ESCUELA DE INGENIERÍA Y ARQUITECTURA

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA Y COMUNICACIONES

Trabajo Fin de Grado:

Título del trabajo: ESTUDIO DE MODULACIONES CAP (CARRIERLES AMPLITUDE PHASE) PARA SU UTILIZACIÓN EN REDES ÓPTICAS PASIVAS

English tittle: ANALYSIS OF CAP (CARRIERLESS AMPLITUDE PHASE) MODULATIONS FOR PASSIVE OPTICAL NETWORKS

Autor:
Jaime Gimeno Ferrer

Director:
Juan Ignacio Garcés Gregorio

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Año
2019-2020

Resumen

En este trabajo de fin de grado se estudia el uso de modulaciones Carrierless Amplitude Phase (CAP) para redes ópticas. El origen de esta propuesta viene de la complejidad y elevado coste que presentan otros tipos de modulaciones propuestas para este tipo de redes, particularmente las que se usan en redes ópticas basadas en multiplexación por división en longitud de onda, que requieren de componentes necesarios para la modulación y demodulación óptica que suponen un incremento en el coste. Es por esto que en este proyecto se estudia la modulación CAP como un sistema flexible, de menor complejidad y de coste reducido en comparación con otras modulaciones avanzadas que precisan de moduladores más caros y complejos.

Una vez explicado todos los principios de funcionamiento de CAP y sus diferentes variantes se procede a la implementación de un modulador y demodulador CAP programados en MATLAB. Si bien estos códigos no se han podido probar en un sistema real en el laboratorio debido a las limitaciones impuestas por el covid-19, han sido implementados para ello sea posible sin ninguna modificación adicional, teniendo en cuenta las problemáticas asociadas al sistema real. Por ello, se ha hecho uso de un toolbox de MATLAB para simular sistemas ópticos llamado OPTILUX, diseñado por el profesor Paolo Serena, con el cual se han diseñado los escenarios para los diferentes tests.

Las principales conclusiones que hemos extraído de este estudio tratan la optimización de parámetros a la hora de diseñar el sistema óptico, la comparación entre las diferentes variantes de CAP (CAP, multiCAP...) y soluciones a los diferentes problemas provenientes de la transmisión por el sistema óptico.

Finalmente, se expone como, gracias a su flexibilidad, bajo coste, baja complejidad y alta eficiente espectral, CAP podría convertirse en una de las modulaciones más competitivas para sistemas de transmisión ópticos. Esto sería una vez se hayan diseñado solucionado ciertos inconvenientes actuales tales como la necesidad de una respuesta frecuencial del canal plana para que funcione correctamente.

Agradecimientos

A mis padres, a mi hermana y a toda mi familia y amigos, gracias a quienes soy quien soy y hacia quienes sólo puedo expresar mi sincero agradecimiento por apoyarme durante esta etapa académica que hoy culmina.

A mi director de este trabajo Nacho junto a sus compañeros Miguel, David y José Antonio, por su acompañamiento, su energía y su apoyo durante esta fase que nos ha unido.

Índice general

Índice de figuras

Capítulo 1

Introducción

La actual demanda de servicios de transmisión de datos requiere sistemas de comunicación que dispongan de un gran ancho de banda junto a una rápida velocidad de acceso. Esto es debido a servicios emergentes tales como videojuegos on-line, servicios multimedia basados en vídeo, servicios de almacenamiento en la nube, etc.

Se prevé que las redes ópticas pasivas (PON) junto a la multiplexación de longitud de onda (WDM) lleguen a cumplir con la gran demanda de capacidad de los sistemas de transmisión de la próxima generación. Gracias a la multiplexación WDM podemos transmitir varias señales de datos independientes por una sola fibra óptica haciendo que cada señal se envíe en un rango de frecuencias distinto. Por ejemplo, es posible conseguir una conexión de 100 Gb/s usando cuatro canales de 25.8 Gb/s [2]. Este tipo de sistema tiene una serie de problemas con los que hay que tratar. En los sistemas WDM los principales problemas son una diafonía no lineal mayor (non-linear crosstalk) y un significante incremento en el coste [3].

Para compensar los inconvenientes de los sistemas WDM-PON se propone usar modulaciones más avanzadas con mejores eficiencias espectrales haciendo uso de un numero de canales WDM mínimo, ahorrando de esta forma costosos componentes ópticos necesarios para la demodulación óptica. Se han explorado múltiples modulaciones (M-PSK, M-QAM, OFDM...) [4], pero la mayoría precisan costosos y complicados sistemas de transmisión. Es por esto que se proponen los sistemas de modulación CAP como una alternativa flexible, de menor complejidad, y coste reducido.

La modulación Carrierless Amplitude and Phase (CAP) es un esquema de modulación multidimensional y multinivel propuesta a mediados de los 70s por Falconer [5]. Tiene muchas similitudes con el esquema QAM dado que transmite dos flujos de datos ortogonales en fase. La principal diferencia está en que CAP no usa osciladores si no que usa filtros ortogonales para transmitir ambos flujos. Esto hace que con un sistema más simple se consiga la misma eficiencia espectral, y es por esta razón que se comenzó a utilizar mucho en los 90s [6]. Más tarde, se intentó explotar el ancho de banda de los cables de cobre ya instalados, pero se descubrió que CAP era muy sensible a canales de espectro no plano, requiriendo ecualizadores muy complejos y por ello se fue haciendo menos popular [7].

Últimamente se han llevado a cabo muchos estudios para la utilización de CAP en sistemas ópticos de corto alcance. Entre ellos destacamos como ejemplo el estudio de un enlace de 15 km con una tasa de 102 Gb/s haciendo uso de un solo canal WDM y detección directa [8]. Otro estudio realizado implicaba una transmisión CAP de 40 Gb/s haciendo uso de filtros transversales económicos demostrando unos resultados satisfactorios [9]. Estos y muchos otros estudios nacen del hecho de ser CAP un buen candidato para convertirse en la modulación por excelencia a la hora de establecer enlaces ópticos de corto alcance y bajo coste.

En este trabajo se va explicar el funcionamiento de esta modulación, así como distintas variantes o modificaciones, y del mismo modo se simularán entornos reales para evaluar el desempeño obtenido.

1.1. Objetivos

El objetivo de este proyecto es implementar unos códigos para modular y demodular señales CAP y ciertas modificaciones. La idea inicial era introducir esas señales generadas en un entorno óptico real y analizar la calidad de la transmisión y de los códigos generados. Debido a la situación dada por el covid-19 ha resultado imposible llevar a cabo la prueba en un escenario real, por lo que se ha optado por simular estos escenarios con un toolbox de simulación de comunicaciones ópticas de MATLAB llamado OPTILUX y probar con éste la calidad de los códigos diseñados en este trabajo.

Así, se crearán los códigos para las diferentes modulaciones CAP y se pondrán a prueba en entornos de simulación de OPTILUX. Como ventaja, eliminamos muchas limitaciones que aparecerían en el laboratorio dado que podemos diseñar entornos con los parámetros que queramos tales como la longitud de la fibra óptica. Por contra, perdemos fiabilidad en los resultados dado que no conocemos completamente cómo OPTILUX se diferenciará frente a un sistema real. No obstante, es un entorno de simulación que se está usando desde el año 2009 y tiene una solidez bastante probada en los últimos años. En cualquier caso, sería interesante evaluar la veracidad de los resultados obtenidos mediante experimentos reales.

Con los resultados obtenidos analizaremos el desempeño de la modulación bajo distintas circunstancias así como solucionaremos diferentes problemas o inconvenientes que aparezcan en una transmisión óptica utilizando modulación CAP simple y mediante alteraciones en el esquema.

1.2. Organización de la memoria

La estructura de la memoria que vamos a seguir es la siguiente.

- En el Capítulo 2 se introducen los conceptos necesarios para entender las modulaciones CAP y sus alteraciones. Además, se habla de las características típicas de un sistema de transmisión óptico.
- En el Capitulo 3 se explica como se han implementado los códigos para las modulaciones CAP, los códigos completos se pueden consultar en el Anexo A.
- La herramienta usada para simular el sistema óptico es introducida en el Capítulo 4.
- En una segunda parte del Capítulo 4 se realizan una serie de tests con OPTILUX para estudiar diferentes aspectos de las modulaciones y el sistema de transmisión.
- Por último, en el Capítulo 5, se exponen las conclusiones obtenidas en este trabajo y se comenta el posible trabajo futuro.

Capítulo 2

Marco teórico

Este capítulo va a abordar los conceptos teóricos sobre las modulaciones CAP que implementaremos y valoraremos en los capítulos posteriores. Además, se verán las características básicas del sistema óptico las cuales influirán en el desempeño final de la transmisión.

2.1. Modulación CAP

En la figura 2.1 podemos ver una representación de un sistema de transmisión CAP en su mínima expresión. Como hemos mencionado anteriormente, este sistema es muy similar a una transmisión con modulación M-QAM, de hecho, el mapeo del bloque "CONSTELLATION MAPPING" es un mapeo típico M-QAM. La principal diferencia se da en la sustitución de osciladores y mezcladores (mixers) por dos filtros que proporcionan la ortogonalidad entre ambas cadenas de datos.

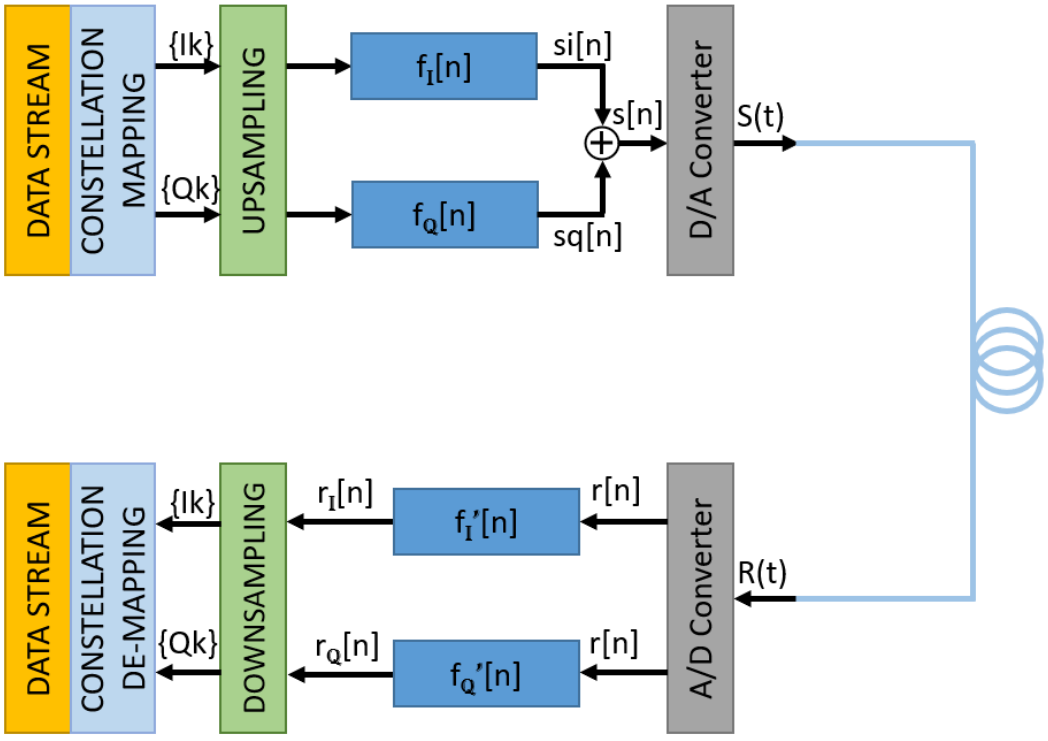

Figura 2.1: Esquema transmisión CAP

A continuación, vamos a ir paso por paso desarrollando los bloques de todo el sistema para poder tener una comprensión general de una modulación CAP.

2.2. Transmisor CAP

En la figura 2.1, la parte correspondiente a la transmisión CAP envuelve la parte superior, desde la cadena de bits (Data Stream) hasta el conversor digital-analógico (D/A Converter) ambos incluidos. En esta sección vamos a explicar el funcionamiento y el papel de cada uno de los bloques correspondientes a la transmisión.

2.2.1. Flujo de bits (Data stream)

El primer paso es generar los bits que vamos a transmitir, estos bits serán la información de mínimo nivel que se querrá recuperar en el receptor. En el caso de que queramos poner a prueba el sistema lo mejor es generar una secuencia binaria aleatoria (PRBS) de una longitud suficiente para que se tenga una visión general.

2.2.2. Mapeo de los bits (Constellation mapping)

Una vez se dispone de una secuencia de bits el siguiente paso es el mapeo de los bits en una constelación M-QAM. Esto implicará transformar los bits en símbolos y colocar esos símbolos ordenadamente en la constelación.

Primero se van a separar los bits en grupos de bits, y, dado que M representa el número de símbolos y estamos tratando con bits (que pueden tomar dos valores), K = $log_2 M$ será el número de bits que forman un símbolo. Así que agruparemos los bits en grupos de K bits.

Cuando ya estén los bits agrupados, se procederá a situarlos en la constelación. Una constelación de dimensión M puede ordenarse de muchas formas, véase la figura 2.2. Sabiendo que el ruido del canal va a afectar a los símbolos desplazándolos sobre la constelación, lo más probable es que, si el error no es excesivamente grande, un símbolo se quede cerca de su sitio ideal o que se desplace a un símbolo adyacente. Si elegimos la ordenación de los símbolos de forma que la diferencia entre símbolos adyacentes sea de un solo bit podemos minimizar el error en el bit (el error en el símbolo seguirá siendo el mismo). Es aquí donde entra la codificación Gray, que consigue específicamente este resultado. Si nos fijamos en la figura 2.2 vemos como, para una constelación con codificación binaria, en muchos de los casos la diferencia entre símbolos adyacentes es mayor a un bit, sin embargo, con la constelación con codificación Gray conseguimos que la distancia sea siempre de un bit para todos los símbolos adyacentes [10].

Para el mapeo se utilizará entonces la constelación con codificación Gray de la figura 2.2. Conseguiremos así para cada símbolo un par de coordenadas en fase y cuadratura (I y Q). Estás coordenadas son los datos que van a ser transmitidos (tras ser procesados en los siguientes bloques). El receptor deberá identificar los símbolos correspondientes a las coordenadas recibidas.

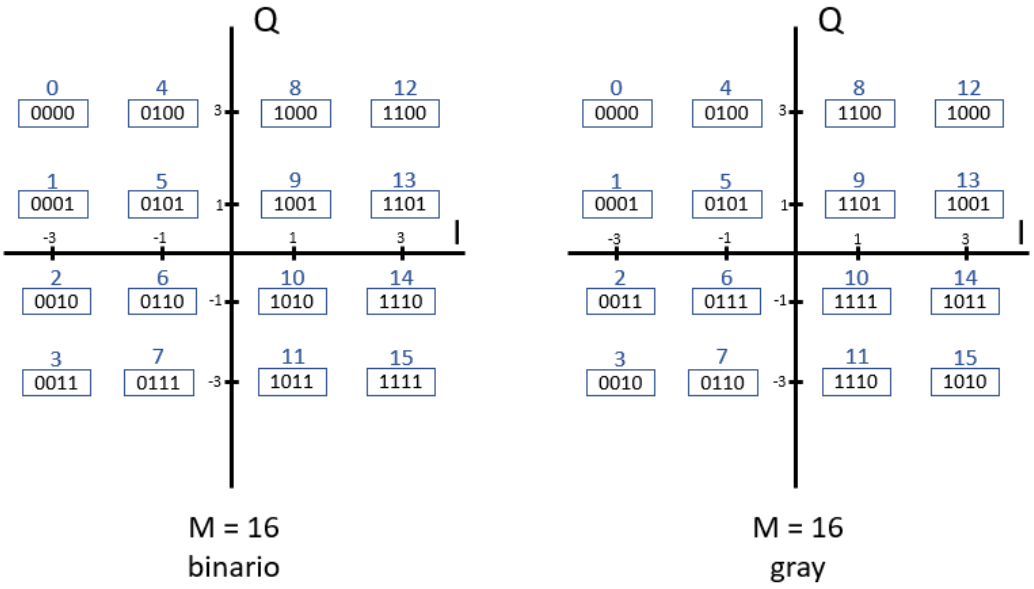

Figura 2.2: Constelaciones con M = 16 usando constelaciones binaria y gray

2.2.3. Sobre-muestreo y filtrado

Vamos a explicar ahora el objetivo de dos bloques que van de la mano. El sobre-muestreo (upsampling) y el filtrado (filtros I y Q). Dado que es necesario primero entender el proceso de filtrado, dejaremos el sobre-muestreo para el final de esta sección.

Esta parte de la transmisión tiene dos objetivos, el primero es dar forma al pulso de símbolo transmitido (no tiene por qué ser un pulso rectangular) para mejorar el desempeño de la transmisión. El segundo es ortogonalizar los dos flujos de datos, coordenadas en fase y cuadratura (I y Q), para poder enviarlos superpuestos y poder separarlos de nuevo en recepción.

Pulso conformador (pulse shaping)

Antes de nada, hay que conocer el criterio de Nyquist para ISI (Interferencia Inter-Simbólica). La ISI es una forma de distorsión en el instante de muestreo de un símbolo producida por otro símbolo adyacente que puede empeorar seriamente la calidad de la transmisión. El criterio de Nyquist para eliminar la ISI puede ser expresado como:

$$h(nT_s) = \begin{cases} 1 & \text{si } n = 0 \\ 0 & \text{si } n \neq 0 \end{cases} \tag{2.1}$$

Esto implica que el pulso de un símbolo sea nulo en los instantes de muestreo del resto de símbolos. Un buen ejemplo de un pulso de Nyquist podemos verlo en la figura 2.3. En los instantes de muestreo (nT_s) el único valor no nulo es el pico del pulso conformador correspondiente.

Además de buscar eliminar la ISI, también buscamos reducir lo máximo posible el ancho de banda para mejorar la eficiencia espectral. Los pulsos que transmitimos en la práctica no son pulsos cuadrados, ya que esto requeriría un ancho de banda enorme (teóricamente infinito). Si usamos pulsos más "suavizados", sin un cambio de pendiente tan brusco, podemos minimizar el ancho de banda necesario para transmitir estos pulsos. Nyquist se dio cuenta de que el mínimo ancho de banda que se podia obtener para una transmisión con un tiempo de símbolo T_s(segundos/símbolo) era $BW_{min} = 1/T_s$.

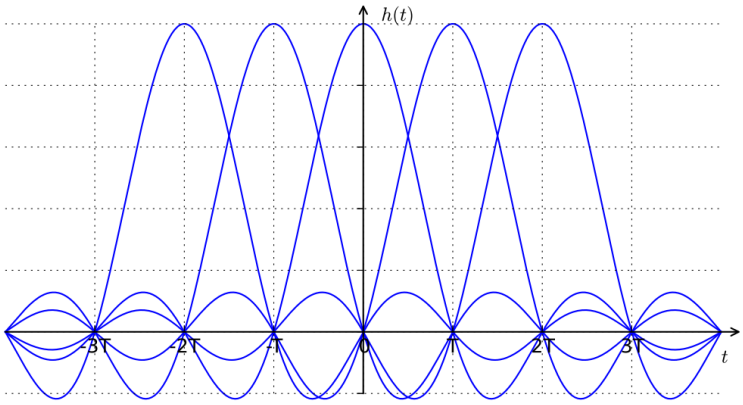

Figura 2.3: Pulsos que cumplen el criterio de Nyquist

Un pulso que cumple tanto con el criterio de Nyquist para la ISI como con el mínimo ancho de banda es el ya mostrado en la figura 2.3, se trata de un pulso conformador sinc.

$$sinc(t) = Ts\frac{\text{sen}\,\frac{\pi}{Ts}t}{\pi t} \tag{2.2}$$

Donde Ts es el periodo de símbolo. Este seno, como ya hemos dicho alcanza los dos requisitos para el ISI y para el ancho de banda. Teóricamente tiene un ancho de banda de $\frac{1}{T_s}$, el menor ancho de banda posible para una transmisión con una tasa $R_s = \frac{1}{T_s}$ manteniendo el criterio de la ISI.

El filtro coseno realzado (raised cosine)

En la práctica es imposible obtener un pulso sinc que se extienda infinitamente en el tiempo, por ello se suele usar una aproximación, el coseno realzado. El coseno realzado tiene la siguiente respuesta impulsional:

$$h(nT_s) = \begin{cases} \frac{\pi}{4T} sinc\left(\frac{1}{2\beta}\right) & \text{si } t = \pm\frac{T}{2\beta} \\ \frac{1}{T} sinc\left(\frac{t}{T}\right) \frac{\cos\frac{\pi\beta t}{T}}{1-\left(\frac{2\beta t}{T}\right)^2} & \text{si } t \neq \pm\frac{T}{2\beta} \end{cases} \tag{2.3}$$

Con $sinc(x) = \sin \pi x / \pi x$

Siendo T el tiempo de símbolo y β un parámetro llamado factor de roll-off que toma un valor entre 0 y 1. β tiene que ver con el acercamiento del coseno realzado a una sinc, si $\beta = 0$, entonces la respuesta impulsional es una sinc. Conforme más pequeño sea el valor, más próximo será el pulso a una sinc y más difícil de implementar será. Valores típicos de β se encuentran entre 0.3 y 0.5.

Como se puede intuir, cuanto menor sea β, menor ancho de banda será necesario. En la figura 2.4 se puede ver el espectro de un coseno realzado dependiendo del valor de β. El ancho de banda es:

$$BW_{rc} = \frac{1+\beta}{T_s} = R_s(1+\beta) \tag{2.4}$$

Con T_s el tiempo de símbolo y R_s la tasa de símbolo. En la figura 2.5 se ve como es cierto que, usando un coseno realzado en lugar de un pulso rectangular, conseguimos un contenido frecuencial mucho más compacto y por tanto una eficiencia espectral mayor.

