PROYECTO FIN DE CARRERA

Estudio de esquemas de diversidad cooperativa en sistemas CDMA

Autor:
Ignacio Mingote Marina

Directora:
Mª Luz de Pablo González

Tutora:
Matilde Pilar Sánchez Fernández

DEPARTAMENTO DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

Leganés, 2009

Agradecimientos

En primer lugar quiero agradecer a la directora del proyecto, Mª Luz de Pablo González, su esfuerzo y colaboración para poder llevar a cabo este proyecto fin de carrera.

También quiero agradecer el esfuerzo dedicado, por todos los profesores de la Universidad Carlos III de Madrid que han participado en mi formación a lo largo de mis estudios de Ingeniería Técnica de Telecomunicación en la especialidad de Sistemas de Telecomunicación.

Por otro lado, agradecer también el incalculable apoyo de mis padres y mis hermanos a lo largo de estos años. Sin duda, todo lo logrado hasta este momento ha sido por su incondicional ayuda. Por supuesto, también a Helen, quien me ha sabido escuchar y comprender en todos los momentos durante estos dos últimos años.

Por último, dar las gracias a todos los compañeros con quienes he compartido grandes momentos a lo largo de estos años. Porque gracias a ellos, la universidad… es la universidad.

Índice

Lista de Figuras

Lista de Tablas

Abstracto

En las comunicaciones móviles de última generación se hace imprescindible la búsqueda de nuevas técnicas que maximicen la calidad de servicio ofrecida al cliente. Los servicios requeridos son cada vez más exigentes y requieren una alta fiabilidad de transmisión. Es necesario que estos servicios se adapten a todos los entornos: urbano, suburbano, vehicular o rural. Por lo tanto se buscan nuevas técnicas capaces de ofrecer mejor calidad en las comunicaciones móviles.

La técnica de acceso al medio *CDMA (Code Division Multiple Access)* y las comunicaciones de espectro ensanchado, son utilizadas en la actualidad en las comunicaciones móviles de tercera generación 3G. Hoy en día nos encaminamos hacia una nueva generación probablemente basada en la cooperación entre usuarios.

En este proyecto se tratan algunas técnicas de diversidad cooperativa como AAF (*Amplify and Forward)* o DAF (*Decode and Forward)*. Se han realizado aplicaciones desarrolladas en *MATLAB*® para simular distintos escenarios en los que combinaremos distintos códigos y modulaciones usadas en aplicaciones *CDMA*, en concreto *UMTS (Universal Mobile Telecommunications System)*. También, simularemos canales que simulen los desvanecimientos provocados por el entorno en el que se encuentren las señales.

Se analizará la tasa de error de bit resultante de cada escenario, y podremos comparar las distintas técnicas y escenarios desarrollados en este proyecto.

1 Introducción

CDMA es una técnica de acceso múltiple donde la información del usuario es ensanchada mediante la multiplicación de esta señal de información por una secuencia de bits aleatorios llamados "chips" [15]. La técnica de CDMA difiere de otras tecnologías como FDMA (*Frequency Division Multiple Access*) o TDMA (*Time Division Multiple Access*), en que esta permite a los usuarios transmitir en la misma frecuencia y en el mismo instante de tiempo. Gracias a CDMA incorporamos una serie de ventajas con respecto a otras tecnologías [15]:

. Tasas de bit mayores de 2Mbps
. Multiplexación de servicios con diferentes requerimientos en una sola conexión.
. Soporta tráfico asimétrico en *Uplink* y *Downlink* (páginas *web*).

Los nuevos avances en la tecnología CDMA introducen algunas mejoras como mecanismos de señalización y control avanzados, mejoradas técnicas contra interferencias y nuevas tecnologías en las antenas como *MIMO* (*Multiple Inputs Multiple Outputs)* y *SDMA* (*Space Division Multiple Access*), con el fin de aumentar las tasas de transmisión y la calidad de servicio[1]. Además en CDMA, a diferencia de otras técnicas de acceso al medio, usaremos códigos digitales para diferenciar a los usuarios. Por lo tanto, todos los usuarios comparten el mismo rango del espectro radioeléctrico en esta técnica.

CDMA está actualmente liderando la tecnología 3G en el mundo y soporta todos los avances tecnológicos que los usuarios requieren como por ejemplo *e-mail, video-streaming*, televisión y redes sociales entre otros[1].

En los foros de estandarización, CDMA ha emergido como la tecnología más ampliamente adoptada en la tercera generación de móviles en cuanto al interfaz radio se refiere. Su especificación ha sido creada en 3GPP (*the third Generation Partnership Project*), la cual incluye las estandarizaciones de Europa, Japón, Corea, USA y China [15].

1.1 Motivación y objetivos

CDMA es una técnica que viene en constante progresión en los últimos años, y está siendo utilizada en la tercera generación de telefonía móvil. Con CDMA se ha aumentado considerablemente la velocidad de las comunicaciones, lo que ha provocado la aparición de nuevos servicios y nuevas aplicaciones requeridas por los usuarios.

El objetivo de este proyecto es analizar el comportamiento de distintas técnicas de diversidad cooperativa en un sistema CDMA, así como las modulaciones y códigos utilizados en estos sistemas. Evaluaremos estas técnicas a través de aplicaciones programadas en MATLAB®.

1.2 Organización del proyecto

El proyecto está estructurado en siete capítulos. El contenido de cada uno de ellos es el siguiente:

El capítulo 1 introduce las motivaciones y objetivos por las cuales se ha realizado este proyecto.

El capítulo 2 explica los conceptos básicos de CDMA, sus ventajas y desventajas, así como los distintos códigos usados y las técnicas de modulación de la señal de espectro ensanchado.

El capítulo 3 introduce el concepto de diversidad cooperativa y explica algunas de las técnicas utilizadas. Veremos como se llevan a cabo y que propiedades conlleva cada una.

El capítulo 4 muestra algunos modelos de canal posibles en la propagación de señales bajo el punto de vista de las señales de espectro ensanchado. En qué consiste cada tipo de canal y bajo que condiciones se usan unos u otros.

El capítulo 5 contiene el análisis práctico de los escenarios desarrollados en MATLAB®. Una detenida explicación de los códigos programados y su relación con la teoría vista en apartados anteriores.

El capítulo 6 incluye los resultados obtenidos de las simulaciones y su consiguiente discusión.

Finalmente, el capítulo 7 contiene la conclusión final del proyecto junto con posibles trabajos futuros a raíz de este.

2 CDMA

La primera generación de telefonía móvil se caracterizaba principalmente por ser analógica y estrictamente para voz. Ofrecía una muy baja velocidad de transmisión, se basaba principalmente en FDMA y la seguridad no existía. La técnica de FDMA consiste en la división del ancho de banda asignado al sistema, en 'porciones' las cuales llamamos canales.

Más adelante entraron las nuevas generaciones de comunicaciones móviles. Estas se caracterizaban por ser digitales, utilizaban protocolos más sofisticados y son los sistemas que se utilizan en la actualidad. La principal tecnología de estas nuevas generaciones fue GSM (*Global System for Mobile Communications*). Los protocolos empleados ofrecen velocidades más altas de voz, pero limitadas en la comunicación de datos. Fuera de Europa también se utilizó la tecnología CDMA en 2G [16].

Fue con la llegada de la tercera generación de móviles cuando CDMA cobró su mayor protagonismo. Gracias a esta tecnología los sistemas soportan velocidades más altas, acceso rápido a Internet y videoconferencia entre otros.

2.1 CDMA en UMTS.

El proceso de especificación del acceso radio de UMTS se llevó a cabo por ETSI (*European Telecommunications Standard Institute*) en 1997, con la creación de varios grupos de trabajo quienes decidieron utilizar CDMA como técnica de acceso radio [4].

El hecho de que varios usuarios del mismo sistema utilicen recursos compartidos, nos puede llevar a una situación de conflicto si dos o más usuarios transmiten al mismo tiempo y en la misma frecuencia sin tomar medidas especiales. Las técnicas de acceso múltiple han sido desarrolladas para resolver posibles interferencias entre usuarios y maximizar la capacidad del sistema, o lo que es lo mismo, que se pueda dar servicio al mayor número de usuarios posible con una determinada calidad [4]. Por lo tanto UMTS no opta por las clásicas técnicas mencionadas anteriormente FDMA y TDMA, sino que utiliza la técnica de CDMA con la que consigue mejores calidades de servicio, como se dijo en apartados anteriores.

Por lo tanto, en los siguientes apartados entraremos en detalle del funcionamiento de CDMA y de sus características.

2.2 Introducción a CDMA.

En CDMA a cada usuario se le asigna un código de secuencia único que usa para codificar su información. El receptor, que conoce los códigos asignados a cada usuario, decodifica la señal recibida y recupera la información inicial. El ancho de banda de la señal original se expande para su transmisión, resultando una señal de espectro ensanchado; por eso es por lo que a esta técnica también se la conoce como modulación de espectro ensanchado (*Spread Spectrum-SS*).

El origen exacto de las comunicaciones de espectro ensanchado es difícil de situar debido a que estas comunicaciones son el resultado de los avances en muchas

direcciones, tales como radares de alta resolución, orientación, filtros adaptados, evitar interferencias, teoría de la información y seguridad en las comunicaciones. Una de las principales características de la tecnología CDMA es el uso de señales de espectro ensanchado, lo cual, como veremos más adelante, tiene algunas ventajas como la seguridad en la comunicación. Esta última cualidad ya empezaba a ser usada con fines militares por Estados Unidos en el *Global Positioning System* (GPS) y el *Joint Tactical Information Distribution System* (JTIDS) [2].

Las modulaciones de espectro ensanchado fueron originalmente desarrolladas para su uso en radares militares y sistemas de comunicaciones debido a su capacidad contra interferencias de señales y a la baja probabilidad de detección. Solo recientemente, con nuevas y más baratas tecnologías emergentes, los investigadores están interesados en aplicar los equipos de espectro ensanchado en aplicaciones civiles.

Los sistemas de espectro ensanchado deben cumplir dos características fundamentales [3]:

1. El ancho de banda de transmisión debe ser mucho mayor que el ancho de banda de la señal que se desea enviar.
2. El ancho de banda utilizado es independiente de la señal de información.

Las modulaciones de espectro ensanchado transforman una señal de información en otra señal con un ancho de banda mucho mayor. Esta transformación se consigue al codificar la señal de información con un código que es independiente de los datos y tiene un ancho espectral mucho mayor que la señal de información. Estos códigos expanden la potencia de la señal original sobre todo el ancho de banda de la señal resultante, dando lugar a una densidad de potencia menor como se muestra en la Figura 1.

