del ecualizador y la mejora en estimación de canal en condiciones de potencias altas de transmisión. Otro aspecto notable es que las transmisiones en modo 8k tienen una BER menor que las transmisiones en 2k, en condiciones de igualdad de modulación PAM. De hecho, un símbolo OFDM en modo 8k tiene muchas más portadoras piloto en proporción a un símbolo OFDM en modo 2k, disponiendo ambos símbolos del mismo ancho frecuencial (6, 7 o 8 Mhz aproximadamente). La interpolación realizada en la estimación de canal mediante portadoras resulta más precisa en modo 8k, por lo que mejor se regenera la señal con la ecualización. El último punto a destacar es que la modulación más robusta para ambos modos en la QPAK. Ésta ofrece mayor protección frente a la presencia de ruido que las modulaciones 16-QAM y 64-QAM. Por contra, la modulación QPSK tiene una tasa de transmisión más lenta ya que se envían menos bits por símbolo. La modulación 64-QAM en cambio tiene una tasa de transmisión mayor pero no ofrece mucha protección frente al ruido. Esto es debido a que sus márgenes de decisión en el detector son más estrechos, por lo que sufre más ISI en canales de baja calidad (desde el punto de vista de la señal transmitida).

6.3.2.3 Código implementado

```
1 function data_out=ecualiz(data_in,canal,fft_mode,D,S,T);
2 fprintf('·Rx: Ecualización en frecuencia para el símbolo %g .',S);
3 fprintf('.');
4 Y=(data_in);
5 H=canal((1+(fft_mode-
6 data_out=H.\(Y.');%Ecualizamos
7 data_out=data_in;
8 fprintf('.\tOK\n');
```

Código 23 ecualiz.m

6.3.3 Extracción de información TPS

En el apartado 4.3.4 se ha comentado la función de las portadoras piloto TPS, sus posiciones dentro del símbolo OFDM y su modulación. El principal objetivo de las portadoras piloto TPS es la de transmitir la información de los parámetros de transmisión al receptor.

6.3.3.1 Localización del bloque en el sistema

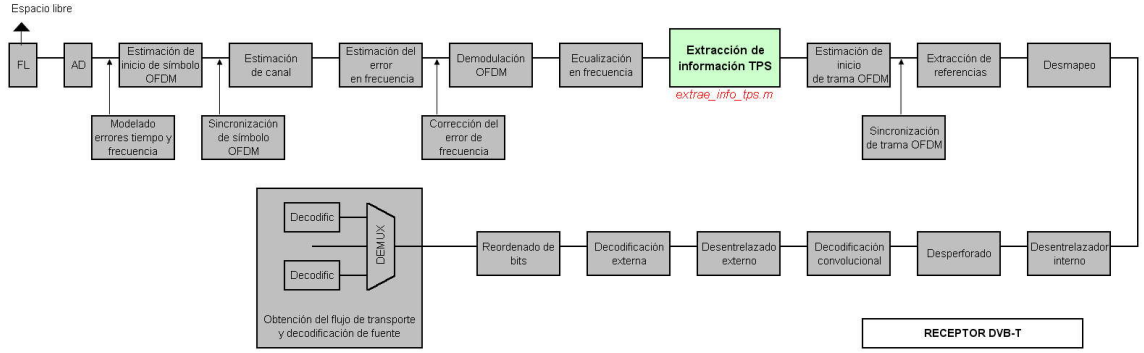


Fig. 6.29 Localización del bloque de extracción de información TPS

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *extrae_info_tps.m*. Utiliza una función auxiliar que se ocupa de la creación del bloque de información (usando decodificación BCH) que es *tps_deinfo.m*.

6.3.3.2 Objetivo y relación con el estándar

Este bloque del receptor tiene como objetivo demodular en DBPSK las portadoras piloto TPS y obtener así el bloque TPS de información una vez procesada una trama OFDM (68 símbolos OFDM). Es decir, cada trama OFDM transmite al receptor la información de diferentes parámetros de transmisión enviando en todas las portadoras TPS de cada símbolo OFDM un bit del bloque de información TPS (que tiene 68 bits=68 portadoras). La Fig. 4.32 ilustra la formación de este bloque TPS de información y la Fig. 4.33 ilustra el proceso de modulación DBPSK de las portadoras TPS de un símbolo OFDM que transmiten un bit del paquete de información TPS de 68 bits.

Cada vez que pasa un símbolo OFDM por este bloque se extraen las portadoras TPS moduladas en DBPSK y se realiza el proceso contrario para demodular. La Fig. 6.30 ilustra el proceso de demodulación partiendo de un ejemplo en el que se está recibiendo 4 símbolos con información TPS en sus portadoras TPS. El vector de información TPS que transmiten los 4 símbolos OFDM es [0 0 0 1], un bit transmitido por símbolo OFDM. El primer símbolo lleva el bit de inicialización cuya modulación (mediante un registro PRBS) sirve de referencia al resto. Si se recibe la misma modulación de TPS que el símbolo anterior se determina que se está transmitiendo un 0. En caso contrario, si la modulación de las TPS es inversa a la del símbolo anterior se determina que se ha recibido un 1.

Una vez obtenido el bloque de información TPS de 68 bits (tras recibir y demodular las portadoras TPS de los 68 símbolos OFDM que forman una trama OFDM) se procede a decodificar el bloque codificado en BCH. Para la decodificación en BCH se ha utilizado la librería *Communications Tools* de Matlab, en especial la función *gf.m* (para transformar el bloque de bits a campo de Galois) y la función *bch_dec.m* (para decodificar en BCH).

El proceso es inverso al de la codificación BCH por lo que se obtiene como resultado un vector de información de 53 bits útiles, cuyos valores corresponden a los de la Tabla 4.17.

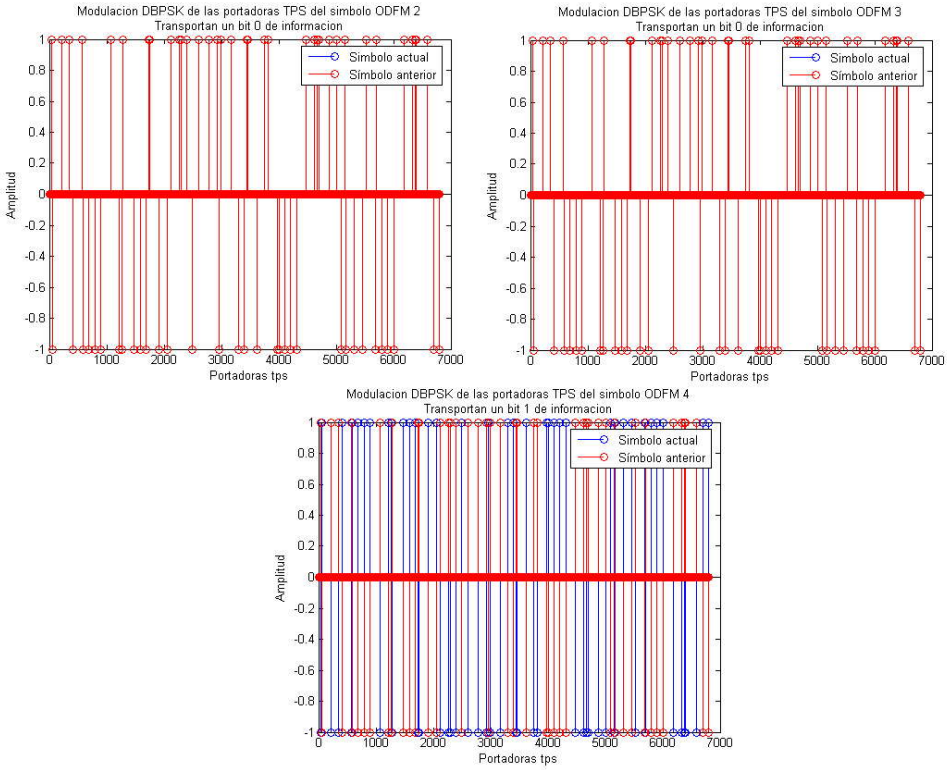


Fig. 6.30 Ejemplo de demodulación DBPSK

A modo de ejemplo, la Fig. 6.31 presenta la información extraída e interpretada de las portadoras TPS de los símbolos OFDM de una trama OFDM recibida. Se puede observar que la información que se transmite es el de número de trama OFDM (O), constelación utilizada (n), parámetro de modulación (alfa), relación de codificación convolucional (C), longitud del intervalo de guarda (D) y modo de transmisión (modo).

```
·Tx: Adición y modulacion de información a las portadoras TPS del simbolo OFDM 68 de la trama OFDM 2 de la si
·Tx: Realización de IFFT de 8192 puntos para el símbolo 68 de modo 8K (6817 portadoras) con intervalo de guar
·C: Paso de los datos por canal ideal h(t)=1 ...    OK
·C: Paso de los datos por canal sin ruido...    OK
            SE HA RECIBIDO LA TRAMA OFDM 1  (68 símbolos OFDM)
·Rx: Realización de FFT de 8192 puntos para el símbolo 68 de modo 8K (de 6817 portadoras) una vez eliminado :
·Rx: Demodulacion y extracción de información a las portadoras TPS del simbolo OFDM 68 de la trama OFDM 2 de
    ·De la trama OFDM 1 se desprende la siguiente informacion tps:
                    Enviado       Recibido
    O                 1             1
    n                 4             4
    alfa              1             1
    C                 0.5           0.5
    D                 0.25          0.25
    modo              8             8
·Rx: Extracción de 769 señales de referencia al simbolo 68 OFDM de 6048 portadoras de datos...  OK
·Rx: Desmapeo del símbolo OFDM 68 desnormalizado con constelación QPSK con alpha=1...   OK
·Tx: Creando trama 253 MPEG-2...    OK
·Tx: Paso de la trama 253 por etapa de scrambling...    OK
```

Fig. 6.31 Ejemplo de extracción de información TPS

Se debe mencionar que para poder transmitir información TPS se ha tenido que llevar una cuenta exhaustiva del número de símbolo OFDM que se envían y sus índices dentro de una trama OFDM. En este sentido, el simulador implementado también resulta completo y acorde con el estándar de transmisión DVB-T.

6.3.3.3 Código implementado

```
function [data_ref,tps_modulados_ant,tps_bch,n_tps,alpha_tps,D_tps,C_tps,modo_tps,O_tps,S_index,sincro] = extrae_info_tps(modo,data_ref,tps_modulados_ant,tps_bch,S_rx,tps_rx,S_index,O_index,SO,T,sincro, const);

fprintf('·Rx: Demodulacion y extracción de información a las portadoras TPS del simbolo OFDM %g de la trama OFDM %g de la supertrama OFDM %g',S_index,O_index,SO);
fprintf('.');
%-------------------------------------------------------------------------------------------------------------
% Apuntamos a las posiciones de los pilotos TPS (índices) definidas por el estandar
%-------------------------------------------------------------------------------------------------------------
   if modo==2                                          Cargamos la primera vez que se ejecuta la función. Son siempre los mismos indices.
     %--------------------TPS-------------------------
     tps=[34 50 209 346 413 569 595 688 790 901 1073 1219 1262 1286 1469 1594 1687];
   elseif modo==8
     %--------------------TPS-------------------------
     tps=[34 50 209 346 413 569 595 688 790 9011073 1219 1262 1286 1469 1594 1687 1738 1754 1913 2050 2117 2273 2299 2392 2494 2605 27772923 2966 2990 3173 3298 3391 3442 3458 3617 3754 3821 3977 4003 4096 4198 4309 4481 4627 4670 4694 4877 5002 5095 5146 5162 5321 5458 5525 5681 5707 5800 5902 6013 6185 6331 6374  6398 6581 6706 6799];
   end
  num_tps=length(tps);
%---------------------------------------
%     Demodulacion tps (DBPSK)
%---------------------------------------
%Las portadoras tps vienen moduladas en BPSK. Lo que hacemos es quedarnos con las portadoras piloto TPS
 pt=1;                                                 Apuntador a la primera posición de la portadora piloto tps
 tps_modulados=zeros(1,length(tps));                   Aquí guardamos los piotos tps modulados
%Generamos secuencia PRBS para mirar si se trata del bit de inicializacion modulado
 w= zeros(1,T);                                        Almacenaremos la secuencia PRBS
 num_reg=11;                                           Numero de  registros
 reg = ones(1,num  reg);
 for m = 1:T
 w(m) = reg(11);
 bit_nuevo = xor (reg(9), reg(11));
 reg = [ bit_nuevo reg(1:10) ];
 end
 %Obtenemos los pilotos tps modulados y preparamos señal de referencia PRBS
 for u= 1:T
   if pt <= length(tps)
     t = 1+tps(pt);
   else
     t = 0;
   end
   if u == t;                                          Piloto tps
   tps_modulados(pt)=round(real(data_ref(u)));         Guardamos el piloto tps modulado
   referencia(pt) =real(2 * (0.5 - w(t)));             Guardamos la modulacion de referencia. La del bit se señalizacion
   pt = pt + 1;                                        VER ETSI 300.744 v1.51 4.6.3 pag 34
   end
 end
fprintf('.');
%Ahora tenemos ya el bit tps modulado  que transmite este simbolo
 if isempty(tps_modulados_ant)                         Miramos si es el primer simbolo que recibimos de la trama OFDM...
    iguales=0;
```

```
47      for a=1:num_tps
48          if sign(tps_modulados(a))==sign(referencia(a))
49              iguales=iguales+1;
50          end
51      end
52      if iguales>=num_tps/2;                          Si la mitad o mas de portadoras son iguales
                                                        determinamos que se trata de un 0
53       %El primer simbolo lleva la modulacion de referencia
54          S_index=1;                                  Sabemos que es el primer simbolo porque lleva
                                                        referencia de modulacion de la señalización
55          sincro=1;                                   Si son iguales activamos la señal de sincro
56      else
57          sincro=0;                                   Desactivamos la señal de sincro.
58          tps_modulados_ant=[];
59          n_tps=[];
60          alpha_tps=[];
61          D_tps=[];
62          C_tps=[];
63          modo_tps=[];
64          O_tps=[];
65          return                                      Descartaremos este simbolo a espera de el de
                                                        señalizacion
66      end
67   else                                               Si no es el primer simbolo empezamos a demodular el
                                                        DBPSK
68      iguales=0;                                      Llevará la cuenta de los bits iguales entre tps anterior y
                                                        actual.
69      for a=1:num_tps
70          if
    sign(tps_modulados(a))==sign(tps_modulados_ant(a))

71              iguales=iguales+1;
72          end
73      end
74      if iguales>=num_tps/2 bit_tps=0;                Si la mitad o mas de portadoras son iguales
                                                        determinamos que se trata de un 0
75      else bit_tps=1;                                 Si no, decimos que es un 1
76      end
77      tps_bch(S_index)=bit_tps;                       Guardamos el valor del bit tps demodulado (codificado
                                                        en BCH) en el vector
78   end
79  tps_modulados_ant=tps_modulados;                    Guardamos la info de la modulación actual para usarla
                                                        como anterior en la siguiente secuencia
80  %Cuando recibimos los 68 simbolos disponemos ya del bloque tps codificado en BCH
81  if S_index==68
82  %---------------------------------
83  % Generacion del bloque tps de info
84  %---------------------------------
85  [n_tps,alpha_tps,D_tps,C_tps,modo_tps,O_tps]=tps_deinfo(tps_bch);
86  else                                                Decodificamos el bloque tps y obtenemos los
                                                        parametros de transmisión
87  n_tps=[];
88  alpha_tps=[];
89  D_tps=[];
90  C_tps=[];
91  modo_tps=[];
92  O_tps=[];
93  end
94  fprintf('.\tOK\n');
```

Código 24 extrae_info_tps.m

```
function [n_tps,alpha_tps,D_tps,C_tps,modo_tps,O_tps]=tps_deinfo(bloque_tps)
%-------------------------------------------------------------
%      Obtención del bloque de info TPS
%-------------------------------------------------------------
%DECODIFICACIÓN BCH (Paso inverso al codificador)
N=127;
K=113;
tps_senyaliz=bloque_tps(1);                    El primer bit es de señalizacion
code_tps_rx=bloque_tps(2:68);                  Datos codificados en BCH
code_tps_rx_ext=[zeros(1,60),code_tps_rx];     Añadimos los 0 para crear vector de N bits
code_tps_rx_ext_galois=gf(code_tps_rx_ext);    Pasamos a formato Galois
[code_tps_dec,err,code_tps_dec] = bchdec(code_tps_rx_ext_galois,N,K);
s_galois_rx=[code_tps_dec(61:K)];              Obtenemos los 53 bits tps de info en formato galois el
                                               vector recibido esta en formato de Galois (cada bit es
 s_rx=zeros(1,54);                             Meteremos aquí nuesto bloque de info+señalización
 s_rx(1)=tps_senyaliz;                         El primer valor de nuestra s es senyalización, que ya lo
 for u=1:53
    if(s_galois_rx(u)==1) s_rx(u+1)=1;
    else s_rx(u+1)=0;
    end
 end
%OBTENCIÓN DE INFO
 if s_rx(2:17)==[0 0 1 1 0 1 0 1 1 1 1 0 1 1 1 0]
 elseif s_rx(2:17)==[1 1 0 0 1 0 1 0 0 0 0 1 0 0 0 1]
 end
 if s_rx(18:23)==[0 1 0 1 1 1];
 end
 if s_rx(24:25)==[0 0] O_tps=1;
 elseif s_rx(24:25)==[0 1] O_tps=2;
 elseif s_rx(24:25)==[1 0] O_tps=3;
 elseif s_rx(24:25)==[1 1] O_tps=4;
 else O_tps=0;
 end
 if s_rx(26:27)==[0 0] n_tps=4;
 elseif s_rx(26:27)==[0 1] n_tps=16;
 elseif s_rx(26:27)==[1 0] n_tps=64;
 else n_tps=0;
 end
 if s_rx(28:30)==[0 0 1] alpha_tps=1;
 elseif s_rx(28:30)==[0 1 0] alpha_tps=2;
 elseif s_rx(28:30)==[0 1 1] alpha_tps=4;
 else alpha_tps=0;
 end
 if s_rx(31:33)==[0 0 0] C_tps=1/2;
 elseif s_rx(31:33)==[0 0 1] C_tps=2/3;
 elseif s_rx(31:33)==[0 1 0] C_tps=3/4;
 elseif s_rx(31:33)==[0 1 1] C_tps=5/6;
 elseif s_rx(31:33)==[1 0 0] C_tps=7/8;
 else C_tps=0;
 end
 if s_rx(34:36)==[0 0 0] C_tps_2=1/2;
 elseif s_rx(34:36)==[0 0 1] C_tps_2=2/3;
 elseif s_rx(34:36)==[0 1 0] C_tps_2=3/4;
 elseif s_rx(34:36)==[0 1 1] C_tps_2=5/6;
 elseif s_rx(34:36)==[1 0 0] C_tps_2=7/8;
 else C_tps_2=0;
 end
  %INTERVALO DE GUARDA
 if s_rx(37:38)==[0 0] D_tps=1/32;
 elseif s_rx(37:38)==[0 1] D_tps=1/16;
 elseif s_rx(37:38)==[1 0] D_tps=1/8;
 elseif s_rx(37:38)==[1 1] D_tps=1/4;
 else D_tps=0;
 end
  %ESTANDAR DE TRANSMISIÓN
 if s_rx(39:40)==[0 0] modo_tps=2;
 elseif s_rx(39:40)==[0 1] modo_tps=8;
 else modo_tps=0;
 end
```

Código 25 tps_deinfo.m

6.3.4 Estimación de inicio de trama OFDM

En el punto en que se encuentra el proceso de demodulación de la señal recibida la sincronización de tiempo y de frecuencia ya ha sido realizada, así como la estimación de canal y la ecualización de la señal. A pesar de que el estándar DVB-T no especifica nada referente a la sincronización de trama OFDM, según aplicaciones, puede existir la necesidad de que el receptor sincronice con la primera trama OFDM de una supertrama OFDM. Para estos casos una solución es utilizar sincronizadores de trama. Éstos están formados por bloques de estimación de inicio de trama basados en detectar una determinada secuencia de bits dentro de una trama mayor, conocida por el receptor, que tenga las mejores propiedades de autocorrelación posibles dentro de esa trama.

James L. Massey propuso en 1972 un esquema de sincronización de trama basada en una regla de decisión óptima [29] [30] para localizar una secuencia de sincronismo transmitida por un canal AWGN. Se debe recalcar que el estimador que propuso se aplica únicamente en una secuencia de sincronismo periódica e integrada dentro de una trama de datos. Uno de los puntos fuertes es que la secuencia de sincronización no tiene porque estar precedida de un conjunto de bits nulos para reforzar la sincronización sino que tiene buenas prestaciones a pesar de que la secuencia de sincronismo esté rodeada de datos aleatorios.

En el apartado anterior se ha presentado el bloque que extrae la información de ciertos parámetros de transmisión de las tramas OFDM. En cada trama OFDM se transmite un bloque TPS de 68 bits, por lo que, en cada uno de los 68 símbolos OFDM que forman la trama OFDM se transmite 1 bit de información. El tipo de información que transmite el bloque TPS de una trama OFDM se presenta en la Tabla 4.17. Entre los parámetros de información, los parámetros de sincronización y de número de trama OFDM son parámetros que se repiten en cada bloque de información TPS, por lo que se pueden considerar fijos y ser usados como una referencia. Ambos parámetros pueden formar conjuntamente lo que se podría denominar una "secuencia de sincronismo TPS", de longitud L=*22* bits, dentro de una trama de información de N=*53 bits*. Los valores que toman éstos bits son {0,1}.

Aprovechando que la estructura de la transmisión de información TPS se asemeja a las condiciones necesarias para poder utilizar el estimador propuesto por Massey, se ha decidido estudiar su comportamiento y aplicarlo al estándar de transmisión del DVB-T aprovechando la estructura del bloque de información TPS.

6.3.4.1 Localización del bloque en el sistema

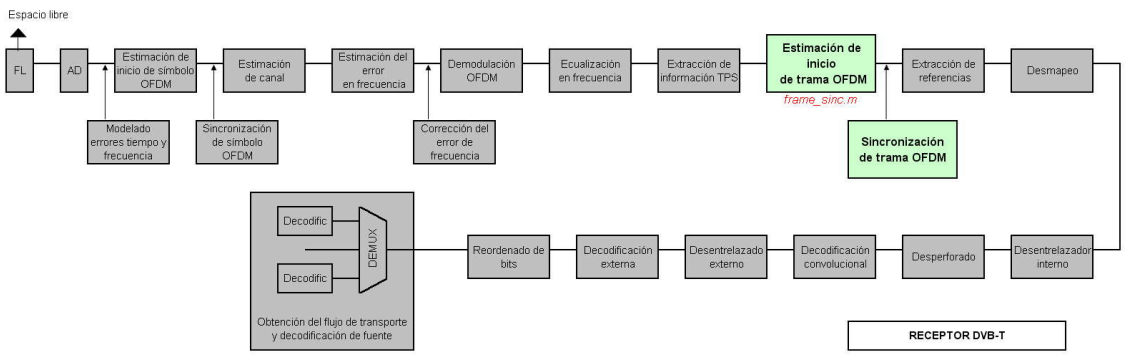


Fig. 6.32 Localización del bloque de estimación de inicio de trama OFDM

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *frame_sincr.m*.