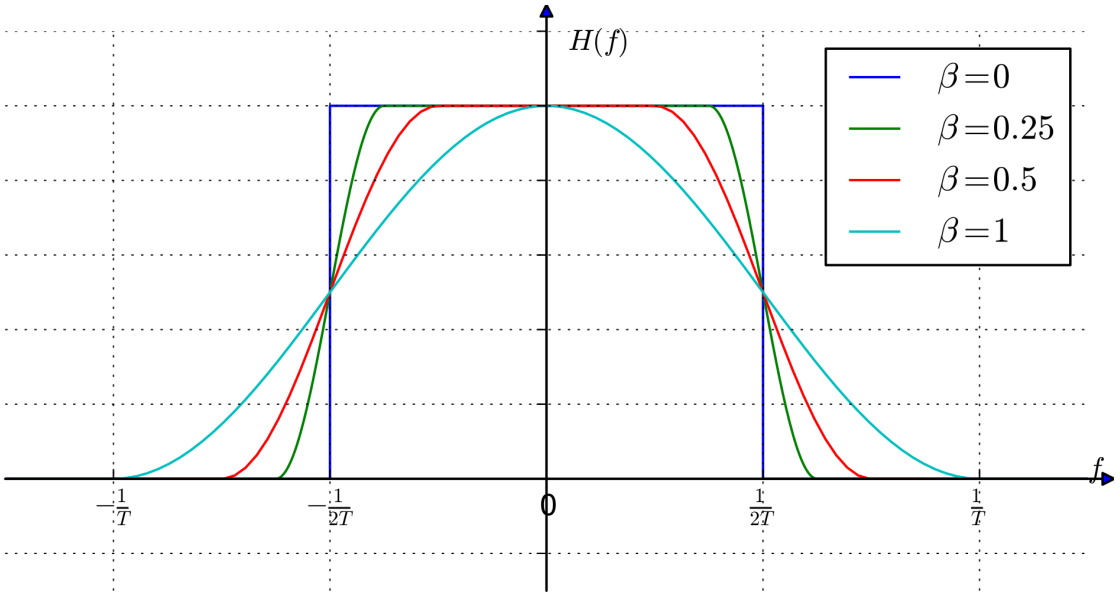

Figura 2.4: Espectro de un filtro coseno realzado en función de β

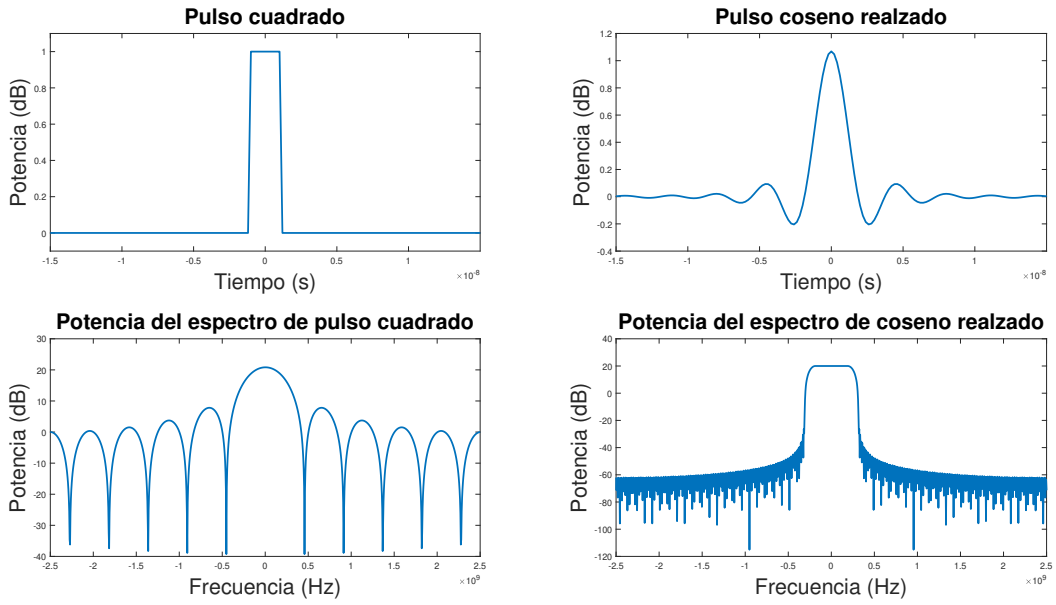

Figura 2.5: Ejemplo pulso cuadrado y pulso coseno realzado ($\beta = 0.25$) en tiempo y frecuencia

Para la transmisión CAP se utilizan los filtros coseno realzado por sus buenas características de ISI y eficiencia espectral que se acaban de explicar. En realidad, se usa el filtro raiz de coseno realzado ya que tendremos dos en cascada, uno en el transmisor y otro en el receptor, ambos conformarán un solo filtro coseno realzado.

Filtros ortogonales

En la figura 2.1, vemos como, a la salida de los filtros, las dos componentes $s_i[n]$ y $s_q[n]$ se suman. Estas dos componentes podrán ser recuperadas de forma independiente en el receptor. Para poder sumar las señales como una sola y poder recuperarlas necesitamos que sean ortogonales entre sí. Conseguiremos la ortogonalidad entre ambas señales haciendo que los filtros I y Q sean, del mismo modo, ortogonales. Siguiendo la misma idea que en la modulación QAM, se van a añadir una sinusoide desfasada $\pi/2$ en cada señal para obtener esa ortogonalidad pero, en lugar de multiplicar la señal directamente con estas sinusoides se incluirán en el filtro CAP, ahorrándonos así osciladores y mezcladores.

La ortogonalidad que se consigue entre los filtros es en el mismo ancho de banda ya que son ortogonales en fase, comparten el modulo del espectro. Esto será interesante para poder usar varias bandas de frecuencia, lo cual se verá en el siguiente apartado (modulaciones multi-cap).

Tras añadir todos los conceptos mencionados, se obtienen los filtros I y Q:

$$f_I(t) = g_{RRC}(t) \cos 2\pi f_c t \tag{2.5}$$

$$f_Q(t) = g_{RRC}(t) \sin 2\pi f_c t \tag{2.6}$$

Siendo g_{RRC} un filtro raíz de coseno realzado (root raised cosine) y f_c la frecuencia portadora de las sinusoides ya mencionadas. Como ya se ha mencionado, se usa la raíz del filtro explicado ya que tenemos dos filtros en cascada (uno en el transmisor y otro en en el receptor) que al ser raíces de coseno realzado, conforman un solo filtro coseno realzado. La figura 2.6 muestra las formas de los filtros en el dominio de tiempo.

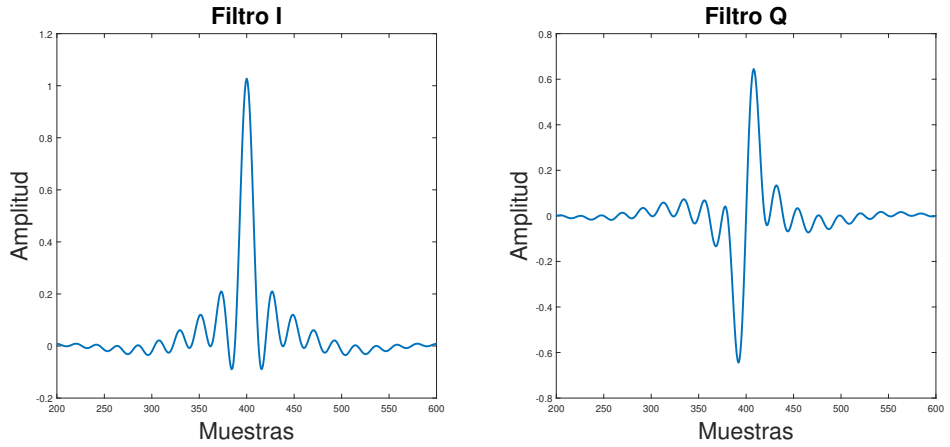

Figura 2.6: Filtros f_I y f_Q

Las señales ortogonales a sumar de la figura 2.1 s_I y s_Q serán:

$$s_I[n] = I_k * f_I[n] \tag{2.7}$$

$$s_Q[n] = Q_k * f_Q[n] \tag{2.8}$$

Donde $*$ denota convolución lineal. La señal enviada será $s[n] = s_I[n] + s_Q[n]$.

Sobre-muestreo (upsampling)

Para poder hacer un filtrado correcto es preciso hacer un sobre-muestreo previamente, ahora veremos cual es el motivo.

Dado que el objetivo es que el filtro actúe de filtro conformador (que dé forma a los símbolos), se debe hacer que este pulso se genere en los periodos de símbolo. Una forma sencilla de verlo matemáticamente para la señal s_I (recíproco para s_Q) es:

$$s_I[n] = \sum_k p[n] * I_k \delta[n - kN_s] \tag{2.9}$$

Siendo $p[n]$ el filtro conformador generado teniendo en cuenta la frecuencia de muestreo f_s, I_k el valor de la muestra k-ésima de la coordenada I y N_s el periodo de símbolo en muestras, $N_s = f_s/R_s$ muestras, con R_s la tasa de transmisión (símbolos por segundo). El significado de esta expresión es que se va a situar un pulso conformador de amplitud I_k cada N_s muestras ($T_s = f_s/R_s$) tal y como vemos en la Figra 2.3.

Debemos tener en cuenta que I_k es un vector de muestras, generado sin tener en cuenta la frecuencia de muestreo. Para que al convolucionar con el filtro se sitúen los pulsos en los instantes correspondientes debemos hacer que las muestras de I_k estén equi-espaciadas el número de muestras correspondientes a un periodo, y es por esto mismo que añadimos $f_s/R_s - 1$ ceros entre cada muestra. En la figura 2.7 se pude ver la importancia del sobre-muestreo. Si no se aplica el sobre-muestreo, no se está cumpliendo el criterio de Nyquist para la ISI, los pulsos no se anulan donde se sitúan el resto de pulsos. Sin embargo, vemos como, si aplicamos un sobremuestreo adecuado, el criterio de Nyquist para la ISI se cumple.

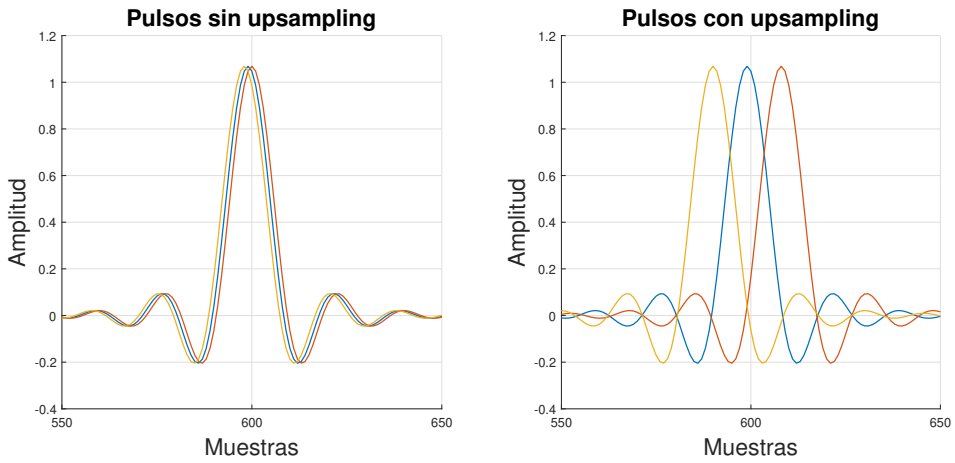

Figura 2.7: Muestras pasadas por un filtro coseno realzado sin y con sobre-muestreo

Cuando ya hayamos aplicado el sobre-muestreo y el filtrado a cada una de las dos cadenas de datos (I y Q), se sumarán en una sola señal $s[n]$. Esta señal es la que va a ser transmitida.

2.2.4. Conversor D/A (D/A Converter)

Para poder transmitir la señal necesitamos convertirla al dominio eléctrico, a una señal analógica. No nos vamos a extender en este apartado, solo se debe saber que se va a transformar la señal digital $s[n]$ en una señal eléctrica empleando la frecuencia de muestreo con la cual hemos generado los filtros y hemos sobre-muestreado las componentes I y Q.

En el apartado del medio (la fibra óptica) se explicará cómo introducimos esta señal eléctrica en la fibra óptica haciendo una conversión electro-óptica.

2.3. Receptor CAP

En esta sección se va a explicar, del mismo modo que para la transmisión, la recepción CAP. Se van a recorrer todos los bloques correspondientes a la recepción los cuales incluyen desde el conversor A/D (A/D Converter) hasta la recuperación de la cadena de datos (Data Stream) de la figura 2.1.

2.3.1. A/D CONVERTER

En el bloque de conversor D/A la señal procesada digitalmente se había convertido a una señal analógica para transmitirla por el canal. En recepción tendremos de nuevo esa señal analógica pero alterada por el canal. El objetivo de este bloque es devolver la señal analógica al dominio digital. Se muestreará la señal con una frecuencia de muestreo que no tiene que ser la misma que la del emisor (por ello habrá que trabajar con esta nueva frecuencia de muestreo) y se enviará al siguiente bloque, los filtros adaptados.

2.3.2. Filtros adaptados (Matched filters)

La señal muestreada se corresponde con la señal enviada $s[n]$ pero con las sufridas alteraciones del canal. Para poder recuperar las dos cadenas de datos por separado (coordenadas I y Q) hay que filtrar la señal con los filtros adaptados a f'_I y f'_Q. Estos filtros deberán ser diseñados conforme a la nueva frecuencia de muestreo que se tiene en recepción, dada por el conversor A/D. La expresión para los filtros adaptados es:

$$f'_I[n] = f_I[-n] \tag{2.10}$$

$$f'_Q[n] = f_Q[-n] \tag{2.11}$$

Dada la ortogonalidad de la señales y los filtros ya mencionada, con este filtrado obtenemos las señales $r_I[n]$ y $r_Q[n]$ tal y como se ve en la ecuaciones 2.12 y 2.13.

$$r_I[n] = r[n] * f'_I[n] \tag{2.12}$$

$$r_Q[n] = r[n] * f'_Q[n] \tag{2.13}$$

Estas señales aun no se corresponden con las coordenadas I_k y Q_k, sino con las señales $s_I[n]$ y $s_Q[n]$. Para obtener las muestras deseadas habrá que pasar estas señales por el siguiente bloque.

2.3.3. Infra-muestreo (Downsampling)

Ya se ha explicado la necesidad del sobre-muestreo, una vez se han entendido este concepto es muy fácil comprender la razón de ser del infra-muestreo. Se trata de quedarse solo con la muestra correspondiente al instante de muestreo óptimo dejando atrás las muestras propias del resto de los pulsos conformadores. Véase la figura 2.8, en esta figura solo nos interesan las muestras en negro, dado que el resto de parte de los pulsos conformadores no nos aportan ninguna información. Es

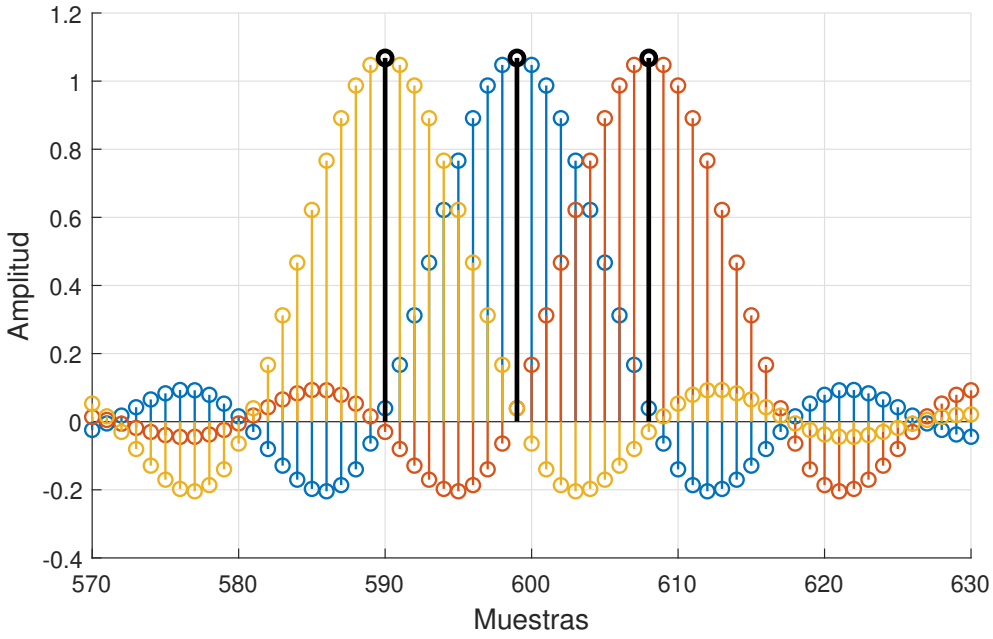

Figura 2.8: Pulsos con instantes donde aplicar downsampling

por esto que se toman solo los valores cada $N_s = f_s/R_s$ muestras, siendo f_s la nueva frecuencia de muestreo impuesta por el conversor A/D.

Así, el infra-muestro es el proceso inverso al sobre-muestreo y se hace para eliminar la información que ya no nos interesa (muestras no pico de los pulsos). Tras haber aplicado el infra-muestreo a las dos señales $r_I[n]$ y $r_Q[n]$, tendremos las muestras que nos interesan que son las coordenadas I_k y Q_k con las cuales podremos deducir qué símbolos han sido transmitidos.

2.3.4. Desmapeo (De-Mapping)

En este punto ya se tienen los pares de coordenadas I_k y Q_k de cada símbolo, esto quiere decir que ya podemos situar cada símbolo en su posición concreta de la constelación y con esto hace la traducción al número de símbolo y a bits tal y como vemos en la figura 2.9. Cuando las coordenadas ya han sido traducidas a bits y estos bits han sido encadenados, ya se habrá recuperado la cadena de bits inicial (Data Stream).

Hasta este punto se ha explicado el funcionamiento más simple de una modulación CAP. A este sistema se le pueden añadir disantos bloques para optimizar ciertas características, en los siguientes apartados se van a ver diferentes formas de aprovechar lo nos ofrece esta modulación, todas ellas son ligeros cambios que afectarán al desempeño final.

2.4. CAP de dimensión alta

En la modulación CAP simple se ha visto como mediante el uso de filtros ortogonales se han transmitido dos cadenas de datos en una misma señal. En este caso eran las coordenadas de una constelación, pero se puede enviar cualquier cadena de datos. Una posible cuestión a plantearse es si es posible aumentar el número de filtros ortogonales sobre el mismo ancho de banda para tener más cadenas de datos en una misma señal y así obtener una mejor eficiencia espectral tal y como

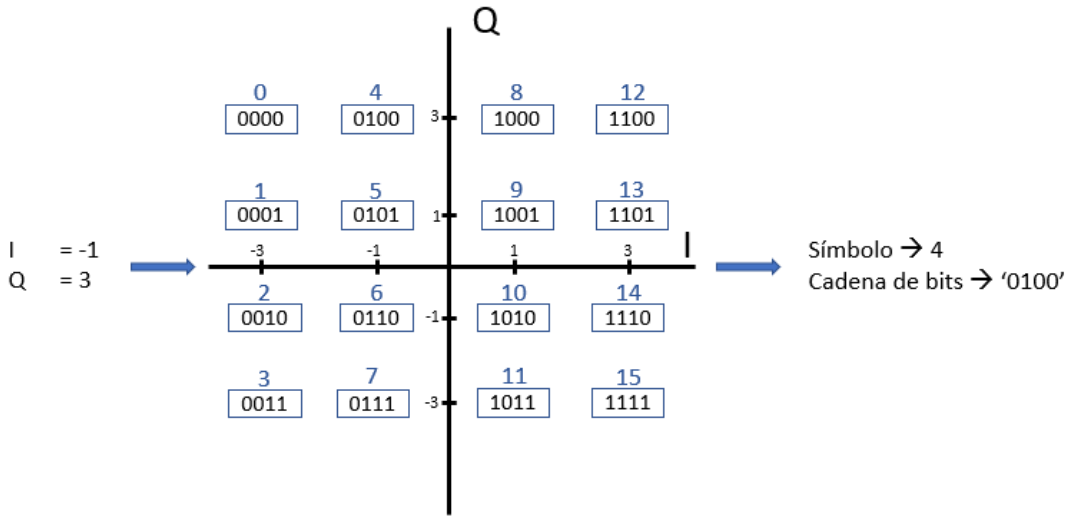

Figura 2.9: Ejemplo desmapeo con coordenada I y Q

vemos en la figura 2.10. El número de filtros ortogonales indicará el número de dimensiones de la modulación.