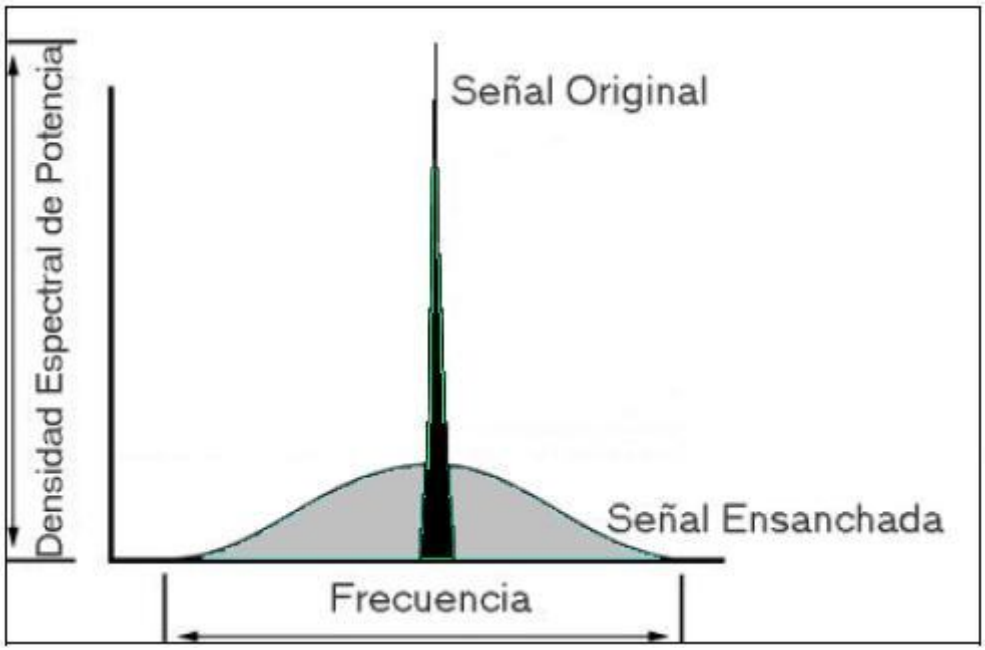


Figura 1. Comparación del ancho de banda de la señal original con la señal ensanchada.

La relación entre el ancho de banda de la señal transmitida y de la señal de información se llama ganancia G_p,

$$G_p = \frac{B_t}{B_i}$$

donde B_t es el ancho de banda de transmisión y B_i es el ancho de banda de la señal de información original.

El receptor correla la señal recibida con una copia del código, con el que ha sido modulada la señal original, generada de forma síncrona, y a continuación recupera la señal de información. Esto implica que el receptor debe conocer los códigos de los usuarios con los que se modelan los datos.

Debido a la codificación y al ancho de banda resultante, las señales de espectro ensanchado tienen las siguientes propiedades [3]:

1. ***Capacidad de acceso múltiple***. Si varios usuarios transmiten sus señales de espectro ensanchado al mismo tiempo, el receptor será capaz de distinguir entre los distintos usuarios, siempre y cuando conozca los códigos utilizados para ensanchar la señal de información. Correlando la señal recibida con el código de un cierto usuario, desensancharemos solo la señal de ese usuario, mientras que las otras señales de los demás usuarios seguirán estando ensanchadas, (Figura 2). De este modo, la potencia de la señal de dicho usuario será mucho mayor que la potencia del resto de señales interferentes, y la señal deseada podrá ser recuperada.

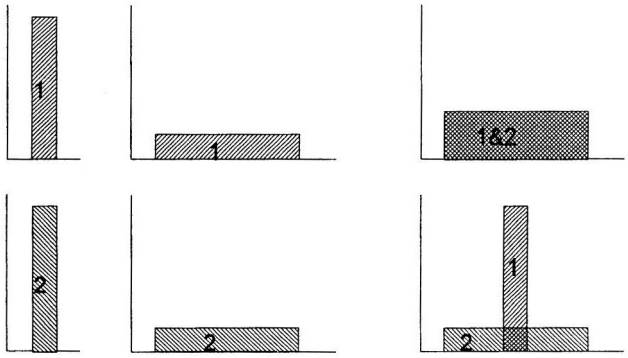


Figura 2. Principio de acceso múltiple en espectro ensanchado.

2. ***Protección contra interferencia multitrayecto***. En un canal de radio no existe solo un camino entre el transmisor y el receptor. Debido a la reflexión (y a la refracción) una señal puede ser recibida por distintos trayectos. Las señales de los diferentes trayectos son copias de la señal transmitida pero con amplitudes y fases diferentes. Las modulaciones de espectro ensanchado pueden combatir esta interferencia multitrayecto; esto dependerá mucho del tipo de modulación usada.
3. ***Privacidad***. La señal transmitida solo puede ser desensanchada y por lo tanto recuperada si el receptor conoce el código.

4. ***Rechazo de interferencias***. Correlando el código con una señal de banda estrecha, ensancharemos la potencia de la señal de banda estrecha por lo que reduciremos la potencia de la interferencia en el ancho de banda de información, Figura 3.

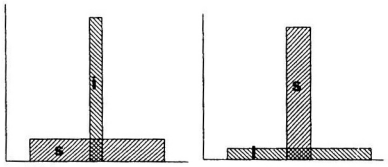


Figura 3. Rechazo de interferencias.

5. ***Capacidad contra interferencias***. El mismo concepto que rechazo de interferencias pero ahora las interferencias están completamente infligidas en el sistema.
6. ***Baja probabilidad de interceptación***. Debido a su baja densidad de potencia, las señales de espectro ensanchado son difíciles de detectar.

Existen varias técnicas de modulación que generan señales de espectro ensanchado como por ejemplo *Frequency Hopping (FH)* o *Time Hopping (TH)* [3]. Aunque la técnica más popular es *direct-sequence spread-spectrum* (DS-SS) [3].

2.3 Espectro ensanchado por secuencia directa (DS-SS *Direct-sequence spread-spectrum*)

En DS-CDMA la señal de información modulada es directamente modulada por un código digital. La señal de información puede ser analógica o digital., pero en la mayoría de los casos será digital. Lo que se ve frecuentemente en el caso de las señales digitales es que se omite la modulación de los datos y la señal se multiplica directamente por el código y la señal resultante modula la portadora. De esa multiplicación directa es de donde este protocolo toma su nombre.

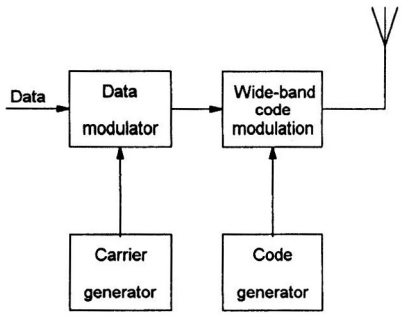


Figura 4. Diagrama de un trasmisor DS-CDMA

En la Figura 4 podemos ver el diagrama de un transmisor de DS-CDMA. La señal binaria de datos modula la portadora. La portadora modulada es posteriormente modulada por el código. Este código consiste en un número determinado de "*chips*" que pueden ser +1 o -1. Para obtener el ensanchamiento deseado de la señal, la tasa de chips del código debe ser mucho mayor que la tasa de bits de la señal de información[3].

Para la modulación del código se pueden usar varias técnicas, pero generalmente se utiliza alguna variación de PSK como la binaria BPSK, diferencial DBPSK o en cuadratura QPSK. Si omitimos la modulación de los datos y se usara una modulación BPSK para el código, obtendríamos el diagrama de la Figura 5:

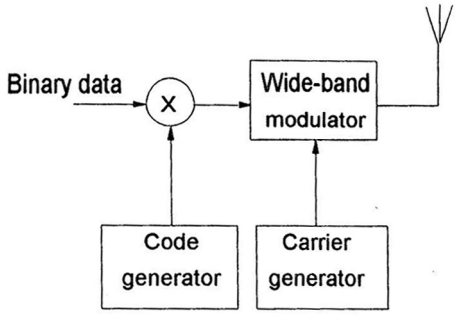


Figura 5. Diagrama modificado de un transmisor DS-CDMA

La señal DS-SS resultante se muestra en la Figura 6. En esta figura, el código está compuesto por 10 chips por cada bit de información.

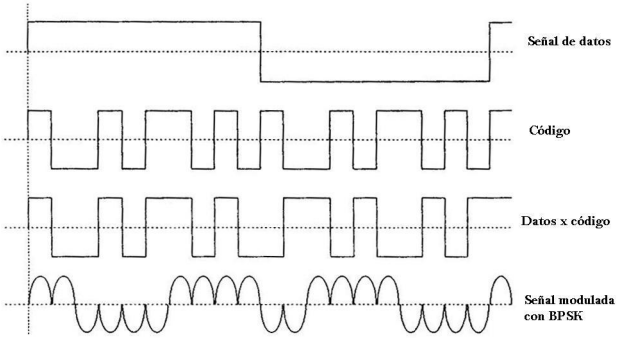


Figura 6. Generación de una señal de espectro ensanchado con BPSK.

La señal es generada mediante una multiplicación directa ente la señal de datos y el código del usuario:

$$s = i \times c$$

donde '*s*' es la señal generada, '*i*' la señal de información del usuario y '*c*' el código utilizado por ese usuario.

Después de la transmisión de la señal, el receptor usará demodulación coherente para desensanchar la señal expandida anteriormente, usando una secuencia del código generada localmente. Para llevar a cabo esta operación, el receptor debe, no solo saber la secuencia del código usada por el usuario, si no que además es necesario que la secuencia recibida y el código generado localmente por el receptor, estén sincronizados. Esta sincronización debe comenzar desde el principio de la recepción y debe ser mantenida hasta que la señal se ha recibido completamente. De esta forma el receptor

multiplicará la señal recibida por el código del usuario que desea obtener la información:

$$s\_rec = s\_in \times c$$

donde '*s_rec*' es la señal recuperada, '*s_in*' es la señal recibida por el receptor y '*c*' es el código del usuario.

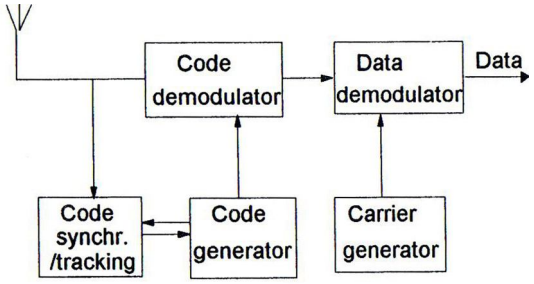


Figura 7. Diagrama de un receptor DS-CDMA

Los protocolos DS-CDMA tienen una serie de propiedades que podemos dividir en ventajas (+) y desventajas (-)[3]:

+ La generación de la señal codificada es fácil ya que se puede hacer con una simple multiplicación.
+ Solo se tiene que generar una frecuencia de portadora.
+ Es posible desensanchar la señal con una demodulación coherente.
+ No es necesaria la sincronización entre los usuarios.