6.3.4.2 Algoritmo propuesto. Massey

Considérese que una secuencia de sincronismo S de L bits:

$$S = [s_0, s_1, s_2, ..., s_{L-1}] \tag{6.47}$$

es insertada con periodicidad N en una trama binaria de datos aleatoria y que el valor de cada s_i es $\{0,1\}$. Consideramos R el vector de datos recibido a ser procesados de longitud N en donde está contenida S:

$$R = [r_0, r_1, r_2, ..., r_{N-1}] \tag{6.48}$$

Entonces se puede expresar R como:

$$R = [d_0, d_1, d_2, ..., s_m, s_{m+1}, ..., s_{m+L-1}, ..., .d_{N-2}, d_{N-1}] \tag{6.49}$$

donde s_m es el primer bit de la secuencia de sincronismo que se encuentra en la posición m del vector R y d_i representan los bits aleatorios que completan el vector R y cuyos valores son equiprobables $\{1,-1\}$.

El parámetro a estimar es m y, a priori, puede tomar cualquier valor dentro del intervalo de [0, N-1] muestras del bloque periódico R. En presencia de ruido complejo AWGN, la trama R se expresa como:

$$R' = [d_0, d_1, d_2, ..., \underbrace{s_m, s_{m+1}, ..., s_{m+L-1}}_{SecuenciaSincronismo}, ..., .d_{N-2}, d_{N-1}] + n_i \text{ con } i=0...N-1 \tag{6.50}$$

donde n_i representa cada contribución de ruido gausiano blanco complejo de media 0 y varianza $N_0/2$.

Teniendo en cuenta estas consideraciones, Massey [30] formuló la regla óptima para la localización de la secuencia de sincronismo (*Optimum Rule*), que se trata de escoger la μ, $0 \leq \mu \leq N-1$, que maximiza el siguiente estadístico:

$$S_O = \sum_{i=0}^{L-1} s_i r_{i+\mu} - \sum_{i=0}^{L-1} f(r_{i+\mu}) \tag{6.51}$$

donde

$$f(x) = \frac{N_0}{2\sqrt{E}} \ln\left[\cosh\left(\frac{2\sqrt{E}x}{N_0}\right)\right] \tag{6.52}$$

La expresión (6.51) suele aproximarse al primer término eliminando el segundo, quedando un estadístico denominado Regla de Correlación (*Correlation Rule*):

$$S_C = \sum_{i=0}^{L-1} s_i r_{i+\mu} \tag{6.53}$$

La expresión (6.51) se puede aproximar de maneras distintas si se tiene en cuenta el valor de la SNR. Para valores de SNR elevados el argumento del *cosh* de la expresión (6.51) es mucho más grande que 1 por lo que se puede aproximar por $1/2 \cdot e^{y}$, resultando la expresión del estadístico llamado Regla óptima para valores de SNR altos (*Optimum Rule for High SNR*):

$$S_H = \sum_{i=0}^{L-1} s_i r_{i+\mu} - \sum_{i=0}^{L-1} \left| r_{i+\mu} \right| \tag{6.54}$$

Para valores de SNR pequeños, la aproximación da lugar a la expresión del estadístico conocido como Regla óptima para valores pequeños de SNR (*Optimum Rule for Low SNR*):

$$S_L = \sum_{i=0}^{L-1} s_i r_{i+\mu} - \frac{\sqrt{E}}{N_0} \sum_{i=0}^{L-1} r^2{}_{i+\mu} \tag{6.55}$$

La figura de mérito que se emplea para evaluar el conjunto de estimadores es la probabilidad de error. En [30] se especifica que la probabilidad de error de estimación de inicio de secuencia en ausencia de ruido se expresa como:

$$P(\varepsilon_S) = \sum_{i=1}^{M} (-1)^{i+1} \frac{1}{i+1} \binom{N-L-(L-1)i}{i} 2^{-L \cdot i} \tag{6.56}$$

donde

$$M = \left\lfloor \frac{N-L}{L} \right\rfloor \tag{6.57}$$

Esta expresión de la probabilidad de error en ausencia de ruido determinará una cota inferior para los resultados de probabilidad de error de las simulaciones con presencia de ruido.

Las simulaciones para la obtención de la probabilidad de error se han realizado con un total de 10.000 realizaciones para cada valor de Es/N_0. El cálculo de la probabilidad de error se ha realizado a partir de:

$$P(\varepsilon) = \frac{\#estimaciones\_erróneas}{\#realizaciones\_totales} \tag{6.58}$$

Un factor determinante en la calidad de la estimación de la primera posición de la secuencia de sincronismo dentro de la trama de *N* bits es la el grado de autocorrelación de la secuencia de sincronismo dentro de *R*. Si *N>>L* las propiedades de autocorrelación de *S* serán bajas debido a que hay una gran posibilidad de que un conjunto de bits de *R* pueda formar aleatoriamente una secuencia igual a la de sincronismo. A medida que *L* tiende a *N* la palabra de sincronismo aumenta sus propiedades de autocorrelación.

La Fig. 6.33 muestra los gráficos de probabilidad de error de estimación y varianza del error de estimación frente a Es/N_0 en la que se estima la posición de inicio de una secuencia conocida *S*=[1,1,1,-1,-1,1,-1] de longitud *L=7* dentro de una trama de *N=28* bits {-1,1] aleatorios. La posición de inicio de la secuencia de sincronismo dentro de la trama ha sido de *m*=7. No se han observado diferencias de comportamiento de los estimadores en función de *m* (debido a que el vector *R* ha sido realizado módulo N) si se encuentra dentro del intervalo $0 \leq m \leq N-1$.

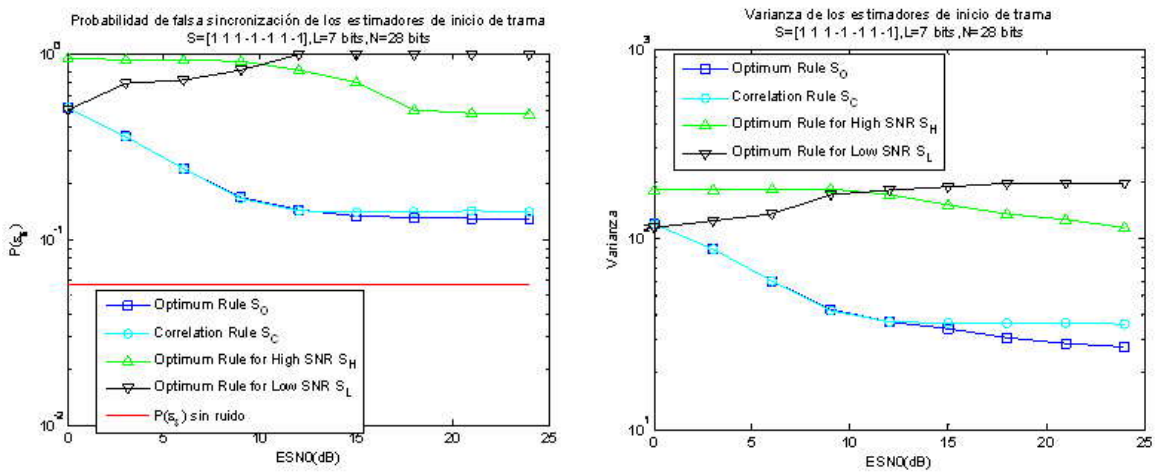


Fig. 6.33 Probabilidad de error y varianza de error de estimación de inicio de secuencia para L=7 y N=28

Se puede comprobar que el funcionamiento de los estimadores son coherentes con sus aproximaciones. El estimador óptimo es el que mejor comportamiento demuestra, seguido del estimador de correlación. Se puede observar también que los estimadores para aproximaciones a bajas y a altas SNR se comportan según lo esperado. El de aproximaciones a bajas SNR, S_L, se satura llegado a un valor de Es/N0 de 10 dB aproximadamente, valor en el que S_H ya empieza a tener un mejor comportamiento.

Para comprobar si el comportamiento de los estimadores va en función de las propiedades de correlación de la secuencia de sincronismo se genera un vector *R* que contiene el vector *S* anterior y el resto de bits son 0. Con esto se pretende hacer que las propiedades de autocorrelación de *S* sean muy elevadas (caso ideal) ya que no existe la posibilidad de que ninguna combinación del resto de bits aleatorios de *R* forme una secuencia igual a *S*. La Fig. 6.34 muestra los gráficos de probabilidad de error y de variancia de error de estimación bajo estas condiciones. Se puede observar que el valor de la probabilidad de error y la varianza reciben valores mucho más pequeños que los obtenidos en las curvas en caso de correlación no ideal, incluso se supera la cota inferior de probabilidad de error expresada en (6.56). Se supera la cota debido a que ésta está calculada en base a la aleatoriedad de los valores del resto de bits de *R*.

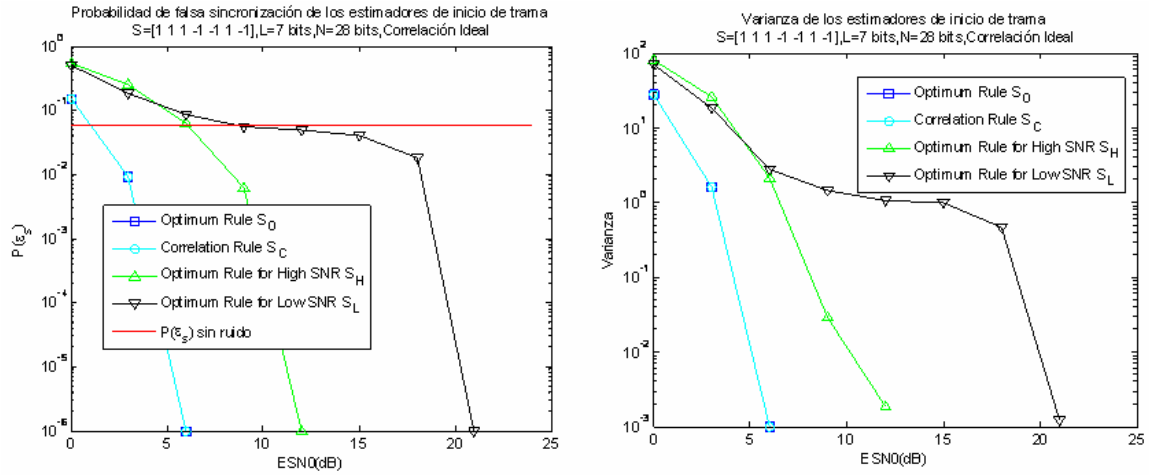


Fig. 6.34 Probabilidad de error y varianza de error de estimación de inicio de secuencia para L=7 y N=28. Caso ideal de correlación

A continuación, en la se muestran los resultados de las simulaciones realizadas con la secuencia de sincronismo TPS de *L=22 bits* obtenidos de la Tabla 4.17, cuyos valores son de:

$$S = [1\text{-}1\text{-}1\ 1\ 1\text{-}1\ 1\text{-}1\ 1\ 1\ 1\ 1\ \text{-}1\ 1\ 1\ 1\ \text{-}1\text{-}1\ 1\text{-}1\ 1\ 1\ 1] \tag{6.59}$$

donde los 0's han sido intercambiados por 1's para satisfacer las consideraciones previas de los estimadores. La longitud de la trama *R* en este caso es de ***N***=53 bits, ya que es el tamaño del bloque TPS de información tras la decodificación BCH. El resto de bits que cumplimentan *R* se han supuesto aleatorios.

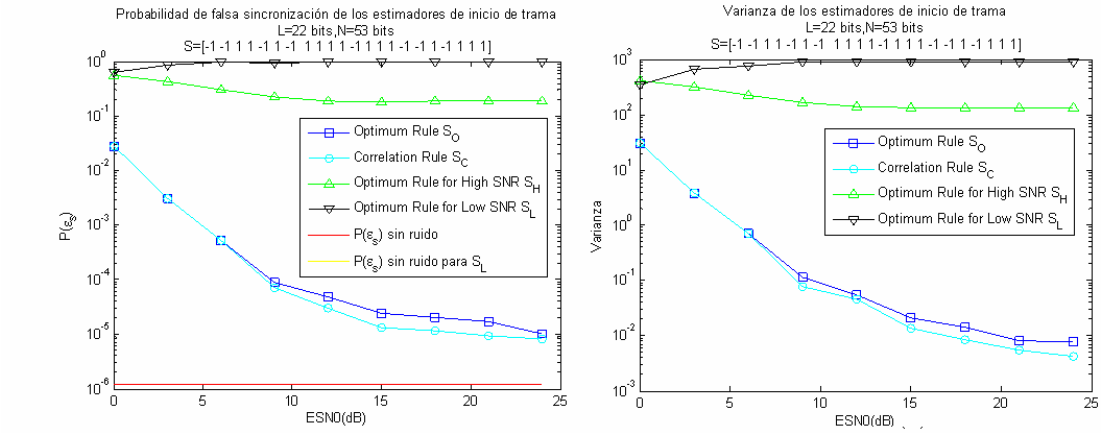


Fig. 6.35 Probabilidad de error y varianza de error de estimación de inicio de secuencia de sincronismo TPS.

Se puede observar en la Fig. 6.35 que la probabilidad de error de estimación de inicio de secuencia es muy baja para los casos S_O y S_C, tendiendo sus valores a la cota inferior teórica de probabilidad de error en ausencia de ruido. Como sucede en el resto de simulaciones realizadas, se obtiene una respuesta esperada para todos los estimadores en relación al comportamiento para cada valor de SNR.

6.3.4.3 Código implementado

```
function [m_est]=frame_sincr(m,estim)
error=0;                                  Inicializamos variables que llevarán la cuenta de los
                                          errores de estimacion para la probabilidad de error
m=4;                                      Posición del primer bit de S en R a estimar
Pn=1/EBN0;                                Potencia de ruido a partir de ESN0 (bit a bit)
L=22;
M=53;
n_r=randn(1,N)*sqrt(Pn/2);                Generamos vector de ruido aleatorio, gausiano de
n_i=randn(1,N)*sqrt(Pn/2);                Generamos vector de ruido aleatorio, gausiano de
n=n_r+j*n_i;                              Ruido complejo
%-----------------------------------------
%Generación del vector de sinconización S
%-----------------------------------------
S=randint(1,L,2);                         Creamos vector inicial S€{0,1}
s=find(S==0);                             Buscamos las posiciones de los 0s
S(s)=-1;                                  Ahora ya si que S€{-1,1}
S=[-1 -1 1 1 -1 1 -1  1 1 1 1 -1 1 1 1 -1 -1 1 -1 1 1 1];  Secuencia de sincronismo
R=randint(1,N,2);                         Creamos vector inicial R€{0,1}
r=find(R==0);                             Buscamos las posiciones de los 0s
R(r)=-1;                                  Ahora ya si que D€{-1,1}
%-----------------------------------------
%Generación del vector de entrada al sincronizador.
%-----------------------------------------
%Se posiciona S dentro de R.Lo hacemos módulo N,circular.
R=[R R];                                  Generamos nuestro vector de entrada al sincronizador a
R(m:m+L-1)=S;                             partir de S y de R
R=R+[n n];                                Añadimos el ruido
%---------------------------------
%Aplicación de tipo de estimador
%---------------------------------
for mu=1:N                                term es una constante a restar
 corr_2(mu)=sum(S(1:L).*R(mu:mu+L-1));
 term_1(mu)=sum((1/(2*sqrt(EBN0)))*log(cosh(pi/180*(2*sqrt(EBN0)*R(mu:mu+L-1)))));
 term_3(mu)=sum((R(mu:mu+L-1)));
 term_4(mu)=sqrt(EBN0)*sum((R(mu:mu+L-1)).^2);
end
if estim==1                               Optimum Rule
    S_estim=corr_2(1:N)-term_1(1:N);
elseif estim==2                           Correlation Rule
   S_estim=corr_2(1:N);
elseif estim==3                           Optimum Rule for High SNR
   S_estim=corr_2(1:N)-term_3(1:N);
elseif estim==4                           Optimum Rule for Low SNR
   S_estim=corr_2(1:N)-term_4(1:N);
end
%-----------------------------------------------------------
%Determinación de la posición del primer bit de sincronismo
%-----------------------------------------------------------
 [nada,frame_est]=max((S_estim));
 frame_dif(i)=(m-frame_est)^2;            Valores de varianza
 if frame_dif(i)~=0 error=error+1; end    Cuenta errores para la probabilidad de error
%---------------------------------
%Cálculo de probabilidad de error
%---------------------------------
for h=1:(floor((N-L)/L))
prob_error_teor(h)=((-1)^(h+1))*(1/(h+1))*(factorial(N-L-
(L-1)*h)/factorial(N-L-(L-1)*h-h))*(2^(-L*h));
end
prob_error_sin_ruido=sum(prob_error_teor);  Probabilidad de falso sincronismo teórico para el caso
                                            sin ruido
prob_error_sin_ruido_sl=(N-1)/N;            Prob error para el estimador Optimum Rule for Low
                                            SNR
prob_error(e)=error/S_vect(e);              Probabilidad de error con ruido
VAR_valores(e)=mean(frame_dif)              ´Calculo de la variancia a partir de la diferencia de
                                            valores al cuadrado
fprintf('·La posición del primer bit de sincronismo S dentro de R es la %g\n',m);
fprintf('·La posición estimada del primer bit de sincronismo S dentro de R es la %g\n',frame_est);
```

Código 26 frame_sincr.m

6.3.5 Extracción de referencias

Este bloque es complementario al presentado en el apartado 4.3.3. Una vez se obtienen los datos en el dominio frecuencial este bloque se ocupa de extraer las portadoras piloto continuas, dispersas y de información TPS para formar una señal OFDM únicamente de datos. En modo 2k el símbolo OFDM vuelve a tener 1512 portadoras y en modo 8k el símbolo OFDM vuelve a tener 6048 portadoras.

6.3.5.1 Localización del bloque en el sistema

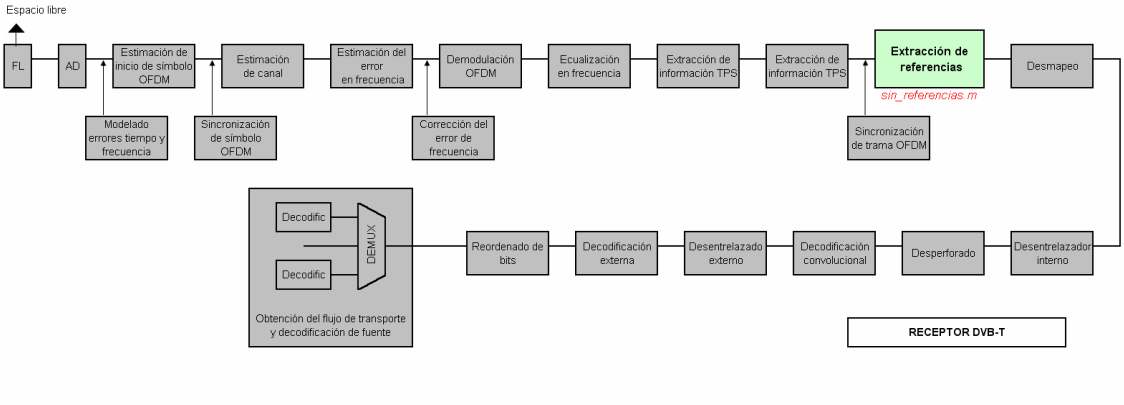


Fig. 6.36 Localización del bloque de extracción de referencias

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *sin_referencias.m*.

6.3.5.2 Código implementado

```
1  function [data_no_ref,tps,continual,w,indices_pilotos] =sin_referencias(data_in,modo,N,P,T,S,tps,continual,
   S_index,w)
2  fprintf('·Rx: Extracción de %g señales de referencia al simbolo %g OFDM de %g portadoras de datos',P,S,N);
3  fprintf('.');
…  MISMO código que referencias.m (líneas 8-70) recogido en el apartado 4.3.3.3
71    %Creación del vector paralelo solo de datos
72   data_no_ref = zeros(T, 1);
73   d=1;
74   ps=1;
75   pt=1;
76   for u = 1:T
77     p = indices_pilotos(ps)+1;
78     if pt <= length(tps)
79       t = 1+tps(pt);
80     else
81       t = 0;
82     end
83     if u == p
84       ps = ps + 1;
85     elseif u == t;
86       pt = pt + 1;
87     else
88       data_no_ref(u) =  data_in(d);
89       d = d + 1;
90     end
91 data_no_ref=data_no_ref.';
92   end
93 fprintf('.\tOK\n');
```

Código 27 sin_referencias.m

6.3.6 Detección y desmapeo

6.3.6.1 Localización del bloque en el sistema

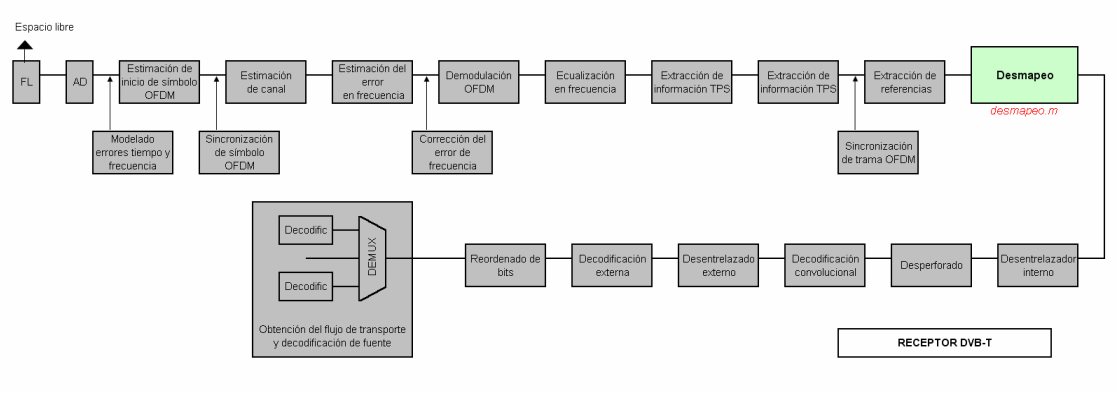


Fig. 6.37 Localización del bloque de detección y desmapeo

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *desmapeo.m*.

6.3.6.2 Objetivo y relación con el estándar

El bloque presentado en esta sección realiza la operación de detección y desmapeo de los símbolos PAM transmitidos en las portadoras del símbolo OFDM. Se trata de la operación complementaria a la presentada en el apartado 4.2.7.