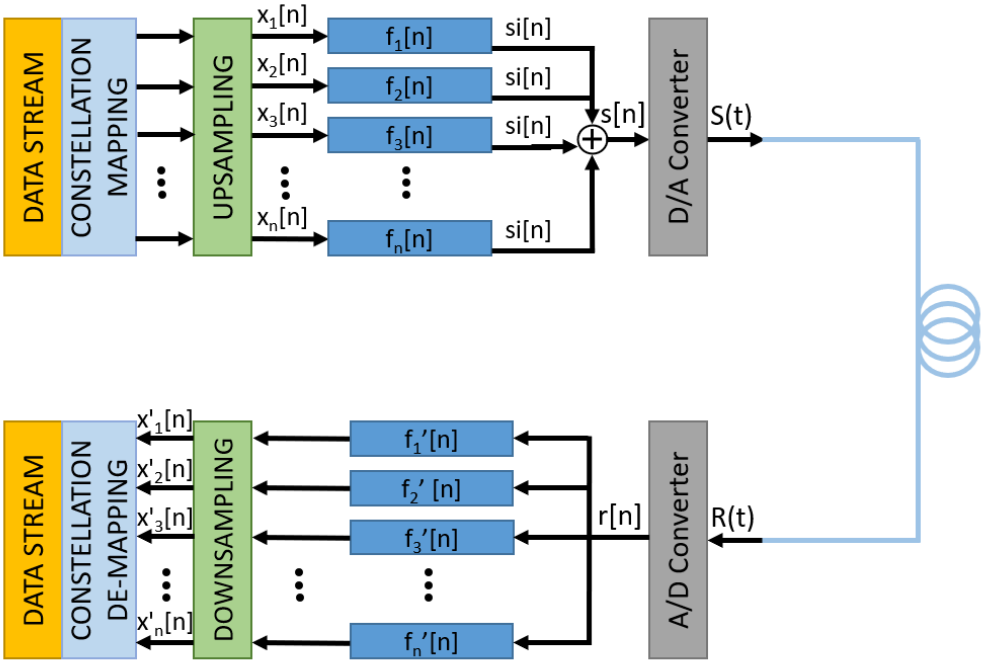

Figura 2.10: Esquema modulación CAP de dimensión alta

Para resolver la cuestión planteada hay que acudir a teorema de dimensionalidad demostrado en [11]. Teniendo un periodo de símbolo T_s, un ancho de banda mínimo W y una dimensionalidad de la modulación N:

$$2WT_s = N \tag{2.14}$$

Se puede ver la relación directamente proporcional que tienen el mínimo ancho de banda y el número de dimensiones $W = \frac{N}{2T}$, esto significa que un incremento en el número de dimensiones

hará que crezca el mínimo ancho de banda necesario para la modulación. Además de esto, el ratio de muestras por símbolo (N_{ss}) es también linealmente proporcional al número de dimensiones N [12], lo que supone que si hay un mayor número de dimensiones se necesitará un mayor factor de sobre-muestreo.

Así, resulta imposible mejorar la eficiencia espectral simplemente incrementando el número de dimensiones dado que para mantener el mismo ancho de banda se debería reducir la tasa de símbolo o aumentar el factor de sobre-muestreo. De hecho, el incremento del ancho de banda que supone aumentar el número de dimensiones compensa el mayor número de símbolos en el alfabeto, obteniendo la misma eficiencia espectral que para el caso de una modulación CAP independientemente del número de dimensiones [13]. Sin embargo, la ventaja que supone usar un mayor número de dimensiones no reside en el las características espectrales.

La razón por la cual podría ser interesante un aumento en la dimensionalidad reside en que es una buena forma de enviar diferentes servicios a diferentes usuarios. Pero la complejidad que implica generar un alto numero de filtros ortogonales en un ancho de banda reducido es alta. Podríamos conseguir esta misma ventaja de otra forma (Modulación MultiCAP).

2.5. Modulación MultiCAP

A continuación se introduce un nuevo concepto, ya se ha visto que no podemos mejorar la eficiencia espectral simplemente añadiendo filtros ortogonales en las mismas bandas de frecuencia, pero sí que podemos enviar distintas modulaciones CAP en distintas bandas de frecuencia. A esta modulación se le llama modulación CAP multibanda o MultiCAP. Un esquema para este tipo de modulación para 2 bandas de frecuencia podemos verlo en la figura 2.11, esta figura puede extenderse para un número de bandas N para N usuarios.

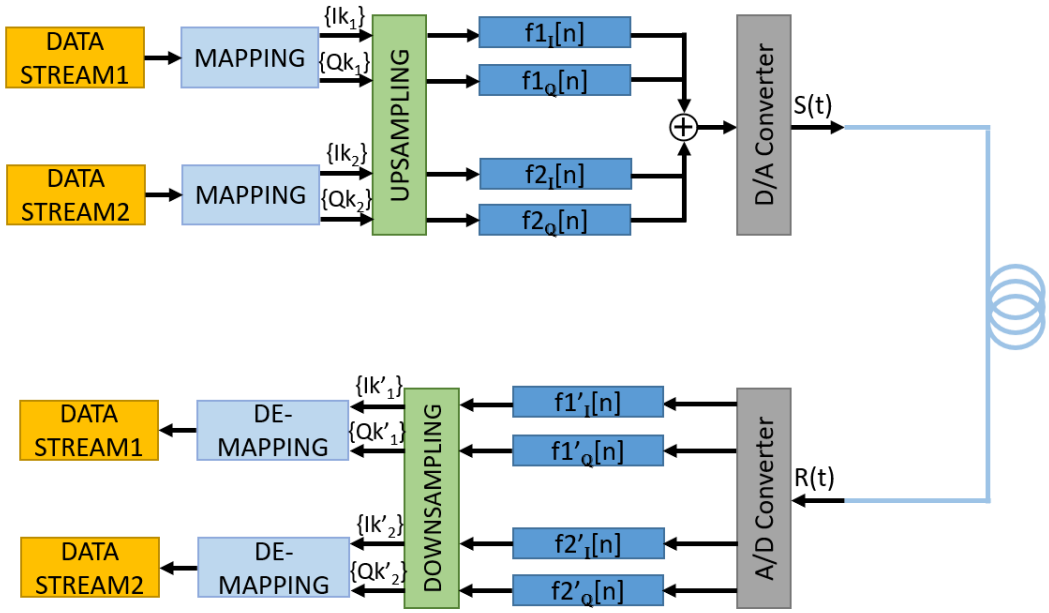

Figura 2.11: Esquema modulación MultiCAP

El esquema de una modulación MultiCAP no es más que una extensión de la modulación CAP en la cual tenemos varias cadenas de datos a transmitir, cada una de estas cadenas será

mapeada independientemente obteniendo sus respectivas coordenadas I y Q, las cuales serán sobre-muestreadas y filtradas.

La siguiente diferencia está en los filtros. Los filtros van a tener el mismo formato que los usados en la modulación CAP, pero ahora nos deberemos preocupar especialmente por la frecuencia portadora ya que la información de cada cadena de datos irá transmitida en una banda de frecuencia que será ortogonal (en frecuencia) con el resto. Podemos ver como los filtros van a dar esa ortogonalidad en frecuencia en la figura 2.12, donde se han simulado tres filtros con distintas portadoras y se ve como cada uno de los tres filtros usados ocupa una banda específica sin interferir al resto de filtros. Esto se consigue considerando el ancho de banda de cada filtro el cual viene dado por la ecuación 2.4, $BW = R_s(1 + \beta)$. Así, la distancia en frecuencia entre las portadoras deberá ser al menos de BW, en la implementación haremos que se cumpla siempre esto añadiendo un parámetro para ajustar la distancia extra (de seguridad) entre las bandas, f_{esp} (figura 2.13).

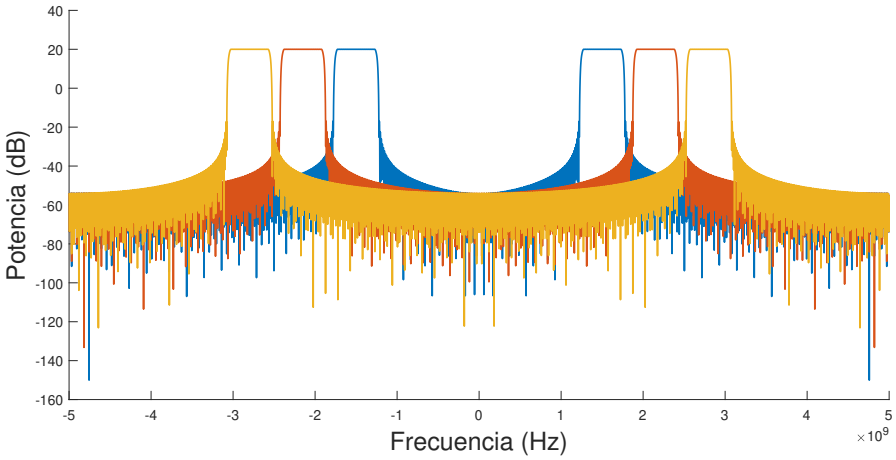

Figura 2.12: Potencia del espectro de los filtros multiCAP (cada banda es un color)

Una vez filtradas todas las señales se sumarán en una sola. Podremos recuperarlas de nuevo en recepción gracias a las dos ortogonalidades conseguidas, en fase para los pares de coordenadas de cada banda y en frecuencia para las distintas bandas. La ecuación que define esta señal a enviar se puede expresar como:

$$s[n] = \sum_{i=1}^{n} \left[I_k^i * f_I^i\right] + \left[Q_k^i * f_Q^i\right] \tag{2.15}$$

Donde $*$ denota convolución lineal, I_k^i son la muestras ya sobre-muestreadas de las coordenadas I del filtro i-ésimo (recíproco para Q_k^i) y f_I^i y f_Q^i son el par de filtros para la banda i-ésima. Una representación frecuencial de esta señal podría verse como en la figura 2.13. Donde cada banda contiene la información de una transmisión.

La recepción es igual que en el sistema CAP, los filtros adaptados actúan como filtros paso banda seleccionando la banda específica y además separarán las componentes en fase y en cuadratura. Después se procederá a hacer el infra-muestreo para recuperar los valores reales de las coordenadas y se aplicara el desmapeo obteniendo así los símbolos y con ello los streams de datos originalmente enviados.

Esta modulación nos permite asignar por separado tanto el orden de la modulación (constelación M-aria) como la potencia, así se puede ajustar la transmisión a la SNR de cada banda de frecuencia, en el punto de el medio, la fibra óptica, veremos como la atenuación que produce la fibra a la señal

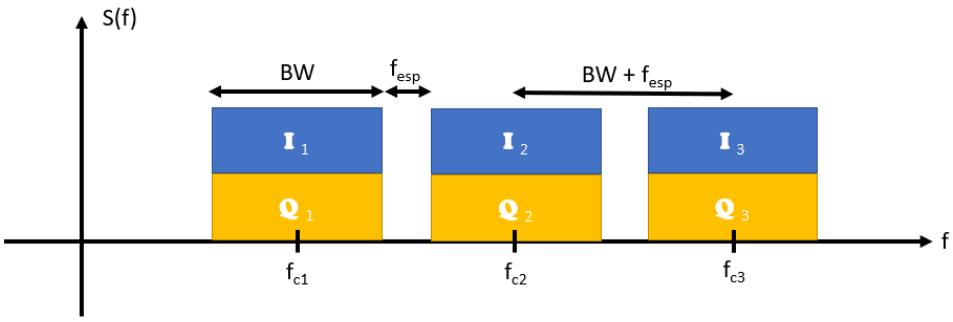

Figura 2.13: Módulo cuadrado de los filtros CAP (cada banda es un color)

transmitida aumenta con la frecuencia por lo que podremos asignar más potencia y/o modulaciones de orden inferior a las bandas de alta frecuencia.

2.6. Modulaciones SSB

Como veremos a continuación, la señal CAP que hemos generado modulará a un láser en amplitud. Esta modulación se traduce a mover el espectro de la señal CAP en banda base a una frecuencia superior, generando así una señal de doble banda tal y como vemos en la figura 2.14.

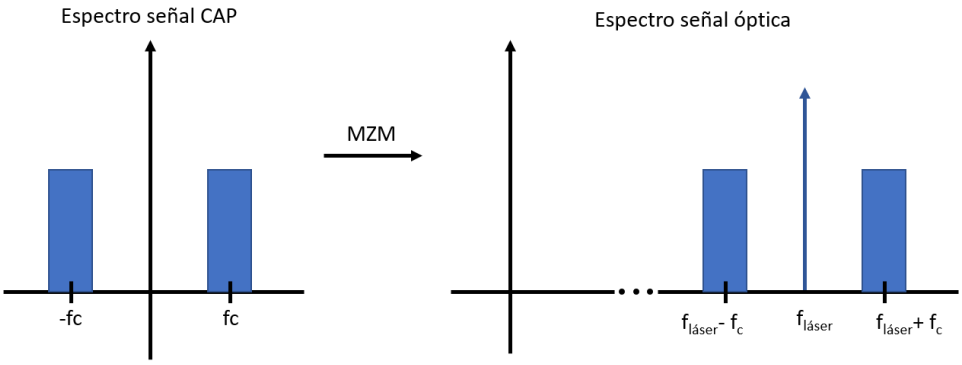

Figura 2.14: Espectro de la señal CAP y de la señal óptica modulada

De esta manera tenemos una banda redundante, así como una delta que no aporta información. Podríamos eliminar una u otra banda para eliminar esa redundancia mediante distintos métodos. De esta forma obtendríamos una mayor eficiencia en cuanto a la potencia que usamos, además podría solventar problemas futuros que mencionaremos tras ver los resultados de la simulaciones. Este concepto resulta importante para futuros apartados.

2.7. El canal óptico

En este apartado se hablará del medio por donde se va a transmitir la señal S(t) (figura 2.1). Nos centraremos en ver como es el sistema de transmisión y recepción y los efectos que produce la fibra sobre la señal transmitida.

El esquema final que tenemos podemos verlo en la figura 2.15, a continuación serán explicados los bloques añadidos.

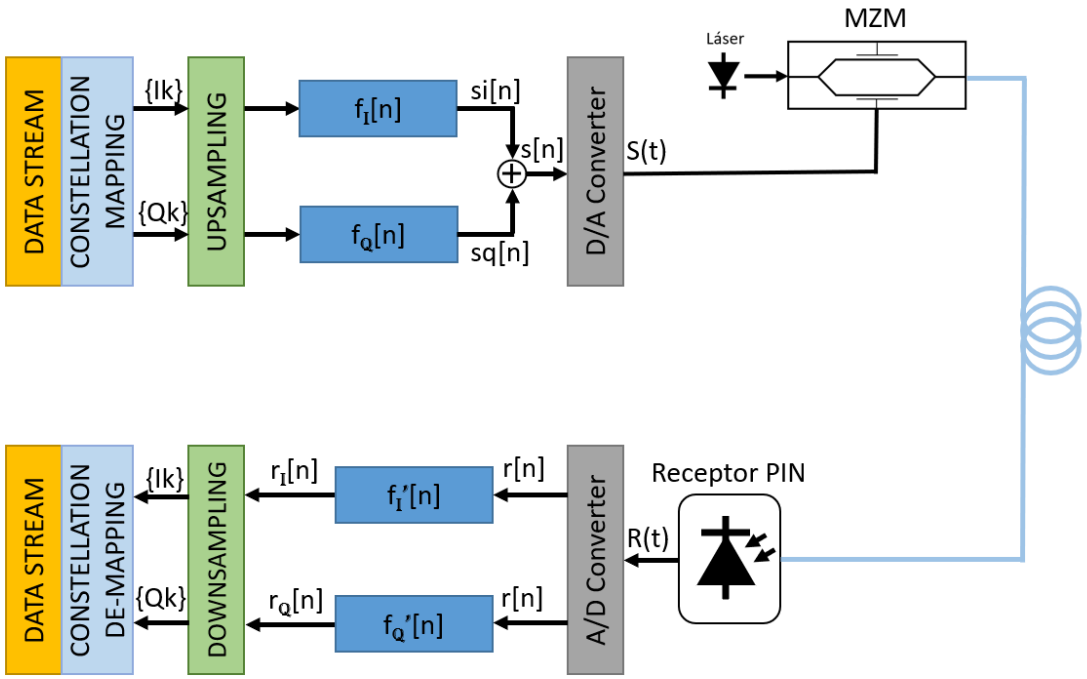

Figura 2.15: Esquema de la transmisión CAP con la parte óptica incluida

2.7.1. Transmisión

El primer paso es introducir la señal en la fibra óptica, para ello vamos a usar un modulador electro-óptico (MOE). Un modulador electro-óptico es un dispositivo en el cual una señal eléctrica se utiliza para modular un láser, esta modulación puede ser en fase, frecuencia, amplitud o polarización. En este trabajo se utiliza un modulador Mach Zehnder, el cual es un MOE que modula el haz de luz en amplitud. El principio que sigue un MZM es simple, un haz de luz se divide en dos caminos con un divisor para más tarde recombinarse. En función de la fase relativa adquirida por los dos caminos el haz saldrá con una potencia entre el 0 % y el 100 % [14].

En la figura 2.16 se puede ver el esquema más típico de un modulador MZM, tenemos dos entradas: Un láser y una señal eléctrica V_{in}. La salida E_{TX} es el campo electromagnético generado al modular el láser en amplitud con la señal eléctrica.

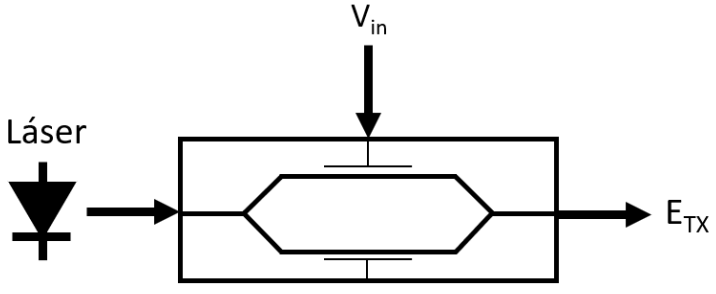

Figura 2.16: Esquema de un modulador Mach Zehnder

La modulación en amplitud no sigue un comportamiento lineal, para ver como funciona debemos ver la forma de la curva propia del MZM (figura 2.17). Esta curva representa la potencia óptica que genera el MZM en función del voltaje de la señal eléctrica de entrada, para entenderla debemos explicar dos parámetros: V_{BIAS} y V_π. V_{BIAS} indica el voltaje de referencia, esto quiere decir que si V_{BIAS} es 0.5 V, cuando el voltaje de entrada sea nulo, realmente se situará en 0.5 V en la curva (V_{BIAS} es sumado al voltaje de entrada). V_π Indica en qué punto se alcanza la potencia óptica máxima a la salida.

Con un V_{BIAS} de 0.5 y un V_π de 1, para aprovechar toda la potencia óptica disponible nuestra señal de entrada debería variar entre -0.5 y 0.5. Pero esto no es siempre lo más recomendable, tal y como vemos en la figura 2.17 las partes de mayor y menor son las que menos lineales son, esto es porque la curva típica de un Mach Zehnder tiene forma de coseno cuadrado. Dependiendo del escenario será mejor usar menos potencia pero ceñirse a la zona lineal o usar toda la potencia con la desventaja de esa no linealidad. En la simulación analizaremos en qué tipo de escenarios es mejor usar más o menos parte de la curva haciendo uso de un parámetro que hemos llamado "m" que indicará la porción de la curva a usar.

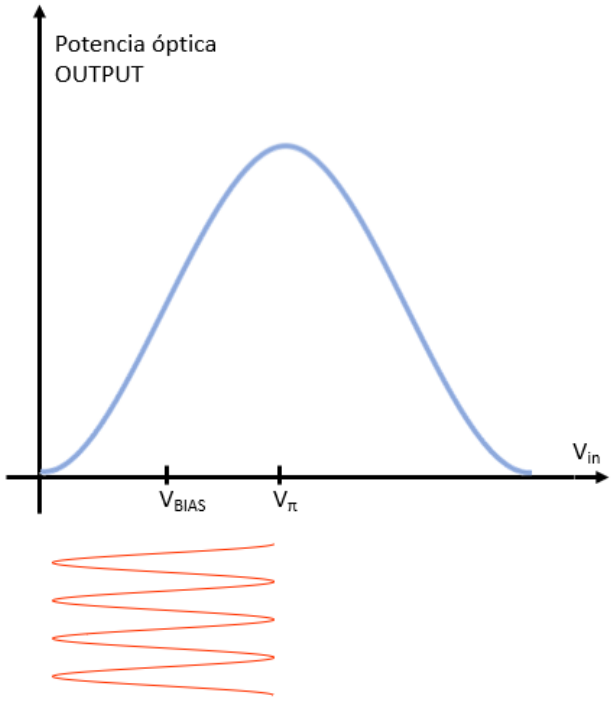

Figura 2.17: Curva de potencia óptica en función de V_{in} de un MZM

Tras introducir la señal en el MZM ya habremos generado el campo electromagnético modulado y podremos transmitirlo por la fibra.

2.7.2. El medio: la fibra óptica

La información se encuentra ahora en el campo electromagnético que se está transmitiendo por la fibra óptica. Las fibras ópticas tienen dos principales limitaciones: la atenuación y la dispersión. La atenuación es la pérdida de potencia óptica en la propagación, depende del material de la fibra y otros factores, los que más influyen son: la absorción, dada cuando la energía luminosa se transforma en otro tipo de energía, el scattering, cuando la energía luminosa guiada pasa a ser radiada debido a la propagación en un medio distinto del vacío, y, la radiación, que ocurre cuando la energía luminosa guiada pasa a ser radiada debido a curvaturas en la fibra o imperfecciones

periódicas [15]. A causa de este fenómeno la potencia óptica que llega al receptor es menor y los distintos ruidos afectan en mayor medida a la señal. La atenuación es distinta en función de la longitud de onda, nosotros usaremos la llamada tercera ventana, situada en torno a 1550 nm en la cual la atenuación es menor.

Por otra parte tenemos la dispersión, la dispersión es la diferencia en la velocidad de propagación de distintas componentes de la potencia óptica [15], ya sean modos, longitudes de onda, polarizaciones, etc. Este fenómeno se traduce en un aumento de la anchura de los pulsos transmitidos contribuyendo a la aparición de la ya mencionada ISI además de la propia distorsión. En la figura 2.18 vemos los dos efectos mencionados.