- Es difícil de conseguir y mantener la sincronización del código generado localmente y de la señal recibida. Esta sincronización debe ser realizada con una precisión de la duración de un "*chip*".
- La potencia recibida de los usuarios que están más cerca del receptor será mucho mayor de los que están más lejos. Si un usuario alejado está continuamente enviando información y uno que está cerca del receptor también, este último esta interfiriendo en la comunicación haciéndola casi imposible para el usuario que está alejado. Este efecto "*near-far*" se puede resolver aplicando un algoritmo de control de potencia, de forma que el receptor reciba a todos los usuarios con la misma potencia media. De todas formas este control es muy complicado de realizar.

2.4 Generación de códigos para sistemas CDMA

Es importante en un protocolo CDMA que los códigos asignados a los usuarios hagan posible una buena separación entre la señal del usuario deseado y la señal de otros usuarios interferentes. Dado que la separación se hace mediante la correlación entre la señal recibida y la señal del código, generada localmente por el receptor, del usuario deseado, se puede afirmar que lo que realmente demanda un código para cumplir los requisitos que deseamos, es que exista una baja correlación cruzada entre los distintos códigos.

La autocorrelación de un código es también un aspecto muy importante, ya que nos dirá como de bueno es ese código en cuanto a sincronización entre la señal recibida

y la señal del código generada localmente en el receptor. También, la habilidad de un sistema CDMA de combatir los efectos de la interferencia multitrayecto, depende de la función de autocorrelación del código.

Así pues, se utilizarán las funciones de autocorrelación y correlación cruzada periódicas para juzgar si un código está cualificado o no para ser utilizado. En particular se utilizan las funciones de autocorrelación y correlación cruzada discretas que se pueden ver a continuación:

- Función de autocorrelación periódica:

$$\phi_{xx}(l) = \sum_{i=1}^{L} c_x(i) c_x[(i+l) \bmod L] \text{ para } 0 \le l \le L-1$$

- Función de correlación cruzada periódica:

$$\phi_{xy}(l) = \sum_{i=1}^{L} c_x(i) c_y[(i+L) \bmod L] \text{ para } 0 \le l \le L-1$$

donde,

- $c_x(i)$ es el i-ésimo chip del código ***x*** de longitud L.
- $c_y(i)$ es el i-ésimo chip del código ***y*** de longitud L.

La función de autocorrelación ideal de una secuencia binaria, sería una función delta centrada en el cero, Figura 8.

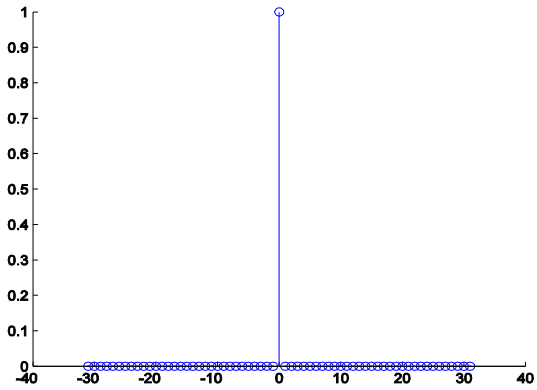


Figura 8. Función pulso centrada en cero.

Es decir, nos interesaría que un código se pareciera mucho más a sí mismo en el mismo instante de tiempo, que desplazado en el tiempo. En cuanto a la correlación cruzada debemos tener en cuenta que debe tener un valor lo más próximo a cero para cualquier valor de '*l*'. Además, la condición de que el valor de la correlación cruzada sea cero para *l*=0, es válida solo para cuando las señales recibidas estén alineadas en el tiempo, lo que solo ocurre en el caso *Downlink*, ya que en el *Uplink* las señales de los usuarios tendrán distintos caminos de propagación hacia el receptor base.

La propiedad de correlación es probablemente la característica más importante en el desarrollo de unos códigos adecuados para una comunicación CDMA. De todas formas, podemos pensar también en otra serie de interesantes propiedades que nos interesaría que tuvieran los códigos. Una lista completa de las propiedades deseables de estos códigos sería:

1. La señal del código debe ser fácil de generar.
2. Debe tener la deseada autocorrelación, correlación cruzada y equiprobabilidad entre 0's y 1's.
3. Largos períodos para obtener una autocorrelación satisfactoria.
4. Para tener transmisiones seguras, la señal del código debe ser difícil de reconstruir a partir de un segmento pequeño.

Existen muchas formas de generar estas secuencias de código [3]. Una de estas es usar registros desplazados y retroalimentados, Figura 9. Un registro de desplazamiento consiste en un número de celdas (numeradas de 1 a 'r') donde cada celda es una unidad de almacenamiento la cual, bajo el control de un pulso de reloj, mueve el contenido hacia su salida mientras que almacena un nuevo dato obteniéndolo de su entrada. En una configuración normal de un FSR (*Feedback Shift Register*), la entrada de la celda 'm' es una función de la salida de la celda 'm-1' y la salida de la celda 'r'.

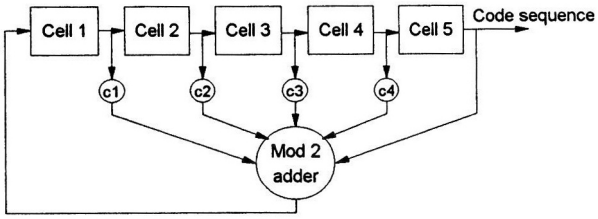


Figura 9. Diagrama de bloques de un FSR lineal simple.

En UMTS, para la transmisión de la señal se utiliza una doble codificación: cada bit se multiplica por el código ortogonal expansor o código de canalización asignado al canal físico correspondiente a ese bit. El resultado es la señal expandida a 3.840 Kchip/s. Esta señal se multiplica, a su vez, a nivel de chip, por un código de aleatorización SC (*Scrambling Code*) que ya no produce expansión [10].

Las siguientes características resumen perfectamente qué buscamos en unos buenos códigos de aleatorización [3]:

1. Buenas características de autocorrelación: la función deberá tener un máximo agudo con lóbulos laterales de pequeña amplitud para conseguir una sincronización inicial fiable. Si esos lóbulos laterales son altos, puede haber decisiones de sincronización erróneas y dificultades para separar y resolver las componentes de propagación multitrayecto.

2. Correlación cruzada pequeña para reducir la interferencia del multiacceso.

3. Elevada cuasi-aleatoriedad, de forma que la señal resultante de la aplicación del SC se asemeje a un ruido blanco gaussiano.

Las características 1 y 2 no pueden darse simultáneamente por lo que la búsqueda de códigos SC debe procurar alcanzar un compromiso razonable entre ambas [3].

En el enlace ascendente estos últimos códigos, establecen la distinción, en el Nodo B, entre los diferentes usuarios puesto que cada uno tiene asignado su propio SC.

A continuación vamos a profundizar en cada uno de los códigos utilizados. Veremos como se generan y analizaremos las ventajas y desventajas que tienen.

2.4.1 Códigos GOLD

Para entender de dónde salen los códigos Gold debemos tener primero presente los tipos de FSR que existen.

Antes dijimos cómo era la configuración normal de un FSR. Pues bien, la función que combina las salidas de las celdas ‘m-1’ y ‘r’ puede ser una función lineal o no lineal.

La ventaja que tienen los no lineales es que son más difíciles de detectar por usuarios no deseados. Pero por otro lado, poseen muchas desventajas que caracterizan a lo no lineal. Primero, no se conoce mucho acerca de las propiedades de los FSRs no lineales, y, segundo, por que muchos de estos registros no pueden generar un ciclo. Por estas razones son mucho más usados en CDMA los FSRs lineales.

Dentro de los FSR lineales, uno muy popular es el ML-FSR (*Maximum Length*). Un FSR es ML si produce la secuencia posible más larga de 2^{r-1} (siendo ‘r’ el número de celdas del registro) antes de que se empiece a repetir la misma secuencia otra vez, es decir, el código empieza a repetirse cuando ya ha alcanzado la longitud máxima posible. La popularidad de estas secuencias es el resultado de la buena autocorrelación y correlación cruzada que poseen[3].

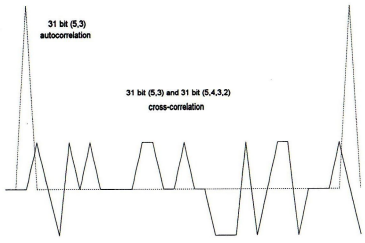


Figura 10. Autocorrelación y correlación cruzada de un código ML

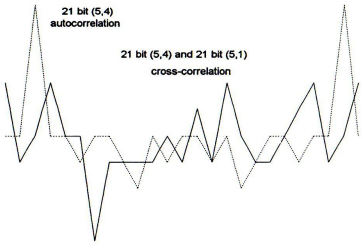


Figura 11. Autocorrelación y correlación cruzada de un código no ML.

Pero los códigos ML tienen la desventaja de que su número es limitado. Esta es una de las razones por la que se buscan códigos no ML que tengan un comportamiento satisfactorio en cuanto a correlación. Una clase de estos códigos es los *códigos Gold.* Estos códigos son generados a partir de la salida de dos códigos ML diferentes, Figura 12. Como podemos ver en las dos gráficas anteriores las propiedades de autocorrelación son cercanas, ver Figura 10 y Figura 11.

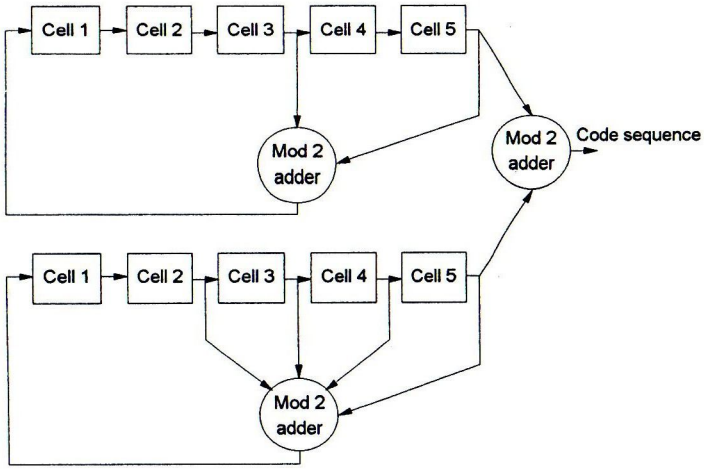


Figura 12. Generador de códigos GOLD usando dos generadores de códigos ML.

Los FSR lineales tienen la mayoría de las propiedades deseadas para nuestros códigos. Son fáciles de generar y tienen unas propiedades de aleatoriedad razonables. Los códigos ML en particular tienen largos periodos.

Debido a que la elección de unos códigos adecuados es crítico en un sistema CDMA, sigue habiendo muchos estudios en el desarrollo de nuevos códigos. Por el momento los códigos GOLD son unos de los más usados y más populares.