Se han realizado simulaciones para determinar el comportamiento del sistema en función del tipo de constelación utilizada en condiciones de presencia de ruido. La Fig. 6.40 muestra las respuestas de BER frente a *Eb/N0* para las constelaciones QPSK, 16-QAM y 64-QAM al transmitir símbolos en modo 2k y se comparan con las curvas teóricas de BER que se presentan en el anexo B.1. Se puede observar como la modulación más robusta al ruido es la modulación QPSK, que tiene mayor separación entre los símbolos de la constelación por lo que los márgenes de decisión son mayores. Se observa como la tendencia de cada curva de simulación cumple lo especificado por la forma teórica

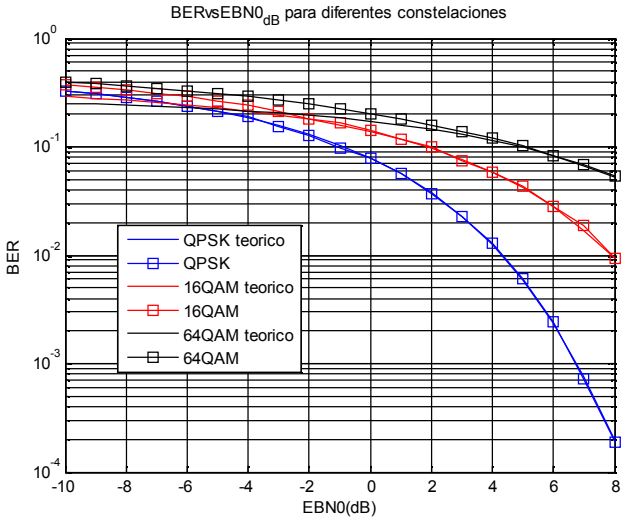


Fig. 6.38 BERvsEb/N0 para las distintas constelaciones

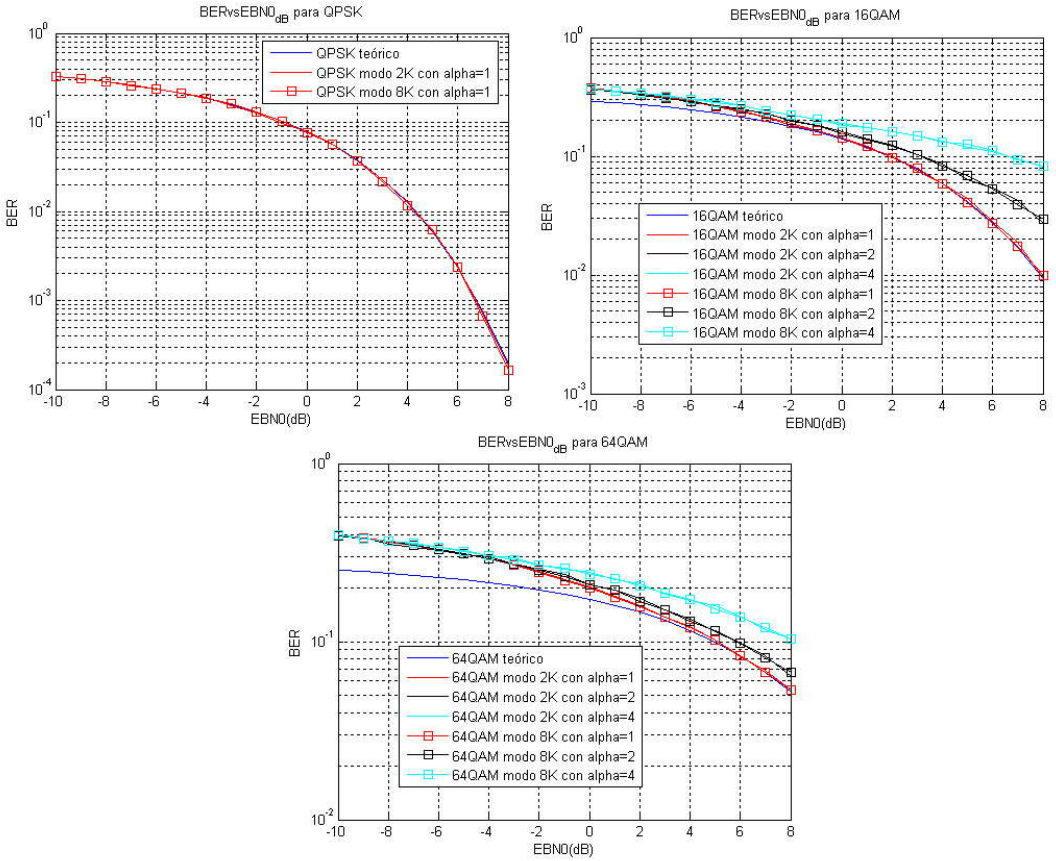


Fig. 6.39 BERvsEb/N0 en función del parámetro α en función de las constelaciones

La Fig. 6.39 muestra los resultados de simulación de BER frente a *Eb/N0* para las distintas modulaciones en función del modo de transmisión (2k u 8k) y en función del parámetro α, todas ellas comparadas con la curva de BER teórica para el caso uniforme (α=1). La primera conclusión que se observa es que en canales ideales con presencia de ruido AWGN el modo de transmisión no repercute en la robustez de la señal frente al ruido, es decir, las curvas correspondientes a los dos modos de transmisión para un mismo esquema de modulación tienen la misma tendencia, sea cual sea el valor del parámetro α.

La característica principal que introduce la no uniformidad en el mapeo de símbolos ($\alpha \neq 1$) destaca en la diferencia de nivel de protección de los bits frente al ruido. Es fácilmente apreciable que en las transmisiones no uniformes el BER empeora y este hecho se agudiza a medida que el parámetro α aumenta. No obstante, hay veces que es imprescindible priorizar un tipo de información respecto de otro y, en estos casos, se puede hacer gracias a esta irregularidad en el nivel de protección de los datos. El mecanismo funciona de la siguiente manera: en todas las constelaciones los dos bits de mayor peso que conforman cada símbolo determinan la polaridad de la parte real y de la parte imaginaria (b3 y b2 en 16QAM y b5 y b4 en 64QAM). Así, en los bits más sencillos de detectar, que son los que determinan el cuadrante al que corresponde el símbolo, se introduce la información más importante y en los demás la menos importante. Resumiendo, en modo no uniforme se conseguirá un BER menor en los dos bits de polaridad que en el caso uniforme (se aprovecha para enviar datos importantes) mientras que los demás bits quedan penalizados (al normalizar en potencia) ofreciendo menor protección contra el ruido (se aprovecha para enviar la información menos relevante).

6.3.6.3 Código implementado

```
function no_map_out = desmapeo(data_in,n,alpha,S,N,const)
if (n==4)
fprintf('·Rx: Desmapeo del símbolo OFDM %g desnormalizado con constelación QPSK con alpha=%g',S,alpha);
else
fprintf('·Rx: Desmapeo del símbolo OFDM %g desnormalizado con constelación %gQAM con
alpha=%g',S,n,alpha);
end
if const==1                                          Para activar al plot poner a 1
figure(1)
hold on
plot(data_in+i*eps, '.r');
title(sprintf('Constelación %gQAM unitaria del simbolo
legend('Indica zona de constelacion','Señales PAM enviadas','Señales PAM recibidas')
hold off
end
%Cálculo de energía media de la constelación QAM
Es=0;
Estot=0;
for u=alpha:2:(ceil(sqrt(n)))-1+alpha
  for u=alpha:2:(ceil(sqrt(n)))-1+alpha
      Es=Es+(u^2+u^2);
   end
   Estot=Estot+Es;
   Es=0;
end
Estotmedia=Estot/(n/4);                              Media entre el número de símbolos de un cuadrante
data_in=data_in*sqrt(Estotmedia);                    Ya tenemos nuestra señal con energía inicial (no
fprintf('.');
%    Detección y Desmapeo de bits
  switch n
    case 4
      orden = 1:2;
    case 16
      orden = 1:4;
    case 64
      orden = 1:6;
  end
  no_map_out = zeros(N,log2(n));                     Inicializamos variable donde se van  bits detectados
  for u = 1:N
    simbolo = [real(data_in(u)) imag(data_in(u))];
    bits = zeros(1,log2(n));
 %Eliminamos el offset provocado por alpha en caso no uniforme
    simbolo = simbolo - (alpha-1).* sign(simbolo);
    amplitud = 2^(log2(n)/2-1);
    confidence = 2^(log2(n)/2-1);
    for v = 1:2:log2(n)
      bit1 = -0.5*(simbolo(1) / confidence) + 0.5;
      bit2 = -0.5*(simbolo(2) / confidence) + 0.5;
      bits(orden(v)) = bit1;
      bits(orden(v+1)) = bit2;
      simbolo = abs(simbolo) - amplitud;
      amplitud = amplitud / 2;
    end
    no_map_out(u,:) = bits;
  end
fprintf('.\tOK\n');
```

Código 28 desmapeo.m

6.4 DECODIFICACIÓN DE CANAL

En el apartado 4.2 se introduce el concepto de codificación de canal y se presentan los bloques encargados de realizarlo. En este apartado se presentan los bloques que realizan la operación complementaria para decodificar las tramas de bits procedentes del demapeador. El bloque de decodificación de canal comprende los sub-bloques de desentrelazado interno y adaptación al perforado aplicado, decodificación convolucional, desentrelazado externo, decodificación externa y, para finalizar, reordenado de bits.

6.4.1 Desentrelazado interno

Este bloque realiza la operación complementaria a la que realiza el bloque de entrelazado interno presentado en el apartado 4.2.6.

6.4.1.1 Localización del bloque en el sistema

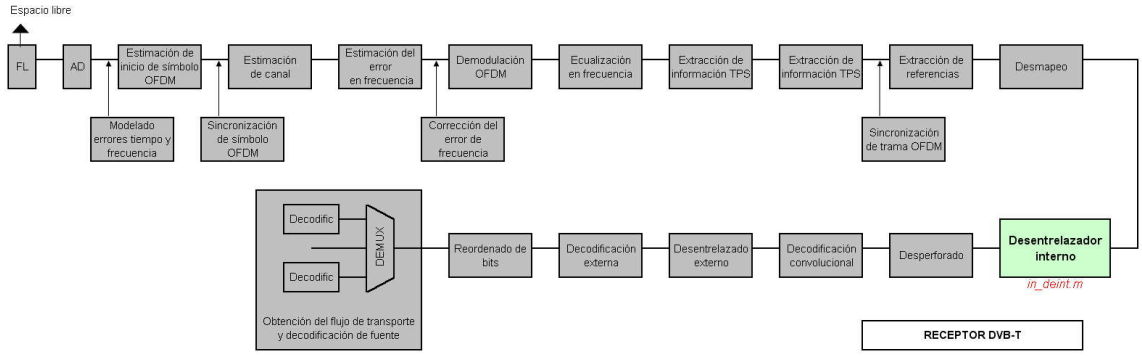


Fig. 6.40 Localización del bloque de desentrelazado interno

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *in_deint.m.*

6.4.1.2 Código implementado

```
1  function simbol_out = in_deint(simbol_in,N,n,S,fft_mode,S_index)
2  fprintf('·Rx: Paso del símbolo %g por etapa de Desentrelazado interno (Inner Deinterleaving)',S);
3  fprintf('.');
4  m=N*log2(n);
5  interliv_bits=126;
6  tam_bloque=interliv_bits*log2(n);
7  [long_simb,nada]=size (simbol_in);
8  num_blocs=long_simb/interliv_bits;
9  simbol_out=zeros(m,1)                          Preparamos el vector que contendrá los datos
10 %Primero calculamos Heq (Ver ETSI 300.744 v1.51 pag 20 Tabla 3)
11 Nmax = N;
12 Mmax = fft_mode;
13 Nr = log2(Mmax);
14 H = zeros(Nmax,1);
15   qq = 0;
16   switch fft_mode
17     case 2048
18       perm_table=[4 3 9 6 2 8 1 5 7 0];
19     case 8192
20       perm_table=[7 1 4 2 9 6 8 10 0 3 11 5];
21   end
22   for ii = 0:Mmax-1
```

```
23  switch ii                                          Cálculo Ri
24   case 0
25         r1 = zeros (1, Nr-1);
26       case 1
27         r1 = zeros (1, Nr-1);
28       case 2
29         r1 = zeros (1, Nr-1);
30         r1(1) = 1;
31       otherwise
32            if fft_mode ==2048
33              r1 = [r1(2:Nr-1) , xor(r1(1+0), r1(1+3))];
34            else
35              xor1= xor(r1(1+0), r1(1+1));
36              xor2= xor(r1(1+4), r1(1+6));
37              r1 = [r1(2:Nr-1) , xor(xor1,xor2)];
38            end
39     end
40     r = zeros(1, Nr-1);
41     for k = 0:Nr-2
42       r(1+perm_table(1+k)) = r1(1+k);
43     end
44         H(1+qq) = rem(ii,2) * 2^(Nr-1);                Cálculo H(q)
45  for jj = 0:Nr-2
46       H(1+qq) = H(1+qq) + r(1+jj) * 2^jj;
47     end
48     if H(1+qq) < Nmax
49       qq = qq + 1;
50     end
51    end
52  %   Revertimos el entrelazador de simbolos
53  y=zeros(m/log2(n),log2(n));
54  for i=1:m/log2(n)
55     if rem(S_index-1,2)==0
56        y(i,:)=simbol_in(1+H(i),:);
57     else
58        y(1+H(i),:)=simbol_in(i,:);
59     end
60  end
61  fprintf('.');
62  for i=1:num_blocs
63     a=y(1+(i-1)*interliv_bits:i*interliv_bits,:);
64  %   Revertimos el entrelazador de bits
65  b=zeros(interliv_bits,log2(n));
66  h_param=[0 63 105 42 21 84];
67  for e=0:log2(n)-1
68     for w=0:interliv_bits-1%                        Cálculo de  H(e,w)
69        h=rem(w+h_param(1+e), interliv_bits);
70        b(1+h,1+e)=a(1+w,1+e);
71     end
72  end
73  %            Multiplexador
74  a = zeros (interliv_bits, log2(n));
75    switch n
76      case 4
77        mapeo=[0 1];
78      case 16
79        mapeo=[0 2 1 3];
80      case 64
81        mapeo=[0 2 4 1 3 5];
82    end
83  for k = 1:log2(n)
84     a(:,k)=b(:,1+mapeo(k));
85  end
86  simbol_out(1+(i-
87  end
88  fprintf('.\tOK\n');
```

Código 29 in_deint.m

6.4.2 Decodificación interna

Este bloque realiza la operación complementaria a la que realiza el bloque de codificación interna formada por la etapa de codificación convolucional (apartado 4.2.4) y por la etapa de perforado (apartado 4.2.5). No se ha implementado decodificación de Viterbi (Hard) para detección y corrección de errores.

6.4.2.1 Localización del bloque en el sistema

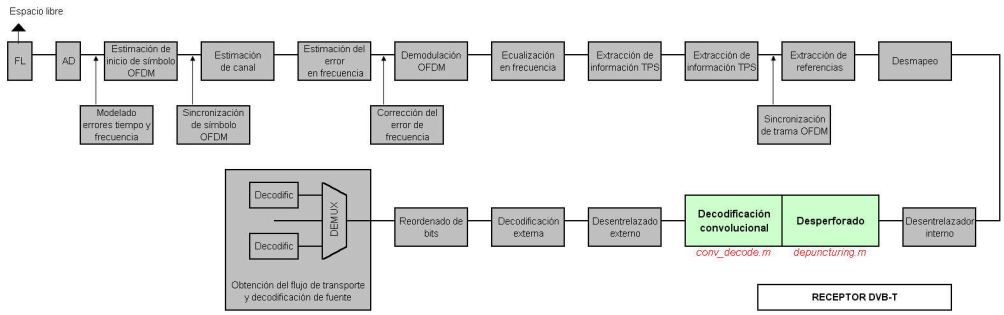


Fig. 6.41 Localización del bloque de decodificación interna

Los ficheros Matlab que implementan el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), son *depuncturing.m* y *conv_decode.m*.

6.4.2.2 Código implementado

```
1  function [X,Y]=depuncturing(data_in,S,C);
2  fprintf('·Rx: Paso del símbolo %g de code rate %g por etapa de Depuncturing para adaptación a code rate de
3  fprintf('.');
4  m=C*length(data_in);
5    switch C                                          VER ETSI EN 300 744 v1.1.1 par 14 Tabla 2
6       case 1/2
7       X=zeros(m,1);
8       Y=zeros(m,1);
9         for i = 1:m
10        X(i)=data_in(2*(i-1)+1);
11        Y(i)=data_in(2*(i-1)+2);
12      end
13     case 2/3
14      X=zeros(m,1);
15      Y=zeros(m,1);
16        for i = 1:2:m
17        X(i)=data_in(3*(i-1)/2+1);
18        X(i+1)=1/2;
19        Y(i)=data_in(3*(i-1)/2+2);
20        Y(i+1)=data_in(3*(i-1)/2+3);
21        end
22     case 3/4
23      X=zeros(m,1);
24      Y=zeros(m,1);
25      for i = 1:3:m
26        X(i)=data_in(4*(i-1)/3+1);
27        X(i+1)=1/2;
28        X(i+2)=data_in(4*(i-1)/3+2);
29        Y(i)=data_in(4*(i-1)/3+3);
30        Y(i+1)=data_in(4*(i-1)/3+4);
31        Y(i+2)=1/2;
```

```
32      end
33      case 5/6
34      X=zeros(m,1);
35      Y=zeros(m,1);
36      for i = 1:5:m
37        X(i)=data_in(6*(i-1)/5+1) ;
38        X(i+1)=1/2;
39        X(i+2)=data_in(6*(i-1)/5+2);
40        X(i+3)=1/2;
41        X(i+4)=data_in(6*(i-1)/5+3);
42        Y(i)=data_in(6*(i-1)/5+4);
43        Y(i+1)=data_in(6*(i-1)/5+5);
44        Y(i+2)=1/2;
45        Y(i+3)=data_in(6*(i-1)/5+6);
46        Y(i+4)=1/2;
47      end
48      case 7/8
49      X=zeros(m,1);
50      Y=zeros(m,1);
51      for i = 1:7:m
52        X(i)=data_in(8*(i-1)/7+1) ;
53        X(i+1)=1/2;
54        X(i+2)=1/2;
55        X(i+3)=1/2;
56        X(i+4)=data_in(8*(i-1)/7+2);
57        X(i+5)=1/2;
58        X(i+6)=data_in(8*(i-1)/7+3);
59        Y(i)=data_in(8*(i-1)/7+4);
60        Y(i+1)=data_in(8*(i-1)/7+5);
61        Y(i+2)=data_in(8*(i-1)/7+6);
62        Y(i+3)=data_in(8*(i-1)/7+7);
63        Y(i+4)=1/2;
64        Y(i+5)=data_in(8*(i-1)/7+8);
65        Y(i+6)=1/2;
66      end
67  fprintf('.');
68  end
69  fprintf('.\tOK\n');
```

Código 30 depuncturing.m

```
1   function [data_out,estado]= conv_decode(X,Y,S,estado)
2   fprintf('·Rx: Paso del símbolo %d por etapa de Inner Decoding (Decodificación convolucional)',S);
3   fprintf('.');
4   registros=6;
5   n=length(X);
6   conv_code_X=[1 2 3 6];
7   conv_code_Y=[2 3 5 6];
8   data_out=zeros(n,1);
9   fprintf('.');
10  for k=1:n
11   data_x=X(k);
12   for v=conv_code_X
13    if estado(v) < 0.5   data_x = data_x;
14    elseif data_x == 0.5     data_x = 0.5;
15    elseif estado(v) > 0.5      data_x = 1-data_x;
16    end
17   end
18   data_y=Y(k);
19   for v=conv_code_Y
20    if estado(v) < 0.5     data_y = data_y;
21    elseif data_y == 0.5     data_y = 0.5;
22    elseif estado(v) > 0.5     data_y = 1-data_y;
23    end
24   end
25   if data_x<0.5||data_y<0.5      data_out(k)=0;
26   elseif data_x>0.5||data_y>0.5  data_out(k)=1;
27   end
28   estado=[data_out(k),estado(1:registros-1)];
29  end
30   fprintf('.\tOK\n')
```

Código 31 conv_decode.m

6.4.3 Desentrelazado externo

Este bloque realiza la operación complementaria a la que realiza el bloque de entrelazado externo presentado en el apartado 4.2.3.

El bloque de desentrelazado tiene un principio similar al bloque de entrelazado con la salvedad de que los índices de las ramas están invertidos, es decir, "j=0" se corresponde al máximo retardo. La sincronización se obtiene haciendo pasar el primer byte SYNC reconocido, invertido o no, hacia la rama "0".

En estas condiciones, en el bloque de desentrelazado cada byte se retarda un total de posiciones igual a

$$(11 - j)\cdot 17 \qquad (6.60)$$

por lo que el retardo total entre emisión y recepción es de (j+11-j)=187, valor que permite recuperar en la recepción el orden original al ser un retardo idéntico para todos los bytes .

Todo este proceso reduce los errores por ráfagas introducidos por el canal de transmisión (errores que afectan a varios bytes consecutivos), ya que, después de la reordenación de los datos en el receptor, estos errores se habrán distribuido entre paquetes sucesivos. Este hecho favorecerá que no se excedan los límites en los que la codificación RS puede recuperar la información original.

6.4.3.1 Localización del bloque en el sistema

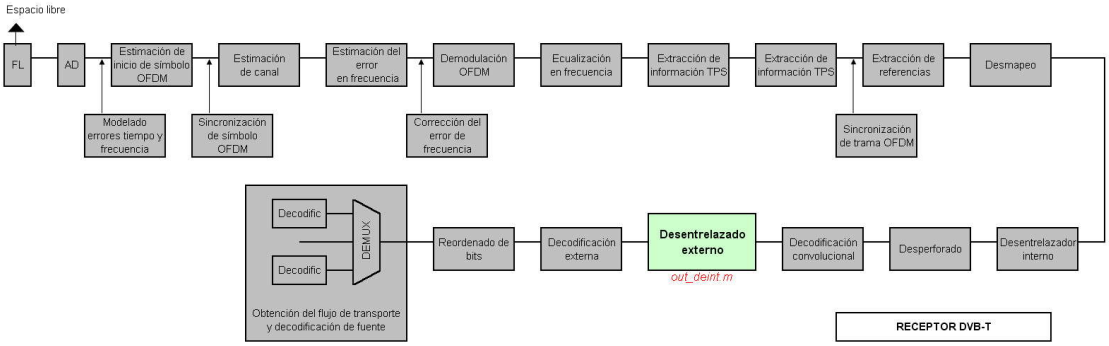


Fig. 6.42 Localización del bloque de desentrelazado externo

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *out_deint.m*.

6.4.3.2 Código implementado

```
1  function [trama_out,cola_out] = out_deint(trama_in,I,M,N,cola_out,B)
2  fprintf('·Rx: Paso del bloque %d por etapa de Outer Deinterleaving (Entrelazado externo)',B);
3  fprintf('.');
4  for u = 1:M
5     for i = 1:I
6       cola_out((I-i)*N + (u-1)*I + i) =trama_in((u-1)*I + i);     %Aquí trama_in es una columna con el valor de cada
7     end                                                          Poner (I-i)*N en vex de (i-1)*N
8  end
9  fprintf('.');
10 trama_out = cola_out(1:N);                                      La info se encuentra al final de la cola
11 cola_out = [cola_out(N+1:(I^2*M)); zeros(N,1)];
12 fprintf('.\tOK\n');
```

Código 32 out_deint.m

6.4.4 Decodificación externa

Este bloque realiza la operación complementaria a la que realiza el bloque de codificación externa presentado en el apartado 4.2.2. Decodifica las tramas de bits recibidas y codificadas con código Reed-Solomon. Del bloque MPEG2 de 204 bytes se queda con los 188 bytes primeros para formar la trama MPEG2.

En este proyecto no se aplica algoritmo de detección y corrección de errores pero existen varios métodos para su detección y corrección en caso de no tener una transmisión libre de errores (Algoritmo Berlekamp-Massey).

6.4.4.1 Localización del bloque en el sistema

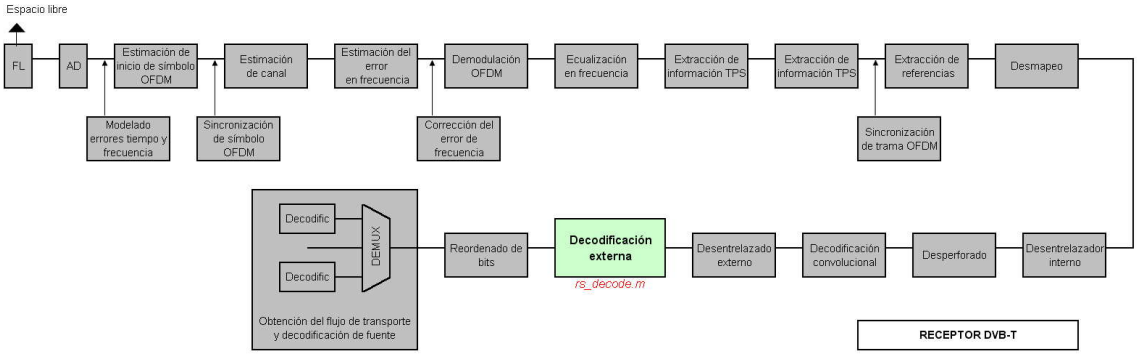


Fig. 6.43 Localización del bloque de decodificación externa

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *rs_decode.m*.