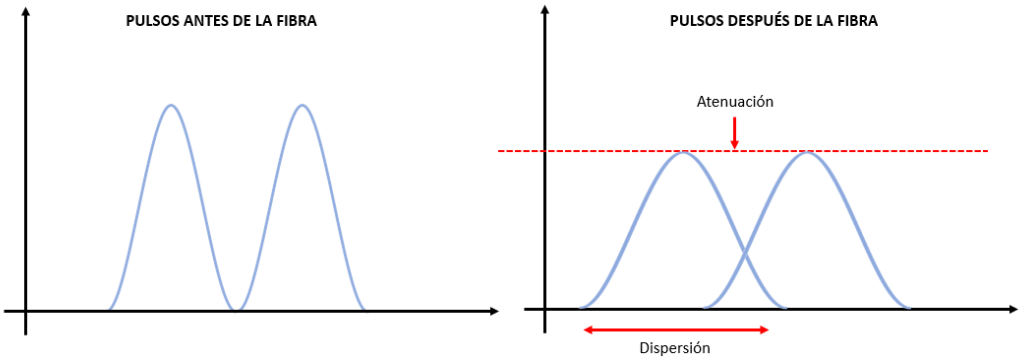

Figura 2.18: Pulsos antes y después de pasar por una fibra óptica

La dispersión se mide en $ps/(km{\cdot}nm)$, lo cual representa que un pulso de anchura espectral 1 nm se ensanchará 1 ps por cada kilómetro recorrido. Del mismo modo que sucede con la atenuación, la dispersión es diferente para cada longitud de onda, en la figura 2.19 [15].

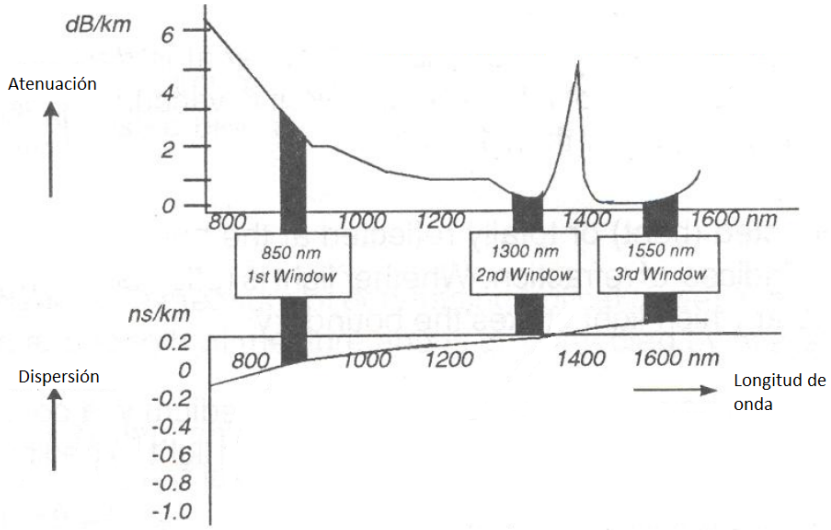

Figura 2.19: Dispersión y atenuación en función de la longitud de onda

Vemos que dependiendo de la ventana (window) que usemos tanto la atenuación como la dispersión toman valores diferentes. Esto afectará a una señal viajando por la fibra atenuando más o menos ciertas frecuencias y haciendo que se propaguen más o menos rápido por la fibra. Las consecuencias a simple vista son que ciertas bandas de frecuencias sufrirán más atenuación que otras y precisarán de mayor potencia. Existen más consecuencias provenientes de la variación en la dispersión que se verán y serán analizadas en el apartado de tests realizados.

Es posible compensar ambos efectos de diferentes formas. Para lidiar con la atenuación es posible hacer uso de amplificadores ópticos, los cuales pueden situarse al principio del sistema como preamplificadores o al final del sistema como un amplificador de línea. Los amplificadores más típicos son los denominados EDFAs, los cuales presentan muy buenas especificaciones en cuando a ganancia, potencia de salida y ruido [15].

Para compensar la distorsión usamos diferentes tipos de fibra óptica, existe la posibilidad de usar para la transmisión una fibra de baja dispersión lo cual acabaría con esta en mayor medida pero no es lo más recomendable como ya se comentará en el apartado de los tests. Otra opción es hacer uso de una fibra de compensación de distorsión al final del sistema, la cual tiene una dispersión negativa que, dado a la propiedad aditiva de la dispersión, contrarrestaría la dispersión acumulada.

2.7.3. Recepción

Para recibir la señal usamos un fotodetector, un fotodetector es el elemento final de un sistema de comunicaciones ópticas en el que la luz recibida se transforma de nuevo en corriente eléctrica. Generalmente esta corriente eléctrica es amplificada, filtrada y detectada para proceder a su demodulación.

En un receptor típico tenemos múltiples características a tener en cuenta, tales como la eficiencia cuántica, el coeficiente de absorción y la responsividad. Estos factores afectarán al desempeño final de un receptor. Existen distintos tipos de fotodetectores, en concreto el fotodetector usado para este proyecto es un diodo PIN, con este tipo de receptores se obtienen mayores eficiencias [15].

Lo que más nos interesa en este proyecto sobre el fotodetector es el ruido que este va a introducir en la señal. Para un diodo PIN existen únicamente dos fuente de ruido, el ruido cuántico y el ruido térmico. Si la potencia óptica en el fotodetector no es lo suficientemente grande estos ruidos podrían llegar a enmascarar la señal.

Capítulo 3

Implementación software de la modulación CAP

Para implementar la teoría relatada en los capítulos anteriores se han recorrido dos fases. Una primera fase de implementación del software donde se generan las diferentes señales a tratar y se simula todo el proceso usando un sistema ideal. Y una fase posterior en la cual se simularán los dispositivos reales tales como los láseres, moduladores y demoduladores ópticos y la propia fibra óptica.

En este capítulo estudiaremos la implementación de la modulación CAP ideal y sus derivadas y comprobaremos que la teoría funciona en la simulación tal y como debería. Como herramienta de software se ha elegido MATLAB dado que, entre otras cosas, es muy versátil para trabajar con vectores y matrices (formato que tendrán las señales a procesar), además, es posible hacer uso de librerías especialmente diseñadas para el procesado de señal.

Al igual que en el marco teórico, se van a recorrer los distintos bloques de la figura 2.1, explicando ahora como los hemos implementado en MATLAB. Se explicará primero la implementación de la modulación CAP en su forma más simple y se irán añadiendo los distintos conceptos explicados en el marco teórico.

Para entender el proceso se va a mostrar el código usado para realizar una modulación CAP con $M = 16$, los valores de las tasas de símbolo y frecuencias de muestreo serán especificados en el código.

3.1. Implementación del transmisor

3.1.1. Flujo de bits (Data Stream)

Queremos enviar información y comprobar que la recibimos correctamente, para ello generaremos una secuencia de bits pseudo-aleatoria (PRBS) lo suficientemente larga para poder extraer conclusiones generales sobre el desempeño del sistema (tales como el BER). Usaremos la función "randi" de MATLAB para su generación tal y como vemos en la siguiente muestra de código.

Código 3.1: Generación flujo de bits

```
1 n = 10*(2^12);   % Numero de bits
2 dataIn = randi([0 1],1,n); % Data Stream
```

3.1.2. Mapeo de bits

Una vez hayamos obtenido una secuencia de bits, el siguiente paso será realizar el mapeo M-QAM. Por razones de complejidad, se han implementado los códigos para modulaciones QAM M-arias cuadradas, i.e, con valores de M que cumplan $M = 2^n$ siendo n un número entero par. De esta forma la constelación tendrá siempre una distribución cuadrada y no en forma de cruz (véase que, en la figura 3.1, solo valdrían la 4-QAM y la 16-QAM).

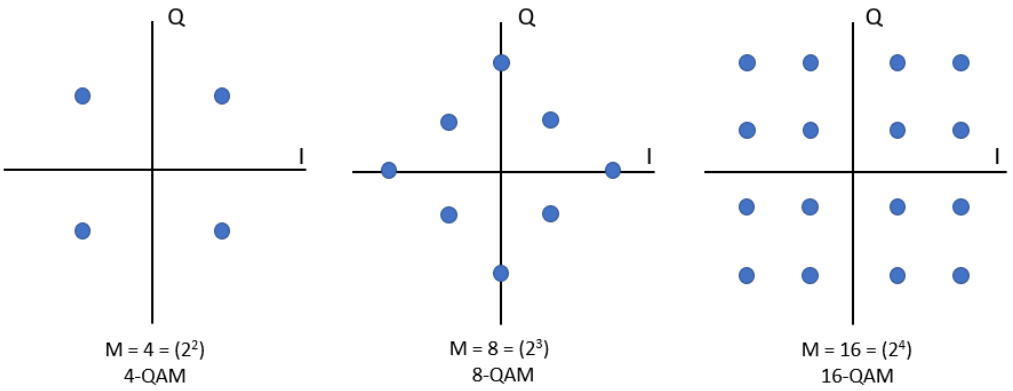

Figura 3.1: Modulaciones M-QAM

Para mapear los bits en la constelación y obtener los símbolos y las coordenadas respectivas se ha implementado la función "qammodulation" (código y explicación en Anexo A), la cual, dado un vector de bits y un orden M de modulación QAM, devuelve un vector con los símbolos (números enteros entre 0 y M-1) y otro vector complejo con las coordenadas, I en la parte real y Q en la parte imaginaria. En el siguiente código se hace uso de esta función. Además, calculamos la potencia de la señal de coordenadas compleja "dataMod" para ajustar la potencia obtenida en recepción más adelante.

Código 3.2: Mapeo de los bits

```
1 M = 16;  % Orden de la modulacion
2 [dataMod, dataSymIn] = qammodulation(dataIn,M);  % Datos a QAM
3
4  % Calculamos potencia para futuro uso.
5 potIn = mean(abs(dataMod).^2);
```

En la figura 3.2 se puede ver como se obtiene esta transformación de bits a símbolos. En este caso, para $M = 16$, se obtiene un símbolo por cada 4 bits ($\log_2 M$).

3.1.3. Sobre-muestreo

Una vez hayamos conseguido los datos mapeados con su parte en fase y parte en cuadratura, el siguiente paso es un sobre-muestreo (upsampling) tal y como se ha visto en el marco teórico. El sobre-muestreo lo hacemos haciendo uso de la función de MATLAB "upsample" que tiene como argumentos la señal a sobre-muestrear y el número de muestras por símbolo. Para saber el número

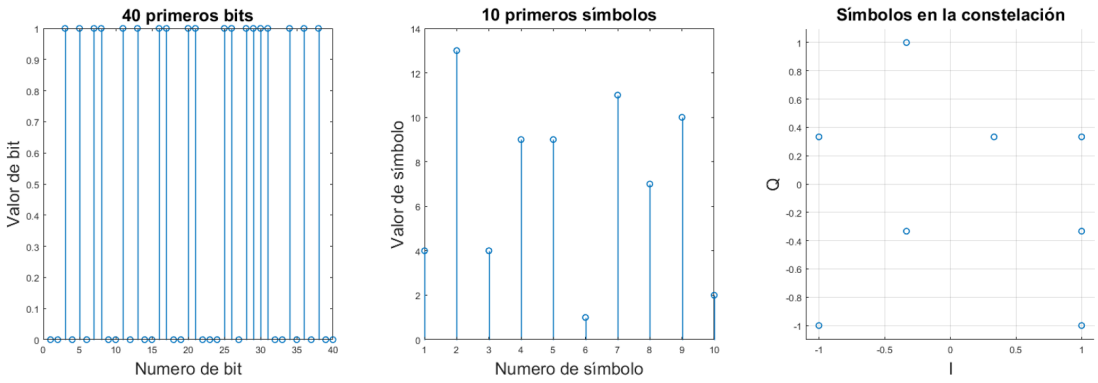

Figura 3.2: Transformación de bits a símbolos y coordenadas

de muestras por símbolo necesitamos saber la tasa de símbolos por segundo de la transmisión y la frecuencia de muestreo. En el siguiente código (3.3) vemos como se implementa.

Código 3.3: Código para el sobre-muestreo

```
1 Rs = 0.5e9;   % Tasa de simbolo [sym/s] [0.5e9]
2 fs = 10e9;    % Frecuencia de muestreo de la salida analogica [10e9]
3 Nss = fs/Rs;  % Muestras por simbolo (Number of samples per symbol)
4
5 dataMod_up = upsample(dataMod,Nss);  % Upsampling
```

En la figura 3.3 se aprecia como se añaden esas muestras de valor nulo entre las muestras originales. En este caso se añaden 20 ceros entre símbolos dado que la frecuencia de muestreo es 10 GHz y la tasa de símbolo es 500 MHz.

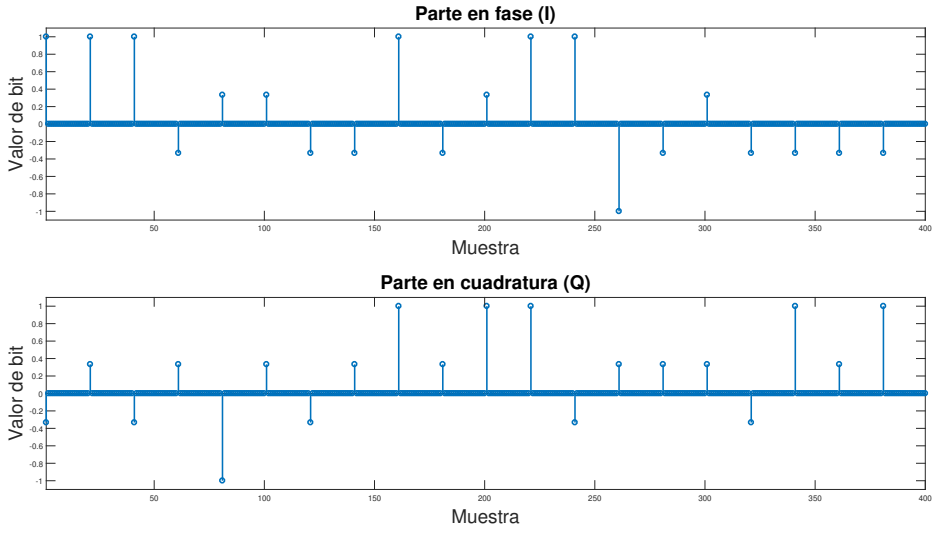

Figura 3.3: Parte en fase y cuadratura tras el sobre-muestreo

3.1.4. Filtros CAP

Tras haber hecho el sobre-muestreo se procede a filtrar las coordenadas. Los filtros los hemos implementado basándonos en la fórmula 2.3 la cual introducimos en la Función A.4 “CAPfilter”

que podemos ver en el Anexo A. Esta función nos devolverá el par de filtros ortogonales CAP dados los parámetros necesarios para su generación, dichos parámetros son especificados en el código y su significado ha sido clarificado en el marco teórico. En el siguiente código vemos cómo los filtros son generados y filtrados con cada componente (I y Q) sumándose después las señales resultantes de dichos filtrados, obteniendo así la señal a enviar.

Código 3.4: Generación y uso de los filtros CAP

```
a = 0.1;     % Roll-off factor (alfa) [0.1]
span = 20;   % Numero de periodos Ts a cada lado del filtro [20]
Ts = 1/Rs;   % Periodo de simbolo
fc = 2e9;    % Frecuencia portadora CAP [2e9]

 % Calculamos filtros ortogonales
[fI, fQ] = CAPfilter( fc , a , Ts , fs , span );

 % Filtramos las coordenadas con los filtros CAP
xi = conv(real(dataMod_up),fI);
xq = conv(imag(dataMod_up),fQ);

 % Sumamos las dos componentes
x = xi + xq;
```

En la figura 3.4 están representados los espectros de los filtros y de la señal a enviar que hemos generado. Los filtros actúan como filtros paso banda haciendo que el espectro de la señal resulte mínimo, algo que ya se había mencionado en el marco teórico.

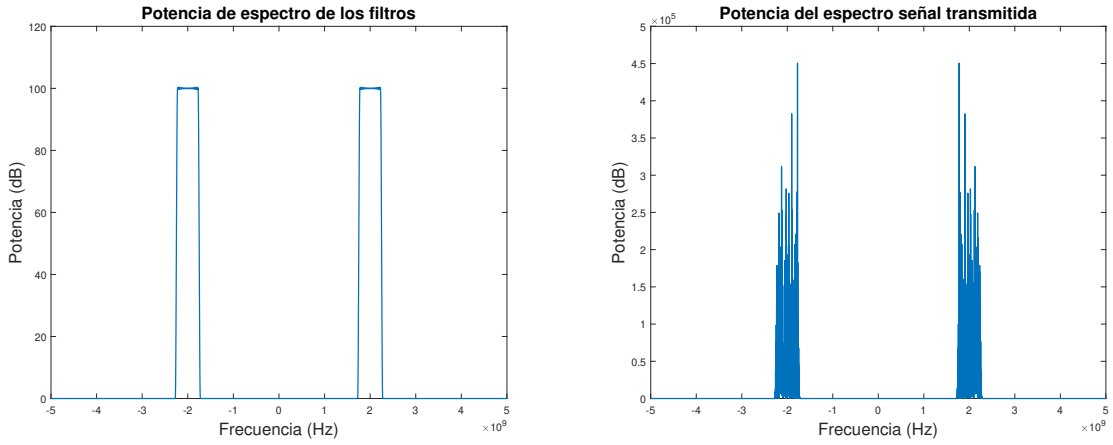

Figura 3.4: Espectro de los filtros y de la señal a enviar

Hay que mencionar que tras el filtrado la longitud de la señal no es la misma, dado que en la convolución con los filtros se ha añadido una parte transitoria al principio y final de la señal. Esto se traduce en añadir un número de muestras a la señal igual a la longitud del filtro menos uno. Podríamos recortar la señal a la parte central y quedarnos con el mismo número de muestras, pero así perderíamos información necesaria para demodular correctamente algunos de los símbolos. Es por esto que enviamos todas las muestras generadas.

3.2. Implementación del receptor

A la hora de recibir la señal hay que tener en cuenta que se va a muestrear con una frecuencia de muestreo distinta a la usada en el transmisor. En el transmisor la frecuencia de muestreo es la de un AWG que genera la señal con una frecuencia de muestreo fs, esta señal es enviada analógicamente y recogida por un osciloscopio que la muestreará a su vez con una frecuencia fsOsc. Es por esto que en la simulación de MATLAB remuestreamos la señal con esta nueva fsOsc usando la función "interp" simulando así este cambio de frecuencia de muestreo en recepción. Esta función muestrea la señal con una frecuencia de muestreo NssOsc veces más grande que la original. Las operaciones tales como la generación de los filtros adaptados se harán con esta nueva frecuencia de muestreo.

Código 3.5: Remuestreo de la señal recibida

```
% Nueva fs (receptor)
NssOsc = fsOsc/fs;
% Remuestreamos la senal recibida con fsOsc
xr = interp(x,NssOsc);
```

3.2.1. Filtros adaptados y corrección de la señal

Para recuperar las coordenadas hay que filtrar la señal recibida con los filtros adaptados. Estos filtros han sido implementados generando los mismos filtros que en transmisión pero dándoles la vuelta en tiempo (reflejándolos) tal y como se explica en las fórmulas 2.10 y 2.11. Tras ello, se filtra la seña recibida con cada uno de estos filtros obteniendo la parte en fase y cuadratura y se suman como parte real e imaginaria tal y como se ve en el siguiente código. Además, ahora que se ha filtrado la señal ya podemos "recortar" la parte sobrante que genera el filtrado ya que no aporta información útil.

Código 3.6: Generación y filtrado con filtros adaptados

```
[f1r, f2r] = CAPfilter( fc , a , Ts , fsOsc , span );

yi = conv(xr,fliplr(f1r));
yq = conv(xr,fliplr(f2r));

y = yi + 1j*yq;

% Nos quedamos con la parte central
y = y(length(f1r)-(fsOsc/fs)+1:end-length(f1r));
```

Ahora nos situamos en un caso ideal, pero en la práctica habría un retardo en la señal al enviarla por el canal. Vamos a tener que calcular ese retardo para poder corregirlo. Para esto se ha implementado la función A.5 "diagramaojo"(ver capítulo A, la cual devuelve el retardo de la señal y representa el diagrama de ojo. El diagrama de ojo no es más que la superposición de todos periodos de símbolos recibidos en una gráfica, véase la figura 3.5. Dado que en los instantes de muestreo la amplitud ha de ser siempre la misma se puede ver donde se encuentra este instante, será donde la señal esté más comprimida. Es así como deducimos el instante óptimo de muestreo y, con él, el retardo para poder corregirlo:

Código 3.7: Generació de diagrama de ojo y corrección del retardo

```
retardo = diagramaojo(real(y(1:10000)),fsOsc/Rs);
y = y(retardo:end);
```

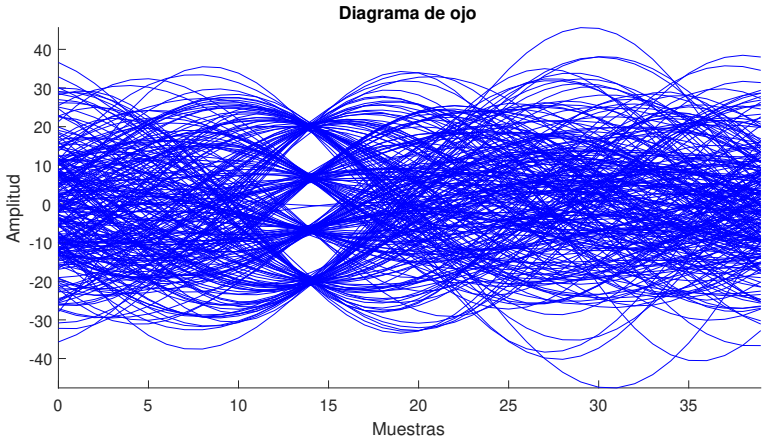

Figura 3.5: Diagrama de ojo de la señal recibida

Tras este proceso, las señales ya habrán sido filtradas con los filtros adaptados y con ello tendremos las deseadas coordenadas en fase y en cuadratura.