Los códigos ortogonales de expansión utilizados en UMTS pertenecen a una familia de códigos de factor de expansión variable OVSF, los cuales trataremos a continuación.

2.4.2 Códigos ortogonales.

Los códigos **Walsh** son unos de los códigos ortogonales más comunes usados en aplicaciones CDMA [14]. Estos códigos corresponden a las columnas de una matriz cuadrada especial conocida como la matriz de *Hadamard*. Para un conjunto de códigos

Walsh de longitud ***'n'***, tendremos una matriz cuadrada de ***'n'*** códigos *Walsh* de forma que tenemos una matriz ***'n x n'***. La primera columna de esta matriz contiene una serie de 1's y las siguiente columnas contienen una serie de -1's y 1's combinados. Cada columna es ortogonal entre sí y tienen la misma aparición de los bits binarios. Esta matriz se define recursivamente de la siguiente manera:

$$W_1 = [1] \qquad W_{2n} = \begin{bmatrix} W_n & W_n \\ W_n & \overline{W_n} \end{bmatrix}$$

donde ***'n'*** es una potencia de 2 que indica la dimensión de la matriz y $\overline{W_n}$ hace referencia al operador lógico NOT en todos los bits de esa matriz. Veamos por ejemplo las matrices W_2 , W_4 y W_8, que muestran la matriz para dimensiones 2,4 y 8:

$$W_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \quad W_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

Se puede ver con facilidad que todas las filas y columnas son mutuamente ortogonales. Las siguientes propiedades se pueden derivar si se define la secuencia de *Walsh* W_i como la i-ésima fila o columna de una matriz de *Hadamard* [14]:

- Las secuencias de *Walsh* son secuencias binarias con valores de +1 y -1.
- La longitud de las secuencias de *Walsh* son siempre potencia de 2.
- Siempre hay 'L' secuencias diferentes de longitud 'L'.
- Las secuencias de *Walsh* son mutuamente ortogonales si están sincronizadas, es decir, $\phi_{xy}(l = 0) = 0$
- Si dos secuencias de *Walsh* tienen desplazamientos en el tiempo, la función de correlación cruzada puede tomar valores mayores que el pico de la función de autocorrelación, el cual es igual a la longitud 'L' de la secuencia. Aunque también es posible que la función de correlación cruzada tome un valor de cero incluso cuando existe cualquier desplazamiento en el tiempo.
- Todas las secuencias de *Walsh* empiezan por +1.

Otro método para generar códigos ortogonales es utilizando estructuras de árbol con un factor de ensanchado variable (**OVSF**):

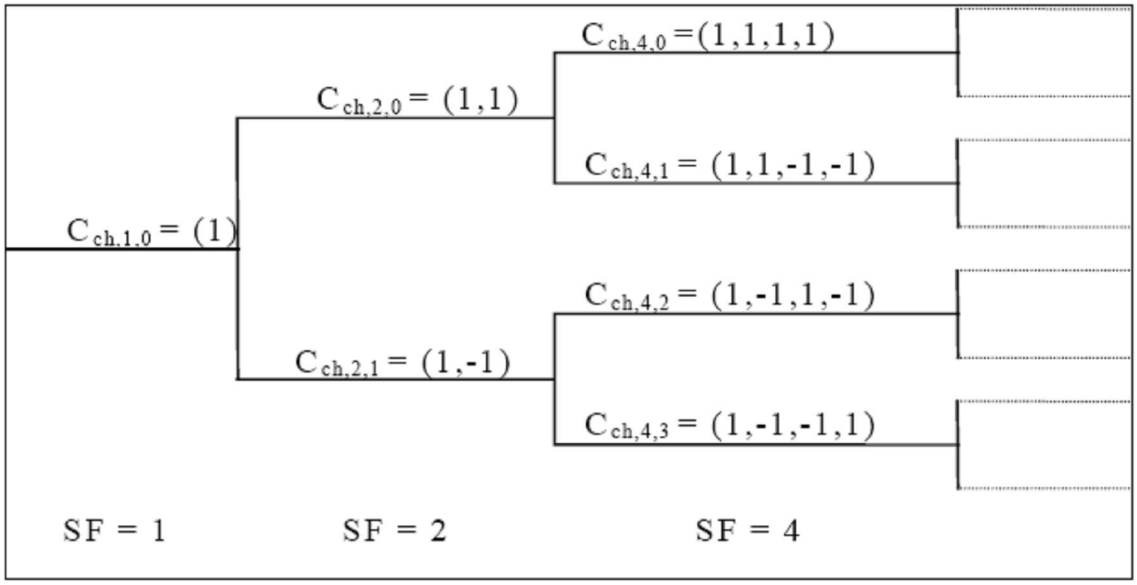


Figura 13. Generación de códigos ortogonales OVSF mediante estructuras en árbol.

La generación de esta estructura de árbol, tal y como se muestra en la Figura 13 se realiza mediante el siguiente proceso recursivo:

$$C_{2n} = \begin{bmatrix} C_{2n,1} \\ C_{2n,2} \\ \cdots \\ C_{2n,2n} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} C_{n,1} & C_{n,1} \\ C_{n,1} & -C_{n,1} \end{bmatrix} \\ \cdots \\ \begin{bmatrix} C_{n,n} & C_{n,n} \\ C_{n,n} & -C_{n,n} \end{bmatrix} \end{bmatrix}$$

donde C_{2n} es un conjunto de códigos ortogonales de tamaño '2n' y SF es el "*spreading factor*" o factor de ensanchamiento. El valor más a la izquierda en cada código es el transmitido en primer lugar [8]. Las propiedades de ortogonalidad de estos códigos son similares a las de los códigos Walsh. El orden de las funciones de la matriz no es el mismo que el de la matriz de *Hadamard*, pero las funciones en sí son las mismas [14]. Las secuencias pertenecientes a la misma rama forman un conjunto de códigos ortogonales, es más, dos secuencias cualesquiera de diferentes ramas son ortogonales excepto si una secuencia es la madre de la otra.

En estos códigos ortogonales, si dos secuencias llegaran al mismo tiempo al receptor, la interferencia que se causarían entre dos usuarios sería nula. Lo que ocurre en el enlace ascendente, es que es muy difícil que los terminales emitan a la vez y los códigos estén alineados en el tiempo. Si existiera algún desplazamiento en el tiempo, las propiedades de correlación cruzada empeorarían mucho, por lo que resultaría poco aconsejable su utilización en el enlace ascendente.

En muchas fuentes podemos encontrar que hablan de los códigos Walsh y de los códigos OVSF por separado, pero si comparásemos los códigos OVSF con los *Walsh-Hadamard*, veríamos que contienen las mismas secuencias de código. La única diferencia entre estos códigos es como están indexados, es decir, el orden en el que se crean las secuencias [14].

3 Propagación de señales de espectro ensanchado

El ruido es una perturbación eléctrica que impone un límite a la calidad de funcionamiento de un sistema radioeléctrico [10]. Las fuentes de ruido pueden ser externas debidas a la radiación producida por elementos naturales como la tierra o el cielo. Las fuentes internas de ruido residen en los elementos de conexión de la antena al receptor y en el propio receptor [10].

En los enlaces zonales, como son los de comunicaciones móviles, las ondas que llegan a las diferentes posiciones en que puede situarse el receptor encuentran distintas condiciones de propagación en su camino [10]. Como la señal recibida por el móvil es el resultado de la suma de componentes que se propagan por múltiples trayectos, las comunicaciones móviles se caracterizan por amplias variaciones del campo eléctrico y de la potencia de recepción en función del espacio (variaciones con la ubicación del receptor) y del tiempo (variaciones temporales).

El impacto de un canal móvil sobre la forma de onda de una señal CDMA, es fácilmente entendido considerando la función de correlación después de la transmisión a través de un escenario similar al de la siguiente figura [9]:

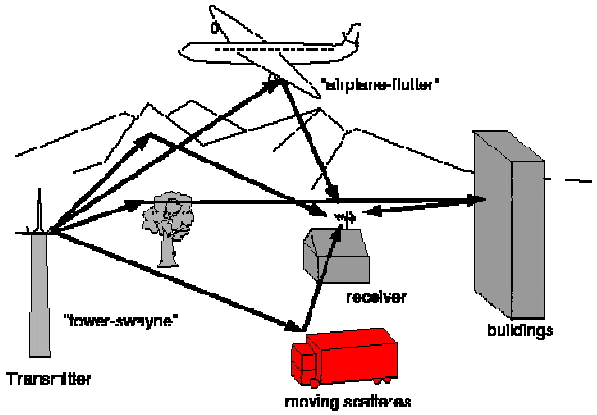


Figura 14. Escenario celular multitrayecto.

En CDMA deben afrontarse y resolverse dos problemas fundamentales y mutuamente relacionados [9]:

1. Separabilidad de las recepciones mediante los códigos de los usuarios.
2. Detección de las señales en un entorno de elevada interferencia.

A continuación introducimos las características de los dos tipos de canales que vamos a tratar: el canal Gaussiano y el canal *Rayleigh*.

3.1 El canal Gaussiano

El canal Gaussiano puede ser considerado el canal ideal, y está solo afectado por un ruido aditivo, blanco y Gaussiano (AWGN), el cual es procesado internamente por el receptor [11]. Siempre se espera conseguir una buena tasa de error en un canal Gaussiano después de haber hecho todo lo posible para mitigar los desvanecimientos; esto puede incluir diversidad o ecualización [11]. El canal Gaussiano ideal es muy

difícil de conseguir en un entorno de radiocomunicaciones. Este canal se ajusta cuando las condiciones de desvanecimientos son lentas [11].

3.2 El canal Rayleigh

El canal Rayleigh se encuentra en el extremo opuesto del anterior canal, ya que nos referimos a un canal en el que nos pondríamos en el peor de los casos. En radiocomunicaciones se utiliza la distribución Rayleigh para describir la variación estadística de la envolvente de la señal resultante de la propagación multitrayecto, la cual, experimenta desvanecimientos rápidos, cuando los diferentes rayos tienen amplitudes similares y fases aleatorias [10].

En la Figura 15, se muestra las características de desvanecimiento aproximadas de un canal Rayleigh.