6.4.4.2 Código implementado

```
1  function [trama_out,cola_out] = out_deint(trama_in,I,M,N,cola_out,B)
2  fprintf('·Rx: Paso de la trama %d por etapa de Outer Decoding (Reed Solomon)',T);
3  trama_out=trama_in(1:k);                    Se queda con los primeros 188 bytes
4  fprintf('.OK\n');
```

Código 33 rs_decode.m

6.4.5 Reordenado de bits. Sincronización de trama MPEG

Este bloque realiza la operación complementaria a la que realiza el bloque de adaptación y dispersión de energía presentado en el apartado 4.2.1. Este bloque se ocupa de recibir las tramas MPEG2 y procesarlas a partir de un generador PRBS para obtener las secuencias de bits transmitidas sin codificar.

6.4.5.1 Localización del bloque en el sistema

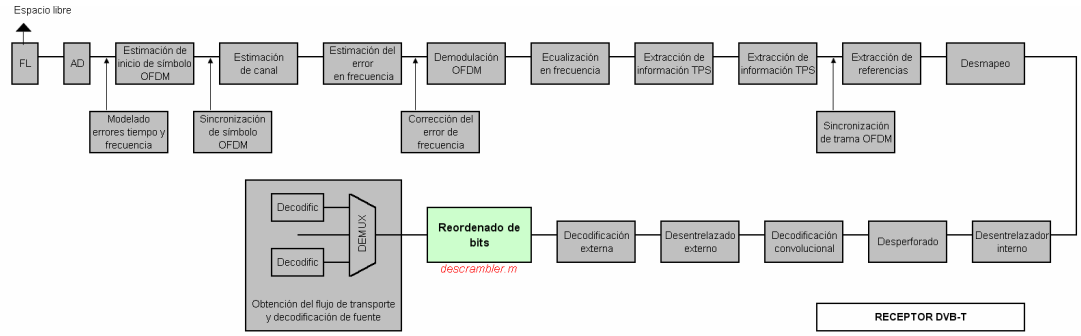


Fig. 6.44 Localización del bloque de reordenado de bits

El fichero Matlab que implementa el algoritmo del bloque que se estudia en este apartado, y que se ejecuta desde el programa principal *DVBT_simulado.m* (Véase Anexo A), es *descrambler.m*.

6.4.5.2 Objetivo y relación con el estándar

Antes de procesar dichas tramas, el bloque de reordenado de bits realiza un proceso de sincronización de trama MPEG2. La estructura de las tramas MPEG2 se presenta en el apartado 4.2.1.2 Las tramas MPEG2 recibidas sin cabecera válida (184_{DEC} o 71_{DEC}) son descartadas. Además, si la primera trama recibida por el bloque no es la trama con cabecera 184_{DEC} se descarta, es decir, el procesado de datos empieza a partir de la recepción de la primera trama, de 188 bytes, cuya cabecera (1 byte) tiene el valor decimal 184. A partir de aquí se procesan las tramas con cabecera válida, es decir, las tramas MPEG2 que tienen el valor decimal 74 en su cabecera o bien, para la primera trama de las ocho tramas MPEG2 que forman una supertrama MPEG2, tienen el valor decimal 184.

El descarte de tramas por cabecera no válida se puede producir debido a que no se ha detectado bien la señal, debido a bajas prestaciones de la transmisión, o bien porque se trata de las primeras tramas que inicializan la cola del bloque desentrelazador externo.

6.4.5.3 Código implementado

```
function [trama_out, estado_r_prbs_out,T_index,T] = descrambler( trama_in, T_index,B, estado_r_prbs_in,T)
SYNC_t=trama_in(1);                                  %Miramos el primer byte de la trama...
switch SYNC_t
 case 184
      if rem((T_index-1),8)~=0              T_index=0;
         fprintf('No se ha empezado desde la primera trama (cabecera 184)-->Resincronizando\n')
      end
   estado_r_prbs_in=[1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0];     Se inicializa registro prbs (184dec)
%Aplicamos scrambler al resto de tramas de datos
     secuencia_prbs=zeros(8*1,1);                    Vector fila en bits
      for i=1:8*1
      bit_nuevo=xor(estado_r_prbs_in(14),estado_r_prbs_in(15));
      estado_r_prbs_in=[bit_nuevo, estado_r_prbs_in(1:14)];
      secuencia_prbs(i)=bit_nuevo;
      end
      secuencia_prbs_bytes=reshape(secuencia_prbs,8,1).';
      secuencia_prbs_bytes=bi2de(secuencia_prbs_bytes,'left-msb');
      estado_r_prbs_out=estado_r_prbs_in;
 otherwise
     trama_out=[];                                    Por si son las primeras tramas que vienen del outer
      estado_r_prbs_out=estado_r_prbs_in;             deinterleaver que han servido para inicializar la cola y
      T_index=T_index;        T=T;                    dar tramas con encabezado
      fprintf('\tLa trama %g no tiene encabezado (184 o 71). Ha inicializado la cola del Outer Dinterleaver\n',B)
      return
end
secuencia_prbs=zeros(1,8*187);                       Vector fila en bits
for i=1:(8*187)
      bit_nuevo=xor(estado_r_prbs_in(14),estado_r_prbs_in(15));
      estado_r_prbs_in=[bit_nuevo, estado_r_prbs_in(1:14)];
      secuencia_prbs(i)=bit_nuevo;
end
secuencia_prbs_bytes=bi2de(reshape(secuencia_prbs,8,187)','left-msb');
estado_r_prbs_out=estado_r_prbs_in;                  Si hemos llegado aqui se supone que es una trama
T_index=T_index+1;  T=T+1;                           recibida útil. No sirve para inicializar la cola del out
%-----------------------------------------------     deint!!!
% Formación de la trama (SYNC+prbs)
%-----------------------------------------------
trama_out=[71; bitxor(trama_in(2:188),secuencia_prbs_bytes)];
fprintf('\tLa trama %g tiene encabezado (184 o 71). Corresponde a la trama %g de información útil \n',B,T)
```

Código 34 scrambler.m

Capítulo 7 CONCLUSIONES Y PROPUESTAS DE FUTURO

7.1 CONCLUSIONES

En este proyecto se ha desarrollado un sistema software de transmisión fiel al estándar DVB-T, siendo el modulador OFDM el sub-sistema en que más interés se ha prestado. Tal y como se ha comentado en numerosas ocasiones, dicho esquema de modulación tiene múltiples ventajas frente a modulaciones de portadora única en aplicaciones de banda ancha. Pero, por contra, presenta unos requisitos de sincronización extremadamente elevados. Por ese motivo, el objetivo principal ha sido el de implementar técnicas de estimación y sincronización que ayuden al sistema OFDM a demodular la señal recibida sin problemas, aprovechando las propiedades inherentes a su estructura combinada con la disposición de las portadoras especificadas por el sistema DVB-T.

Tras una extensa recopilación bibliográfica y posterior estudio se ha conseguido entender y desarrollar un sistema transmisor DVB-T fiel al estándar, teniendo en cuenta todas sus especificaciones para todos los modos de operación. Una vez implementado el sistema transmisor y validado en condiciones ideales se ha procedido ha modelar los problemas típicos de los enlaces inalámbricos: ruido AWGN, modelo de canal y errores de sincronización y fase. Cabe la pena mencionar que el modelado de éstos no ha sido una tarea sencilla puesto que, desde el punto de vista de implementación software, ha requerido el diseño de una estructura de flujo de datos muy elavorada (por ejemplo el uso de buffers que posibilitaran modelar el error de tiempo). A partir de aquí se ha podido realizar un conjunto de simulaciones, configuradas fácilmente desde un fichero de texto, a partir de las cuales se han podido determinar los efectos negativos provocados por los diferentes aspectos comentados anteriormente en las señales y su posterior corrección a partir de las implementaciones realizadas.

Se ha podido determinar que el uso de la estructura de las señales DVB-T, que utilizan un conjunto de símbolos pilotos, es del todo práctica para la implementación de algoritmos estimadores y de sincronización, haciendo que se puedan aplicar algoritmos sencillos y, en relación, eficientes.

Además se ha podido ver que, además de la sincronización de tiempo y fase requerida, se especifica la necesidad de un conjunto de sincronizaciones de trama OFDM (a nivel de modulación OFDM por portadoras piloto) y de trama MPEG (a nivel de decodificación de canal, en el bloque de ordenado de bits o *descrambler*). Debido a la experiencia adquirida al realizar las diferentes simulaciones (costosas en tiempo y procesado) se ha podido de terminar que el más mínimo de sincronización de error de frecuencia y tiempo imposibilitan la sincronización a este nivel puesto que es una

sincronización basada en secuencias de sincronismo (o cabeceras) y los valores que se reciben de éstas resultan confusas para el receptor en estos casos (imposibilitando la sincronización de trama OFDM o de trama MPEG hasta llegar a valores elevados de SNR)

Como resultado de este proyecto se ha obtenido una plataforma de simulación DVB-T/OFDM muy portable y de fácil configuración que deja abierta la posibilidad de aplicar nuevos estudios teniendo en cuenta múltiples efectos de degradación de la señal.

7.2 PROPUESTAS DE FUTURO

El sistema de modulación OFDM tiene muchísimas aplicaciones gracias a las ventajas que tiene y a las facilidades que ofrece para solucionar problemas inherentes a su estructura. Debido a esto, son muchas las áreas de estudio que se pueden tener en cuenta y en las que se puede profundizar a partir de la implementación de nuevos algoritmos. Entre estas áreas se encuentran el modelado de canal y su estimación, comparación de diferentes estimadores de tiempo y frecuencia (gruesa y fina), comparación entre diferentes técnicas de sincronización de símbolo y de trama. En definitiva, profundizar de una manera más notable en áreas específicas de OFDM.

En el caso de este proyecto se ha partido de la estructura de datos y de señales del estándar DVB-T, cuyas especificaciones de diseño se cumplen con totalidad. De hecho hay múltiples estándares de transmisión que utilizan OFDM en su esquema de transmisión por lo que las especificaciones de diseño varían según necesidades. Partiendo del sistema OFDM como núcleo central de nuevas plataformas software, se pueden extender los estudios propuestos anteriormente a nuevos estándares, pudiendo comparar las prestaciones que ofrece OFDM en cada uno de ellos. Por ejemplo, si se usa el estándar DVB-H gran parte de la implementación software está realizada pero se debe adaptar el diseño a las especificaciones de su estándar. Por otro lado, al ser un esquema de transmisión terrestre para receptores móviles, permite el estudio de sus prestaciones en unas condiciones diferentes a las que se han estudiado en este proyecto (diferentes tipos de canales, codificaciones particulares...).

Otra propuesta que ayudaría a completar el proyecto es el estudio e implementación de bloques de codificación para la protección contra errores. Por ejemplo, en el bloque de decodificación convolucional se puede implementar el algoritmo de Viterbi para la detección y corrección de errores, así como en el bloque de decodificación externa (Reed-Solomon) se puede implementar un algoritmo para detectar y corregir errores que el bloque anterior no ha podido corregir.

En este proyecto se ha desarrollado, entre otras cosas, una plataforma software basada en el esquema de transmisión DVB-T e implementada en el lenguaje de programación basado en el entorno Matlab®. Partiendo de esta base existen muchísimas posibilidades a la hora de realizar propuestas de futuro.

Referente al entorno de programación se propone implementar el procesador mediante otro lenguaje de programación que resulte más versátil, portable y que ofrezca mayor velocidad de procesado, tal como puede ser C++ o IDL. En cuanto a los algoritmos implementados, éstos pueden ser nuevamente implementados en lenguaje de programación hardware para ser ejecutados en FPGAs. Además, sea cual sea el entorno de programación, se propone una tarea de optimización de código y posterior creación de una interfaz de usuario (GUI) que facilite la configuración de las simulaciones y recoja y almacene de manera ordenada los resultados obtenidos de una manera automatizada.

Como proyecto estrella se podría depurar el simulador implementado de manera que permitiera hacer frente a la demodulación y decodificación de una señal real capturada a la entrada de un receptor. Se habla de depurar debido a que en el paso de un software de

simulación con datos sintéticos a un software de procesado de datos reales siempre aparecen problemas que no se han modelado en el simulador o que necesitan de nuevos algoritmos para poder regenerar correctamente la señal recibida. No es una idea descabellada el utilizar un software para la reproducción de los datos de video y audio recibidos por radiodifusión, de hecho existen varios software que realizan lo comentado con las señales GPS.

Anexo A MANUAL DE USUARIO

Uno de los objetivos principales de este proyecto es el de obtener una plataforma software de transmisión DVB-T en la que poder implementar y estudiar el comportamiento y las prestaciones tanto de nuevas aplicaciones de estimación y sincronización como de codificación de canal. Con la finalización de este proyecto queda realizado todo el sistema de transmisión y recepción DVB-T con la inclusión de diferentes bloques de estimación y sincronización, ya presentados y estudiados a estas alturas.

Para que el sistema de simulación implementado pueda ser reutilizado para continuar con nuevas implementaciones o, incluso, llegar a decodificar secuencias reales partiendo de tramas OFDM en recepción, es necesario que la estructura y el código del simulador quede bien definido y presentado. Por ello, en este anexo se ha querido recoger el conjunto de prestaciones del programa, su estructura y diagrama de flujo, información de variables de entrada y de salida, de los parámetros de configuración de las simulaciones y, finalmente, el código implementado.

Para facilitar la comprensión de la estructura y del código se utilizan herramientas visuales, es decir, se utiliza un código de colores que relaciona cada parte de la estructura del simulador con cada parte del código implementado. Se recomienda consultar la Fig A.1 a la vez que se consulta el código presentado en el apartado A.5.

El simulador ha sido programado con lenguaje MATLAB®. Se ha utilizado la versión 7.0.24704 (R14) Service Pack 1, con fecha de lanzamiento el 13 de Septiembre de 2004. Para el funcionamiento del simulador, además de las funciones implementadas en este proyecto, únicamente son necesarias las librerías básicas de Matlab, la librería específica *Communication Toolbox* (para poder utilizar las funciones *bchenc.m, bchdec.m, gf.m* y *bchgempoly.m* en la etapa de generación y extracción de información TPS).

El simulador ha sido diseñado para que la inclusión de nuevos bloques sea sencilla y para que el flujo de datos sea rápido y óptimo.

Para cada *M-file* implementada se ha seguido también una misma estructura, ya sea para la función principal como para las funciones auxiliares que desempeñan la labor de cada bloque implementado. Se parte de una descripción teórica de la función en cuestión (que no se incluye en los códigos presentados en este documento, puesto que aquí ya se recoge toda la teoría necesaria), preparación de variables que intervienen, presentación por pantalla de la trama MPEG o símbolo OFDM que se procesa por el bloque, implementación del algoritmo y por último se imprime por pantalla es estado de la operación (OK o fallido).

A continuación se presenta una descripción más meticulosa del programa implementado.

A.1 ESTRUCTURA DEL SIMULADOR DVB-T IMPLEMENTADO

El simulador, ejecutado con el fichero principal *DVBT_simulado.m*, modela un sistema trasmisor-receptor DVB-T. En la Fig A.1 se ilustra un esquema en el que se presenta la arquitectura del programa. Se destacan 5 bloques:

- Bloque de SETUP: encargado de preparar la simulación de transmisión. Obtiene los parámetros de transmisión del fichero de texto *parámetros.txt*. (Véase A.2) y configura la transmisión: generación de bits suficientes para realizar la transmisión, inicialización y preparación de variables y constantes. Además valida los parámetros de transmisión introducidos por el usuario y si no se han insertado de manera correcta imprime un aviso identificando el problema y la solución.
- Bloque TRANSMISOR: realiza todas las funciones de los bloques del transmisor DVB-T que aparecen en el Capítulo 4.
- Bloque CANAL: realiza las funciones de los bloques de modelado de canal que aparecen en el Capítulo 5.
- Bloque RECEPTOR: realiza todas las funciones de los bloques del receptor DVB-T que aparecen en el Capítulo 6.
- Bloque TEST: encargado de almacenar vectores de información de prestaciones del sistema (Véase Anexo B), cálculo de BER y VAR, presentación por pantalla de un resumen de la transmisión, graficado de resultados y guardado de datos de interés en ficheros almacenados en carpetas (según tipo de datos).

Se ha hecho especial hincapié en diseñar el flujo de los datos que recorren el sistema. Inicialmente se generan los bits que van a ser utilizados para realizar la transmisión (puesto que uno de los parámetros de transmisión que define el usuario es el número de tramas MPEG o de símbolos OFDM a transmitir). Éstos quedan guardados en un contenedor o *buffer* que se va vaciando a medida que se van obteniendo y procesando las tramas MPEG que surgen de él. Éste proceso se repite con otro *buffer* que almacena tramas MPEG con codificación de canal para que el sistema OFDM extraiga de allí los bits para formar símbolos OFDM. El proceso inverso ocurre en el receptor. Existe otro *buffer* que almacena los símbolos OFDM que van a ser transmitidos por el canal. Dicho *buffer* se vacía a medida que éstos van pasando por el canal, con política FIFO, para pasar al primer bloque del receptor. En caso de que alguno de estos *buffers* esté vacío el flujo de los datos se dirige al *buffer* anterior para extraer nuevos datos a procesar.

Se han programado *buffers* de registro de datos en los que se copia el contenido de los datos procesados. Éstos, junto con las variables que llevan la cuenta de las tramas y símbolos procesados, sirven para poder realizar comparaciones de bits transmitidos y recibidos y poder determinar el BER o la varianza. Se puede decir que se utilizan estos *buffers* como puntos de test para evaluar una parte o todo el sistema.

Esta estructura que marca el recorrido del flujo de datos permite que no se carguen las variables con demasiada información, puesto que el número de bits que se transmiten para realizar simulaciones de BER y varianza es muy elevado. Vaciando los buffers a medida que se adquieren datos este problema de carga computacional se soluciona. El hecho de procesar símbolo a símbolo permite llevar un control del estado de la transmisión gracias a que cada bloque imprime por pantalla el número de la trama o símbolo que se está procesando tanto en transmisión como en recepción. De esta manera, además, se facilita la localización de los errores que surgen cuando se está programando una nueva aplicación.

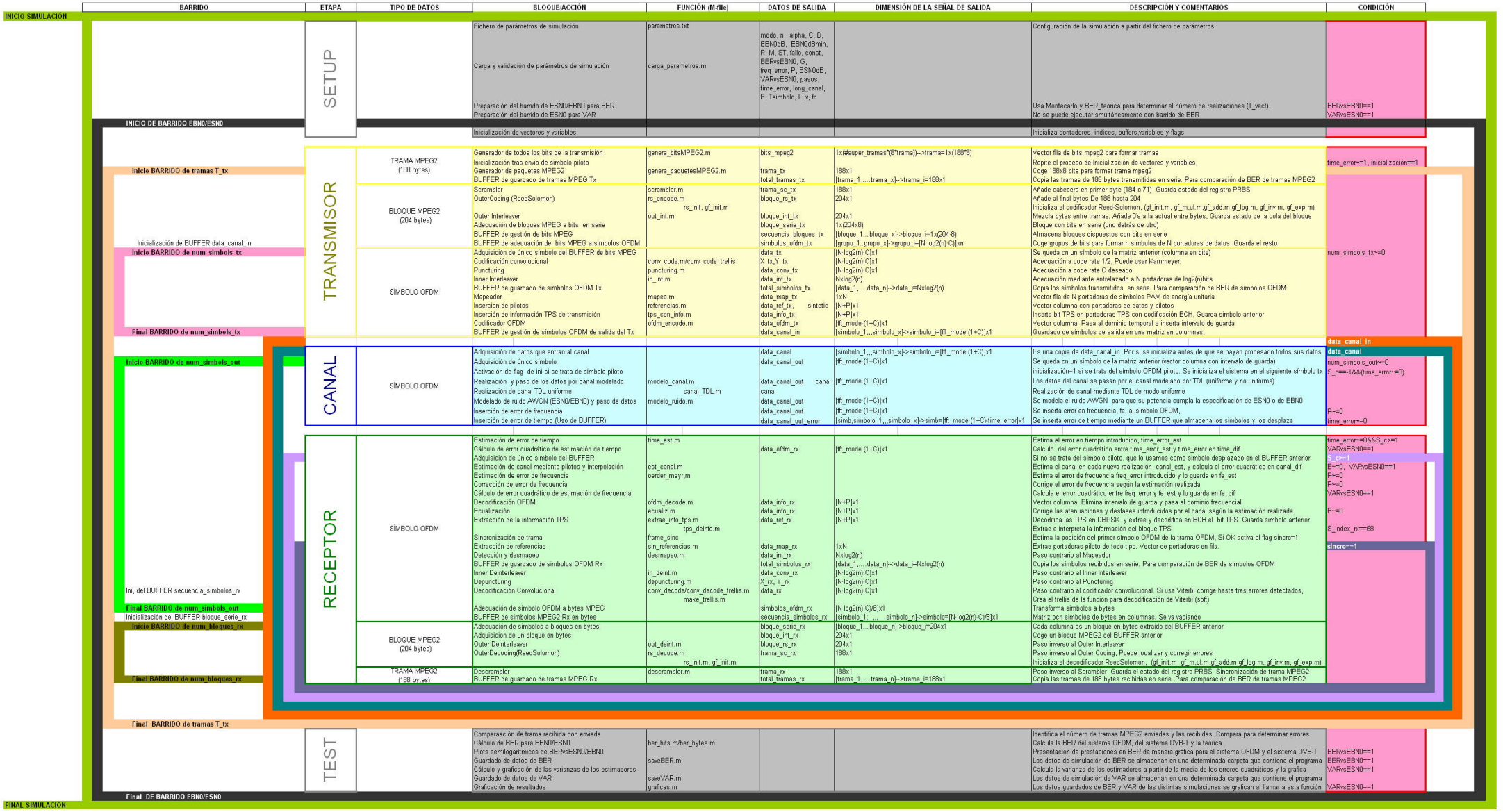


Fig A.1. Mapa conceptual del flujo de datos MPEG-2 del simulador DVBT_simulado.m

A.2 ESTRUCTURA DEL FICHERO DE PARÁMETROS DE SIMULACIÓN

Tabla A.1 Estructura del fichero de parámetros parametros.txt

Parámetro	Variable	Activación	Descripción	Condición
Ancho de banda del canal	B	5 6 7 8	Canales de 5 MHz de ancho de banda Canales de 6 MHz de ancho de banda Canales de 5 MHz de ancho de banda Canales de 8 MHz de ancho de banda	
Tipo de datos que define el bucle de simulación	G	0 1 2	Tramas MPEG2 Supertramas MPEG2 Símbolos OFDM	
Número de tramas a transmitir según tipo de datos (G)	ST	"X"	Determina el número de tramas MPEG2 a transmitir	Depende del valor de G
Modo de transmisión DVB-T	modo	2 8	Modo 2K Modo 8K	
Tipo de constelación	n	4 16 64	QPSK 16-QAM 64-QAM	
Parámetro de modulación	alpha	1 2 4	Uniforme No uniforme No uniforme	
Relación de codificación convolucional	C	1 2 3 4 5	1/2 2/3 3/4 5/6 7/8	
Longitud porcentual del intervalo de guarda	D	1 2 3 4	1/4 1/8 1/32 1/32	
Tipo de ruido de canal	R	0 1	Sin ruido Ruido AWGN	
Relación de energía de datos a ruido AWGN	ESN0dB	0 1	Enería de bit a energía de ruido Enerígia de símbolo PAM a energía de ruido	R==1 R==1
Valor de la relación de energía de datos a ruido	EBN0dB	"X"	Si BERvsEBN0 o VARvsEBN0 están activados es el valor máximo de barrido	R==1
Valor de la relación de energía de datos a ruido mínimo	EBN0dBmin	"X"	Es el valor mínimo de barrido	BERvsEBN0==1 o VARvsEBN0==1
Tipo de canal TDL	M	0 1 2 3 4	Ideal TDL uniforme urbano Autopista (1 transmisor) Interior (2 transmisores) Ciudad (3 transmisores)	
Número de taps del modelo TDL niforme	L	"X"		M==1
Velocidad del receptor móbil (Km/h)	v	"X"	Para el cálculo del desplazamiento Doppler	M==1
Frecuencia de la primera portadora OFDM	fc	"X"	Para el cálculo del desplazamiento Doppler	M==1
Longitud del canal en símbolos OFDM	long_canal	"X"	Determina cada cuantos símbolos el canal cambia	M~=0
Estimación y Ecualización de canal	E	0 1	No estima el canal ni ecualiza los datos Estima el canal con LS y ecualiza los datos	M~=0
Error de frecuencia	freq_error	"X"	Dentro del intervalo (-0,5,0,5)	
Estimador de error de frecuencia	P	0 1 3 4	No se introduce error de frecuencia Exponente lineal del NL Exponente cúbico del NL Exponente 4 del NL	
Error de tiempo de inicio de simbolo	time_error	0 "X"	No añade error de inicio de simbolo Añade error de inicio de simbolo, lo estima y lo corrige	
Graficación de estados de simulación	const	0 1	No grafica nada Grafica constelaciones, correlaciones y otros datos de simulación	ST<8
Simulación por barrido de EBN0/ESN0 para gráfico	BERvsEBN0	0 1	No realiza el barrido Realiza el barrido	VARvsEBN0==0
Simulación por barrido de EBN0/ESN0 para gráfico de VAR	VARvsEBN0	0 1	No realiza el barrido Realiza el barrido para determinación de varianza de estimadores	BERvsEBN0==0
Intervalo de separación entre valores de EBN0	pasos	"X"	Indica el salto de valor entre EBN0/ESN0 máximo y mínimo para la simulación	BERvsEBN0==1 o VARvsEBN0==1