3.2.2. infra-muestreo y normalizado

Hemos dicho que en este punto ya tenemos las coordenadas I y Q, pero esto no es realmente cierto. Para obtener las coordenadas debemos aplicar un infra-muestreo a la señal tal y como hemos especificado en el marco teórico para quedarnos solamente con las muestras de los instantes de muestreo dejando atrás la parte sobrante de los símbolos que no aportan más información (ver figura 2.8). Es por esto que usamos la función "downsample"de MATLAB para quedarnos con las muestras óptimas, de esta forma ya tendríamos las coordenadas. Pero no hay que olvidarse de que la señal ha pasado por una serie de filtrados y por el canal lo cual seguramente ha alterado su potencia y por ende la amplitud de los símbolos. Para solucionar esto vamos a normalizar la potencia con la potencia que hemos calculado en emisión con la función de MATLAB “modnorm” la cual, dada una señal de entrada y una potencia objetivo, devuelve un factor por el que hay que multiplicar a señal para obtener esa potencia. Todo este proceso se resume en las siguientes lineas de código.

Código 3.8: Infra-muestreo y normalizado

```
% Infra-muestreo
yd = downsample(y,fsOsc/Rs);

% Si ponemos demasiadas muestras de retardo, se consideraran como
% informacion y habra bits de mas al principio, eliminamos esos bits.
yd = yd(end-length(dataMod)+1:end);

% Ajustamos potencia teniendo cuenta la potencia enviada
P = modnorm(yd,'AVPOW',potIn);
yn = yd*P;
```

3.2.3. Demodulación

En este punto ya se deberían tener las coordenadas con sus correctas amplitudes. El siguiente y último paso se trata de la demodulación, proceso que desmapea las coordenadas y las traduce a símbolos y a bits. Para demodular hemos creado la función A.3 "qamdemodulation"(Anexo A) que toma como parámetros de entrada el vector de coordenadas obtenidas y el orden de la modulación (M = 16 en este caso) y devuelve los símbolos además de la cadena de bits correspondientes a esos símbolos.

Código 3.9: Código para demodular las coordenadas

```
1 [dataOut, dataSymOut] = qamdemodulation(yn,M);
```

Este es el final del sistema, en este momento ya se tienen la cadena de bits resultante de la transmisión. En un caso real puede que la cadena de bits demodulada no sea exactamente igual a la enviada debido a posibles ruidos e interferencias, es por esto que podemos medir una serie de prestaciones del sistema con referencia a la calidad de la transmisión.

3.2.4. Cálculo de prestaciones

Como ya se ha comentado, una vez hemos obtenido la señal demodulada podemos comprobar diferentes medidas de calidad en función de la veracidad de los datos recibidos. En concreto, vamos a calcular la tasa de error en el bit (BER), la tasa de error en el símbolo (SER) y la magnitud del vector de error (EVM) la cual indica el porcentaje de desplazamiento de los símbolos recibidos respecto a su posición ideal en la constelación. Estas tres funciones las hemos implementado en el Anexo A (funciones A.6, A.7 y A.8 respectivamente) y las utilizamos de la siguiente forma:

Código 3.10: Código para calcular prestaciones

```
1 [ber, NbitErr] = BER(dataIn, dataOut, M);
2 [ser, NsymErr] = SER(dataSymIn, dataSymOut);
3 EVM = evm(dataMod,yn);
```

Además de estas medidas numéricas podemos ver una representación de la constelación recibida. Para que se vea como afecta el ruido a la constelación se va a añadir ruido blanco en la recepción (figura 3.6).

En la figura 3.6, ninguno de los símbolos tiene el suficiente ruido como para moverse a una posición incorrecta así que el SER y BER serán cero, siendo que está claro que existe una componente de ruido. Es por esto que medidas como el EVM o representaciones de la constelación aportan información relevante en muchas ocasiones.

En el Anexo A, podemos ver el código A.9 en el cual están recogidos todos los fragmentos de código mostrados abarcando una transmisión CAP completa, donde se simula un canal con un ruido blanco y un retardo.

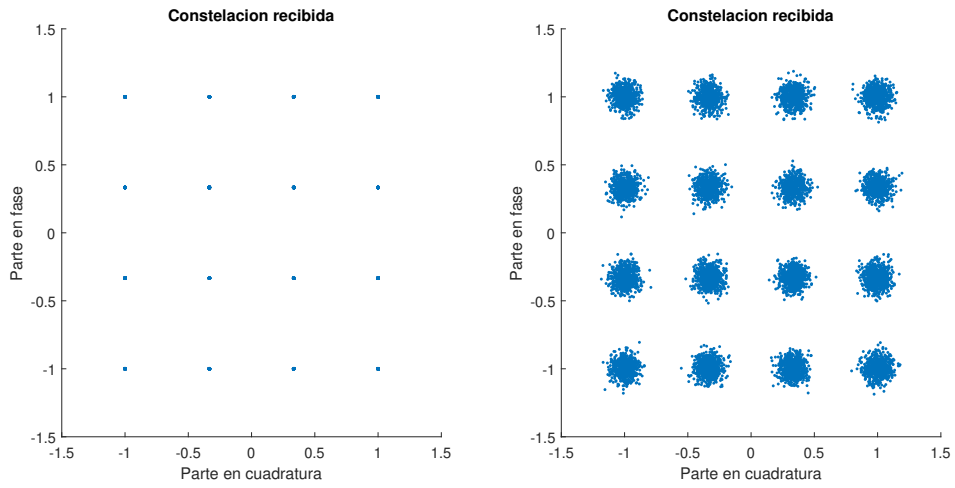

Figura 3.6: Constelación recibida sin ruido y con un ruido blanco

3.3. Implementación de un sistema multiCAP

Como ya se ha aclarado, la modulación multiCAP no es más que el uso de varias modulaciones CAP con distintas portadores sumadas de forma que los espectros sean ortogonales entre sí (en frecuencia). Por lo tanto una vez se tiene el código para modulación CAP es sencillo extrapolarlo para multiCAP.

El código va a ser prácticamente el mismo que en CAP salvo por el hecho de que hay que generar N modulaciones que serán sumadas y por tanto N pares de filtros y filtros adaptados. Dado que es redundante escribir todos los bloques del código multiCAP por su parecido al CAP no se van a detallar de nuevo, el código se puede ver en el capítulo A en el fragmento de código A.10.

La función que sí que vamos a comentar de multiCAP es "multi_CAPfilter" función que se ha implementado con propósito de generar los pares de filtros CAP para cada canal. Tiene como parámetros de entrada los mismos parámetros que la función "CAPfilter" pero añade alguno más: la frecuencia de la primera portadora ($fc1$), la distancia de seguridad en frecuencia entre canales ($fesp$) y el número de canales (N). Así, si el ancho de banda de cada canal es BW y queremos una separación extra en frecuencia de f_{esp} la distancia entre las frecuencias de portadora deberá ser $BW + f_{esp}$ tal y como se ve en la figura 2.13. Dado que conocemos BW ($BW = \frac{1+\beta}{T_s}$), se puede implementarlo en la función de la siguiente forma haciendo uso de la función "CAPfilter":

Código 3.11: Función multi_CAPfiler

```
function [ f ] = multi_CAPfilter( fc1,  fesp, N, B , Ts , fs , span )
% multi_CAPfilter: Funcion para generar pares de filtros CAP ortogonales,
% un par para cada banda CAP. Los pares de filtros se devuelven en f usando
% una matriz de 3 dimensiones. N filas, cada fila contiene dos dimensiones
% que contienen los respectivos filtros CAP.

% a: Roll off/ Excess bandwidth factor
% Ts: Tiempo de simbolo
% fs: frecuencia de muestreo de la salida (≥ 1/tau)
% span: Tamano del filtro. Numero de periodos Ts a cada lado del pico
% f1: frecuencia primera portadora
% fesp: espaciado entre bandas
% N: Numero de bandas
% Ancho de banda BW = (1+a)/Ts;
% Elegir fs > 2*BW
```

```
16
17 BW = (1+B)/Ts; % Ancho de banda cada banda
18 fc = zeros(1,N);
19 fc(1) = fc1; % Primera portadora
20 for i = 2:N % El resto de portadoras
21     fc(i) = fc(i-1)+BW+fesp;
22 end
23 muestras_filtro = 2*span*Ts*fs-1;
24 f = zeros(N,muestras_filtro,2);
25 for i = 1:N
26     [f(i,:,1), f(i,:,2)] = CAPfilter( fc(i) , B , Ts , fs , span );
27 end
28 end
```

Los filtros los almacenamos en una matriz de tres dimensiones, conteniendo en cada fila los pares de filtros para cada canal (matriz de NxMUESTRASx2, siendo MUESTRAS el número de muestras de cada filtro).

El resto de funciones no han de ser modificadas y las podemos usar mediante bucles "for" de N iteraciones. Con el sistema completo obtenemos el mismo tipo de prestaciones pero ahora para cada canal. Si simulamos de nuevo ahora un sistema multiCAP de N = 6 canales con ruido y retardo obtenemos las siguientes constelaciones.

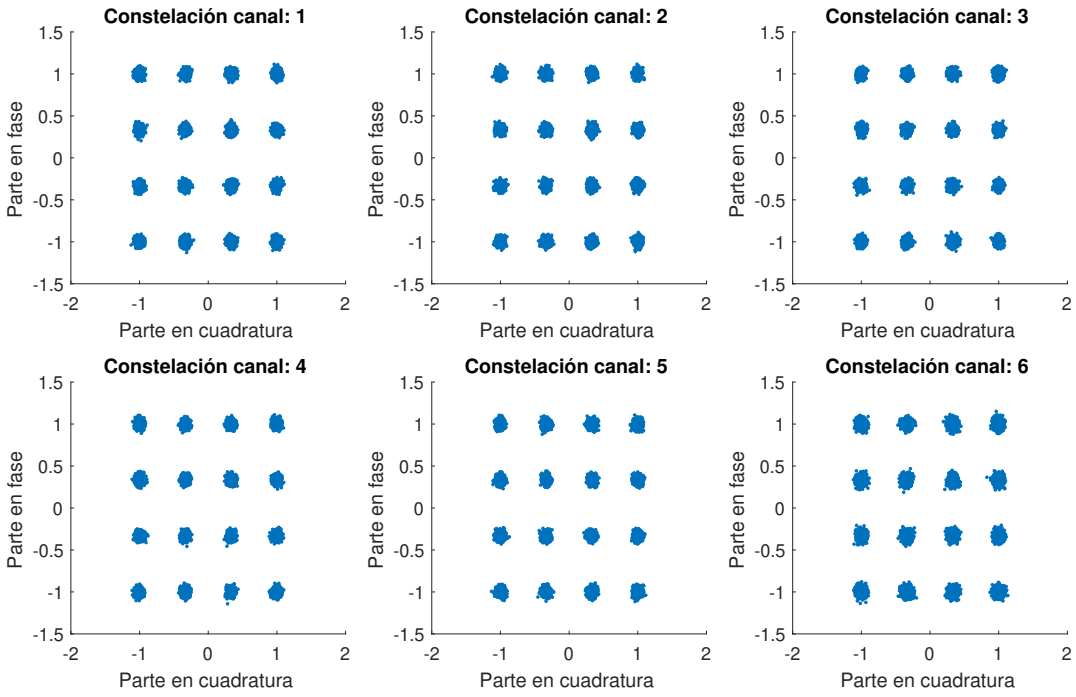

Figura 3.7: Constelación recibida para cada canal tras añadir ruido blanco

Los valores de BER y SER para cada canal son 0 con un ruido no demasiado grande, así que ya se ha comprobado el correcto funcionamiento de ambos sistemas CAP y multiCAP, el siguiente paso es simular un entorno real para poder obtener verdaderas conclusiones de las prestaciones obtenidas (BER, SER, EVM y las constelaciones).

Capítulo 4

Simulación del entorno real: OPTILUX

Para simular el entorno real sobre el que se trabajaría hemos hecho uso de un toolbox de MATLAB para comunicaciones ópticas llamado OPTILUX, creado por el profesor Paolo Serena [16]. Este toolbox provee tecnicas avanzadas para diseñar, simular y analizar sistemas de comunicaciones ópticas. Está implementado para MATLAB/OCTAVE e incluye distintas rutinas para describir los principales aspectos de los sistemas ópticos:

- Generación de patrones de bits
- Formatos de modulación multi-nivel
- Fibras ópticas en el régimen no lineal
- Métodos Karhunen-Loéve para análisis de rendimiento
- Estimación Monte Carlo
- Optimización del rendimiento
- Efectos de polarización

OPTILUX trabaja con modulaciones OOK, DPSK y DQPSK. Para poder hacer uso de la modulación CAP que se ha implementado se ha tenido que adaptar el código al formato de OPTILUX tras un estudio de éste. Además, dado que este toolbox no ha sido diseñado para modulaciones fuera de las mencionadas, no podemos hacer uso de las funciones de análisis de rendimiento. Por lo tanto, el análisis de los resultados lo diseñaremos nosotros sobre la plataforma. De este modo usaremos OPTILUX exclusivamente para simular los diferentes efectos dados en la transmisión, propagación y recepción en un sistema óptico.

4.1. Uso de OPTILUX

En esta sección vamos a comentar las herramientas de OPTILUX que usamos, las cuales engloban la transmisión donde se inyecta la señal en una fibra, la propagación por la fibra y la recepción de señales.

4.1.1. Transmisión: Modulador Mach-Zehnder

Teniendo ya la señal CAP a transmitir (señal S(t) de la figura 2.1) ésta se va a introducir en un modulador Mach-Zehnder (MZM) modulando así un láser en amplitud.

El láser que se va a modular lo generamos con la función "lasersource" de OPTILUX, especificando únicamente la longitud de onda y la potencia en nuestro caso. La función que hace de MZM se llama "mz_modulator" y tiene como parámetros de entrada la señal del láser, la señal CAP y una estructura para modificar parámetros como V_{bias}, V_π y el extinction ratio. Esta función devuelve el campo que generaría el MZM con esos parámetros. Esta función estaba diseñada de forma que los valores introducidos estaban normalizados a V_π, haciendo confuso su uso, por ello se ha modificado ésta haciendo que los valores introducidos sean los reales (ver función A.12).

OPTILUX trabaja con una variable global donde almacena las distintas variables y sobre la cual se aplican las diferentes funciones de la fibra, la transmisión y la recepción. Por ello lo primero será generar esta variable global. Para ello se hace uso de la función "reset_all" en la cual especificamos el número de símbolos que vamos a transmitir, el número de muestras por símbolo y el número de canales WDM, aunque éste será siempre uno en nuestro caso. Tras ello, tenemos que introducir el campo en esa variable global, lo haremos con la función "create_field" indicando como parámetro el campo obtenido con el MZM, además se añaden dos parámetros más para indicar que la potencia indicada es la potencia media (average) y que queremos usar solo un canal (unique).

En el siguiente fragmento de código vemos cómo se implementaría la transmisión en OPTILUX (la señal "X" se corresponde con S(t)).

Código 4.1: Código transmisor OPTILUX

```
% Creamos la variable global
reset_all(Nsymb,Nt,Nch);
% Generamos el laser de potencia media Pavg y longitud de onda lam
E    = lasersource(Pavg, lam);
% Modulamos el laser E con la senal X
Eopt = mz_modulator_mod(E,X,struct('exratio',exratio,'amplitude',1,'bias',0.5));
% Introducimos el campo electromagnetico obtenido en el canal
create_field('unique',Eopt,[],struct('power','average'));
```

4.1.2. Propagación: Fibra

Para simular la propagación del campo electromagnético por la fibra necesitamos hacer uso de una sola función, pero antes debemos especificar los parámetros de la fibra los cuales serán almacenados en una variable struct llamada tx (Tabla 4.1).

Una vez especificados los parámetros de la fibra simularemos la propagación del campo creado por la fibra mediante la función "fiber". En esta función indicamos los parámetros de la fibra (tx) y una serie de FLAGS para indicar si queremos que se simulen o no distintos efectos: dipersión de velocidad grupal, dispersión de modo de polarización, automodulación de fase y mezcla de cuatro ondas. No se va a entrar en detalle para ver lo que implica cada uno de ellos, simplemente diremos que la parte no lineal no se tiene especialmente en consideración. En el código 4.2 vemos cómo usamos esta función.

Parámetro	Descripción	Unidades
tx.length	Longitud de la fibra	m
tx.alphadB	Atenuación	dB/Km
tx.aeff	Área efectiva fibra	μm^2
tx.n2	Índice de no linealidad	M^2/w
tx.lambda	Longitud de onda	nm
tx.disp	Dispersión	ps/nm/km
tx.slope	Pendiente de dispersión	$ps/nm^2/km$
tx.dphimax	Rotación en fase no lineal máxima por paso	rad

Tabla 4.1: Parámetros de la fibra óptica para OPTILUX

Código 4.2: Código propagación OPTILUX

```
1 fiber(tx,'g-sx')
```

4.1.3. Recepción: fotodiodo PIN

Tras haberse propagado el campo por la fibra, el último paso es la recepción. La función proporcionada por OPTILUX para el receptor la hemos considerado elemental dado que no consideraba todas las contribuciones de ruido del receptor. Así, se ha redactado en colaboración con el grupo de comunicaciones ópticas de la UPC una función modificada para este receptor llamada "PIN_receiver" en la cual se ha añadido la posibilidad de introducir una corriente de oscuridad y ruidos 'shot' y térmico.

Al igual que para la fibra necesitamos indicar ciertos parámetros que definirán el comportamiento del receptor (Tabla 4.2).

Parámetro	Descripción	Unidades
x.BW	Ancho de banda del fotodetector	Hz
x.R	Responsividad del fotodetector	-
x.oftype	Tipo de filtro óptico	gauss/ideal/...
x.obw	Ancho de banda del filtro óptico	Hz
x.photonoise	Photo-noise	true/false
x.darkcurrent	Corriente de oscuridad	A

Tabla 4.2: Parámetros del receptor

Con el receptor definido, para obtener la señal de corriente eléctrica se llama a la función "PIN_receiver", la cual devuelve la señal a la salida del receptor. Esta señal ya puede ser pasada por el resto de bloques de recepción del esquema de la figura 2.1, obteniendo así la señal demodulada.

Código 4.3: Código recepción OPTILUX

```
1 Iric = PIN_receiver(1,x); % Obtenemos la senal electrica Iric
```

4.2. Tests realizados

Se han diseñado diferentes escenarios para estudiar los distintos aspectos del sistema óptico que afectarán a la transmisión. Para ello se ha hecho uso del mencionado toolbox OPTILUX. En este capítulo se presentan estos escenarios y se exponen las conclusiones correspondientes.

4.2.1. Curva de sensibilidad

El primer estudio que se ha llevado a cabo es la representación de la curva de sensibilidad para una transmisión CAP. La sensibilidad indica el alcance del sistema, el nivel mínimo de señal óptica que se necesita para un correcto funcionamiento. Este "correcto funcionamiento" se mide en términos de Bit Error Rate, BER. De forma general se representa la potencia recibida en función del BER obtenido y el punto en el que con la potencia recibida se obtenga un BER de 2×10^{-3} será el nivel de sensibilidad. Este BER de 2×10^{-3} puede parece un nivel de error grande (2 de cada 1000 bits), pero realmente éste no sera el BER final dado que, por lo general, se usa Forward Error Correction (FEC). No se va a explicar en detalle el funcionamiento de esta técnica, basta con saber que permite disminuir considerablemente la tasa de error por medio de añadir redundancia a la señal enviada. De acuerdo a determinados estándares, usando una redundancia del 7 % se consigue pasar de un BER de 2×10^{-3} a uno de 10^{-12}, lo cual resulta en un error mínimo [17].

Se va a representar esta curva de sensibilidad para una transmisión CAP de 5 Gb/s con los parámetros de la tabla 4.3.

Parámetro fibra	Valor	Unidades
tx.alphadB	0.2	dB/km
tx.aeff	80	μm^2
tx.n2	2.7e−20	M^2/w
tx.lambda	1550	nm
tx.disp	17	$ps/nm/km$
tx.slope	0	$ps/nm^2/km$
tx.dphimax	3e−3	rad
Parámetro receptor	**Valor**	**Unidades**
x.BW	7	GHz
x.R	0.8	-
x.oftype	gauss	-
x.obw	30	GHz
x.photonoise	true	-
x.darkcurrent	5e−9	A

Tabla 4.3: Parámetros del sistema de transmisión

Además, la señal CAP se ha generado con los parámetros de la tabla 4.4.

Simulando el sistema para distintas longitudes de fibra óptica hemos conseguido las curvas de la figura 4.1.

Hemos conseguido estas curvas por medio de atenuadores ópticos implementados como amplificadores con ganancia negativa (OPTILUX no tiene atenuadores) acoplados a una fibra de longitud variable tal y como se ve en el fragmento de código 4.4.