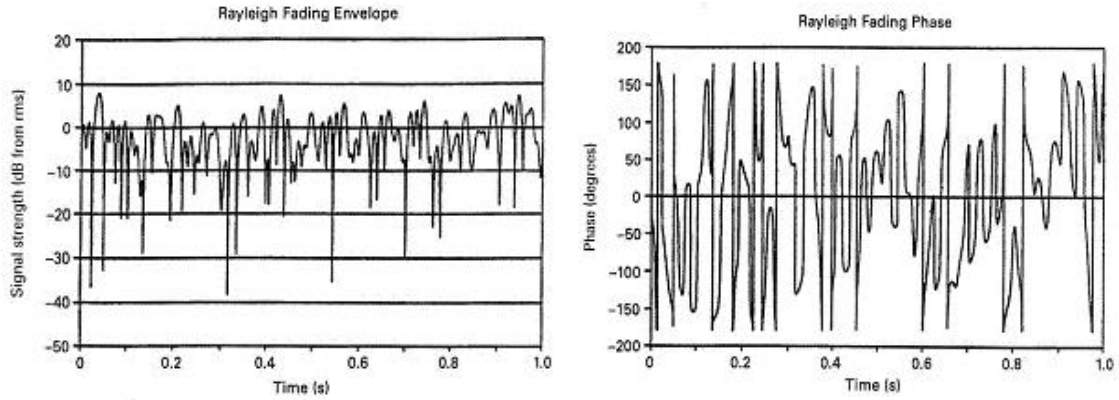


Figura 15. "Típica" envolvente y fase de un desvanecimiento Rayleigh en un escenario móvil.

Los canales pueden ser caracterizados por una función 'K', la cual se define de la siguiente manera:

$$K = \frac{potencia\_en\_el\_trayecto\_directo}{potencia\_en\_los\_trayectos\_atenuados}$$

En una celda más pequeña, las componentes con LoS (línea de vista) son más dominantes. Existen muchos casos en los que existe una componente directa y otras atenuadas lo que sería un escenario típico de multitrayecto. Volviendo a la ecuación anterior, cuando K=0, el canal es *Rayleigh* (el numerador es 0 y toda la energía recibida viene de trayectos atenuados)[11]. Cuando K= ∞, el canal es Gaussiano y el denominador es cero.

En la Figura 16 [11], encontramos la tasa de error de bit para algunos valores típicos de 'K'. Los valores intermedios de 'K' corresponden a un canal con una distribución de *Rice*. Para un escenario móvil los valores de 'K' suelen variar entre 5 y 30 [11]. Para celdas grandes se tiende a valores menores de 'K'.

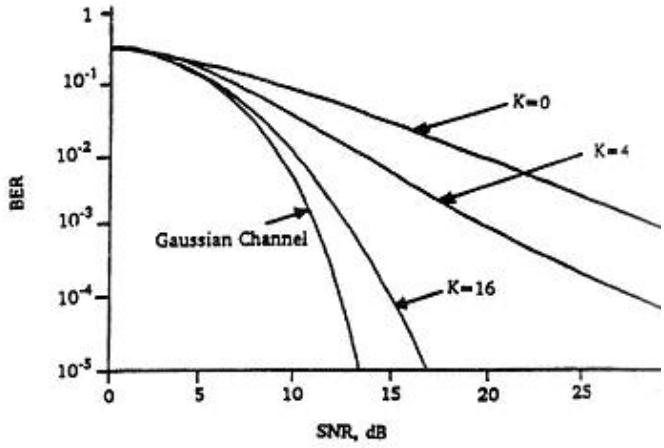


Figura 16. BER teórica para varios valores de K, y modulación -FSK

4 Comunicación cooperativa

El fenómeno fundamental que hace difícil la fiabilidad de la transmisión inalámbrica es la atenuación multitrayecto [5]. Este es el fenómeno que produce la principal diferencia en comparación a la fibra, cable coaxial, guía de ondas o incluso transmisiones de satélites.

Teóricamente, la técnica más efectiva para mitigar la atenuación multitrayecto en un canal inalámbrico es el control de potencia en el transmisor [6]. Si las condiciones del canal que experimenta el receptor en un lado de la comunicación son conocidas por el transmisor del otro lado, el transmisor puede '*predistorsionar*' la señal para superar el efecto del canal en el receptor. Existen dos problemas fundamentales con esta aproximación [6]. Para que el transmisor pueda superar un cierto nivel de atenuación, debe incrementar su potencia al mismo nivel, lo cual en muchos casos no es práctico debido a las limitaciones en la potencia de radiación y al tamaño y coste de los amplificadores.

El segundo problema es que el transmisor no tiene ningún conocimiento de lo que experimenta el receptor del canal, a excepción de sistemas en los que el sentido *uplink* y *downlink* utilizan la misma frecuencia. De este modo, la información del canal tiene que ser realimentada desde el receptor al transmisor, lo que provoca una gran complejidad en el sistema. Además, en muchos casos no existirá un enlace de realimentación entre el receptor y el transmisor [6].

Otras técnicas efectivas son la diversidad en tiempo y frecuencia [6]. La división temporal, junto con un código de corrección de errores, puede proveer mejoras en la diversidad, aunque producirá grandes retrasos si el canal varía lentamente.

En estos escenarios inalámbricos, la aplicación de la diversidad en las antenas, es un método efectivo y, por lo tanto, es una técnica usada para reducir el efecto de atenuación multitrayecto [6].

4.1 Introducción a la comunicación cooperativa

Al introducir el concepto de comunicación cooperativa en las comunicaciones inalámbricas, generalmente se suele pensar en el uso de varias antenas colocadas en el transmisor. De cualquier forma, muchos dispositivos inalámbricos están limitados a una sola antena ya sea por tamaño, por precio o por la complejidad del *hardware* [5].

En comunicación cooperativa existen diferentes métodos. Estos métodos proponen que los sistemas inalámbricos con una antena en un escenario multiusuario puedan compartir sus antenas, y de este modo, generar un escenario virtual de transmisores con varias antenas [5]. Para una primera explicación de las ideas que hay detrás de este concepto hacemos referencia a la Figura 17. Esta figura muestra dos móviles comunicándose hacia el mismo destino. Cada móvil tiene una antena y no puede generar individualmente una diversidad espacial. En este caso uno de los móviles puede actuar como receptor de la información del otro móvil, y de esta forma, puede reenviar esa información junto con sus propios datos. Debido a que los caminos de los dos móviles son independientes se genera una diversidad espacial [5].

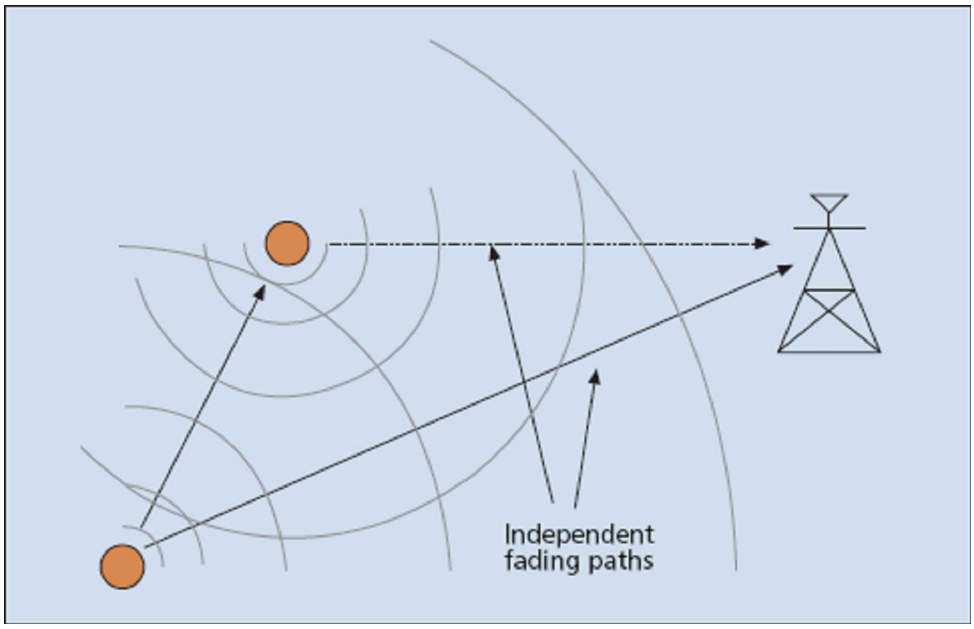


Figura 17. Comunicación cooperativa.

En comunicaciones inalámbricas cooperativas, refiriéndonos a una red inalámbrica, los usuarios pueden incrementar la calidad de servicio (medido en la capa física por la tasa de error de bit o probabilidad de pérdida) mediante la cooperación [5].

En un sistema de comunicaciones cooperativo, como podemos ver en la Figura 18, cada usuario inalámbrico transmite sus datos, a la vez que actúa como un agente de cooperación con otro usuario.

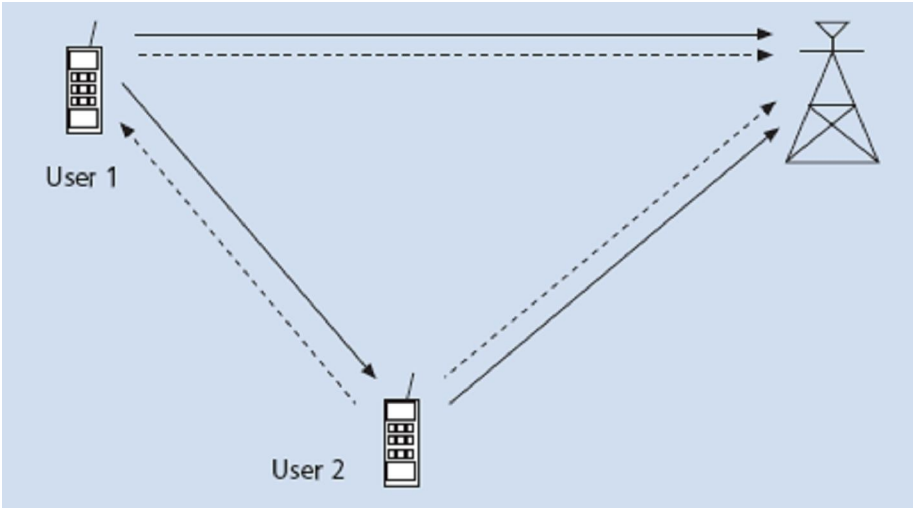


Figura 18. Los usuarios actúan también de repetidores en este caso.

La cooperación nos conduce a una interesante compensación en cuanto a la tasa de los códigos y a la potencia transmitida [5]. En el caso de la potencia, se puede discutir por un lado que un usuario necesitará mas potencia debido a que, cuando se encuentre en modo cooperativo, transmitirá por ambos usuarios. Pero por otro lado, la potencia base transmitida por ambos usuarios se verá reducida debido a la diversidad [5].

De un modo parecido ocurre con la tasa de bits del sistema. En modo cooperativo cada usuario transmite tanto sus propios bits como información de su compañero, por lo que se podría pensar que se produce una pérdida de tasa de bits en el sistema. De cualquier forma, la eficiencia espectral de cada usuario mejora porque, debido a la diversidad cooperativa, la tasa de codificación del canal se puede mejorar [5]. De nuevo se observa una compensación. La pregunta clave es si merece la pena la cooperación a pesar de los "costes" que hay que asumir. La respuesta ha sido contestada positivamente por muchos estudios [5].