El fichero de texto *parametros*.txt es el fichero a partir del cual el usuario define la simulación que se va a realizar. Debido a la estructura del código implementado hay múltiples configuraciones que se pueden realizar:

- Simulaciones con único valor de EBN0 o ESN0 y simulaciones con barrido de EBN0 o ESN0. El primer tipo de simulación resulta útil para estudiar el comportamiento del sistema con unos parámetros fijos y el segundo sistema de simulación resulta útil para caracterizar el sistema para diferentes valores de potencia de ruido. En este segundo sistema de simulación se obtienen los gráficos correspondientes.
- El método de simulación con barrido permite obtener curvas de BER o curvas de varianza, no ambas a la vez. Las curvas de BER se utilizan para caracterizar el sistema y las curvas de varianza se utilizan para caracterizar el comportamiento de los estimadores.
- Las simulaciones pueden ser muy completas. Se puede lanzar simulaciones con múltiples configuraciones. Lo usual, al programar un bloque nuevo es lanzar una simulación en estado ideal de canal (sin canal) y ruido (sin ruido) para comprobar el correcto funcionamiento del sistema. Una vez se ha validado el funcionamiento lo usual es lanzar simulaciones en las que el ruido interviene y, cuando se ha caracterizado el funcionamiento del bloque, finalmente se lanzan simulaciones con presencia de canal. La estructura del programa permite configurar la simulación para que se realice con las condiciones que el usuario precise. Es decir, se contemplan todas las posibilidades de simulación.
- El simulador permite definir el número de bits a transmitir, definiendo éste número a partir de diferentes unidades de transporte. Se puede definir el número de bits a transmitir mediante un número de tramas MPEG2 a enviar, o bien un número de símbolos OFDM a enviar o bien un número de supertramas OFDM a enviar. Ésta opción resulta muy útil a la hora de optimizar tiempo. Si únicamente se quiere observar el comportamiento de un bloque recién implementado se enviarán pocas tramas MPEG2 o pocos símbolos OFDM (según el bloque implementado). En cambio, si se quiere caracterizar el sistema mediante gráficas de BER y varianza se deberán transmitir un elevado número de bits, según la expresión recogida en (C.4)
- El formato de barrido puede ser definido por el usuario desde el mismo fichero de parámetros de simulación. Se pude definir el intervalo de barrido de EBN0 o ESN0 e incluso el intervalo de salto entre cada valor. Los gráficos que resultan están preparados para ofrecer información del tipo de gráfico, unidades en los ejes e incluso leyenda. Se trata de un proceso automatizado.
- Para el caso de los bloques estimadores, se puede definir el tipo error que se añade a la señal (fase, tiempo, canal) con independencia de si su estimación y corrección están activadas. Esto permite que se puedan observar las prestaciones del sistema con o sin efectos de los errores provocados en la señal.
- Se puede activar una opción que permite graficar diferentes estados del proceso, entre ellos las constelaciones de los datos transmitidos y recibidos o la respuesta frecuencial del canal utilizado.
- Se puede definir el tipo de canal utilizado, ya sea ideal, multicamino, ruidoso o multicamino y ruidoso. Para el caso del modelado multicamino se puede escoger entre diferentes modelos implementados con TDL de distintos entornos.

La estructura del programa invita al usuario a estudiar el sistema implementado con diferentes configuraciones que permiten muchas posibilidades de una manera intuitiva.

La Tabla A.1 recoge el conjunto de posibilidades de configuración de la simulación a partir de sus variables de control y sus valores. Se añaden comentarios referentes al parámetro que activa cada variable y las condiciones de uso. En la Fig A.2 se muestra la estructura del fichero de parámetros de configuración de la simulación, *parametros.txt.*

```
-------------------------------------------------------------------------------------------------------
                            PARÁMETROS_DE_SIMULACIÓN_SISTEMA_DVB-T
-------------------------------------------------------------------------------------------------------
B           8   %Ancho de banda de canal (5, 6, 7, 8)MHz
ST          1   %Número de supertramas/tramasMPEG2/Simbolos OFDM que queremos enviar según parámetro G.
G           2   %Se refiere a 0-tramas MPEG2 1-supertramas MPEG2 2-simbolos OFDM
modo        8   %Modo de transmisión K (2 o 8)
n           4   %Tipo de nQAM (4,16 o 64)
alpha       1   %Modulación jerárquica/uniforme o no según parámetro alpha (1, 2 o 4)
C           1   %Relación de codificación convolucional o puncturing (1-1/2, 2-2/3, 3-3/4, 4-5/6, 5-7/8)
D           0   %Intervalo de guarda (0-Nada, 1-1/4, 2-1/8, 3-1/16 o 4-1/32 de la longitud de símbolo)
R           0   %Para habilitar el ruido Gausiano (referencia a Eb/N0 o Es/N0 deseada) poner a 1
ESN0dB      1   %Si ESN0dB=1-->Energia de simbolo; Si ESN0dB=0-->Energia de bit
EBN0dB      30   %Relación de Energía (bit o simbolo) en referencia a la energía de ruido.
EBN0dBmin   0   %Si hay barrido activado, EBN0dBmax=EBN0dB. Si no, EBN0dBmin=EBN0dB
M           1   %Modelo canal con linea de retardos. 0-Nada 1-TDL uniforme urbano 2-Autopista 3-Interior
4-Ciudad
L           10  %Número de taps del modelo TDL uniforme
v           100  %velocidad del receptor móbil (Km/h). Para el cálculo del desplazamiento doppler
fc          750 %Frecuencia de la portadora (MHz). Para el cálculo del desplazamiento doppler
long_canal  1   %Longitud de canal en símbolos OFDM
E           1   %Estimación y Ecualización de canal. 0-Nada 1-LS
-------------------------------------------------------------------------------------------------------
                                        ERRORES
-------------------------------------------------------------------------------------------------------
freq_error  0.2 %Error en frecuencia. La frecuencia en el intervalo de un simbolo es (-0.5,0.5)
P           0   %Exponente del NL (Valor absoluto). Si 0 no se introduce error en frequencia ni
estimación.Debe ser 1,3 o 4. Nunca 2.
time_error  0   %Error en muestras de inicio de símbolo. Es la primera portadora del primer símbolo que
se recibe
-------------------------------------------------------------------------------------------------------
                                        GRÁFICAS
-------------------------------------------------------------------------------------------------------
const       1   %Para realizar plots de las constelaciones Tx y Rx, correlación de pilotos poner a 1.
BERvsEBN0   0   %Para realizar grafica de barrido poner a 1. No grafica constelaciones. EBN0=[-10,8];
ESN0dB=[-10,24]
VARvsESN0   0   %Para realizar grafica de barrido (solo para ESN0) en intervalos de "pasos" poner a 1.
pasos       5   %Intervalo de separación entre valores del eje X de las gráficas de barrido
_______________________________________________________________________________________________________
```

Fig A.2 Estructura del fichero de configuración parametros.txt

La función que se ocupa de leer, validar y preparar los parámetros de configuración de la simulación es la función *carga_parametros.m*. A continuación se muestra el código básico empleado para la lectura de los datos. No se añade el código que valida que los parámetros de configuración sean correctos puesto que su extensión es muy grande (se trata de un conjunto de sentencias condicionales que comprueban que la opción de activación de cada parámetro esté contemplada en el programa)

```
1  function [modo, n , alpha, C, D, EBN0dB,EBN0dBmin, R, M, ST,fallo,const,BERvsEBN0,G, freq_error,P
   ,ESN0dB,VARvsESN0,pasos,time_error,long_canal,E,Tsimbolo,L,v,fc] = carga_parametros( fichero )
2  A = textread(fichero,'%*s%f','commentstyle','matlab');
3  fprintf('.');
4  B=A(4);ST=A(5);G=A(6);modo=A(7);n=A(8);alpha=A(9);C=A(10);D=A(11);R=A(12);
5  ESN0dB=A(13);EBN0dB=A(14);EBN0dBmin=A(15);M=A(16);L=A(17);v=A(18);fc=A(19);
6  long_canal=A(20);E=A(21);freq_error=A(25);P=A(26);time_error=A(27);const=A(31);
7  BERvsEBN0=A(32);VARvsESN0=A(33);pasos=A(34);
8  clear A;
9  if (C==1) C=1/2; elseif (C==2) C=2/3; elseif (C==3) C=3/4; elseif (C==4) C=5/6; elseif (C==5) C=7/8; end
10 if (D==1) D=1/4; elseif (D==2) D=1/8; elseif (D==3) D=1/16; elseif (D==4) D=1/32; end
11 if (P==0) freq_error=0; end
12 if (BERvsEBN0==0&&VARvsESN0==0) EBN0dBmin=EBN0dB;end
13 if(BERvsEBN0==1) const=0; end
14 if(VARvsESN0==1) ESN0dB=1; const=0; n=4; modo=2;long_canal=1; end
15 if((ST>1)&&G==1)||((ST>8)&&G==0)||(G==2&&ST>5)
16    const=0;
17 end
18 if M==0 E=0; end %No ecualiza si no hay canal
```

Código 35 carga_parametros.m

A.3 ESTRUCTURA DEL FICHERO DE FUNCIÓN M-FILE

Todos los ficheros *M-file* implementados siguen la misma estructura, con el objetivo de facilitar la comprensión del algoritmo utilizado en cada función. Se parte de una explicación de los objetivos de la función en la que se añaden referencias e información de otras funciones implicadas. A continuación se incluye el código de la función, totalmente comentado y estructurado por partes. Para poder llevar un seguimiento del estado del proceso se incluye en cada *M-file* una sentencia que imprime en pantalla el nombre del bloque por el que se procesan los datos en ese momento y el número de trama MPEG o símbolo OFDM que se procesa. En la Fig A.3 se ilustra el formato de fichero *M-file* que se ha comentado en este apartado.

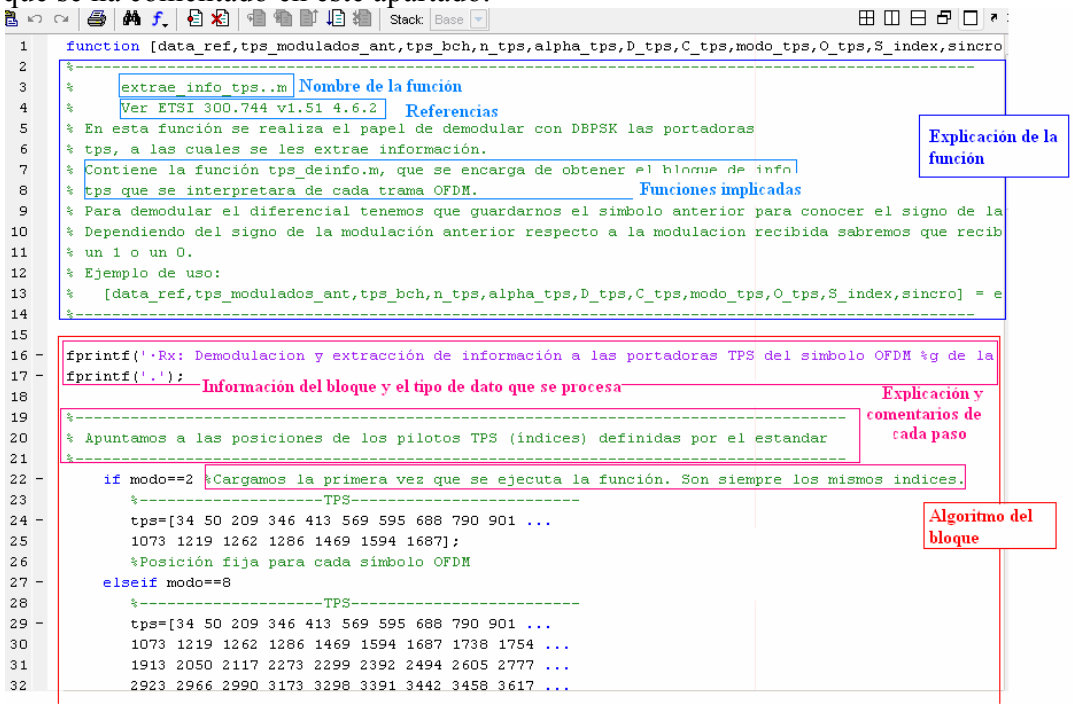


Fig A.3 Estructura del fichero de función M-file

A.4 INTERFAZ DE USUARIO

Una vez configurada la simulación mediante el fichero *parametros.txt* y ejecutada la función principal *DVBT_simulado.m,* se presenta en pantalla el estado de la simulación en cada momento. Inicialmente se muestra por pantalla un resumen de la configuración de la simulación. A continuación se imprime por pantalla la información del estado de la simulación según el flujo de datos, permitiendo así que el usuario pueda realizar un seguimiento del mismo. Al terminar el proceso de simulación se imprime en pantalla un resumen de la simulación ofreciendo resultados de BER y estimaciones realizadas.