Parámetro	CAP	Unidades
F_s	50	GHz
M	16	-
Bandas	1	-
R_s	5	GHz
f_c	4.2	GHz
α	0.1	-
span	25	T_s

Tabla 4.4: Parámetros para evaluar m

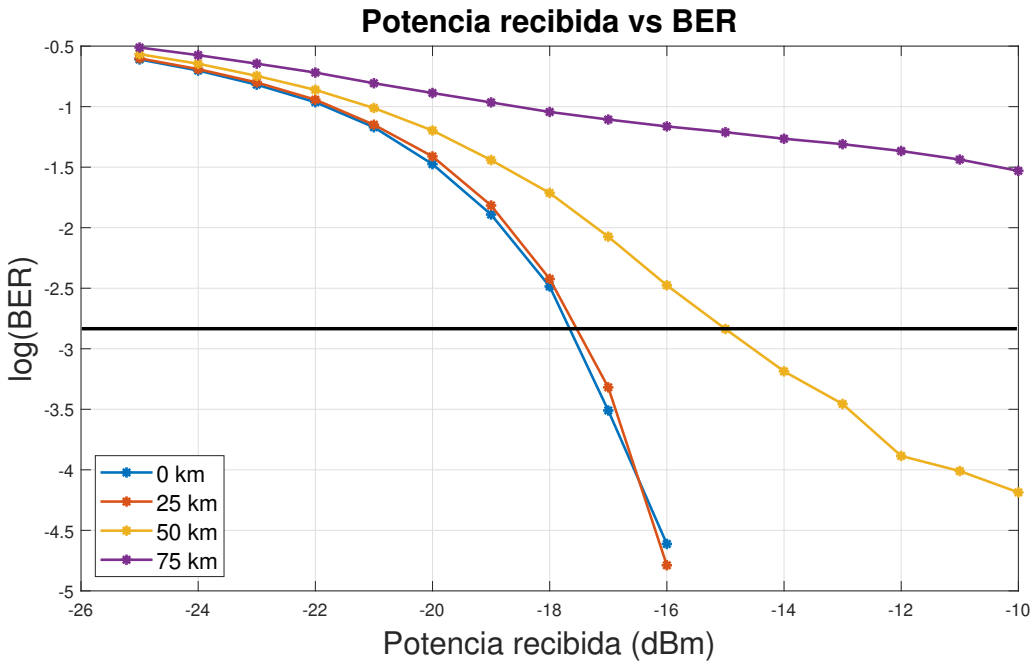

Figura 4.1: Curvas sensibilidad CAP para distintas longitudes de fibra

Código 4.4: Código usado para atenuar la señal

```
ampliflat(-att,'gain')
fiber(tx,'g-sx')
```

Vemos en la curva como, para longitudes de fibra corta, la sensibilidad se encuentra en torno a -17.5 dBm. En cuanto la longitud de la fibra se va haciendo mayor se nota cómo empeoran los resultados: para obtener el mismo BER que antes se necesitan mayores potencias. Esto es debido principalmente a la dispersión cromática de la fibra. Dado que tenemos una transmisión con un gran ancho de banda no nos podemos permitir usar longitudes de fibra tan largas, la causa de este fenómeno se verá en la sección 4.2.4.

4.2.2. CAP vs multiCAP

Vamos a estudiar las diferencias que encontramos en el desempeño de una modulación CAP frente a una multiCAP que transmita la misma información. Para que la prueba tenga sentido se configurarán las modulaciones para que ocupen el mismo contenido frecuencial.

Parámetro	CAP	multiCAP	Unidades
F_s	16	16	GHz
M	16	16	-
Bandas	1	5	-
R_s	5	1	GHz
f_c	4.2	4.2	GHz
α	0.1	0.1	-
span	25	25	T_s

Tabla 4.5: Parámetros para evaluar m

Se va a implementar el mismo sistema de transmisión de 5 Gb/s del apartado anterior. Para CAP será una única banda de 5 GHz, para multiCAP usaremos cinco bandas de 1 GHz. En la tabla 4.5 vemos los parámetros específicos de las modulaciones. Los parámetros del sistema de transmisión son los mismos que en la tabla 4.3.

Con estas configuraciones hemos calculado las curvas de sensibilidad para ambas modulaciones. En la figura 4.2 vemos los resultados.

Podemos ver como los resultados son similares. Las bandas multiCAP de mayor frecuencia funcionan un poco peor pero en un principio no parecería haber demasiada diferencia entre usar una u otra modulación, salvo el añadido de complejidad de la modulación multiCAP. Ahora, la pregunta sería cómo podríamos decir si es mejor usar una modulación u otra.

Las modulaciones multiCAP son muy versátiles, se pueden diseñar las bandas de forma flexible donde se desee y con la anchura que deseemos, además, se puede elegir el orden de la modulación para cada banda. Por esto, si hay más interferencias en una banda de frecuencia se podría saltar o asignar una modulación de orden inferior en esa banda. Es por ello que en general es más útil usar la modulación multiCAP adaptada al sistema en concreto.

4.2.3. Curva Mach Zehnder

Como hemos mencionado en la sección 2.7.1, dependiendo del escenario puede ser útil usar más o menos parte de la curva del MZM (figura 2.17). Al tener una función de transferencia no lineal

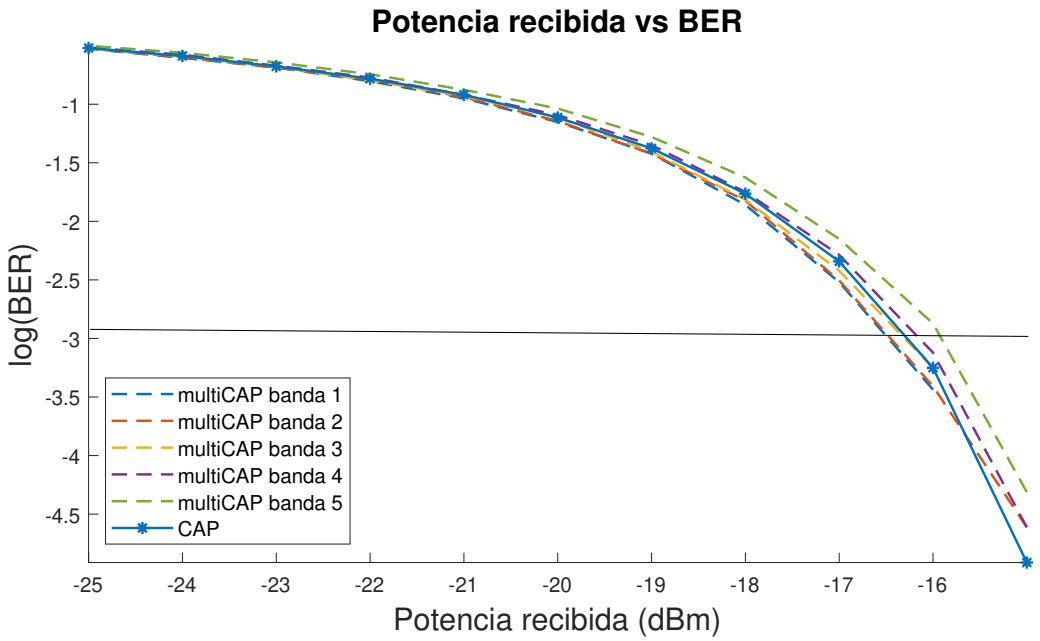

Figura 4.2: Curvas BER vs potencia recibida para CAP y multiCAP

usar una menor parte de la curva se traduce en usar una zona más lineal por lo que la señal es distorsionada en menor medida por el MZM, pero tendríamos la desventaja de que el rango de potencia usado sería mucho menor. Si por el contrario usamos toda la curva, el rango de potencias usado es el máximo, pero contaríamos con una distorsión mayor en las partes de mayor voltaje de la señal. En la figura 4.3 podemos ver este fenómeno plasmado en una curva típica de un MZM.

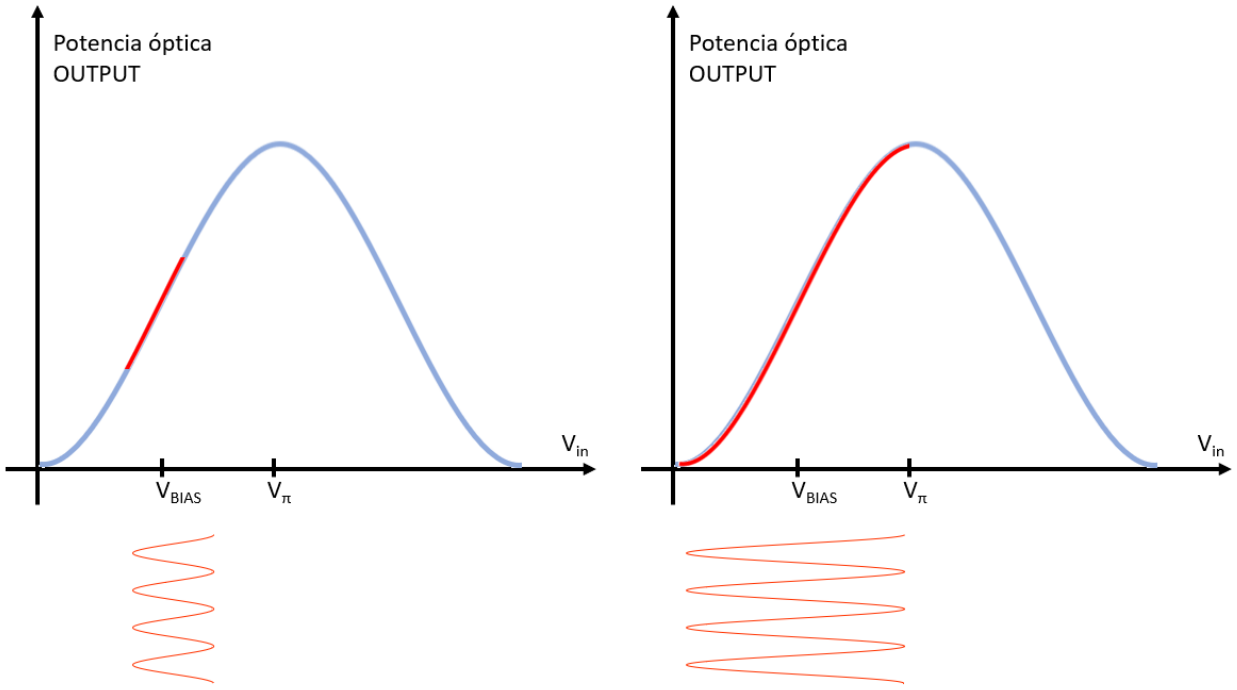

Figura 4.3: Distintos usos del Mach Zehnder usando una mayor o menor porción de la curva

Para ajustar la porción de la curva a usar hemos creado un parámetro "m" el cual indica la fracción del total de la curva que se usa. En la figura 4.3 se usa un m = 0.2 para la primera curva y un m = 1 para la segunda.

La distorsión que provoca el uso de toda la curva la podemos ver reflejada en la constelación.

Los símbolos que se transmiten con los voltajes máximos y mínimos son los que más se verán afectados por esas no linealidades. En la figura 4.4 se ve una comparación para distintos valores de m. Se comprueba como ciertamente los símbolos más alejados del centro de la constelación, es decir los que más cerca están de los extremos de la curva, son los que mayor distorsión presentan cuando usamos una gran parte de la curva.

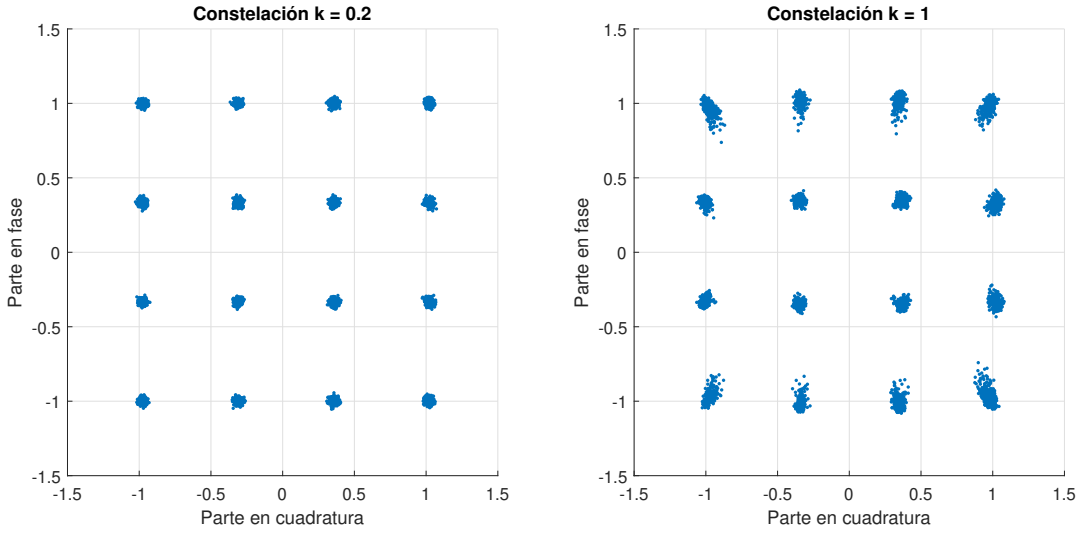

Figura 4.4: Constelaciones para distintos valores de m

Que haya una mayor distorsión cuando usamos un valor de m alto no significa que siempre vaya a ser mejor usar valores de m pequeños. Esto va a depender del canal y la potencia transmitida, si tenemos un canal corto y que no atenúa demasiado la señal si que podemos usar valores pequeños de m dado que la relación señal a ruido será grande y no necesitaremos una gran variación de potencia. Si por el contrario tenemos un canal largo y que atenúe mucho la señal, el ruido añadido por elementos como el receptor será mucho mayor en proporción lo cual se traduce en una peor relación señal a ruido. En estos casos es mejor usar una mayor parte de la potencia que disponemos pese a que se añada esa distorsión.

Para estudiar la dependencia de m con el canal se va a generar una gráfica donde se verá el EVM (magnitud del error vectorial) en función del factor m para distintas potencias recibidas. De esta forma se podrá observar el valor óptimo de m en función de la potencia recibida. Para generar esta gráfica se ha usado un escenario con las características mostradas en la tabla 4.6 y se ha evaluado el EVM obtenido para distintas longitudes de fibra las cuales influían decremente en la potencia recibida.

Así, se han obtenido las curvas de la figura 4.5. Podemos ver como se cumple lo predicho, cuando la potencia recibida es alta, el valor óptimo de m es menor, por ejemplo para la curva de potencia recibida (Prec) = - 5 dB el valor óptimo de m está en torno a 0.2. Sin embargo, cuando la potencia recibida es pequeña, el valor óptimo de m es mayor. Con Prec = -20 el valor de m óptimo es 0.7. Llega un punto a partir del cual si recibimos menos potencia el valor de m óptimo es el máximo, 1, vemos como esto pasa para la curva de P_rec = -25.

Otra forma de ver esta dependencia de m con la potencia recibida es mediante la representación del EVM en función de la potencia recibida usando distintos valores de m. En la figura 4.6 vemos esta representación. De nuevo, se pueden sacar las mismas conclusiones, para valores altos de potencia recibida se encuentra que el menor EVM se obtiene con valores de m bajos y, cuando la potencia comienza a ser menor, los valores de m para los que se obtiene el menor EVM son mayores.

Parámetro CAP	Valor	Unidades
F_s	16	GHz
M	16	-
R_s	0.5	GHz
f_c	0.5	GHz
α	0.1	-
span	25	T_s
Parámetro MZM	Valor	Unidades
V_π	1	-
V_{BIAS}	0.5	-
Parámetro fibra	Valor	Unidades
tx.alphadB	0.2	dB/Km
tx.aeff	80	μm^2
tx.n2	2.7^{-20}	M^2/w
tx.lambda	1550	nm
tx.disp	17	$ps/nm/km$
tx.slope	0	$ps/nm^2/km$
tx.dphimax	3e−3	rad
Parámetro receptor	Valor	Unidades
x.BW	0.4	GHz
x.R	0.8	-
x.oftype	gauss	-
x.obw	5	GHz
x.photonoise	true	-
x.darkcurrent	5e−9	A

Tabla 4.6: Parámetros para evaluar m

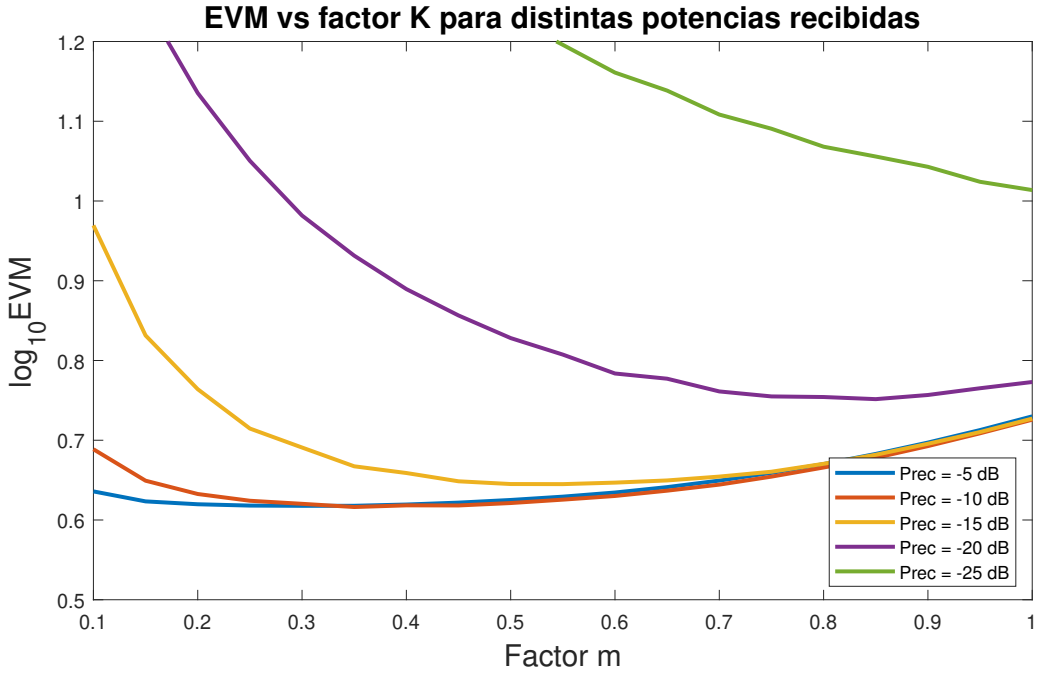

Figura 4.5: EVM en función de m para distintas potencias recibidas

Estas curvas pueden obtenerse para cualquier sistema. Si conocemos la potencia media que va a llegar al receptor podremos escoger el valor de m óptimo para esa transmisión.

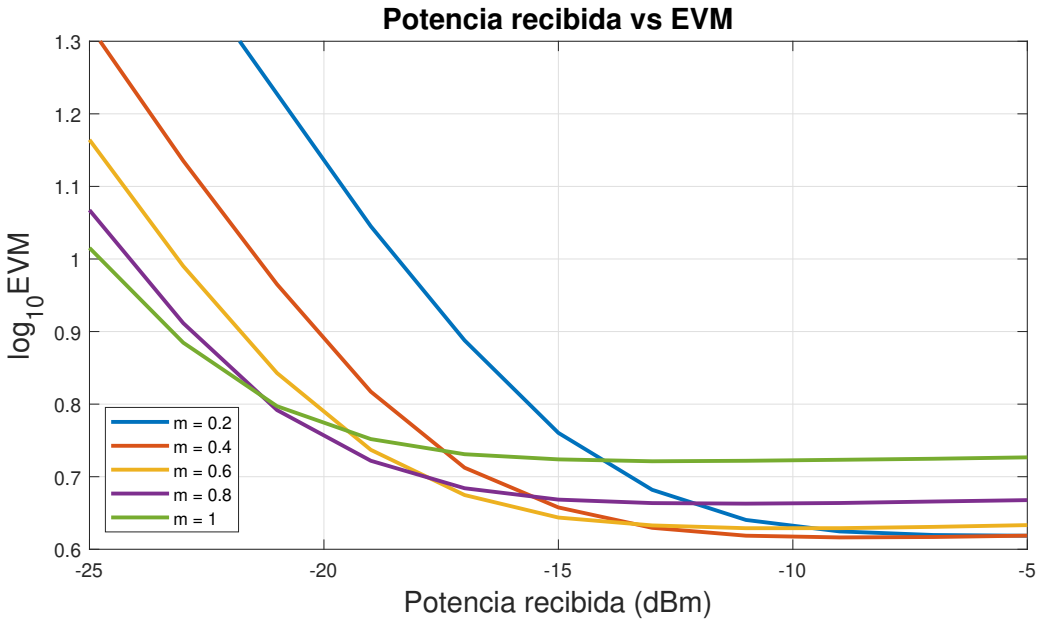

Figura 4.6: EVM en función de la potencia recibida para distintos valores de k

4.2.4. Variación de dispersión

Como ya hemos comentado en el marco teórico, la dispersión varía en función de la longitud de onda. Para señales digitales esto se traduce finalmente en un ensanchamiento de los pulsos transmitidos. Para las señales CAP, además, se traduce en que las distintas componentes de la señal a enviar viajan a distintas velocidades por la fibra, lo cual supone un desfase entre diferentes frecuencias en el espectro. Dado que para recibir estamos usando detección directa, lo que implica hacer el módulo cuadrado del campo eléctrico de la señal en banda base, las fases laterales del espectro óptico se suman, generando una cierta distorsión y además, si la diferencia entre las fases se acerca a π, aparecerá un nulo en el espectro.

En la figura 4.7 se ha transmitido una señal multiCAP de 10 bandas llegando a frecuencias muy altas. Esta señal se ha transmitido por una fibra de 80 km. En la figura de la izquierda tenemos la señal óptica antes de pasar por el detector. Cuando la señal es detectada, las fases de la señal CAP por debajo y por encima de la portadora se suman produciendo los mencionados nulos en ciertas frecuencias que podemos ver en la figura de la derecha. Este fenómeno que empeora la señal solo pasa si tenemos frecuencias altas y una longitud de fibra considerablemente grande debido a que la dispersión acumulada será mayor. Para prevenir que suceda esto, se pueden usar diferentes técnicas.