A continuación analizaremos algunos de los más importantes métodos de cooperación [5].

4.2 Métodos *Decode and Forward*.

Este tipo de métodos es quizás el que más se aproxima a la idea de un repetidor tradicional. En este caso un usuario intenta detectar los bits del compañero y después retransmitirlos, (Figura 19). Como se puede observar en ese esquema, hay dos usuarios que cooperan entre sí.

Supongamos como caso particular que cada usuario posee su propio código de ensanchado, que lo llamaremos $c_1(t)$ y $c_2(t)$. Los bits de los dos usuarios se denotan como $b_i^{(n)}$ donde i = 1, 2 son los índices de los usuarios y *n* denota el índice del tiempo de los bits de información. Los factores $a_{i,j}$ denotan las amplitudes de las señales, de ahí que represente la asignación de la potencia a las distintas partes de la señal. Supongamos que cada período de señal consiste en tres intervalos de bits. Denotamos la señal del usuario 1 como $X_1(t)$ y la señal del usuario 2 $X_2(t)$,

$$X_1(t) = [a_{11}b_1^{(1)}c_1(t), a_{12}b_1^{(2)}c_1(t), a_{13}b_1^{(2)}c_1(t) + a_{14}B_2^{(2)}c_2(t)]$$

$$X_2(t) = [a_{21}b_2^{(1)}c_2(t), a_{22}b_2^{(2)}c_2(t), a_{23}B_1^{(2)}c_1(t) + a_{24}b_2^{(2)}c_2(t)]$$

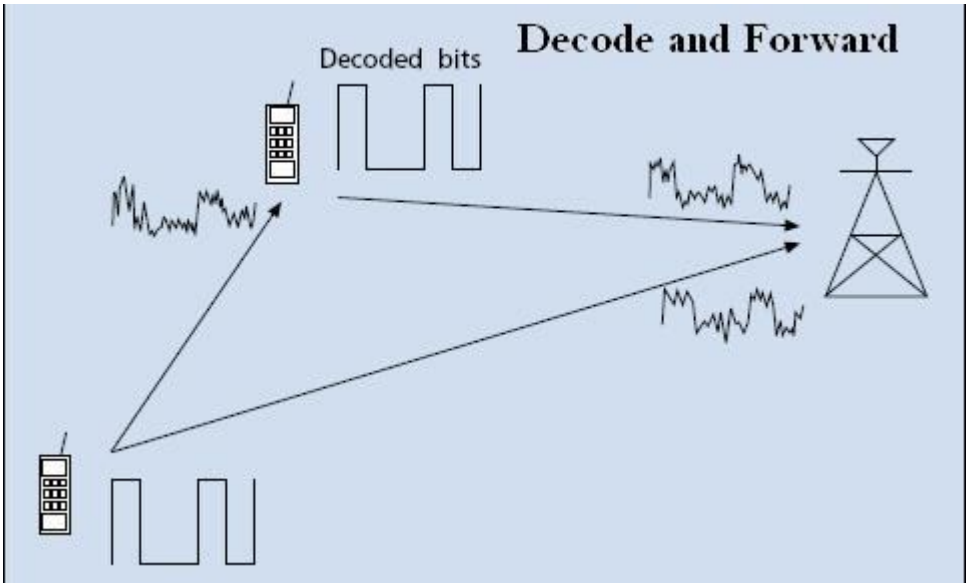


Figura 19. El usuario decodifica la señal, y lo retransmite hacia el receptor.

En otras palabras, en el primer y segundo intervalo cada usuario transmite sus propios bits. Cada usuario después detecta los segundos bits del otro usuario (la estimación del bit del otro usuario se denota como B_i). En el tercer intervalo, ambos usuarios transmiten una combinación lineal de su propio segundo bit y del segundo bit del compañero, cada uno multiplicado por su correspondiente código de ensanchado. Es decir, el usuario 1 retransmitirá la información del usuario 2 codificando los datos con el código de ensanchado del usuario 2, y viceversa.

4.3 Métodos *Amplify and Forward*

Otro de los métodos para establecer una comunicación cooperativa es el método *Amplify and Forward*. En este método, cada usuario recibe una versión con ruido de la señal transmitida por su compañero. Tal y como el nombre implica, el usuario después amplifica y retransmite la señal ruidosa (Figura 20). El receptor base combinará la información enviada por el usuario y su compañero, y tomará una decisión sobre el bit transmitido.

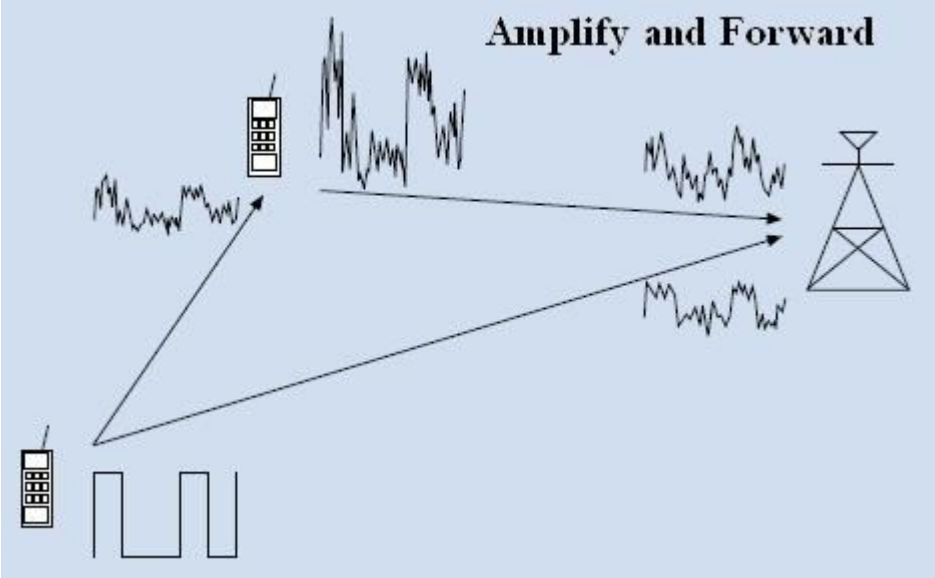


Figura 20. Ejemplo del método de Amplify and Forward.

Aunque el ruido es amplificado mediante la cooperación, el receptor base recibe dos versiones de la señal independientemente atenuadas y podrá tomar mejores decisiones en la detección de la información [5].

4.4 Alamouti

En primer lugar explicaremos el modelo MRRC (*Maximal-Ratio Receive Combining*) en el receptor para poder entender más adelante el nuevo modelo de transmisión diversificada.

4.4.1 *Maximal-Ratio Receive Combining.*

Supongamos que el transmisor envía una señal s_0. El canal, incluyendo el esquema del transmisor y del receptor, puede ser modelado como una distorsión

compuesta de una respuesta en amplitud y una respuesta en fase [6]. El canal entre la antena del transmisor y la antena 0 del receptor lo denotamos como h_0 y entre la antena del transmisor y la antena 1 del receptor como h_1 donde,

$$h_0 = \alpha_0 e^{j\theta_0}$$
$$h_1 = \alpha_1 e^{j\theta_1} \quad (1)$$

El ruido y las interferencias se añaden en las dos antenas receptoras. Las señales resultantes recibidas son:

$$r_0 = h_0 s_0 + n_0$$
$$r_1 = h_1 s_0 + n_1 \quad (2)$$

Donde n_0 y n_1 representan el ruido complejo y la interferencia.

Asumiendo n_0 y n_1 distribuidas como una Gaussiana, la regla de decisión de máxima verosimilitud en el receptor para estas señales recibidas es elegir la señal s_i si y solo si,

$$d^2(r_0, h_0 s_i) + d^2(r_1, h_1 s_i) \le d^2(r_0, h_0 s_k) + d^2(r_1, h_1 s_k) \qquad \forall i \ne k \quad (3)$$

donde $d^2(x, y)$ es la distancia Euclidea entre las señales 'x' e 'y' calculadas por la siguiente expresión:

$$d^2(x, y) = (x - y)(x^* - y^*) \quad (4)$$

Entonces para un sistema MRRC de dos antenas receptoras el esquema de la Figura 21 queda de la siguiente forma [6]:

$$S_0 = h_0^* r_0 + h_1^* r_1 = h_0^*(h_0 s_0 + n_0) + h_1^*(h_1 s_0 + n_1) = (\alpha_0^2 + \alpha_1^2) s_0 + h_0^* n_0 + h_1^* n_1 \quad (5)$$

Desarrollando (3) y usando (4) y (5), y simplificando llegamos a:

Elegir s_i si y solo si,

$$d^2(S_0, s_i) \le d^2(S_0, s_k) \qquad \forall i \ne k \quad (6)$$

El sistema puede entonces construir la señal S_0 usando un detector de máxima verosimilitud.

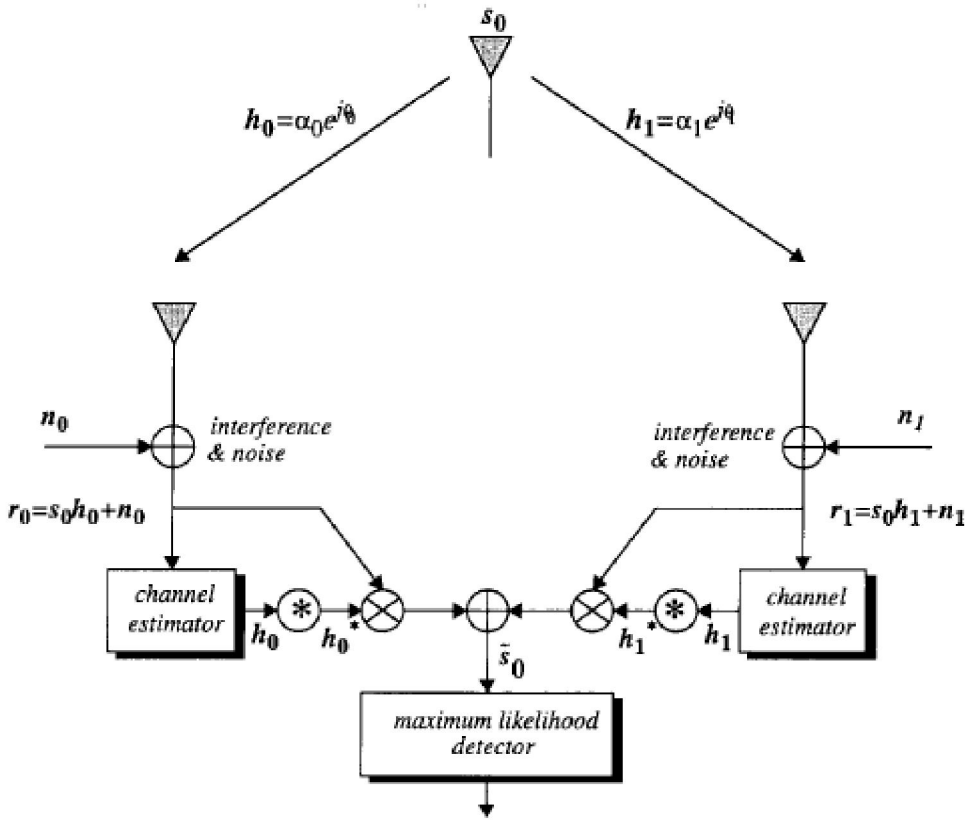


Figura 21. Esquema MRRC con dos antenas receptoras.