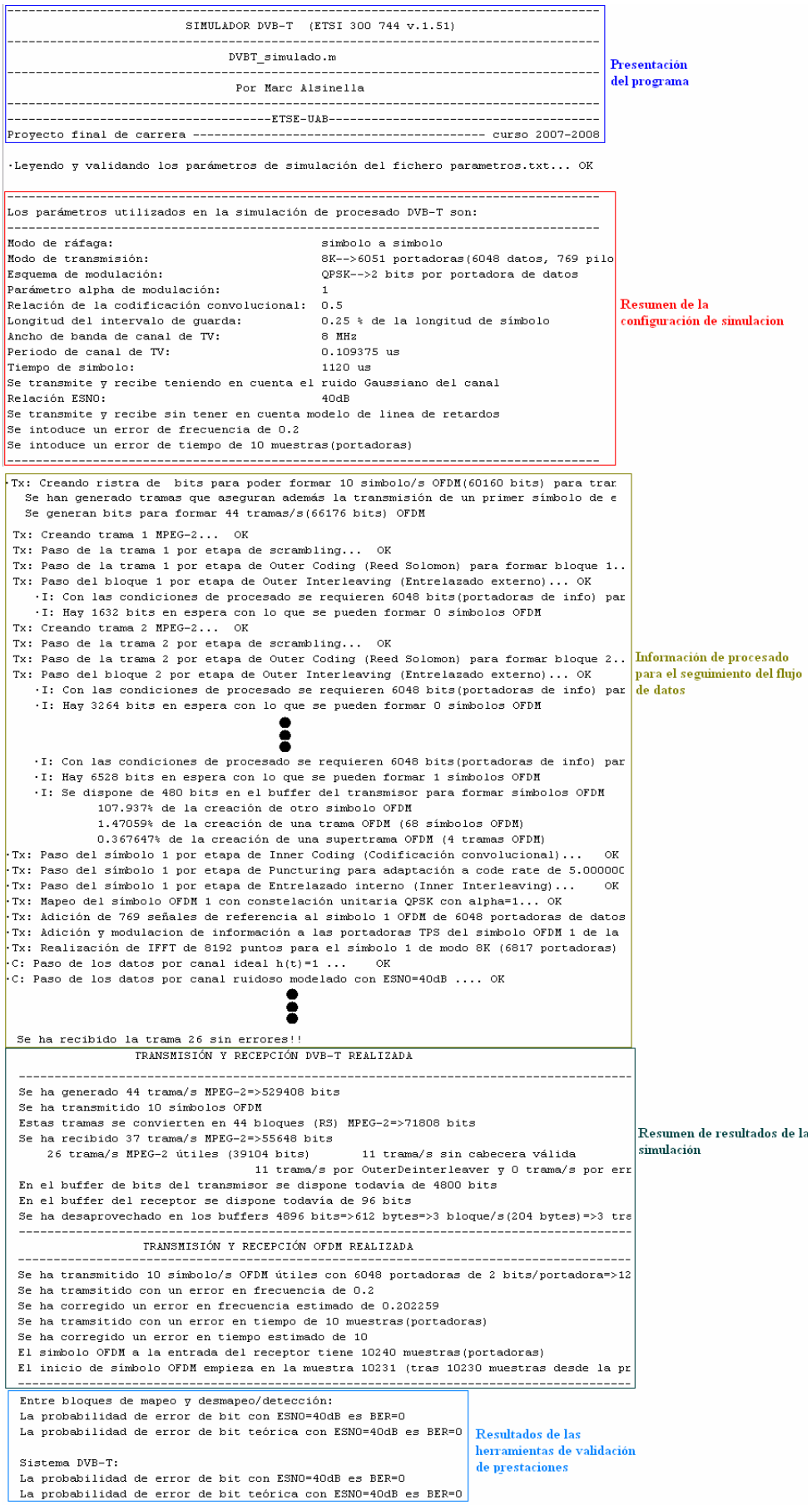


Fig A.4 Estructura de la interfaz de usuario

A.5 CÓDIGO IMPLEMENTADO

```
1  %--------------------------------------------------------------------
2  %                DVBT_simulado.m
3  %--------------------------------------------------------------------
4  clc
5  clear all
6  close all
7  %--------------------------------------------------------------------
8  %*****INICIO DEL PROGRAMA**********
9  %--------------------------------------------------------------------
10 %-------------------
11 %Bienvenida
12 %-------------------
13 disp('-----------------------------------------------------');
14 disp(' SIMULADOR DVB-T  (ETSI 300 744 v.1.51)")
15 disp('-----------------------------------------------------');
16 disp('               DVBT_simulado.m         ')
17 disp('-----------------------------------------------------');
18 disp('       Por Marc Alsinella       ')
19 disp('-----------------------------------------------------');
20 disp('------------ETSE-UAB----------------------');
21 disp('Proyecto final de carrera ----- curso 2007-2008');
22 disp(' ');
23 %--------------------------------------------------------------------
24 %Carga y validación de parámetros de simulación
25 %--------------------------------------------------------------------
26 [modo, n , alpha, C, D, EBN0dB, EBN0dBmin, R,
   M,ST,fallo,const,BERvsEBN0,G,freq_error,P,ESN0dB,VARvsESN0,pasos,time_error,long_canal,E,Tsimbol
   o,L,v,fc]=carga_parametros('parametros.txt');
27 if fallo==1
28    return
29 end
30 %--------------------------------------------------------------------
31 % Configuración del barrido de BERvsEBN0.Montecarlo
32 %--------------------------------------------------------------------
33 switch modo
34    case 2
35       N_portadoras=1512;
36       fft_mode=2048;
37    case 8
38       N_portadoras=6048;
39       fft_mode=8192;
40 end
41 if BERvsEBN0==1                          Para el caso de barrido EBN0 comparado con la BER
                                            teórica
42 G=0;                                     Con Montecarlo forzamos que se generen bits para
                                            formar tramas (no supertramas ni simbolos)
43 BER=theorical_BER(ESN0dB,n,EBN0dBmin,EBN0dB,p  para obtener los valores teóricos de BER (en función de
   asos);                                   EBN0 o ESN0)
44 EBN0dB_vect=EBN0dBmin:pasos:EBN0dB;      EBN0dB teorico para el cual la BER es menor a 10^-4
45 bits_graf=(100./BER);                    Número de bits necesarios a comparar para poder
                                            cumplir Montecarlo
46 T_bits=204*8;                            numero de bits de cada trama (en bloques RS)
47 simbol_bits=N_portadoras*log2(n);        número de bits por simbolo comparado para la BER por
                                            el canal
48 num_simbol_vect=ceil((bits_graf./simbol_bits));  Conversión del numero necesario de bits a simbolos
49 num_tramas_simbol_vect=ceil(simbol_bits*C/T_bits);  Conversión de numero de tramas para cada simbolo
50 T_vect=ceil(num_simbol_vect.*num_tramas_simbol_ve  Conversión del numero de simbolos a tramas,
   ct);                                     (sumamos una mas si querermos ir sobrados, por lo
                                            que se queda en el buffer)
```

```
%-------------------------------------------------------------------
% Configuración del barrido de VARvsEBN0
%-------------------------------------------------------------------
elseif VARvsESN0==1                                 Para el caso de barrido para obtener Varianza en
                                                    función de ESN0 Aquí EBN0dB indica ESN0dB ya que
                                                    ESN0==1,es decir, se usa energia de simbolo en vez de
                                                    energía de bit
G=0;                                                Con Montecarlo forzamos que se generen bits para
                                                    formar tramas (no supertramas ni simbolos)
pasos=5;                                            Forzamos pasos de 5
EBN0dBmin=0;                                        Forzamos inicio en 0
EBN0dB=25;                                          Forzamos final en 25
EBN0dB_vect=EBN0dBmin:pasos:EBN0dB;                 valores de ESN0 para graficar Varianza.
MCRBfreq_vect=3/(2*pi^2*fft_mode)./(10.^(EBN0dB_ve  Valores de MCRB para usar Montecarlo
ct/10));
T_vect=[100,100,100,200,200,200];                   Número de tramas a enviar para cada ESN0dB
                                                    especificada
else
   EBN0dB_vect=EBN0dB;
end
%-----------------------------------------------
%       Inicio de barrido en EBN0
%-----------------------------------------------
e=1;                                                Para apuntar al vector que guarda los resultados de
                                                    BER y EBN0dB si hay barrido activado
for EBN0=min(EBN0dB_vect)*(BERvsEBN0+VARvsESN0): pasos:max(EBN0dB_vect) * (BERvsEBN0 +
VARvsESN0)
   if BERvsEBN0==1||VARvsESN0==1
   EBN0dB=EBN0;
   ST=T_vect(e);                                    ST ahora lleva el numero de tramas (no de
                                                    supertramas, debemos poner a 0 el flag de G en
                                                    genera_bitsMPEG2.m)
   end
%-----------------------------------------------------------
% Inicializacion de vectores y variables
%-----------------------------------------------------------
total_tramas_tx=[];                                 Aqui guardaremos todas las tramas en una matriz, una
                                                    por columna(de 8 subcolumnas de 8 bits)
total_tramas_rx=[];                                 Almacenaremos aquí las tramas recibidas al final del
                                                    proceso de recepción
total_simbolos_tx=[];                               Aqui guardaremos todas las tramas en una matriz, una
                                                    por columna(de 8 subcolumnas de 8 bits)
total_simbolos_rx=[];                               Almacenaremos aquí las tramas recibidas al final del
                                                    proceso de recepción
S_tx=0;                                             Inicializamos variable que cuenta el numero de
                                                    simbolos que se procesan en el transmisor
O_tx=0;                                             Inicializamos variable que cuenta las tramas OFDM que
                                                    se van creando con los simbolos el transmisor
SO_tx=0;                                            Inicializamos variable que cuenta las supertramas
                                                    OFDM que se van creando en el transmisor
S_rx=0;                                             Inicializamos variable que cuenta el numero de
                                                    simbolos que se procesan en el receptor
O_rx=0;                                             Inicializamos variable que cuenta las tramas OFDM que
                                                    se van creando con los simbolos el receptor
SO_rx=0;                                            Inicializamos variable que cuenta las supertramas
                                                    OFDM que se van creando en el receptor
B_rx=0;                                             Inicializamos variable que cuenta los bloques MPEG2
                                                    recibidos.
T_rx=0;                                             Inicializamos variable qe cuenta los bloques/tramas
                                                    MPEG2 recibidos.
T_rx_index=0;                                       Inicializamos variable que lleva el indice de las tramas
                                                    MPEG2 que se reciben
T_tx_index=0;                                       Inicializamos variable que lleva el indice de las tramas
                                                    MPEG2 que se envian
S_index_tx=0;                                       Inicializamos variable que lleva el indice de los simbolos
                                                    OFDM que se envian para formar una determinada
                                                    trama OFDM
S_index_rx=0;                                       Inicializamos variable que lleva el indice de los simbolos
                                                    OFDM que se reciben
S_desincr=0;                                        Inicializamos variable que lleva la cuenta de simbolos
                                                    deshechados en recepcion por falta de sincronismo
t_rx=1;                                             Para llevar la cuenta de las tramas que se van
                                                    comparando
data_canal_out_error=[];                            Inicializamos buffer de almacenamiento de símbolos
                                                    para meter los que se van a recibir por el canal con
                                                    error de tiempo de inicio de simbolo
S_c=0;                                              Cuenta los simbolos que pasan por el canal
C_c=0;                                              Cuenta el numero de canales que se van realizando
freq_pos=1;                                         Indexa al vector que contiene los valores cuadraticos
                                                    medios para el calculo de varianza en frecuencia
time_pos=1;                                         Indexa al vector que contiene los valores cuadraticos
                                                    medios para el calculo de varianza en tiempo
if time_error~=0
   S_c=-1;                                          Descontamos el simbolo piloto para realizar la
                                                    estimacion de tiempo
end
```

```
%-------------------------------------------------------------
%   Generador de bits MPEG-2 (sin cabecera)
%-------------------------------------------------------------
[bits_mpeg2,ST,N_portadoras,P_portadoras,T_portadoras,fft_mode]=genera_bitsMPEG2(ST, modo, n,
G,C,time_error,M);
%-------------------------------------------------------------------------------------------
% AHORA TRABAJAMOS PAQUETE A PAQUETE (TRAMA A TRAMA)
%-------------------------------------------------------------------------------------------
%*********************************************************
%      AQUÍ EMPIEZA EL TRANSMISOR
%*********************************************************
inicializacion=0;                                   Indicará cuando inicializar.
for T_tx=1:ST                                       Indica la trama/bloque que se va preparando y
                                                    enviando. Se hace tramas MPEG-2 una detras de otra

if inicializacion==1&&(time_error~=0)               Inicializamos todo una vez se ha procesado el simbolo
                                                    piloto
total_tramas_tx=[];                                 Aqui guardaremos todas las tramas en una matriz, una
                                                    por columna(de 8 subcolumnas de 8 bits)
total_tramas_rx=[];                                 Almacenaremos aquí las tramas recibidas al final del
                                                    proceso de recepción
total_simbolos_tx=[];                               Aqui guardaremos todas las tramas en una matriz, una
                                                    por columna(de 8 subcolumnas de 8 bits)
total_simbolos_rx=[];                               Almacenaremos aquí las tramas recibidas al final del
                                                    proceso de recepción
S_tx=0;                                             Inicializamos variable que cuenta el numero de
                                                    simbolos que se procesan en el transmisor
O_tx=0;                                             Inicializamos variable que cuenta las tramas OFDM que
                                                    se van creando con los simbolos el transmisor
SO_tx=0;                                            Inicializamos variable que cuenta las supertramas
                                                    OFDM que se van creando en el transmisor
S_rx=0;                                             Inicializamos variable que cuenta el numero de
                                                    simbolos que se procesan en el receptor
O_rx=0;                                             Inicializamos variable que cuenta las tramas OFDM que
                                                    se van creando con los simbolos el receptor
SO_rx=0;                                            Inicializamos variable que cuenta las supertramas
                                                    OFDM que se van creando en el receptor
B_rx=0;                                             Inicializamos variable que cuenta los bloques MPEG2
                                                    recibidos.
T_rx=0;                                             Inicializamos variable qe cuenta los bloques/tramas
                                                    MPEG2 recibidos.
T_rx_index=0;                                       Inicializamos variable que lleva el indice de las tramas
                                                    MPEG2 que se reciben
T_tx_index=0;                                       Inicializamos variable que lleva el indice de las tramas
                                                    MPEG2 que se envian
S_index_tx=0;                                       Inicializamos variable que lleva el indice de los simbolos
                                                    OFDM que se envian para formar una determinada
                                                    trama OFDM
S_index_rx=0;                                       Inicializamos variable que lleva el indice de los simbolos
                                                    OFDM que se reciben
S_desincr=0;                                        Inicializamos variable que lleva la cuenta de simbolos
                                                    deshechados en recepcion por falta de sincronismo
t_rx=1;                                             Para llevar la cuenta de las tramas que se van
                                                    comparando
T_tx=1;                                             Para que empiece a contar desde el principio otra vez
                                                    las tramas una vez haya pasado el simbolo piloto

inicializacion=0;                                   Desactivamos la inicialización tras el paso de primer
                                                    simbolo util de datos
end
T_tx_index=T_tx_index+1;
%-------------------------------------------------------------------
%   Generador de paquetes estandar MPEG-2
%-------------------------------------------------------------------
trama_tx=genera_paquetesMPEG2(bits_mpeg2,T_tx);
%-------------------------------------------------------------------
% Guardamos cada trama enviada en una matriz de tramas
%-------------------------------------------------------------------
if isempty(total_tramas_tx)
   total_tramas_tx=bi2de(trama_tx,'left-msb');      Nos servirá para comparar lo que ha llegado con lo que
                                                    se ha enviado
else
    total_tramas_tx=[total_tramas_tx,bi2de(trama_tx,'left-msb')];
end
```

```
%----------------------------------------
%           Scrambler
%----------------------------------------
if T_tx==1                                          Solo inicializamos en la primera trama a procesar
registro_prbs_tx=[1 0 0 1 0 1 0 1 0 0 0 0 0 0 0];   Secuencia inicial del registro prbs
end
[trama_sc_tx,registro_prbs_tx,T_tx_index]=scrambler(trama_tx,T_tx_index,T_tx,registro_prbs_tx);

trama_tx=bi2de(trama_tx,'left-msb');                Aquí la salida es un vector columnna con 188 filas
                                                    (bytes en decimal!!) para poder compararlo con la salida
                                                    (mismas "unidades") de 188x1
%-------------------------------------------------
%    OuterCoding (Reed Solomon)
%-------------------------------------------------
trama_sc_tx=bi2de(trama_sc_tx,'left-msb');          Ahora se trata de un vector columnna de 188
                                                    posiciones en vez de matriz de 8*188
bloque_rs_tx = rs_encode(trama_sc_tx,204,188,8,T_tx);
%----------------------------------------
%    Outer-Interleaver
%----------------------------------------
if T_tx==1                                          Solo inicializamos en la primera trama/bloque a
                                                    procesar
cola_out_int_tx=zeros(12^2*17,1);                   (12^2*17) es la longitud de la cola del outer interleaving,
                                                    Esta cola es de todo 0's(max retardo) inicialmente. Se
                                                    puede hacer cola random o de sec prbs (Ver la función
                                                    dvbt_send_init.m Part 3 de código abierto DVBT)

end
[bloque_int_tx,cola_out_int_tx]= out_int(bloque_rs_tx,12,17,204,cola_out_int_tx,T_tx);

%-------------------------------------------------------------------------- La salida vuelve a ser en forma de bytes (Vector
---                                                 columna con bytes)
%    Adecuación de los bytes P/S, Guardado de trama
%-------------------------------------------------------------------------- Adecuamos la trama/bloque de bytes puestos en
---                                                 paralelo a serie (todos los bits de una trama/bloque
                                                    seguidos)
bloque=de2bi(bloque_int_tx,8,'left-msb');           Volvemos a poner cada byte (uno por fila) en forma
                                                    binaria (8 bits).
bloque_serie_tx=reshape(bloque',1,204*8);           Aquí se guardan en una matriz las tramas/bloques en
                                                    fila que se van procesando.
if T_tx_index==8
   T_tx_index=0;
end
if(T_tx==1)                                         Inicializamos la matriz de guardado de tramas y
                                                    contador de tramas
secuencia_bloques_tx=[];
end
secuencia_bloques_tx=[secuencia_bloques_tx,         buffer de bits!!!!vector fila!
bloque_serie_tx];
%-------------------------------------------------------------
% Adecuación de la secuencia de tramas/bloques en serie a un símbolo OFDM
%-------------------------------------------------------------
num_bits_tx=length(secuencia_bloques_tx);
simbol_bits_tx=(N_portadoras*log2(n))*C;            longitud del simbolo OFDM
num_simbols_tx=floor(num_bits_tx/simbol_bits_tx);
simbolos_ofdm_tx=zeros(simbol_bits_tx,              Aqui se guardan los simbolos ofdm en columna en
num_simbols_tx);                                    cuanto ya se pueden formar
fprintf('\t·I: Con las condiciones de procesado se requieren %g bits(portadoras de info) para formar 1 símbolo
OFDM\n',simbol_bits_tx);
fprintf('\t·I: Hay %g bits en espera con lo que se pueden formar %g símbolos
OFDM\n',num_bits_tx,num_simbols_tx);
for i=1:num_simbols_tx                              Vaciamos el buffer "secuencia_tramas_en_serie" de bits
                                                    si ya se pueden crear simbolos y los almacenamos en
                                                    simbolos_ofdm
   simbolos_ofdm_tx(:,i)=secuencia_bloques_tx(1+(i-1)*simbol_bits_tx:i*simbol_bits_tx);

end
if ~isempty(secuencia_bloques_tx)                   lo que ha quedado del buffer
                                                    "secuencia_tramas_en_serie" lo continuamos
                                                    guardando a la espera de mas tramas
   secuencia_bloques_tx=secuencia_bloques_tx(1+num_simbols_tx*simbol_bits_tx:num_bits_tx);
end
```

```
%-------------------------------------------------------------------
%-------------------------------------------------------------------
%   AHORA EMPEZAMOS A TRABAJAR CON
UNIDADES DE SÍMBOLO OFDM (en bits)
%-------------------------------------------------------------------
%-------------------------------------------------------------------
data_canal_in=[];                                   Inicializamos buffer de almacenamiento de símbolos
                                                    para meter los que se van a enviar por el canal
for k=1:num_simbols_tx                              Trabajamos con unidades de bit correspondientes a un
                                                    simbolo OFDM (cada uno por separado)
   data_tx=simbolos_ofdm_tx(:,k);                   Símbolo en columna
S_tx=S_tx+1;                                        Incrementamos el contador de simbolos OFDM que se
                                                    procesan
S_index_tx=S_index_tx+1;                            Incrementamos el indice del simbolo OFDM
if S_tx==1
S_index_tx=1;                                       Índice de símbolo entrante [1:68] para formar una trama
                                                    OFDM
O_index_tx=1;                                       'Índice de trama OFDM entrante [1:4];
end
fprintf('\t·I: Se dispone de %g bits en el buffer del transmisor para formar símbolos
OFDM\n',length(secuencia_bloques_tx));
fprintf('\t\t\t %g%% de la creación de otro simbolo OFDM\n',num_bits_tx/simbol_bits_tx*100);
fprintf('\t\t\t %g%% de la creación de una trama OFDM (68 símbolos OFDM)\n',S_tx/68*100-O_tx*100);
fprintf('\t\t\t %g%% de la creación de una supertrama OFDM (4 tramas OFDM)\n',S_tx/(68*4)*100-SO_tx*100);
if (mod(S_tx,68)==0&&(S_tx)~=0)
   O_tx=O_tx+1;
   O_index_tx=O_index_tx+1;
fprintf('\t\t\t SE HA CREADO LA TRAMA OFDM %g  (68 símbolos OFDM)\n',O_tx)
end
if (mod(S_tx,4*68)==0&&O_tx~=0)
   SO_tx=SO_tx+1;
fprintf('\t\t\t SE HA CREADO SUPERTRAMA OFDM %g  (4 tramas OFDM)\n',SO_tx)
end
%-------------------------------------------------------------------
%Inner Coding (Codificación convolucional)
%-------------------------------------------------------------------
if S_tx==1
estado_tx=zeros(1,6);                               Inicializamos estado con 6 registros a 0
end                                                 Esperamos como salida dos vectores columna. Por
                                                    cada bit salen X e Y
[X_tx,Y_tx,estado_tx] = conv_code(data_tx,S_tx,estado_tx);
%---------------------------------------
%         Puncturing
%---------------------------------------
data_conv_tx= puncturing(X_tx,Y_tx,C,S_tx);         Vector con simbolo de longitud N_carriers*log2(n)
%---------------------------------------
%        Inner Interleaver
%---------------------------------------
data_int_tx = in_int(data_conv_tx,N_portadoras,n,S_tx,fft_mode,S_index_tx);
%-----------------------------------------------------Ha separado en log2(n) columnas la señal anterior
% Guardamos cada simbolo enviado en una matriz de
%-----------------------------------------------------
if isempty(total_simbolos_tx)
  total_simbolos_tx=(data_int_tx);                  Nos servirá para comparar lo que ha llegado con lo que
                                                    se ha enviado
else
   total_simbolos_tx=[total_simbolos_tx,data_int_tx];
end
%---------------------------------------
%             Mapeado
%---------------------------------------
data_map_tx=mapeo(data_int_tx,n,alpha,S_tx,N_portadoras,const);
%---------------------------------------             Vector fila de N_portadoras símbolos PAM(bloqueS/P)
%    Inserción de señales de referencia
%---------------------------------------
if S_tx==1
   tps_tx=[];                                       Creamos el vector donde ubicaremos las posiciones de
                                                    los pilotos tps para referencias.m
   continual_tx=[];                                 Creamos el vector donde ubicamos las posiciones de
                                                    los pilotos continuos
   W_tx=[];                                         Vector donde se almacenará la secuencia PRBS usada
                                                    en la modulación de las portadoras piloto
end
[data_ref_tx,tps_tx,continual_tx,W_tx,indices_pilotos_tx,sintetic] = referencias(data_map_tx,modo,
N_portadoras,P_portadoras,T_portadoras,S_tx,tps_tx,continual_tx,S_index_tx,W_tx);
```

```
%-------------------------------------------------
%  Insercion de informacion en portadoras tps
%-------------------------------------------------
if S_index_tx==1
   data_ref_tx_ant=[];
end
[data_info_tx,data_ref_tx_ant] = tps_con_info(data_ref_tx,data_ref_tx_ant,n,alpha,D,C,modo, O_index_tx,
S_tx, tps_tx, S_index_tx, SO_tx, T_portadoras, const);
%-------------------------------------------
% Codificación OFDM (IFFT+Intervalo de guarda)
%-------------------------------------------
data_ofdm_tx=ofdm_encode(data_info_tx, D, fft_mode,modo, T_portadoras,S_tx);
%-----------------------------------------------------------------
%   Inserción de datos en el buffer de salida. Salida del transmisor
%-----------------------------------------------------------------
if isempty (data_canal_in)
   data_canal_in=[data_ofdm_tx];                    Aquí nuestra señal está en paralelo (vector columna)
else
   data_canal_in=[data_canal_in, data_ofdm_tx];     Aquí nuestros símbolos están en paralelo (1 columna
                                                    por simbolo)
end
if S_index_tx==68
  S_index_tx=0;                                     Se inicializa cada vez que empieza una nueva trama de
                                                    68 simbolos OFDM
end
if O_index_tx==4
  O_index_tx=0;                                     Se inicializa cada vez que empieza una nueva
                                                    supertrama de 4 tramas OFDM
end
end                                                 Del for de num_simbols_tx (el primero del transmisor)

%*********************************************************
%        FINAL DEL TRANSMISOR
%*********************************************************
if ~isempty (data_canal_in)                         Empezamos la transmisión por el canal si hay simbolos
                                                    para enviar!!
data_canal=data_canal_in;
%*********************************************************
%       PASO POR EL CANAL
%*********************************************************
if ~isempty(data_canal)
[nada,num_simb_out]=size(data_canal);               Cada columna es un simbolo
for k=1:num_simb_out
data_canal_out=data_canal(:,k);                     Cogemos de uno en uno
if S_c==-1&&(time_error~=0);                        Cuando se haya transmitido el simbolo piloto
                                                    inicializaremos los buffers (menos canal) y registros.
inicializacion=1;                                   Marcamos que en el siguiente
end
if mod(S_c,long_canal)==0||S_c==-1
C_c=C_c+1;                                          Incrementamos el numero de canales realizados hasta
                                                    el momento.
canal=1;
end
[data_canal_out,canal]=modelo_canal(data_canal_out,C_c,S_c,M,fft_mode,long_canal,canal,Tsimbolo,D,
L,v,fc,T_portadoras,const);
data_canal_out=modelo_ruido(data_canal_out,R,EBN0dB,n,fft_mode,ESN0dB);

S_c=S_c+1;                                          Incrementamos el valor de los simbolos que pasan por
                                                    el canal
%-------------------------------------------------
%      Inserción de error en frecuencia
% -------------------------------------------------
if P~=0                                             Si P=0 no se introduce error en frecuencia ni se realiza
                                                    estimacion.
Af=1/fft_mode;                                      Separación entre portadoras en muestras normalizado
                                                    (sin intervalo de guarda)
fe=(Af)*freq_error;                                 Conversión a error en muestras.
data_canal_out=data_canal_out.*(exp(j*2*pi*fe*(0:length Datos con error en frequencia.
(data_canal_out)-1))).';
end
```

```
%-------------------------------------------------------------------
%              Inserción de error de timing                        Ponemos los simbolos en serie (pero en columna), uno
                                                                   detras de otro. (todas las portadoras seguidas)
%-------------------------------------------------------------------
if time_error~=0
if isempty(data_canal_out_error)
data_canal_out_error=[data_canal_out(time_error+1:en               Empezamos a recibir con error de tiempo de simbolo
d)];
else
data_canal_out_error=[data_canal_out_error;data_canal_out];
end
elseif time_error==0
if isempty(data_canal_out_error)
data_canal_out_error=[data_canal_out];
else
data_canal_out_error=[data_canal_out_error;data_canal_out];
end
time_error_est=1;                                                  Para apuntar a la primera muestra (inicio) del simbolo
                                                                   OFDM (sin error de tiempo)
end
%***********************************************************
%               AQUÍ EMPIEZA EL RECEPTOR
%***********************************************************
%-------------------------------------------------------------------
%          Estimación del error de timing
%-------------------------------------------------------------------
if time_error~=0&&S_c>=1 E                                         Estimamos cuando tenemos el simbolo piloto+el primer
                                                                   simbolo de datos
time_error_est=time_est(fft_mode,D,T_portadoras,sintetic,data_canal_out_error,S_c,const);
if VARvsESN0==1                                                    Calculamos la varianza de la estimacion de tiempo
time_dif(time_pos)=((time_error-((fft_mode*(1+D)+1)-               Valores de error cuadratico medio
(time_error_est)))/(fft_mode*(1+D)))^2;
time_pos=time_pos+1;
end
end
%-------------------------------------------------------------------
%      Adquisición de datos del buffer
%-------------------------------------------------------------------
if S_c>=1                                                          Empezamos a coger datos del canal cuando está el
                                                                   simbolo 1 en el buffer
data_ofdm_rx=data_canal_out_error(time_error_est:(tim              Nos quedamos con un símbolo (columna) por separado
e_error_est+fft_mode*(1+D)-1));
data_canal_out_error=data_canal_out_error(fft_mode*(               Vaciamos del buffer lo que hemos cogido
1+D)+1:end);
S_rx=S_rx+1;                                                       Incrementamos el valor del contador de simbolos que
                                                                   llegan al receptor y se procesan
S_index_rx=S_index_rx+1;                                           Incrementamos el contador de indice de simbolo [1:68]

if S_rx==1
S_index_rx=1;                                                      Índice de símbolo entrante [1:68] para formar una trama
                                                                   OFDM
O_index_rx=1;                                                      'Índice de trama OFDM entrante [1:4];
end
if (mod(S_rx,68)==0&&(S_rx)~=0)
   O_rx=O_rx+1;
   O_index_rx=O_index_rx+1;
fprintf('\t\t\t SE HA RECIBIDO LA TRAMA OFDM %g  (68 símbolos OFDM)\n',O_rx)
end
if (mod(S_rx,4*68)==0&&O_rx~=0)
   SO_rx=SO_rx+1;
fprintf('\t\t\t SE HA RECIBIDO SUPERTRAMA OFDM %g  (4 tramas OFDM)\n',SO_tx)
end
%-------------------------------------------------
%          Estimación de canal
%-------------------------------------------------
if E~=0&&(mod((S_c-1),long_canal)==0)                              Modelado de canal cada "long_canal" simbolos
[canal_est,canal_dif] =est_canal(M,data_ofdm_rx,sintetic,canal,fft_mode,T_portadoras,D, C_c,const,
indices_pilotos_tx,VARvsESN0);
end
%-------------------------------------------------
%   Estimación de error de frecuencia. Oerder&Meyr
%-------------------------------------------------
if (P~=0&&S_c>=1)
fe_est=oerder_meyr(data_ofdm_rx(1:fft_mode),4,P,S_c);
```

```
%-------------------------------------------------
%   Correccion de error en frecuencia
%-------------------------------------------------
freq_error_est=fe_est*Af;                          Conversión de error a muestras
data_ofdm_rx=data_ofdm_rx.*(exp(-                  Datos con error en frequencia corregido.
j*2*pi*freq_error_est*(0:length(data_ofdm_rx)-1))).';
if VARvsESN0==1                                    Calculamos la varianza de estimación de frecuencia
fe_dif(freq_pos)=(freq_error-fe_est)^2;            Valores de varianza
freq_pos=freq_pos+1;
end
end
%--------------------------------
%   Decodificación OFDM
%--------------------------------
[data_info_rx]=ofdm_decode(data_ofdm_rx, D, fft_mode, modo, T_portadoras,S_rx);
%--------------------------------
%       Ecualización
%--------------------------------
if E~=0
data_info_rx=ecualiz(data_info_rx,canal_est,fft_mode,D,S_rx,T_portadoras);
end
%-----------------------------------------
%   Extraccion de info TPS+sincronización
%-----------------------------------------
if S_index_rx==1
   tps_rx=[];                                      Creamos el vector donde ubicaremos las posiciones de
                                                   los pilotos tps para referencias.m
end
if S_index_rx==1
   sincro=1;
   tps_modulados_ant=[];
   tps_bch=zeros(1,68);                            Aqui guardamos la cadena tps codificada a medida que
                                                   llegan los simbolos
end
[data_ref_rx,tps_modulados_ant,tps_bch,n_tps,alpha_tps,D_tps,C_tps,modo_tps,O_tps,S_index_rx,sincr
o] =  extrae_info_tps(modo,data_info_rx,tps_modulados_ant,tps_bch,S_rx,tps_rx,S_index_rx,
O_index_rx, SO_rx,T_portadoras,sincro,const);
if sincro==0                                       %Llevamos la cuenta de los simbolos deshechados por
                                                   falta de sincronismo
S_desincr=S_desincr+1;
S_index_rx=0;
S_rx=0;
end
SOLO PASA LA INFO SI HAY SINCRONIZACIÓN!!
if (sincro==1&&S_desincr~=0)||S_desincr==0         Si la sincronización ha sido establecida correctamente
                                                   mediante las tps, continuamos. Si no, descartamos el
                                                   simbolo y vamos a por otro
if S_index_rx==68
fprintf('\t·De la trama OFDM %g se desprende la siguiente informacion tps:\n',O_index_rx-1);
fprintf('\t\t              Enviado      Recibido\n')
fprintf('\t\tO               %g            %g \n',O_index_rx-1,O_tps)
fprintf('\t\tn               %g            %g \n',n,n_tps)
fprintf('\t\talfa            %g            %g \n',alpha,alpha_tps)
fprintf('\t\tC               %g           %g \n',C,C_tps)
fprintf('\t\tD               %g           %g \n',D,D_tps)
fprintf('\t\tmodo              %g             %g \n',modo,modo_tps)
end
%-----------------------------------------
%  Extracción de referencias
%--------------------------------
if S_rx==1||(S_index_rx==1&&S_desincr~=0)
   tps_rx=[];                                      Creamos el vector donde ubicaremos las posiciones de
                                                   los pilotos tps para referencias.m
   continual_rx=[];                                Creamos el vector donde ubicamos las posiciones de
                                                   los pilotos continuos
   S_index_rx=1;                                   Índice de símbolo entrante [1:68] para formar una trama
                                                   OFDM
   W_rx=[];                                        vector donde se almacenará la secuencia PRBS usada
                                                   en la modulación de las portadoras piloto
end
[data_map_rx,tps_rx,continual_rx,W_rx,indices_pilotos_rx] = sin_referencias(data_ref_rx,modo,
N_portadoras,P_portadoras,T_portadoras,S_rx,tps_rx,continual_rx,S_index_rx,W_rx);
```

```
437 %------------------------------
438 %  Detección y desmapeo
439 %------------------------------                    Contiene los log2(n) bits de cada portadora por filas
440 data_int_rx=desmapeo(data_map_rx,n,alpha,S_rx,N_portadoras,const);
441 %-------------------------------------------------------------------
442 %       Guardamos cada simbolo recibido en una matriz de tramas
443 %-------------------------------------------------------------------
444 if isempty(total_simbolos_rx)
445     total_simbolos_rx=data_int_rx;
446 else
447     total_simbolos_rx=[total_simbolos_rx, data_int_rx];
448 end
449 if S_index_rx==68
450     S_index_rx=0;                                  Se inicializa cada vez que empieza una nueva trama de
                                                       68 simbolos OFDM
451 end
452 if O_index_rx==4
453     O_index_rx=0;                                  Se inicializa cada vez que empieza una nueva
                                                       supertrama de 4 tramas OFDM
454 end
455 %---------------------------------------
456 %       Inner Deinterleaver
457 %---------------------------------------           Vector columna con N*log2(n) bits
458 data_conv_rx = in_deint(data_int_rx,N_portadoras,n,S_rx,fft_mode,S_index_rx);
459 %---------------------------------------
460 %             Depuncturing
461 %---------------------------------------
462 [X_rx,Y_rx]=depuncturing(data_conv_rx,S_rx,C);
463 %-----------------------------------------
464 % Inner Decoding(Decodificación convolucional)
465 %-----------------------------------------
466 [data_rx]=conv_decode(X_rx,Y_rx,S_rx);             Vector columna con las N portadoras (1 bit por
                                                       portadora)
467 Metemos los datos en una cadena de bytes en decimal
468 if S_rx==1
469 secuencia_simbolos_rx=[];                          Buffer de simbolos en bytes,vector columna
470 end
471 simbolos_ofdm_rx=bi2de(reshape(data_rx,8,(N_portado Vector fila de N portadoras en bytes (decimal)
    ras*log2(n)*C/8)).','left-msb');
472 if isempty(secuencia_simbolos_rx)
473     secuencia_simbolos_rx=simbolos_ofdm_rx;
474 else
475     secuencia_simbolos_rx=[secuencia_simbolos_rx;simbolos_ofdm_rx];
476 end
477 if S_index_rx==68
478     S_index_rx=0;                                  Se inicializa cada vez que empieza una nueva trama de
                                                       68 simbolos OFDM
479 end
480 end                                                Del primer FOR donde empieza el receptor que realiza
                                                       las acciones según número de simbolos num_simb_out

481 %-----------------------------------------------------
482 % Adecuación de los símbolos OFDM a bloques de 204 bytes
483 %-----------------------------------------------------
484 num_bytes_rx=length(secuencia_simbolos_rx);
485 bloque_bits_rx=204;                                Cada trama ahora sera de 204 bits para paso por
                                                       ReedSolomon
486 num_bloques_rx=floor(num_bytes_rx/bloque_bits_rx);
487 bloque_serie_rx=zeros(bloque_bits_rx,num_bloques_rx);
488 fprintf('\t·I: Se dispone de %g bytes en el buffer del receptor para formar bloques de 204bytes\n',num_bytes_rx);

489 fprintf('\t\t\tSe pueden formar %g bloque/s MPEG-2 de 204 bytes para la entrada al Desentrelazador
    Externo\n',num_bloques_rx);
490 for k=1:num_bloques_rx
491     bloque_serie_rx(:,k)=secuencia_simbolos_rx(1+(k-1)*bloque_bits_rx:k*bloque_bits_rx);
492 end
493 if ~isempty(secuencia_simbolos_rx)                 Vaciamos del buffer lo que hemos cogido y dejamos el
                                                       resto
494     secuencia_simbolos_rx=secuencia_simbolos_rx(1+num_bloques_rx*bloque_bits_rx:num_bytes_rx);
495 end
496 fprintf('\t\t\tEn el buffer del receptor se dispone todavía de %g bytes\n',length(secuencia_simbolos_rx));
```

```
%-------------------------------------------------------------
% AHORA EMPEZAMOS A TRABAJAR CON UNIDADES DE BLOQUES MPEG2(204 bytes)

%-------------------------------------------------------------
for k=1:num_bloques_rx
B_rx=B_rx+1;                                    Aumentamos el contador de bloques que se procesan

bloque_int_rx=bloque_serie_rx(:,k);
%-------------------------------------------------------------
%   Outer Deinterleaver
%-------------------------------------------------------------
if B_rx==1%Solo inicializamos en la primera
trama/bloque a procesar
cola_out_int_rx=zeros(12^2*17,1);               (12^2*17) es la longitud de la cola del outer interleaving,
                                                Esta cola es de todo 0's(max retardo) inicialmente. Se
                                                puede hacer cola random o de sec prbs (Ver la función
                                                dvbt_send_init.m Part 3 de código abierto DVBT)

end                                             Salida en forma de bytes (vector columna)
[bloque_rs_rx,cola_out_int_rx]= out_deint(bloque_int_rx,12,17,204,cola_out_int_rx,B_rx);
%-------------------------------------------------------------
%   Outer Decoding (Reed Solomon)
%-------------------------------------------------------------
trama_sc_rx=rs_decode(bloque_rs_rx,204,188,8,B_rx);
%-------------------------------------------------------------
%               Descrambler
%-------------------------------------------------------------
if B_rx==1                                      Solo inicializamos en la primera trama/bloque a
                                                procesar
registro_prbs_rx=[1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0];  Secuencia inicial del registro prbs
end
[trama_rx,registro_prbs_rx,T_rx_index,T_rx]=descrambler(trama_sc_rx,T_rx_index,B_rx,registro_prbs_rx,
T_rx);
%-------------------------------------------------------------
%       Guardamos cada trama recibida en una matriz de tramas
%-------------------------------------------------------------
if isempty(total_tramas_rx)&&~isempty(trama_rx)
   total_tramas_rx=trama_rx;
else
   total_tramas_rx=[total_tramas_rx, trama_rx];
end
if T_rx_index==8
   T_rx_index=0;
end
end                                             Del bucle FOR de num_bloques_rx
end                                             Del de sincro==1
end                                             Del if de S_c>=1
% ************************************************************
%                   FINAL DEL RECEPTOR
% ************************************************************
end                                             Del IF que mira si hay data_canal (tiene que ser el
                                                antepenúltimo!)
end                                             Del IF que mira si hay data_canal_in (tiene que ser el
                                                penúltimo!)
end                                             Del bucle de creacion de tramas (primero). Tiene que
                                                ser el último!!!
Comparamos lo que nos ha llegado con lo que hemos enviado mientras va llegando
if ~isempty(total_tramas_tx)&&~isempty(trama_rx)    Si viene una trama vacía del out_deint la descartamos
                                                (ha sido para si inicializar su cola)
   [nada,tx]=size(total_tramas_tx);             número de tramas enviadas
   [nada,rx]=size(total_tramas_rx);             número de tramas recibidas
   total_rx=min(tx,rx);                         mínimo número de tramas en común
   for k=t_rx:total_rx
      tramas_enviadas=total_tramas_tx(:,k);
      tramas_recibidas=total_tramas_rx(:,k);
            if all(all(tramas_enviadas==tramas_recibidas))
         fprintf ('·I: Se ha recibido la trama %g sin errores!!\n',t_rx);
      else
         fprintf ('·I: Se ha recibido la trama %g con errores!!\n',t_rx);
      end
   t_rx=t_rx+1;
   end
end
a=length(secuencia_bloques_tx);
b=length(secuencia_simbolos_rx)*8;
fprintf('\n\n\t\t\t\t\tTRANSMISIÓN Y RECEPCIÓN DVB-T REALIZADA\n\n\n');
fprintf('\t----------------------------------------\n');
```

```
fprintf('\tSe ha generado %g trama/s MPEG-2=>%g bits\n',ST,ST*8*188*8);
fprintf('\tSe ha transmitido %g símbolos OFDM\n',S_tx);
fprintf('\tEstas tramas se convierten en %g bloques (RS) MPEG-2=>%g bits\n',ST,ST*204*8);
fprintf('\tSe ha recibido %g trama/s MPEG-2=>%g bits\n',B_rx,B_rx*188*8);
fprintf('\t\t%g trama/s MPEG-2 útiles (%g bits)\t\t%g trama/s sin cabecera válida\n',T_rx,T_rx*188*8,B_rx-T_rx);

fprintf('\t\t\t\t\t\t\t\t\t\t 11 trama/s por OuterDeinterleaver y %g trama/s por errores\n',B_rx-T_rx-11);
fprintf('\tEn el buffer de bits del transmisor se dispone todavía de %g bits\n',a);
fprintf('\tEn el buffer del receptor se dispone todavía de %g bits\n',b);
fprintf('\tSe ha desaprovechado en los buffers %g bits=>%g bytes=>%g bloque/s(204 bytes)=>%g
trama/s\n',a+b,(a+b)/8,(a+b)/8/204,(a+b)/8/204);
fprintf('\t------------------------------\n\n')
[a,b]=size(total_simbolos_rx);
fprintf('\n\n\t\t\t\t\tTRANSMISIÓN Y RECEPCIÓN OFDM REALIZADA\n\n\n');
fprintf('\t-----------------------------------\n');
fprintf('\tSe ha transmitido %g símbolo/s OFDM útiles con %g portadoras de %g bits/portadora=>%g
bits\n',b/log2(n),N_portadoras,log2(n),a*b);
if P~=0
fprintf('\tSe ha tramsitido con un error en frecuencia de %g\n',freq_error);
fprintf('\tSe ha corregido un error en frecuencia estimado de %g\n',fe_est);
end
if time_error~=0
fprintf('\tSe ha tramsitido con un error en tiempo de %g muestras(portadoras)\n',time_error);

fprintf('\tSe ha corregido un error en tiempo estimado de %g\n',(fft_mode*(1+D)+1)-(time_error_est));
fprintf('\tEl simbolo OFDM a la entrada del receptor tiene %g muestras(portadoras)\n',(fft_mode*(1+D)));
fprintf('\tEl inicio de símbolo OFDM empieza en la muestra %g (tras %g muestras desde la primera
end
fprintf('\t----------------\n\n')
if BERvsEBN0==1&&ESN0dB==0
fprintf('\tMontecarlo especifica que para esta BER teórica de %g  (EBN0=%g dB) se necesitan %g
fprintf('\t%g bits=>%g símbolos MPEG2 =>%g bloques (RS) MPEG2=>%g tramas
end
if BERvsEBN0==1&&ESN0dB==1
fprintf('\tMontecarlo especifica que para esta BER teórica de %g  (ESN0=%g dB) se necesitan %g
fprintf('\t%g bits=>%g símbolos MPEG2 =>%g bloques (RS) MPEG2=>%g tramas
end
if ~isempty (total_simbolos_rx)
%-----------------------------------------------------
%   Cálculo de BER.Sistema OFDM                    Calcula la BER de la transmision del sistema
%-----------------------------------------------------
disp('Entre bloques de mapeo y desmapeo/detección:');
[ber_teorica,ber_sistema_ofdm]=ber_bits(total_simbolos_tx,total_simbolos_rx,EBN0dB,R,n,ESN0dB,paso
%-----------------------------------------------
%   Graficación de BERvsEBN0. Sistema OFDM
%-----------------------------------------------
if BERvsEBN0==1
    EBN0dB_valores(e)=EBN0dB;
    BER_valores_ofdm(e)=ber_sistema_ofdm;
    e=e+1;
    if EBN0==max(EBN0dB_vect)
        berteorica=theorical_BER(ESN0dB,n,min(EBN0dB_vect),max(EBN0dB_vect),pasos);
        hold on
        semilogy(EBN0dB_valores,BER_valores_ofdm,'r.-');
        if n~=4
        if ESN0dB==0
        title (sprintf('BERvsEB/N0 de la modulación %gQAM. Sistema OFDM',n));
        xlabel('EBN0(dB)'); ylabel ('BER'); legend(sprintf('%gQAM teórica',n),sprintf('%gQAM en modo %gK con
alpha=%g, C=%g y D=%g',n,modo,alpha,C,D));
        elseif ESN0dB==1
        title (sprintf('BERvsES/N0 de la modulación %gQAM. Sistema OFDM',n));
        xlabel('ESN0(dB)'); ylabel ('BER'); legend(sprintf('%gQAM teórica',n),sprintf('%gQAM en modo %gK con
alpha=%g, C=%g y D=%g',n,modo,alpha,C,D));
        end
        else
        if ESN0dB==0
        title (sprintf('BERvsEB/N0 de la modulación QPSK. Sistema OFDM'));
        xlabel('EBN0(dB)'); ylabel ('BER'); legend(sprintf('QPSK teórica'),sprintf('QPSK en modo %gK con
        elseif ESN0dB==1
        title (sprintf('BERvsES/N0 de la modulación QPSK. Sistema OFDM'));
        xlabel('ESN0(dB)'); ylabel ('BER'); legend(sprintf('QPSK teórica'),sprintf('QPSK en modo %gK con
        end
        grid on
        hold off
        end
    saveBER(ESN0dB,BER_valores_ofdm,EBN0dB_valores,modo,n,alpha,C,D);
    end
end
else
fprintf('-------------------------------------------------\n')
fprintf('\t No se ha recibido ningún simbolo OFDM. Prueba aumentando el SNR de la transmisión\n');
fprintf('-------------------------------------------------\n')
end                                                Del If de antes del cálculo de BER del sistema OFDM
if ~isempty (total_tramas_rx)
```

```
639 %-------------------------------------------------------
640 %   Graficación de VARvsESN0
641 %-------------------------------------------------------
642 if VARvsESN0==1&&P~=0
643    ESN0dB_valores(e)=EBN0dB;
644    VAR_valores(e)=mean(fe_dif);                    calculo de la variancia a partir de la diferencia de
                                                       valores al cuadrado
645    e=e+1;
646    if EBN0==max(EBN0dB_vect)
647       saveVAR(VAR_valores,ESN0dB_valores,MCRBfreq_vect,P,0,0,0);
648       semilogy(ESN0dB_valores,MCRBfreq_vect,'b')
649       hold on
650       semilogy(ESN0dB_valores,VAR_valores,'rv-');
651       title (sprintf('VARvsES/N0 Oerder&Meyr en canal AWGN '));
652       xlabel('ESN0(dB)'); ylabel ('VAR'); legend(sprintf('MCRB teórica'),sprintf('Oerder&Meyr de NL con orden
653       grid on
654       hold off
655    end
656 end
657 end                                                Del FOR de barrido EBN0
658 %-------------------------------------------------------------------------
659 %**********FINAL DEL EL PROGRAMA**************
660 %-------------------------------------------------------------------------
```