Una opción simple es usar fibras especiales para paliar los efectos de la dispersión. Existen dos tipos de fibras que nos pueden ayudar para este caso: las fibras de compensación de dispersión y las fibras de dispersión desplazada. Las fibras de compensación de dispersión son fibras con una dispersión negativa que se añaden en un extremo de la transmisión para compensar la dispersión acumulada durante el resto de la transmisión. Por otro lado, las fibras de dispersión desplazada son unas fibras que se diseñan para tener una dispersión casi nula en la banda de frecuencias usadas,

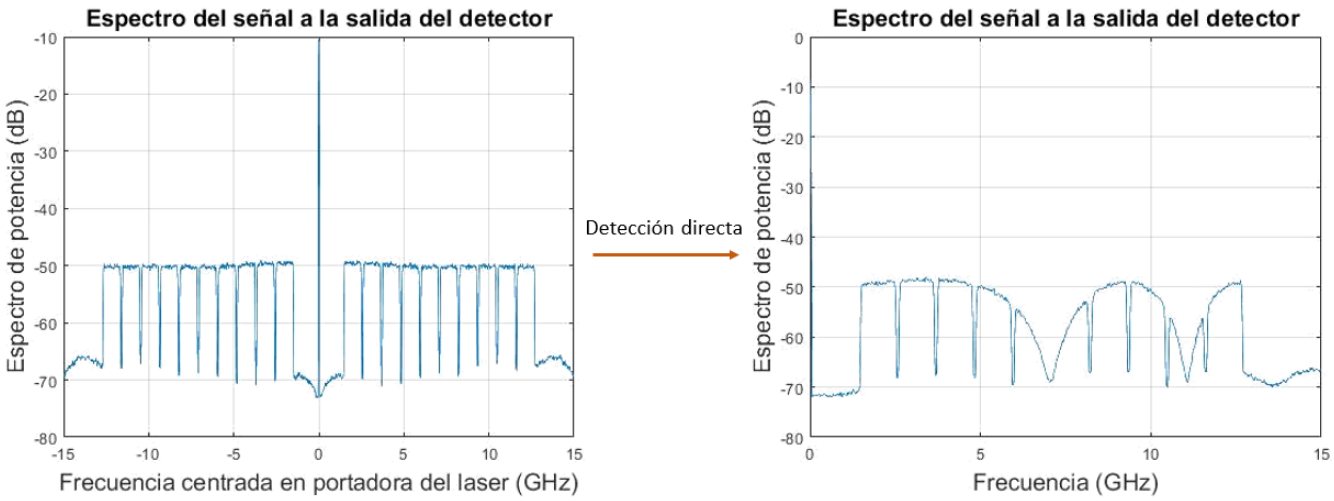

Figura 4.7: Influencia de la dispersión y la detección directa en la señal

de este modo si usamos esta fibra en vez de una fibra convencional los efectos producidos por la dispersión serán mínimos. Esta última solución no es muy popular debido a que se descubrió que estas fibras son muy sensibles a las no linealidades [18].

En la figura 4.8 vemos como, usando una fibra de compensación, conseguimos paliar este efecto de forma completa. La contrapartida es que debemos añadir una longitud considerable de fibra de compensación lo cual añade complejidad, una fuente extra de atenuación a compensar y un coste incremento del coste económico.

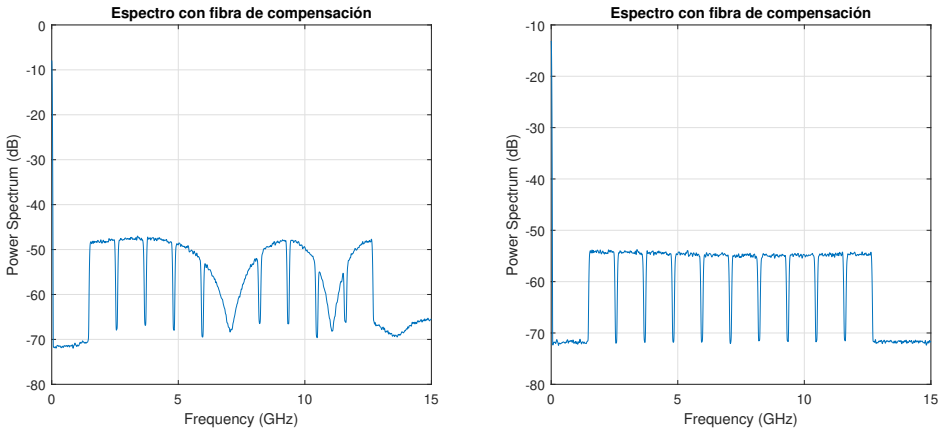

Figura 4.8: Espectro de la señal recibida con y sin fibra de compensación

Otras alternativas vienen del uso de modulaciones multiCAP, ya que éstas permiten situar los distintos canales donde uno diseñe, y conociendo la función de transferencia podemos evitar situar bandas en aquellos puntos donde aparezcan estos nulos [19]. Esta técnica es efectiva pero tiene la contrapartida de que debemos conocer la función de transferencia del canal a priori.

Una última opción, como ya hemos explicado en el marco teórico, sería enviar solo una de las dos partes simétricas del espectro óptico de la figura 4.7. Esto lo conseguimos haciendo uso de una modulación SSB de la señal. De este modo no existiría ese desfase entre las bandas dado que solo existe una de las dos bandas y no aparecerían esos nulos. Para implementar esta modulación SSB hay dos alternativas, la más directa consiste en filtrar ópticamente una de las dos bandas (superior o inferior), pero puede complicarse si queremos hacerlo bien ya que el filtro deberá ser muy restrictivo. La otra opción consiste en usar el esquema de la figura 4.9 que podemos encontrar

en [1]. En este esquema se hace uso de una red de desplazamiento de fase de 90^{o} con la cual se consigue el espectro Y_2 al filtrarla con la señal de entrada y multiplicarla por un seno. Así, si sumamos este espectro con el de la señal multiplicada por un coseno, obtendremos la modulación SSB deseada centrada en la frecuencia del seno y el coseno tal y como se ve en los espectros de la figura 4.9.

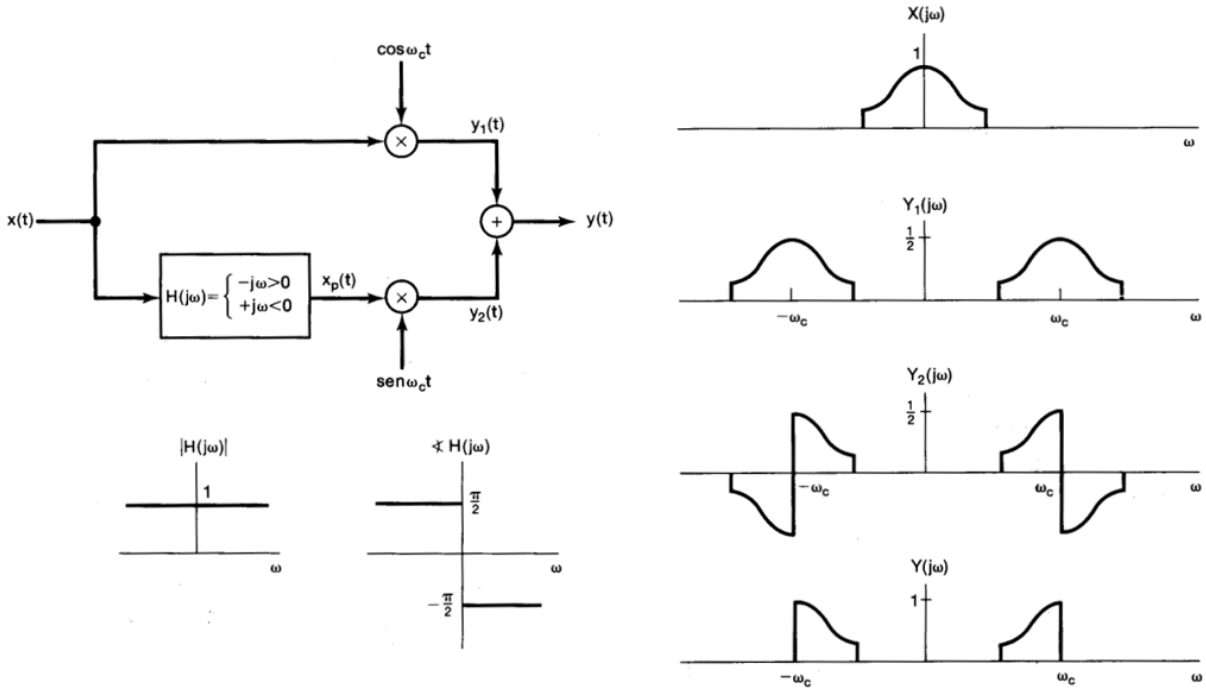

Figura 4.9: Método para modular una señal en SSB [1]

Capítulo 5

Conclusiones y trabajo futuro

En este proyecto hemos conseguido implementar una versión funcional de modulación y demodulación CAP y multiCAP en MATLAB, los códigos los podemos ver en el anexo A. Además hemos podido simular entornos ópticos mediante el toolbox de OPTILUX, con lo que hemos conseguido evaluar distintos factores de una transmisión real. Hemos conseguido representar e interpretar las curvas de sensibilidad para una transmisión CAP cualquiera. Además, hemos visto cómo la diferencia entre CAP y multiCAP es mínima a la hora de comparar sus curvas de sensibilidad en la figura 4.2. Pero también se ha mencionado la versatilidad de multiCAP, y como podemos con multiCAP mandar las bandas que queramos en el rango de frecuencias que queramos y a su vez escoger el orden de modulación de cada banda. Permitiéndonos asignar modulaciones de menor orden a las bandas que tengan una peor sensibilidad.

Hemos concluido que para cada transmisión existe un valor correcto de amplitud de modulación para el MZM al que hemos llamado "m" y que es posible calcularlo para cada escenario. Sería posible simular las curvas vistas con OPTILUX para el escenario real que vayamos a usar y aplicar el valor de "m" óptimo estimado a este.

Por último, hemos estudiado como afecta la dispersión a señales con un gran rango de frecuencia, hemos visto que en cuanto se acumula mucha dispersión empiezan a aparecer nulos en el espectro debido a la diferencia de fase entre la banda superior e inferior de la señal óptica y la detección directa. Para solucionarlo se ha visto que podemos usar fibras especiales con dispersión desplazadas o de baja dispersión, también podemos aprovecharnos de las modulaciones multiCAP y no poner bandas donde haya nulos. La última opción vista es haciendo uso de una modulación óptica SSB para que solo haya una banda en el dominio óptico.

En un futuro sería interesante comprobar todos los tests realizados en el laboratorio para comprobar tanto la veracidad como la precisión de las simulaciones con OPTILUX, toolbox que podría convertirse en una herramienta muy útil para hacer prácticas de esta índole.

Las modulaciones CAP prometen ser un gran candidato para sistemas de transmisión ópticos. En los últimos años ha ganado interés gracias a su eficiencia espectral, escalabilidad de la modulación y bajo coste. Además, las modulaciones CAP podría ser muy útiles para dar servicios a distintos usuarios (multiCAP). CAP se ha comparado con otros formatos de modulación y los resultados muestran a CAP como una alternativa muy competitiva.

Las principal desventaja que aparece en CAP es la ya mencionada dependencia con la respuesta frecuencial del canal, ésta ha de ser muy plana o los resultados empeoran de forma drástica. Es por esto que se precisa de futuras investigaciones para desarrollar ecualizadores que puedan paliar este fenómeno. Por ende, concluimos que CAP merece una mayor atención dado que, una vez se

mitiguen estos efectos, podría llegar a ser una de las modulaciones más competitivas para sistemas de transmisión ópticos.

Apéndice A

Códigos MATLAB

A.1. Mapeo M-QAM

Para poder hacer una modulación CAP primero necesitamos poder hacer un mapeo y desmapeo QAM. En esta sección lo implementamos.

A continuación se muestra la función para la modulación de la cadena de bits, esta función transformará la cadena de bits en una de símbolos dividiéndola en tramos de $log_2 M$. Tras ello, mapeará estos símbolos en una constelación M-QAM usando codificación Gray para obtener un menor error en el bit y devolverá tanto los valores enteros de los símbolos enviados como el vector complejo de los símbolos mapeados (con parte en fase y en cuadratura propia de modulaciones M-QAM).

Código A.1: Función Matlab Modulación M-QAM

```
function [ dataMod, dataSymb ] = qammodulation( dataIn , M)
% qammodulation Funcion para modular una cadena dataIn de bits en qam de M
% simbolos.

% qammodulation( dataIn, M ): devuleve los datos modulados en un vector
% complejo dataMod y los simbolos generados dataSymb resultantes a la
% modulacion M-QAM de el vector fila de bits dataIn con un orden M.

k = log2(M);  % Bits por simbolo

% Anadir ceros a la cadena si no es multiplo de k para que haya un numero
% entero de simbolos.
if mod(length(dataIn),k)~=0
    dataIn = [dataIn zeros(1,k-mod(length(dataIn),k))];
    warning('Zero padding en modulacion');
end
% Reordenamos los bits en columnas de log2(M)
matrixDataIn = reshape(dataIn,k,length(dataIn)/k)';
dataSymb = bi2de(matrixDataIn);  % Convertimos bits en numeros decimales

% Codificamos en gray para menos error en el bit
dataSymGray = bin2gray(dataSymb,'qam',M);

% Creamos la constelacion
const = CreateConstQAM(M);

```

```
% Mapeamos segun el valor en gray
dataMod = const(dataSymGray+1);
dataMod = complex(real(dataMod), imag(dataMod)); % Asegurar num complejo

end
```

Los símbolos generados en la Función A.1 son mapeados en la misma haciendo uso de la Función A.2 (Mostrada a continuación), la cual genera una constelación M-QAM. Devuelve la constelación como un vector con los símbolos ordenados de arriba abajo y de izquierda a derecha, esto es para hacer posible el acceso a esta según el número de símbolo en Gray.

Código A.2: Función para crear una constelación M-QAM

```
function [ constellation ] = CreateConstQAM( M )
% CreateConstQAM Funcion para crear una contelacion M-QAM haciendo uso de
% codificacion Gray.

% CreateConstQAM( M ): devuelve un vector con los valores de la
% constelacion de arriba a abajo y de izquierda a derecha. Esta vector sera
% usado para mapear los distintos simbolos en la contelacion.
if(mod(log2(M),2) ~= 0)
    warning('M debe ser tal que M = 2^n con n par');
end
const = [];
% Creamos un cuadrante de la constelacion.
for iIndex = 1 : 2 : sqrt(M) - 1
    for qIndex = 1 : 2 : sqrt(M) - 1
        const = [const; iIndex+1i*qIndex];
    end
end
% Anadimos el resto de cuadrantes.
newConst = [const; conj(const); -const; -conj(const)];

% Ordenamos el vector de forma que se sigue la constelacion por columnas de
% arriba a abajo y cambiando de columnas de izquierda a derecha. Asi
% conseguimos que se pueda acceder correctamente con el orden Gray.
constellation = zeros(1,M);
for k = 1:M
    % Componente con menor parte real
    ind1 = find(real(newConst) == min(real(newConst)));
    % De ese, el de mayor parte imaginaria
    tmpArray = -1i*inf * ones(size(newConst));
    tmpArray(ind1) = newConst(ind1);
    ind2 = find(imag(tmpArray) == max(imag(tmpArray)));
    % Lo almacenamos en la siguiente posicion
    constellation(k)= newConst(ind2);
    % Eliminamos el punto ordenado
    newConst(ind2) = [];
end

constellation = constellation/(sqrt(M)-1); % Normalizar
end
```

Código A.3: Función para la demodulación QAM

```
function [ dataOut, dataSym ] = qamdemodulation( dataMod, M )
% Funcion para demodular la entrada dataMod modulada con qam de M simbolos.
% Basada en asignacio de indices a las partes real e imaginaria.
```

```
4
5  % qamdemodulation( dataMod, M ): devuelve el vector de simbolos demodulados
6  % dataSym y el vector de bits correspondientes a esos simbolos dataOut.
7
8  % Asegurarse de la correcta orientacion de la entrada
9  if(size(dataMod,1) == 1)
10     dataMod = dataMod(:);
11 end
12
13 sqrtM = sqrt(M);
14
15 dataMod = dataMod*(sqrtM - 1);  % Desnormalizar
16
17 % En fase/real
18 % Obtener indices de la parte real del 0 a log2(M)
19 rIdx = round( ((real(dataMod) + (sqrtM-1)) ./ 2) );
20 % Truncar valores que se salgan del rango
21 rIdx(rIdx <= -1) = 0; rIdx(rIdx > (sqrtM-1)) = sqrtM-1;
22
23 % Cuadratura/imaginario
24 % Obtener indices de la parte imaginaria del 0 a log2(M)
25 iIdx = round(((imag(dataMod) + (sqrtM-1)) ./ 2));
26 % Truncar valores que salgan del rango
27 iIdx(iIdx <= -1) = 0;
28 iIdx(iIdx > (sqrtM-1)) = sqrtM-1;
29
30 % Computar la salida con valores ideales de la constelacion
31 dataDemod = sqrtM-iIdx-1 +  sqrtM*rIdx;
32
33 dataDemod = dataDemod';
34 dataSym = gray2bin(dataDemod,'qam',M);    % Decodificacion Gray
35 dataOutMatrix = de2bi(dataSym,log2(M));
36 dataOut = reshape(dataOutMatrix',1,length(dataDemod)*log2(M));
37 dataSym = dataSym';
38
39 end
```

A.2. Filtros CAP

Código A.4: Función para generar los filtros CAP

```
1  function [ f1, f2 ] = CAPfilter( fc , a , Ts , fs , span )
2  % CAPfilter Funcion para generar pares de filtros ortogonales dados una
3  % frecuencia de portadora fc y los parametros para generar un filtro coseno
4  % realzado.
5
6  % a: Roll off/ Excess bandwidth factor
7  % Ts: Tiempo de simbolo
8  % fs: frecuencia de muestreo de la salida (>= 1/tau)
9  % span: Tamano del filtro. Numero de periodos Ts a cada lado del pico
10 % fc: frecuencia portadora
11 % Ancho de banda BW = (1+alpha)/Ts;
12 % Elegir fs > 2*BW
13
14 % Generar coseno realzado
15 t_positive = eps:(1/fs):span*Ts;  % Reemplazar 0 con eps para evitat NaN
16 t = [-fliplr(t_positive(2:end)) t_positive];
```

```
17 tpi = pi/Ts; amtpi = tpi*(1-a); aptpi = tpi*(1 + a);
18 ac = 4*a/Ts; at = 16*a^2/Ts^2;
19 f = (sin(amtpi*t) + (ac*t).*cos(aptpi*t))./(tpi*t.*(1-at*t.^2));
20
21 % Multiplicar por las portadoras generando filtros ortogonales
22
23 f1 = f.*cos(2*pi*fc*t);
24 f2 = f.*sin(2*pi*fc*t);
25
26 end
```

A.3. Funciones de control

Código A.5: Función que calcula retardo y dibuja el diagrama de ojo

```
1  function [ muestreo ] = diagramaojo( x, N )
2  % diagramaojo: Esta funcion devuelve el desfase del diagrama de ojo respecto
3  % al instante de muestreo. Ademas, hace una representacion de este diagrama
4  % de ojo para ver como se ve el desfase reflejado en el.
5
6  % Vamos a enventanar la senal recibida en vectores fila de N muestreas.
7  L = length(x)/N;          % Numero de filas
8  y = reshape(x,[N,L])';    % Guardamos en 'y' las ventanas de N muestras
9
10 % Sacamos el maximo y minimo de cada muestra usando todas las ventanas y
11 % en el punto de menor compresion sera el de muestreo.
12 sup = max(y,[],1);
13 inf = min(y,[],1);
14
15 compress = sup-inf;
16
17 muestreo = find(compress == min(compress));
18
19 % Representacion diagrama de ojo
20 t = 0:size(y,2)-1;
21 figure, hold on,
22 for i = 1:size(y,1)
23     plot(t,y(i,:),'b')
24 end
25
26 end
```

Código A.6: Función para el cálculo de la BER

```
1  function [ ber , num ] = BER( dataIn , dataOut , M)
2  %BER Calcula la tasa de error en el bit entre dos senales y el numero total
3  %de bits erroneos.
4
5  k = log2(M);
6  % Anadir ceros a la cadena in si no es multiplo de k = log2(M)
7  if mod(length(dataIn),k)~=0
8      dataIn = [dataIn zeros(1,k-mod(length(dataIn),k))];
9      warning('Zero padding BER');
10 end
11
12 num = sum(mod(dataIn+dataOut,2));
```

```
13 ber = num/length(dataIn);
```

Código A.7: Función para el cálculo de la SER

```
1  function [ SER, num ] = SER( dataSymIn, dataSymOut)
2  %BER Calcula la tasa de error en el simbolo entre dos senales y el numero
3  % total de simbolos erroneos
4
5  if(size(dataSymIn)>1)  % Para ver si es CAP o multiCAP
6      i = 2;
7  else
8      i = 1;
9  end
10 num = sum(dataSymIn ≠ dataSymOut,i);
11 SER = num/length(dataSymIn);
12
13 end
```

Código A.8: Función para el cálculo del EVM

```
1  function [ evm ] = evm( ref, rec )
2  %EVM Calcula el modulo del vector error para CAP o multiCAP
3
4  evm = zeros(1,size(ref,1));
5  for i = 1:size(ref,1)
6
7      errorvector=rec(i,:)-ref(i,:);
8      p=sqrt(mean(abs(ref(i,:).^2)));
9      evnorm = errorvector/p;
10
11     evm(i) = sqrt(mean(abs(evnorm(:)).^2))*100;
12
13 end
14
15 end
```