4.4.2 Un nuevo esquema de diversidad en la transmisión.

El nuevo esquema (Figura 22) usa dos antenas transmisoras y una antena receptora y puede ser definido por las siguientes tres funciones [6]:

- la codificación y la secuencia de transmisión de los símbolos de información en el transmisor.
- El esquema combinado en el receptor.
- La regla de decisión de máxima verosimilitud.

1. ***La codificación y la secuencia de transmisión****:* en un período de símbolo dado, dos señales se transmiten simultáneamente desde dos antenas. La señal transmitida por la antena 0 se denota como s_0 y desde la antena 1 s_1. Durante el siguiente período de símbolo, la señal $(-s_1^*)$ se transmite desde la antena 0, y la señal (s_0^*) desde la antena 1, donde * es el conjugado complejo. Esta operación se muestra en la siguiente tabla.

----------	Antena **0**	Antena **1**
Time **t**	s_0	s_1
Time **t + T**	$(-s_1^*)$	(s_0^*)

Tabla 1. Símbolo que se envía en función del *time slot* y el usuario.

El canal durante el instante **t** es modelado por una distorsión multiplicativa compleja $h_0(t)$ para el transmisor 0 y $h_1(t)$ para el transmisor 1. Asumiendo que la atenuación es constante durante los símbolos consecutivos, podemos escribir,

$$h_0(t) = h_0(t+T) = h_0 = \alpha_0 e^{j\theta_0}$$
$$h_1(t) = h_1(t+T) = h_1 = \alpha_1 e^{j\theta_1} \quad (7)$$

donde **T** es la duración del símbolo. La señal recibida puede ser expresada como,

$$r_0 = r(t) = h_0 s_0 + h_1 s_1 + n_0$$
$$r_1 = r(t+T) = -h_0 s_1^* + h_1 s_0^* + n_1 \quad (8)$$

donde r_0 y r_1 son las señales recibidas en los instantes **t** y **t + T** y n_0 y n_1 son variables aleatorias complejas que representan el ruido y la interferencia.

2. ***El esquema combinado:*** El combinador mostrado en la Figura 22 construye las siguientes dos señales que son enviadas al detector de máxima verosimilitud;

$$S_0 = h_0^* r_0 + h_1 r_1^*$$
$$S_1 = h_1^* r_0 - h_0 r_1^* \quad (9)$$

Es importante notar que este esquema es distinto del MRRC en (5). Sustituyendo (7) y (8) en (9) tenemos,

$$S_0 = (\alpha_0^2 + \alpha_1^2) s_0 + h_0^* n_0 + h_1 n_1^*$$
$$S_1 = (\alpha_0^2 + \alpha_1^2) s_1 - h_0 n_1^* + h_1^* n_0 \quad (10)$$

3. ***La regla de decisión de máxima verosimilitud:*** Estas señales combinadas son después enviadas al detector de máxima verosimilitud el cual, para cada una de las señales s_0 y s_1, usa la regla de decisión expresada en (9) para señales PSK.

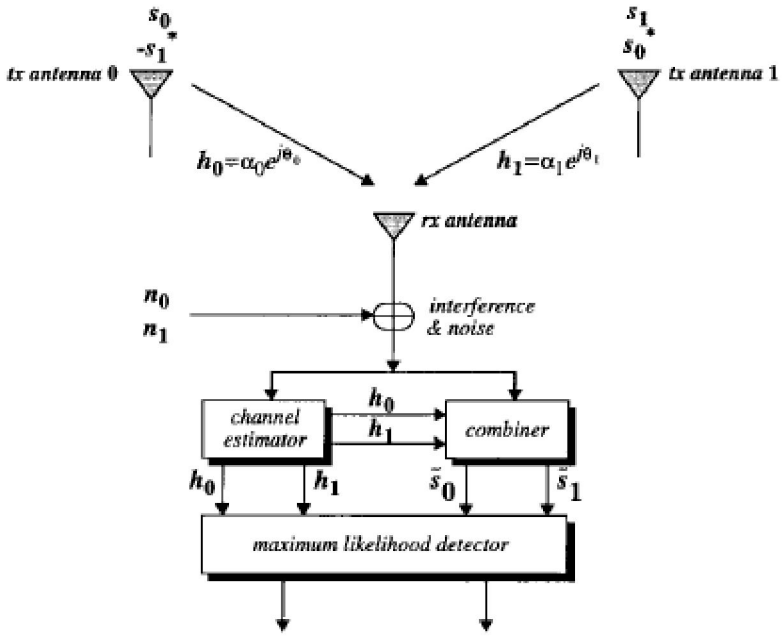


Figura 22. Un nuevo esquema de diversidad en transmisión con un receptor.

5 Análisis práctico

En esta parte vamos a analizar los conceptos desarrollados anteriormente. Hemos desarrollado una serie de escenarios los cuales nos permitirán conocer las prestaciones de los distintos códigos usados en CDMA y de las técnicas de comunicación cooperativa anteriormente mencionadas. Para la realización de este análisis se ha utilizado el programa MATLAB®.

MATLAB® es un programa de cálculo numérico orientado a matrices. Por tanto, los programas son más eficientes si se diseñan los algoritmos en términos de matrices y vectores.

Los escenarios que se han simulado son los siguientes:

- Sistema CDMA sin cooperación con N usuarios y un receptor.
- Sistema CDMA con el método de diversidad cooperativa *Amplify and Forward*, con diferentes factores de amplificación.
- Sistema CDMA con el método de diversidad cooperativa *Decode and Forward*.
- Sistema CDMA con el método de diversidad cooperativa *Alamouti*.

Todos los escenarios propuestos van a ser analizados bajo diferentes condiciones. Vamos a utilizar dos modulaciones diferentes: BPSK y QPSK, que son dos modulaciones típicas usadas en los sistemas CDMA [3]. Pondremos a prueba dos códigos distintos: secuencias Gold y códigos OVSF. Como ya mencionamos en otro apartado, trataremos los códigos OVSF y los códigos *Walsh* por igual [14]. Y por último vamos a tener en cuenta dos posibles entornos; trabajaremos con el canal de ruido aditivo AWGN y con el canal de ruido multiplicativo *Rayleigh*.

Para la explicación de las herramientas programadas para analizar los escenarios propuestos, vamos a dividir el sistema CDMA en 3 grupos como veremos más adelante: Transmisión, Canal y Recepción.

Se explicarán a continuación los escenarios y las aplicaciones; además para cualquier duda del código de las aplicaciones, al final se encuentra un anexo (Anexo i) con todos los códigos explicados.

5.1 Escenarios sin diversidad cooperativa.

A continuación analizaremos el escenario en el que no se ha utilizado diversidad cooperativa. En él simularemos un sistema CDMA con varios usuarios que transmiten su información al mismo tiempo, y un receptor.

5.1.1 Escenario 1: Análisis de las prestaciones de códigos CDMA sin utilizar cooperación.

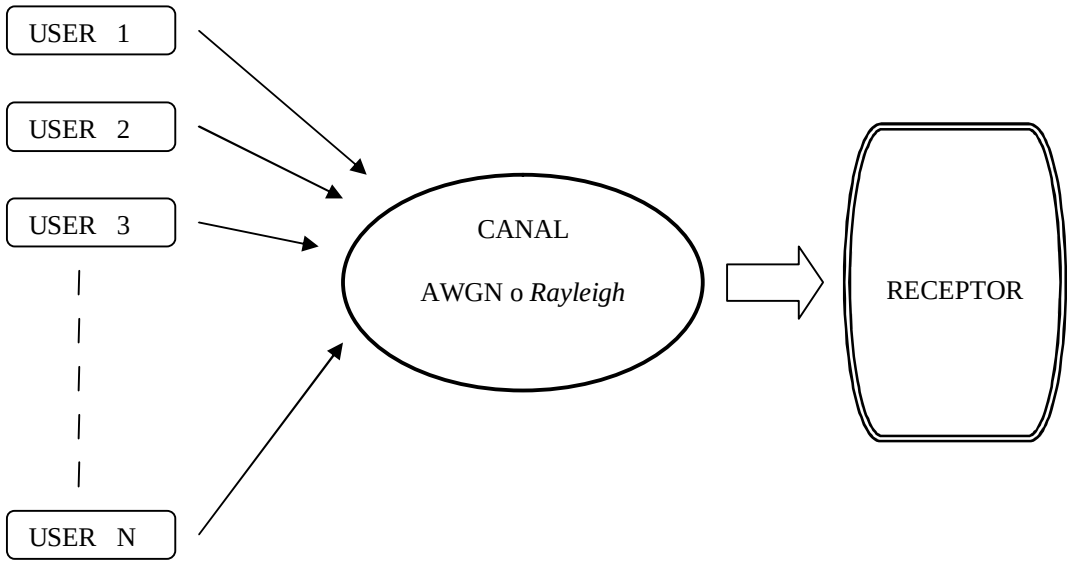


Figura 23. Escenario CDMA para N usuarios y un receptor.

En la Figura 23 se muestra el primer escenario que vamos a estudiar. Para ello hemos programado una aplicación en MATLAB® *cdma.m* la cual nos va a analizar la probabilidad de error del escenario teniendo en cuenta varias variables que podemos cambiar. Veremos la probabilidad de error en función de la SNR del canal, del número de usuarios que emite su información o por ejemplo del tipo del canal utilizado.

5.1.1.1 Transmisión

En la transmisión veremos todos los pasos que llevan los usuarios que "participan" en el sistema hasta que envían la información. Ocurren varias fases importantes que vamos a ir explicando paso por paso.

5.1.1.1.1 Códigos utilizados

Lo primero que vamos a tener en cuenta es la elección de los códigos de ensanchado por parte del transmisor. Vamos a trabajar con códigos *Gold* o con códigos *OVSF* con un factor de ensanchado de 32.