Código 36 DVBT_simulado.m

Anexo B HERRAMIENTAS PARA EL ANÁLISIS DE PRESTACIONES

Para analizar el comportamiento de un sistema de comunicaciones digital y validar el correcto funcionamiento del mismo existen diferentes herramientas que permiten cuantificarlo de cierta manera. Al ser un sistema de comunicaciones digital y, por tanto, ser el objetivo principal de la comunicación el conseguir enviar y recibir tramas de bits, una de las herramientas utilizadas es la de la probabilidad de error de bit, cuantificada con el parámetro BER (*Bit Error Rate*). Además, en este proyecto se ha implementado diferentes bloques estimadores de parámetros de corrección. En este caso, el interés radica en conocer la varianza de estimación de los bloques implementados en función de las características del enlace (por ejemplo, la potencia de ruido con la que se ve contaminada la señal transmitida).

B.1 BIT ERROR RATE

El rendimiento de un sistema de comunicaciones digital es medido en términos de la tasa de bits en error, o lo que es lo mismo, la relación de bits erróneos detectados en el receptor. En general, la BER depende del método de modulación, del esquema de codificación, del tipo de forma de onda usada, de la potencia del transmisor, de las características del canal y del esquema de demodulación.

La representación convencional del rendimiento de un sistema digital en un canal lineal contaminado por ruido muestra el BER frente a E_b/N_0, donde E_b representa la energía de bit y N_0 representa la densidad espectral de potencia del ruido introducido en un canal. De este modo se obtiene la eficiencia del sistema. Para un nivel de ruido dado, el BER puede ser reducido incrementando la energía asociada a cada bit, transmitiendo con mayor potencia o con un mayor periodo de bit. La meta en las comunicaciones digitales es obtener un determinado valor de BER con la mínima energía de bit posible.

Otra posible representación del rendimiento puede ser la que nos da el comportamiento de la BER frente a la relación señal a ruido, SNR. Esta representación indica la sensibilidad del sistema y suele ser usada en comunicaciones analógicas. En el caso de las comunicaciones digitales, en vez de hablar de BER frente a SNR se debería hablar de SER (*Simbol Error Rate*) frente a SNR.

Los valores de BER frente a E_B/N_0 para distintas modulaciones paso banda están cuantificadas cumpliendo la siguiente expresión:

$$BER_{M-QAM} = \frac{4(\sqrt{M}-1)}{b\sqrt{M}} \cdot \left[\frac{1}{2} - \frac{1}{2} erf\left(\sqrt{\frac{3b}{(M-1)} \frac{E_b}{N_0}} \frac{1}{\sqrt{2}} \right) \right] \qquad \text{(C.1)}$$

donde M representa el número de puntos, o niveles, de la constelación (M=4 para QPSK, M=16 para 16-QAM y M=64 para 64QAM), b representa el número de bits empleados en cada modulación PAM cumpliéndose $n=log_2(M)$, y erf representa una función de error, muy relacionada con el mundo de las comunicaciones digitales.

En caso de querer representar el BER en función de E_S/N_0, la expresión (C.1) cambia a:

$$BER_{M-QAM} = \frac{4(\sqrt{M}-1)}{b\sqrt{M}}\left[\frac{1}{2}-\frac{1}{2}erf\left(\sqrt{\frac{3}{(M-1)}\frac{E_S}{N_0}}\frac{1}{\sqrt{2}}\right)\right] \qquad \text{(C.2)}$$

ya que $E_S=b\cdot E_B$.

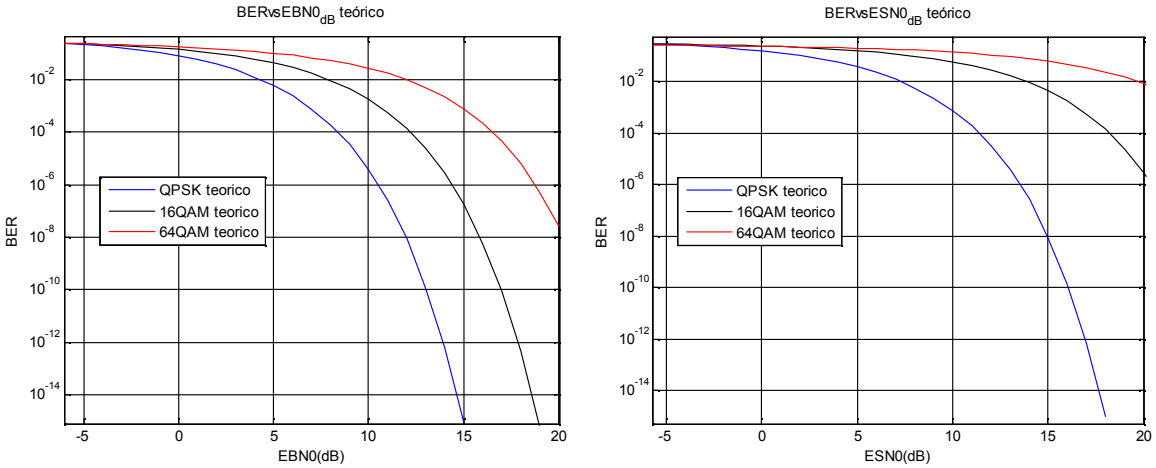


Fig B.1 Valores teóricos de BERvsEBN0 y de BERvsESN0 de las modulaciones PAM

Los valores de BER reflejados en la Fig B.1 son valores de referencia a partir de los cuales se evalúa el funcionamiento del simulador implementado. Se ha creado una función llamada *theorical_BER*.m que calcula y representa el BER teórico de las modulaciones PAM utilizadas en el simulador implementado.