A.4. Códigos de transmisiones completas

Código A.9: Código completo de una transmisión CAP

```
1  %% ESQUEMA CAP COMPLETO SIN OPTILUX
2
3  clear
4  clc
5
6  %% GENERACION SENAL A ENVIAR
7
8  M = 16;          % Orden de la modulacion
9  n = 10*(2^12);   % Numero de bits
10
11 rng('default')
12 dataIn = randi([0 1],1,n);  % Data Stream
13 [dataMod, dataSymIn] = qammodulation(dataIn,M);  % Datos a QAM
14
```

```
15 % Calculamos potencia para futuro uso
16 potIn = mean(abs(dataMod).^2);
17
18 %% UPSAMPLING
19
20 Rs = 0.5e9; % [sym/s] [0.5e9]
21 fs = 10e9; % Frecuencia de muestreo de la salida analogica (Fawg) [10e9]
22 Nss = fs/Rs; % Muestras por simbolo (Number of samples per symbol)
23
24 dataMod_up = upsample(dataMod,Nss); % Upsampling por diferencia entre tasas
25
26 %% CONVOLUCION CON FILTROS
27
28 a = 0.1;     % Roll-off factor [0.1]
29 span = 20;   % Numero de periodos Ts a cada lado del filtro [20]
30 Ts = 1/Rs;   % Periodo de simbolo
31 fc = 2e9;    % Frecuencia portadora CAP [2e9]
32
33 % Calculamos filtros ortogonales
34 [fI, fQ] = CAPfilter( fc , a , Ts , fs , span );
35
36 xi = conv(real(dataMod_up),fI);
37 xq = conv(imag(dataMod_up),fQ);
38 x = xi + xq;
39
40 %% CANAL simulamos ruido blanco y retardo
41
42 % % Anadimos algo de ruido
43 xr = awgn(x,15);
44 % % Anadimos un retardo
45 xr = [zeros(1,46) xr];
46
47 %% RECEPCION
48 % Nueva fs (receptor)
49 NssOsc = 1;
50 fsOsc = NssOsc*fs;
51 % Remuestreamos la senal recibida con fsOsc
52 xr = interp(xr,NssOsc);
53
54 [f1r, f2r] = CAPfilter( fc , a , Ts , fsOsc , span );
55
56 yi = conv(xr,fliplr(f1r));
57 yq = conv(xr,fliplr(f2r));
58
59 y = yi + 1j*yq;
60
61 % Eliminamos las etapas transitorias de los filtros al principio y al final.
62 y = y(length(f1r)-(fsOsc/fs)+1:end-length(f1r));
63
64 % Compensamos retardo
65 retardo = diagramaojo(real(y(1:10000)),fsOsc/Rs);
66 y = y(retardo:end);
67
68 %% MUESTREO y NORMALIZADO
69
70 yd = downsample(y,fsOsc/Rs);
71
72 % Eliminamos bits de mas causados por el retardo.
73 yd = yd(end-length(dataMod)+1:end);
74
75 % Ajustamos potencia teniendo cuenta la potencia enviada
```

```
P = modnorm(yd,'AVPOW',potIn);
yn = yd*P;

%% Demodulacion

[dataOut, dataSymOut] = qamdemodulation(yn,M);

%% Rendimiento

[ber, NbitErr] = BER(dataIn, dataOut, M);
[ser, NsymErr] = SER(dataSymIn, dataSymOut);

EVM = evm(dataMod,yn);

%% Constelaciones

scatterplot(yn)
```

Código A.10: Código completo de una transmisión multiCAP

```
%% ESQUEMA MULTICAP SIN OPTILUX
% El multicap lo he implementado con funciones nuevas que o bien son
% conceptos nuevos o utilizan un 'for' para aplicar las funciones cap a
% cada canal.

clear
clc

%% GENERACION SENAL A ENVIAR

M = 16;  % Orden de la modulacion
k = log2(M);  % Numero de bits por simbolo
n = 10*(2^12);  % Numero de bits
N = 6;  % Numero de transmisiones CAP (bandas)

rng('default');
dataIn = prbs(N,n);

for i = 1:N % Mapeamos cada canal
    [dataMod(i,:), dataSymIn(i,:)] = qammodulation(dataIn(i,:),M);
end

% Calculamos potencia para futuro uso
potIn = mean(abs(dataMod).^2,2);

%% UPSAMPLING

Rs = 0.5e9;  % [sym/s] [0.5e9]
fs = 10e9;  % Frecuencia de muestreo de la salida analogica (Fawg) [10e9]
Nss = fs/Rs;  % Muestras por simbolo (Number of samples per symbol)

dataMod_up = (upsample(dataMod',Nss))';  % Upsampling

%% CONVOLUCION CON FILTROS

B = 0.1;  % Roll-off [0.1]
span = 20;  % Numero de muestras filtro [20]
Ts = 1/Rs;  % Periodo de simbolo

fc1 = 0.5e9;       % Primera portadora[2e9]
```

```
41 fesp = 100e6;    % Distancia de seguridad entre bandas
42
43 % Matriz NxMuestrasx2 (Filtros ortogonales en fcap(i,:,1) y fcap(i,:,2))
44 fcap = multi_CAPfilter( fc1,  fesp, N, B , Ts , fs , span );
45
46 % Calculamos todas las componentes I y Q
47 xi = zeros(N,length(dataMod_up)+size(fcap,2)-1);
48 xq = xi;
49 for i = 1:N
50     xi(i,:) = conv(real(dataMod_up(i,:)),fcap(i,:,1));
51     xq(i,:) = conv(imag(dataMod_up(i,:)),fcap(i,:,2));
52 end
53
54 % Senal a enviar
55 x = sum(xi,1)+sum(xq,1);
56
57 %% CANAL
58 % Anadimos algo de ruido
59 xr = awgn(x,20);
60 % Anadimos un retardo
61 xr = [zeros(1,47) xr];
62 %% DEMODULACION
63 % Nueva fs (receptor)
64 NssOsc = 2;
65 fsOsc = NssOsc*fs;
66 % Remuestreamos la senal recibida con fsOsc
67 xr = interp(xr,NssOsc);
68
69 fcapr = multi_CAPfilter( fc1,  fesp, N, B , Ts , fsOsc , span ); % Matriz ...
      filtros
70
71 for i = 1:N
72     yi(i,:) = conv(xr,fliplr(fcapr(i,:,1)));
73 end
74 for i = 1:N
75     yq(i,:) = conv(xr,fliplr(fcapr(i,:,2)));
76 end
77
78 y = yi + 1j*yq;
79
80 y = y(:,length(fcapr)-(fsOsc/fs)+1:end-length(fcapr));
81
82 retardo = diagramaojo(real(y(1,1:10000)),fsOsc/Rs);
83
84 y = y(:,retardo:end);
85
86 %% MUESTREO y NORMALIZADO
87
88 yd = downsample(y',fsOsc/Rs)';
89
90 yd = yd(:,end-length(dataMod)+1:end);
91
92 for i = 1:N
93     P = modnorm(yd(i,:),'AVPOW',potIn(i));
94     yn(i,:) = yd(i,:)*P;
95 end
96
97 %% Recuperar bits
98
99 for i = 1:N
100     [dataOut(i,:), dataSymOut(i,:)] = qamdemodulation(yn(i,:),M);
```

```
101 end
102
103 %% Rendimiento
104
105 for i = 1:N % Calculo del BER
106     [ber(i), num(i)] = BER(dataIn(i,:),dataOut(i,:),M);
107 end
108
109 [ser, NsymErr] =  SER( dataSymIn, dataSymOut); % Calculo del SER
110
111 EVM = evm(dataMod,yn); % Calculo del EVM
112
113 %% Constelaciones
114
115 for i = 1:N
116     subplot(2,ceil(N/2),i),
117     scatter(real(yn(i,:)),imag(yn(i,:)),'.')
118     title(['Constelacion canal: ',num2str(i)])
119     ylabel('Parte en fase'), xlabel('Parte en cuadratura')
120 end
```

A.5. OPTILUX

Para OPTILUX ya se ha mencionado en la parte teórica cómo hay que usar el código. Vamos a añadir en este anexo un código completo con un ejemplo para un mejor entendimiento de cómo funciona. Mostramos solo el código para CAP, para multiCAP es extrapolable. Solo hay que añadir el canal óptico entre modulación y demodulación.

Código A.11: Código transmisión completa CAP con OPTILUX

```
1
2  clear
3  clc
4
5  %%%%Parametros del campo
6
7  Nsymb = 5*1024;   % Numero de simbolos a transmitir
8  Nt    = 20;       % Puntos por simbolo
9  Nch   = 1;        % Numero de canales
10 global GSTATE     % Creamos la variable global de OPTILUX
11
12 %% GENERACION SENAL A ENVIAR
13
14 M = 16;              % Orden de la modulacion
15 n = Nsymb*log2(M);   % Numero de bits
16
17 rng('default')
18 dataIn = randi([0 1],1,n); % Data Stream
19 [dataMod, dataSymIn] = qammodulation(dataIn,M); % Datos a QAM
20
21 % Calculamos potencia para futuro uso.
22 potIn = mean(abs(dataMod).^2);
23
24 % UPSAMPLING
25
26 Rs = 0.5e9; % [sym/s] [0.5e9]
```

```
fs = 10e9; % Frecuencia de muestreo de la salida analogica (Fawg) [10e9]
Nss = fs/Rs; % Muestras por simbolo (Number of samples per symbol)

dataMod_up = upsample(dataMod,Nss); % Upsampling por diferencia entre tasas

% CONVOLUCION CON FILTROS

a = 0.1;     % Roll-off [0.1]
span = 20;   % Numero de muestras filtro [20]
Ts = 1/Rs;   % Periodo de simbolo
fc = 0.7e9;    % Frecuencia portadora [2e9]
% Calculamos filtros ortogonales
[f1, f2] = CAPfilter( fc , a , Ts , fs , span );

xi = conv(real(dataMod_up),f1);
xq = conv(imag(dataMod_up),f2);
X = xi + xq;

% Modificamos la senal para el MZM. Ha de estar entre -k y k.
k = 0.8;
X = X/(2*max(abs(X)));
X = X';
X = X*k;

% Modificamos la longitud para cuadrar con los simbolos que enviamos
% con OPTILUX (Realmente los bits iniciales y finales no son simbolos si
% no las muestras de mas generadas en el filtrado que seguimos necesitando)
padd = Nt-mod(length(X),Nt);
X = [X; ones(padd,1)*X(end)];

Nsymb = length(X)/Nt;

%% CANAL OPTICO

%%%% Parametros del pulso

exratio  = 13;           % extinction ratio [dB]
lam      = 1550;         % longitud de onda central [nm]
symbrate = Rs/(1e9);     % baudrate [Gbaud]

Fs = Nt*symbrate*1e9;    % Frecuencia de muestreo
GSTATE.Fs = Fs;

%%%%Parametros del enlace

Din   = 0;          % in-line dispersion x span [ps/nm]
Nspan = 1;          % numero de spans

%%%% Fiber 1 (Tx)
tx.length  = 1e5;      % longitud [m]
tx.alphadB = 0.2;      % atenuacion [dB/km]
tx.aeff    = 80;       % area efectiva [um^2]
tx.n2      = 2.7e-20;  % indice no lineal
tx.lambda  = 1550;     % longitud de onda [nm] @ dispersion
tx.disp    = 17;       % dispersion [ps/nm/km] @ wavelength
tx.slope   = 0;        % pendiente [ps/nm^2/km] @ wavelength
tx.dphimax = 3E-3;     % maxima rotacion no linear de fase por step
tx.dzmax   = 2E4;      % maximo SSFM step

%%%%Receptor
x.BW = 1e9;            % Ancho de banda del fotodetector [Hz]
```

```
x.R = 0.8;              % Responsividad
x.plot = 'ploteye';
x.ts     = 0;
x.oftype = 'gauss';     % Tipo de filtro optico
x.obw    = 100;         % Ancho de banda del filtro optico
x.delay  = 'theory';

% Activamos ruido del fotodetector
x.photonoise = true;
x.darkcurrent = 5e-9; % Corriente oscura del fotodetector [A].

%%%%Tx side

reset_all(Nsymb,Nt,Nch);
GSTATE.SYMBOLRATE = symbrate;

Pavg = 1; % mW
E    = lasersource(Pavg, lam);

Eopt = mz_modulator_mod(E,X,struct('exratio',exratio,'amplitude',1,'bias',0.5));

create_field('unique',Eopt,[],struct('power','average'));

%%%%Enlace optico

fiber(tx,'g-sx')

[Iric,x] = PIN_receiver(1,x); % Recibimos en Iric la senal electrica

Iric = Iric(1:end-padd); % Quitamos los bits de mas anadidos

%% DEMODULACION
%Nueva fs (receptor)
NssOsc = 1;
fsOsc = NssOsc*fs;
% % Remuestreamos la senal recibida con fsOsc

xr = interp(Iric',NssOsc);

[f1r, f2r] = CAPfilter( fc , a , Ts , fsOsc , span );

yi = conv(xr,fliplr(f1r));
yq = conv(xr,fliplr(f2r));

y = yi + 1j*yq;

y = y(length(f1r)-(fsOsc/fs)+1:end-length(f1r));

% eyediagram(y,20)

% MUESTREO y NORMALIZADO

yd = downsample(y,fsOsc/Rs);
yd = yd(1:length(dataMod));

% Ajustamos potencia para pico = 1
P = modnorm(yd,'AVPOW',potIn);
yn = yd*P;

% Recuperar bits
[dataOut, dataSymOut] = qamdemodulation(yn,M);
```

```
149
150 % Rendimiento
151 [ber, NbitErr] = BER(dataIn, dataOut, M);
152 [ser, NsymErr] = SER(dataSymIn, dataSymOut);
153
154 EVM = evm(dataMod,yn);
155
156 % Constelaciones
157 scatterplot(yn)
```

Código A.12: Función modificada de modulador MZM en OPTILUX

```
1  function Eout=mz_modulator_mod( Ein, modsig, options)
2  %MZ_MODULATOR modulate the optical field with a Mach-Zehnder Interferometer
3  %   E=MZ_MODULATOR(E,MODSIG, OPTIONS) modulates the optical field E using a
4  %   Mach-Zehnder interferometer [1].
5  %   The parameter MODSIG is the electrical driving signal produced by
6  %   ELECTRICSOURCE. OPTIONS is an optional structure whose fields can be:
7  %
8  %       - exratio : extinction ratio [dB] (default = inf)
9  %       - bias  : bias of the modulator (default = 0)
10 %       - amplitude: Vpi of the modulator (default = 1)
11 %       - nochirp: reduce effect of chirp due to finite exratio [2]
12 %         (default = false)
13 %
14 %   See also LASERSOURCE, ELECTRICSOURCE, LINEAR_MODULATOR, PHASE_MODULATOR
15 %
16 %   [1] S. Walklin and J. Conradi, "Effect of Mach-Zehnder modulator DC
17 %   extinction ratio on residual chirp-induced dispersion in 10-Gb/s binary
18 %   and AM-PSK duobinary lightwave systems," IEEE Photon. Technol. Lett.,
19 %   vol. 9, pp. 1400-1402, Oct. 1997.
20 %   [2] K. Hoon and A. H. Gnauck, "Chirp characteristics of dual-drive
21 %   Mach-Zehnder modulator with a finite DC extinction ratio," IEEE Photon.
22 %   Technol. Lett., vol. 14, pp. 298-300, Mar. 2002.
23 %
24 %   Author: Massimiliano Salsi, 2009
25 %   Contributed by: Marco Bertolini, 2009
26 %   University of Parma, Italy
27
28 %    This file is part of Optilux, the optical simulator toolbox.
29 %    Copyright (C) 2009  Paolo Serena, <serena@tlc.unipr.it>
30 %
31 %    Optilux is free software; you can redistribute it and/or modify
32 %    it under the terms of the GNU General Public License as published by
33 %    the Free Software Foundation; either version 3 of the License, or
34 %    (at your option) any later version.
35 %
36 %    Optilux is distributed in the hope that it will be useful,
37 %    but WITHOUT ANY WARRANTY; without even the implied warranty of
38 %    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
39 %    GNU General Public License for more details.
40 %
41 %    You should have received a copy of the GNU General Public License
42 %    along with this program.  If not, see <http://www.gnu.org/licenses/>.
43
44
45 bias      = 0;
46 amplitude = 1;
47 exratio     = inf;
48 nochirp   = false;
```

```
49
50 if exist('options','var')
51     checkfields(options,{'bias','amplitude','exratio','nochirp'});
52     if isfield(options,'bias');
53         bias = options.bias;
54     end
55     if isfield(options,'amplitude');
56         amplitude = options.amplitude;
57     end
58     if isfield(options,'exratio');
59         exratio = options.exratio;
60     end
61     if isfield(options,'nochirp');
62         nochirp = options.nochirp;
63     end
64 end
65
66 exr_lin = 10^(-exratio/10);
67 gamma = (1-sqrt(exr_lin))/(sqrt(exr_lin)+1);
68
69  % MODIFICADA 12/5/2020
70
71  % Definicion de variables
72 Vpi = amplitude;
73
74 if nochirp
75     Phi_U = ( pi*(modsig+bias)/((1+gamma^2)*Vpi));
76     Phi_L = (-pi*gamma^2*(modsig+bias)/((1+gamma^2)*Vpi));
77 else
78     Phi_U = ( pi*(modsig+bias)/(2*Vpi));
79     Phi_L = (-pi*(modsig+bias)/(2*Vpi));
80 end
81
82  % Any phase shift due to the interferometric structure is neglected for the
83  % sake of simplicity
84 Eout  = j*Ein.*(fastexp(Phi_L)-gamma*fastexp(Phi_U))/(1+gamma);
```

Bibliografía

[1] Alan V Oppenheim, Alan S Willsky, and S Hamid Nawab. *Señales y sistemas*. Pearson Educación, 1998.

[2] NM Ridzuan, MFL Abdullah, MB Othman, and MB Jaafar. A carrierless amplitude phase (cap) modulation format: Perspective and prospect in optical transmission system. *International Journal of Electrical and Computer Engineering*, 8(1):585, 2018.

[3] Li Li, Wei Jian-yi, Zhang Xiu-tai, and Li Hong-an. Research on mixed data rate and format transmission in wdm networks. *TELKOMNIKA (Telecommunication, Computing, Electronics and Control)*, 11(1):127–136, 2013.

[4] BU Rindhe, Jyothi Digge, and SK Narayankhedkar. Implementation of optical ofdm based system for optical networks. In *2014 International Conference on Advances in Communication and Computing Technologies (ICACACT 2014)*, pages 1–9. IEEE, 2014.

[5] DD Falconer. Carrierless am/pm. *Bell Laboratories, NJ, USA, Bell Laboratories Technical Memorandum, Tech. Rep*, 1975.

[6] Ahmed F Shalash and Keshab K Parhi. Multidimensional carrierless am/pm systems for digital subscriber loops. *IEEE Transactions on Communications*, 47(11):1655–1667, 1999.

[7] Jason Gao and Yee-Hong Leung. A new adaptive equalizer for carrierless amplitude and phase (cap) receivers. In *ISCAS'99. Proceedings of the 1999 IEEE International Symposium on Circuits and Systems VLSI (Cat. No. 99CH36349)*, volume 3, pages 90–93. IEEE, 1999.

[8] Miguel Iglesias Olmedo, Zuo Tianjian, Jesper Bevensee Jensen, Zhong Qiwen, Xu Xiaogeng, and Idelfonso Tafur Monroy. Towards 400gbase 4-lane solution using direct detection of multicap signal in 14 ghz bandwidth per lane. In *2013 Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC)*, pages 1–3. IEEE, 2013.

[9] Jonathan D Ingham, Richard Penty, Ian White, and David Cunningham. 40 gb/s carrierless amplitude and phase modulation for low-cost optical datacommunication links. In *Optical Fiber Communication Conference*, page OThZ3. Optical Society of America, 2011.

[10] Kenneth William Cattermole. Principles of pulse code modulation. 1969.

[11] John G Proakis and Masoud Salehi. *Digital communications*, volume 4. McGraw-hill New York, 2001.

[12] Ahmed F Shalash and Keshab K Parhi. Multidimensional carrierless am/pm systems for digital subscriber loops. *IEEE Transactions on Communications*, 47(11):1655–1667, 1999.

[13] Grzegorz Stepniak. Comparison of efficiency of n-dimensional cap modulations. *Journal of lightwave technology*, 32(14):2516–2523, 2014.

[14] KP Zetie, SF Adams, and RM Tocknell. How does a mach-zehnder interferometer work? *Physics Education*, 35(1):46, 2000.

[15] Ignacio Garcés Javier Mateo, María Ángeles Losada. Dispositivos y sistemas de transmisión Óptica. *unizar*, 1:188, 2000.

[16] P Serena, M Bertolini, and A Vannucci. Optilux toolbox. *available at optilux. sourceforge. net/Documentation/optilux_ doc. pdf*, 2009.

[17] Aloke Guha. Adaptive forward error correction system and method, December 16 1997. US Patent 5,699,369.

[18] Long Zheng, Xia Zhang, Xiaomin Ren, Huifang Ma, Lei Shi, Yamiao Wang, and Yongqing Huang. Dispersion flattened photonic crystal fiber with high nonlinearity for supercontinuum generation at 1.55 μm. *Chinese Optics Letters*, 9(4):040601, 2011.

[19] RC Srinivasan and JC Cartledge. On using fiber transfer functions to characterize laser chirp and fiber dispersion. *IEEE Photonics Technology Letters*, 7(11):1327–1329, 1995.