Para obtener el BER para la evaluación del sistema implementado se recurre a comparar los bits enviados con los bits recibidos, dada una E_B/N_0 o una E_S/N_0, de la siguiente manera:

$$BER = \frac{\#\ bits\ erróneos}{\#bits\ enviados} \qquad \text{(C.3)}$$

En el entorno de simulación implementado se realiza una aproximación de este valor puesto que para que sea comparable con los valores teóricos de BER se debería realizar una transmisión de infinitos bits. Por ello, la aproximación realizada se hace en base a la denominada regla de Montecarlo, que indica el número de bits a transmitir para obtener un BER de determinada resolución:

$$\#\ bits\ a\ generar \approx \frac{100}{BER} \qquad \text{(C.4)}$$

Por ejemplo, si se requiere poder obtener un valor de BER=10^{-2} el número de bits a comparar es de 10^4 pero si se requiere obtener un valor de BER=10^{-6} se deben comparar 10^8 bits.

B.1.1 Código implementado

```
function theorical_BER( ESN0dB, n, EBN0min, EBN0max,pasos )
if ESN0dB==0
u=1;
b=log2(n);
AQAM=4*(sqrt(n)-1)/(b*sqrt(n));
for i=EBN0min:pasos:EBN0max
EBN0(u)=i;
EBN0lin(u)=10^(EBN0(u)/10);
Q_QAM(u)=sqrt(3*b*EBN0lin(u)/(n-1));
Q(u)=(1/2)*(1-erf(Q_QAM(u)/sqrt(2)));
BER(u)=AQAM*Q(u);
u=u+1;
end
elseif ESN0dB==1
u=1;
b=log2(n);
AQAM=4*(sqrt(n)-1)/(b*sqrt(n));
for i=EBN0min:pasos:EBN0max
EBN0(u)=i;
EBN0lin(u)=10^(EBN0(u)/10);
Q_QAM(u)=sqrt(3*EBN0lin(u)/(n-1));
Q(u)=(1/2)*(1-erf(Q_QAM(u)/sqrt(2)));
BER(u)=AQAM*Q(u);
u=u+1;
end
end
%-------------------------------------
%    Graficamos resultados
%-------------------------------------
if(EBN0min~=EBN0max)
figure;
semilogy(EBN0,BER);
if ESN0dB==0
title (sprintf('BERvsEB/N0 teórica de la modulación %gQAM',n));
xlabel('EBN0(dB)'); ylabel ('BER')
elseif ESN0dB==1
title (sprintf('BERvsES/N0 teórica de la modulación %gQAM',n));
xlabel('ESN0(dB)'); ylabel ('BER')
end
grid on
end
```

Código 37 theorical_BER.m

```
function [ berteorica, perror ] = ber_bits(total_tramas_tx,total_tramas_rx,EBN0dB,R,n,ESN0dB,pasos)
[nada,tx]=size(total_tramas_tx)
[nada,rx]=size(total_tramas_rx);
total_rx=min(tx,rx);                                    %mínimo número de tramas en común
dif=xor(total_tramas_tx(:,1:total_rx),total_tramas_rx(:,1:total_rx));
Bitsmal=sum(sum(dif));                                  %primero sumamos por columnas y luego por filas
[a,b]=size(total_tramas_rx);
perror=Bitsmal/(a*b);
disp(sprintf('La probabilidad de error de bit con EBN0=%gdB es BER=%g',EBN0dB,perror));
berteorica=theorical_BER(ESN0dB,n,min(EBN0dB),max(EBN0dB),pasos);
disp(sprintf('La probabilidad de error de bit teórica con EBN0=%gdB es BER=%g\n',EBN0dB,berteorica));
```

Código 38 ber_bits.m

B.2 VARIANZA

La varianza es un cálculo utilizado para caracterizar estadísticamente los bloques estimadores implementados. La varianza es una medida de la dispersión de una variable aleatoria *X* respecto a su esperanza *E[X]*. Se calcula como:

$$\mathrm{var}[\tilde{\tau}] = E[(\tilde{\tau} - E[\tilde{\tau}])^2] \tag{C.3}$$

siendo $\tilde{\tau}$ el parámetro a estimar y $E[\tilde{\tau}]$ el parámetro estimado.

En el caso que ocupa el proyecto el interés radica en observar la variabilidad de la estimación del parámetro en cuestión frente unas condiciones dadas de EB/N0 o ES/N0.

Para obtener el valor de varianza se parte de que el parámetro a estimar es conocido y se compara dicho valor con el estimado. Tras múltiples realizaciones, que deben seguir también la regla de Montecarlo descrita en el apartado anterior, se obtiene un vector con un gran número de diferencias cuadráticas, $(\tilde{\tau} - E[\tilde{\tau}])^2$ y, para obtener el valor cuadrático medio o varianza, se realiza la media de dicho vector.

En algunos casos se compara esta curva de varianza respecto EB/N0 o ES/N0 con una cota inferior llamada cota de Cramér-Rao para la varianza de un estimador insesgado, basado en la información de Fisher. En lo que acontece al proyecto presentado únicamente se ha presentado gráficas de varianza comparadas con su respectiva cota inferior en los casos en que se ha podido encontrar el valor de dicha cota sin tener que realizar los cálculos a partir de las bases teóricas de Cramér, ya que se considera que es un aspecto que se escapa de los objetivos del proyecto.

El cálculo de las varianzas de cada estimador se ha implementado directamente sobre el código del programa principal, *DVB-T_simulado.m*, presentado en el anexo A.

Anexo C LA CODIFICACIÓN PARA EL CONTROL DE ERRORES

C.1 INTRODUCCIÓN

En el estudio de los codificadores de canal se presenta el código Reed-Solomon, el cual resulta ser muy ventajoso. Su probabilidad de error en relación con la señal a ruido está cercana al límite de Shannon [9] y presenta mayor eficiencia sobre otros códigos correctores de error en cuanto a ganancia del código. La clave para hacer del código Reed-Solomon una aplicación tecnológica fue la implementación de un algoritmo eficiente de decodificación desarrollado por Berlekamp [10].

El código Reed_solomon es un código corrector de errores basado en bloques en donde el codificador procesa un bloque de símbolos de datos, a los que agrega redundancia para producir un bloque de símbolos codificados. En la actualidad, los códigos Reed-Solomon se utilizan para corregir errores en varios sistemas incluyendo los dispositivos de almacenamiento (discos compactos, DVD, códigos de barras,...), comunicaciones inalámbricas o móviles (telefonía celular, enlaces de microondas, comunicaciones satelitales, televisión digital DVB, módems de alta velocidad....).

C.2 PROPIEDADES DE LOS CÓDIGOS REED SOLOMON

El código Reed Solomon es un subconjunto de los códigos BCH, códigos cíclicos que presentan entre sus parámetros (n,k,t) una relación entre los símbolos de datos k, del código total n y del número máximo de errores por ser corregidos t, y son de bloques lineales. Un código Reed.Solomon se especifica como RS (n,k) con símbolos de s bits, es decir, el codificador toma k símbolos de los s bits y añade símbolos de paridad para hacer una palabra código de n símbolos. Por tanto, existen n-k símbolos de paridad de s bits cada uno. Un decodificador puede corregir hasta t símbolos que contienen errores en una palabra código, siendo ($2t$=n-k), [10]. En la Fig C.1 se muestra una palabra típica de la codificación RS que se conoce como un código sistemático puesto que los datos se dejan inalterados y los símbolos de paridad se anexan.

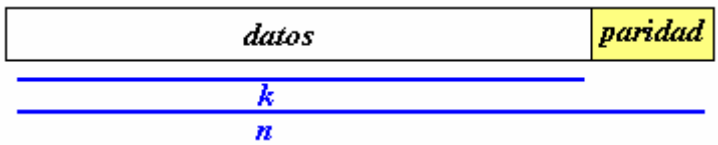


Fig C.1 Palabra código Reed Solomon

Para codificar la trama con esta estructura, ésta se debe procesar a través de un circuito digital o un software que opere bajo los fundamentos de campo finito de Galois.

C.3 CAMPOS DE GALOIS APLICADOS A LA CODIFICACIÓN RS

Los códigos RS se basan en un área especializada de la matemática llamada campos de Galois o campos finitos. Un campo finito tiene la propiedad de que las operaciones aritméticas sobre elementos del campo siempre tienen un resultado en el campo. Un codificador o decodificador Reed-Solomon debe ser capaz de realizar estas operaciones aritméticas [11].

C.4 GENERADOR POLINOMIAL

Una palabra de código RS se genera usando un polinomio especial. Todas las palabras de código válidas son divisibles exactamente por el polinomio generador:

$$g(x) = (x - \alpha^{i})(x - \alpha^{i+1})...(x - \alpha^{i+2t}) \tag{C.1}$$

La palabra código c(x) se genera de

$$c(x) = g(x) \cdot i(x) \tag{C.2}$$

donde *g(x)* es el polinomio generador expresado en (C.1), *i(x)* es el bloque de información, *c(x)* es una palabra de código válida y α es un elemento primitivo del campo o raíz.

El primer paso corresponde a la definición del campo de Galois para la codificación, el cual estará definido en función de la longitud del símbolo, *m*, el cual permite conocer el polinomio reducible del campo de Galois:

$$GF(2^{m}) \tag{C.3}$$

Las bases teóricas que sustentan este codificador están dadas por el polinomio en su forma general, expresada como:

$$g(x) = \prod_{i=0}^{n-k-1} \left(x - \alpha^{h \cdot \alpha \cdot (1+i)}\right) \tag{C.4}$$

Al expandir el polinomio expresado en (C.4), se obtiene:

$$g(x) = G_{n-k-1}x^{n-k-1} + G_{n-k-2}x^{n-k-2} + ... + G_1 + G_0 \qquad \text{(C.5)}$$

donde *n* es la longitud de la palabra codificada (en símbolos), *k* es la longitud del mensaje codificado (en símbolos) y *m* es la longitud del símbolo.

C.5 BLOQUES FUNCIONALES DE UN DECODIFICADOR RS

Para el estudio de un decodificador se consideran los siguientes bloques [12]:

- Cálculo de síndrome: se trata de un cálculo similar al cálculo de paridad. Un código de palabra RS tiene 2*t* síndromes que dependen solamente de los errores, no de la palabra transmitida. Los síndromes pueden ser calculados al sustituir las *2t* raíces del polinomio generador *g(x)* en *r(x)*

- Localizador de errores: encontrar el lugar del símbolo erróneo implica resolver de forma simultánea ecuaciones con *t* incógnitas. Existen varios algoritmos rápidos para hacerlo, los cuales toman ventaja de la estructura matricial especial de los código RS reducen de gran forma el esfuerzo computacional requerido.

- Polinomio localizador del error: Este polinomio de error se puede lograr utilizando el algoritmo Berlekamp-Massey o el algoritmo de Euclides. Para localizar los símbolos erróneos se debe seguir un procedimiento que implica resolver unos sistemas de ecuaciones, el primero de los cuales se representa mediante la siguiente expresión:

$$S_{t+j} = f_1 \cdot S_{t+j-1} + ... + f_{t-1} \cdot S_{j+1} + f_t \cdot S_j \qquad \text{(C.6)}$$

- Corrección de errores: Este paso también implica resolver ecuaciones con *t* incógnitas para poder encontrar los valores verdaderos que deberán ser sustituidos en las posiciones correspondientes, para así poder reproducir el mensaje correcto que se intentó enviar. Esto se realiza con el algoritmo de búsqueda de Chien, donde en $1 \leq j \leq t$ los f_i son las incógnitas y S_i los componentes del síndrome calculado. La posición en la que están los errores se obtiene de los exponentes de las raíces de un polinomio *f(x)*, cuya expresión es:

$$f(x) = x^t + f_1 \cdot x^t + ... + f_{t-1} \cdot x + f_t \qquad \text{(C.7)}$$

Anexo D MODELO ESTADÍSTICO DE CANAL. RAYLEIGH

La potencia recibida en comunicaciones móviles se puede caracterizar según tres velocidades de variación temporal:

- A largo plazo: Son las variaciones que experimenta la señal recibida cuando se promedia un periodo de tiempo grande. Por tanto, modelan las pérdidas de propagación según la distancia. Caracterizado por una ley de propagación, frecuencia, distancia y es dependiente del entorno. Uno de los modelos más conocidos es el de Okuura-Hata que pude ser aplicable a distancias de 1 a 100 Km y en rangos frecuenciales de 150 hasta 1500 MHz.

- A medio plazo: Son las variaciones de potencia recibida debido a las zonas oscuras (*shadowing*) producidas por las ondulaciones del terreno o por grandes objetos que se interponen entre las antenas. Siguen una distribución Log-Normal.

- A corto plazo: Son variaciones de la potencia recibida en un periodo de tiempo muy corto. Están producidas por el multicamino que se produce por objetos cercanos al receptor (del orden de 150 λ). Dichas variaciones están caracterizadas por modelos estadísticos ya que son muchas las reflexiones y difracciones que ocurren en un entorno próximo al receptor. Se modelan mediante distribuciones estadísticas como la distribución Rayleigh (NLOS) o la Ricean (LOS+NLOS).

La siguiente figura ilustra las diferentes velocidades de variación temporal de la potencia de la señal recibida.

Fig D.1 Velocidades de variación de la potencia recibida en función del tiempo

El interés, desde el punto de vista de este proyecto, radica en la propagación multicamino, que generan variaciones de potencia a corto plazo y que se ha supuesto NLOS (entorno de ciudad). Por este motivo, se tratará únicamente la función de densidad de probabilidad que permite evaluar estadísticamente las variaciones de este tipo de canales NLOS.

D.1 DISTRIBUCIÓN RAYLEIGH

La distribución de Rayleigh se utiliza en radiocomunicaciones para describir la variación estadística de la envolvente (amplitud) de la señal recibida en un escenario de propagación multicamino. Esta distribución se puede aplicar en el caso de que existan desvanecimientos de la señal causados por las reflexiones de los rayos de la propagación multicamino, existiendo un número grande de ondas reflejadas y ninguna onda directa (canal NLOS). En este escenario, si r representa la envolvente de la señal, la función densidad de probabilidad viene dada por la ecuación siguiente:

$$p(r) = \frac{r}{\sigma^2} e^{-\frac{r^2}{2\sigma^2}} \quad \text{para } r \geq 0 \tag{D.1}$$

donde σ^2 es la varianza y donde el valor de σ, desviación típica, es la mitad del valor cuadrático medio:

$$\sigma = \sqrt{\frac{r_{rms}^{\ 2}}{2}} \quad \text{para } r \geq 0 \tag{D.2}$$

La siguiente figura muestra la función de probabilidad de la envolvente Rayleigh para diferentes valores de la varianza:

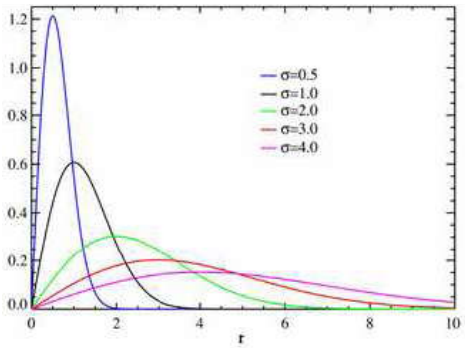


Fig D.2 Función de densidad de probabilidad de la envolvente Rayleigh

Para producir una distribución Rayleigh se generan dos distribuciones gausianas (I(t), fase, y Q(t), cuadratura) de media 0 y una determinada desviación típica σ. La distribución Rayleigh surge de la evaluación del módulo de la envolvente compleja, r:

$$r = \sqrt{I(t)^2 + Q(t)^2} \tag{D.3}$$

La relación utilizada para determinar la σ de las componentes gausianas en fase y cuadratura a partir de las potencias de los taps de un modelo TDL es:

$$P_{tap} = r_{rms}^{\ 2} = 2\sigma^2 \tag{D.4}$$

REFERENCIAS

[1] S. Haykin y Moher. *Modern Wireless Communications*. Editorial Pearson Prentice Hall. Ontario, Canada, Febrero 2005. ISBN: 9-7801-30224-729

[2] Fuquin Xiong. *Digital Modulation Techniques*. Second Edition. Editorial Artech House, 2006. ISBN: 1-58053-863-0.

[3] D. Roldán. *Comunicaciones inalámbricas, un enfoque aplicado*. Editorial Ra-Ma 2005, ISBN: 84-7897-621-3.

[4] V. Paladino. *Introducción a la compresión de video bajo el estándar MPEG-2*. Monografía para el curso de Codificación de Imágenes y Video. Instituto de Ingeniería Eléctrica de la Facultad de Ingeniería. Noviembre de 2002

[5] Dr Mary Ann Ingram. *OFDM simulation using Matlab*. Smart Antenna Research Laboratory. Agosto 2000. http://www.ece.gatech.edu/research/labs/sarl/tutorials/OFDM/Tutorial_web.pdf

[6] ETSI; EN 300 744 v1.5.1. *Digital Video Broadcasting (DVB); Framing Structure, channel coding and modulation for Digital Terrestrial Television (DVB-T)*. Octubre 2004

[7] ISO/IEC 13818: *Information technology - Generic coding of moving pictures and associated audio information*.

[8] ETSI; EN 300 421: *Digital Video Broadcasting (DVB); Framing Structure, channel coding and modulation for 11/12 GHz Satellite Services*. Octubre 2004

[9] C. E. Shannon. *A Mathematical Theory of Communication*. Bell System Technical Journal, vol. 27, 1948.

[10] B. S. Wicker y V. K Bhargava. *Reed Solomon Codes and their applications*. Qiley-IEEE Press. 1999. 336 p. ISBN: 0-7803-5391-9.

[11] F. J. López Martínez. *Diseño de transmisor y receptor para redes inalámbricas W-MAN*. Tesis de Grado. Escuela Técnica de Telecomunicación. Universidad de Málaga, 2005

[12] Xilinx. *Reed-Solomon Solutions with Spartan-II*. Xilinx System Generator v.2v1 for Simulink.

[13] G. Santella, *OFDM with guard interval and sub-channel equalization in a 2-resolution transmission scheme for digital television broadcasting*, Communications, 1994. ICC 94, SUPERCOMM/ICC '94, Conference Record, Serving Humanity Through Communications. IEEE International Conference on,V.1, 374-380 1994

[14] T. Rappaport, *Wireless Communications: Principle and Practice*. Second Edition Editorial Penitence Hall, 2002

[15] Sing Sin Hie, *Radio Channel modeling for mobile ad hoc wireless networks*. Thesis. Naval Postgraduate School, Monterey, California. Junio 2004.

[16] Marelli, D. Minyue Fu, *Subband methods for OFDM equalization*,

Communications, 2003. ICC '03. IEEE International Conference, V4, 2350-2354 (2003)

[17] H. Parviainenm P. Kyösti, X.Zao, *Novel radio channell models for evaluation of DVB-H Broadcast Systems*. 17th Annual IEEE International Symposium in Personal, Indoor and Mobile Radio Communications. 2006.

[18] Š.Matějka; *SFN channel model derivation based on impulse response measurement for DVB-T;* Department of Radio Engineering,Faculty of Electrical Engineering, Czech Technical University in Prague. 2003

[19] J.C.Silva, N.Souto*; Efficient simulation of tapped delay channel models using converted discrete time channel models.* Ericsson EuroLab, Netherlands, 2004.

[20] Jian-Hua, Wang Yong; *Symbol Synchronization Technique in COFDM Systems*; IEEE transactions on Broadcasting, Vol.50 No.1; Marzo 2004.

[21] Zhi-xing Yang; *On the synchronization Techniques for Wireless OFDM Systems*, IEEE Transactions on Broadcasting, Vol 52 No2; Junio 2006

[22] L. Jingyi; *The effect of Filling Unique Words to Guard Interval for OFDM System*; Samsung Electronic. IEEE 802.16 Broadband Wireless Access Working Group. Septiembre 2002

[23] J.Granado, A.Torralba; *Estimadores de desviación de frecuencia y tiempo en OFDM.* Dpto. Ingeniería Electrónica, Universidad de Sevilla.

[24] Jan-Jaap van de Beek, O. Edfors; *On channel Estimation in OFDM Systems*. IEEE Internacional Symposium. 1995

[25] M. Pukkila, *Channel Estimation Modelling.* Nokia Research Center. 2000.

[26] P. Corral, R. García; *Técnicas de estimación de canal en el estándar 802.16-2004*; Área de Teoría de la Señal y Comunicaciones, Universidad Miguel Hernández, Alicante.

[27] Yushi Shen, E. Martinez; *Channel Estimation in OFDM Systems*; Freescale Semiconductor Application Note; AN3059 Rev. 0, 1/2006

[28] M. Luise, M. Marselli; *Low-Complexity blind carrier frequency recovery for OFDM signals over frequency selective radio channels*. IEEE transactions on communications. Vol. 50, No 7. Julio 2002.

[29] J. L. Massey; *Optimum Frame Synchronization*. IEEE transactions on communicaations, Vol-Com 20, No.2, Abril 1972

[30] P. Tolstrup; *Some Optimum and Suboptimum Frame Synchronizers for Binary Data in Gaussian Noise.* IEEE Transactions on commnicatons. Junio 1973.

Marcos Alsinella Fernández

18 de Septiembre de 2008

El sistema de modulación ODFM es utilizada en diversas aplicaciones de banda ancha, tanto en comunicaciones por cable como en aplicaciones inalámbricas. Presenta numerosas ventajas frente a sistemas de banda ancha de portadora única ya que permite una alta eficiencia espectral, una fácil ecualización y una reducción del ISI. Por el contrario, presenta dificultades inherentes a su estructura, que son de vital importancia solventar, entre las cuales se encuentran los altos requisitos de sincronización. Este proyecto presenta métodos de sincronización de tiempo y frecuencia implementados y evaluados sobre una plataforma software basada en Matlab®, que recoge el sistema completo de transmisión basado fielmente en el estándar DVB-T. Tras una presentación de los principios de la modulación OFDM, en este documento se presenta un estudio detallado de este sistema de transmisión y su implementación, formando conjuntamente una plataforma de simulación para la evaluación de los estimadores implementados.

El sistema de modulació OFDM és utilitzat en diverses aplicacions de banda ampla, tant en comunicacions per cable com en aplicacions sense fils. Presenta nombrosos aventatges davant sistemes de banda ampla de portadora única ja que permet una alta eficiència espectral, una fàcil equalització i una reducció de l'ISI. D'altra banda, presenta dificultats inherents a la seva estructura, que són de vital importància resoldre, entre els quals es troben els alts requisits de sincronització. Aquest projecte presenta mètodes de sincronització de temps i de freqüència implementats i avaluats sobre una plataforma software basada en Matlab®, que recull el sistema complet de transmissió basat fidelment en l'estàndard DVB-T. Després de realitzar una presentació dels principis de la modulació OFDM, en aquest document es recull un estudi detallat d'aquest sistema de transmissió y de la seva implementació, formant conjuntament una plaraforma de simulació per la avaluació dels estimadors implementats.

OFDM system is used in several wide band applications, in communications by cable as well as in wireless applications. OFDM presents numerous advantages in front of single carrier wide band systems since it allows a high spectral efficiency, an easy equalization and ISI reduction. On the other hand, it presents inherent difficulties of its structure, that they are from vital importance to be solved, among which the high requirements of synchronization are found. This project presents methods of synchronization of time and frequency implemented and evaluated by means of a software platform based on Matlab®, which picks up the complete transmission system based faithfully on the standard DVB-T. After carrying out a presentation of the OFDM principles, in this document a detailed study of this system of transmission and its implementation is picked up, forming together a simulation platform for the evaluation of the implemented estimators.