Como ya mencionamos con anterioridad en otros apartados, las propiedades de autocorrelación y correlación cruzada de los códigos son muy importantes en los sistemas CDMA. Para ello veamos las dos aplicaciones con las que analizaremos estas propiedades: *autocorr_codes.m y corr_codes.m*

Para calcular la autocorrelación de las secuencias hemos usado la función '*xcorr*' [13]:

autocor = xcorr(codigos(i,:));

A continuación se muestra alguna representación de la función de autocorrelación para ambos tipos de códigos:

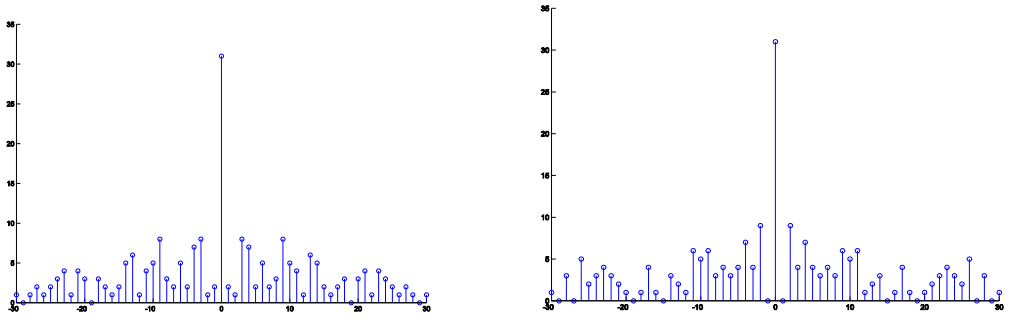


Figura 24. Funciones de autocorrelación para dos secuencias GOLD

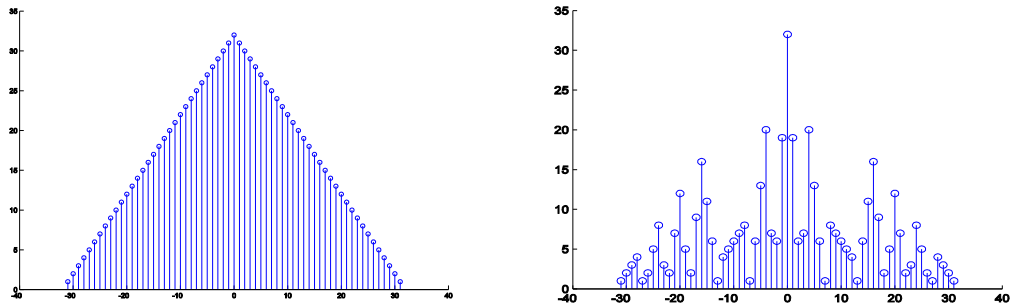


Figura 25. Funciones de autocorrelación para dos códigos OVSF con SF 32

Observamos que las funciones de autocorrelación son mejores para los códigos de aleatorización GOLD, ya que tienen un aspecto más parecido a lo que sería una delta centrada en el origen.

Para la correlación cruzada entre códigos hemos usado el comando '*corr*' [13]. Este comando realiza la correlación cruzada de cada columna con cada una de las demás columnas de la matriz de códigos introducida, y devuelve un valor por cada correlación cruzada entre dos columnas. Este valor está normalizado a 1. Como es de prever, la diagonal de la matriz que devuelva será todo 1's, ya que equivale a la correlación de un código consigo mismo.

El valor medio de todas las correlaciones cruzadas de todos los códigos Gold de los que disponemos es de **0.1516.** Sabemos que está normalizado a 1, por lo que es un valor muy aceptable para usar este tipo de códigos en un sistema CDMA.

En el caso de los códigos OVSF ese valor, lógicamente, es **0**, ya que se trata de códigos ortogonales, por lo que el producto entre ellos es cero.

Como ya adelantamos en apartados anteriores, tanto los códigos OVSF como las secuencias Gold nos vendrán muy bien a la hora de separar la información de los distintos usuarios o de mitigar los efectos del multitrayecto.

Por lo tanto, hemos comprobado como las secuencias Gold obtienen un buen compromiso en cuanto a la relación autocorrelación y correlación cruzada. Los códigos OVSF nos van a permitir insertar una interferencia nula entre los usuarios, siempre y cuando, como se vio en capítulos anteriores, estén las secuencias totalmente sincronizadas.

Si usamos códigos OVSF usaremos la siguiente sentencia en MATLAB®:

codigos_walsh = hadamard(32);
codigos = codigos_walsh(2:32,:);

El comando '*hadamard*' genera una matriz de *Hadamard* que se forma como vimos en el capítulo 2. Con la segunda línea eliminamos la primera fila de esa matriz ya que no la vamos a usar como código al ser utilizada solo para canales piloto [14].

En cuanto a los códigos Gold, disponemos de 33 secuencias de 31 chips cada uno.

A continuación pasamos al ensanchamiento de los bits de información de los usuarios.

5.1.1.1.2 Ensanchamiento de la señal de bits y modulación.

Los usuarios generan una serie de bits que será la información que van a enviar.

bits = round(rand(n_users, n_bits));

A continuación la información va a ser modulada por una de las técnicas que hablamos anteriormente. En el caso de utilizar una modulación BPSK, aplicación *bpsk.m,* sus bits 0s y 1s pasarían a estar polarizados a ±1 de la siguiente forma:

Busco en qué posiciones están los bits que son 0 y los que son 1. Después convierto los bits '0' a '1' y los bits '1' a '-1'.

A continuación la señal modulada resultante es ensanchada por el código correspondiente, como vemos en la Figura 26:

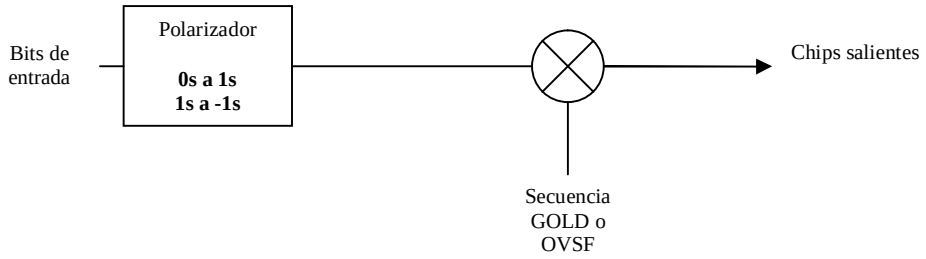


Figura 26. Modulación BPSK

Por lo tanto, disponemos de un número de chips salientes equivalente a:

*n_chips_salientes = n_bits_entrantes*longitud_secuencia*

En el caso de usar una modulación QPSK actuaremos de la siguiente forma:

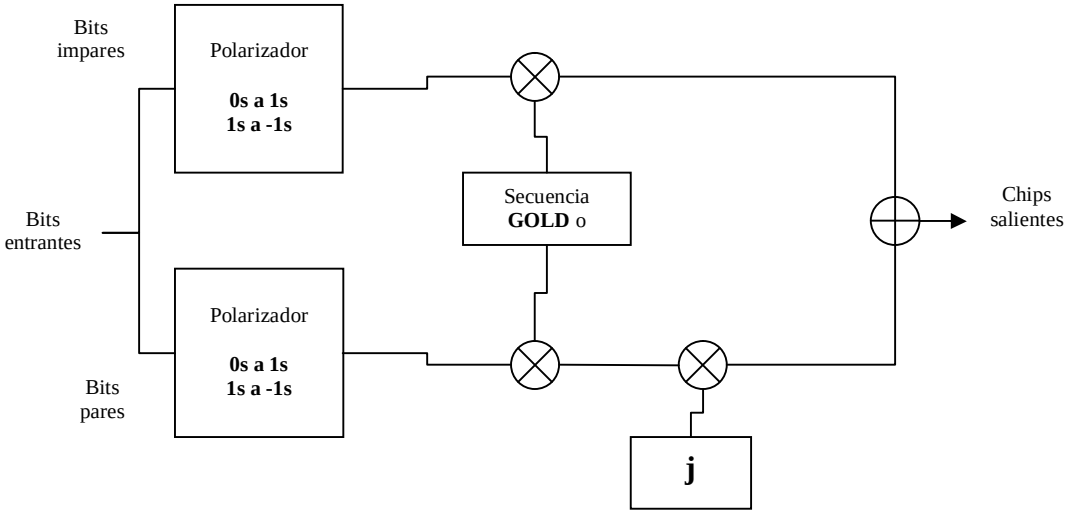


Figura 27. Modulación QPSK, ver aplicación *qpsk.m*

En este caso diferenciamos a los bits impares de los pares. En los bits pares introducimos un desfase de '$\pi/2$', y luego sumamos los chips de los bits pares con los impares en orden, de forma que vamos a obtener una secuencia de chips, con una longitud la mitad del caso de usar una BPSK. Además en este caso el resultado es una secuencia con chips con parte real e imaginaria. Por lo tanto se va a enviar un número de símbolos igual a la mitad de la longitud inicial.

Después de ser modulada y ensanchada la información, cada usuario dispondrá de su secuencia de chips lista para ser enviada.

En la matriz de chips que transmitirán los usuarios definida con el nombre '*tx*', cada fila corresponde a un usuario:

tx = bits_expand.*fin;

Antes de pasar al siguiente apartado es interesante comentar el funcionamiento de una aplicación cuya función es simplemente alternar una secuencia de bits tal y como se muestra en la siguiente figura:

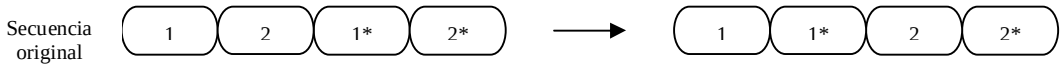


Figura 28. Función de la aplicación *intercalar_simple.m*

A continuación vamos a seguir viendo los pasos más importantes en la siguiente fase de la comunicación, el canal.

5.1.1.2 Canal

Una vez lista la información de cada usuario, es enviada y por lo tanto consideraremos que se le añade el ruido correspondiente a cada señal.

A continuación vamos a usar dos tipos de canal. Una de las opciones es probar con un canal AWGN, y la otra es usar un canal *Rayleigh* al que a continuación le incluiremos el ruido gaussiano.

Como ya vimos en apartados anteriores el canal AWGN es un canal de ruido aditivo.

Primero voy a calcular la potencia de la señal. La potencia de la señal la calculamos como el sumatorio del valor absoluto de todos los chips entrantes al cuadrado, dividido entre el número total de chips:

$$p\_tx = \frac{\sum \left| chips\_entrantes^2 \right|}{num\_chips}$$

La SNR del canal la tengo que pasar a unidades naturales ya que se introduce en dB's. A continuación ya puedo calcular la potencia del ruido que será la varianza de la gaussiana:

$$N = \frac{p\_tx}{SNR}$$

$$n = \sqrt{N} \times G(0,1)$$

Donde '*n*' es el ruido aditivo y $G(0,1)$ es una distribución gaussiana de media 0 y varianza 1. Por lo tanto los chips salientes será la señal total emitida por los usuarios más el ruido.

A cada señal de cada usuario se le aplicará un ruido AWGN. En la siguiente figura podemos ver el funcionamiento del canal AWGN:

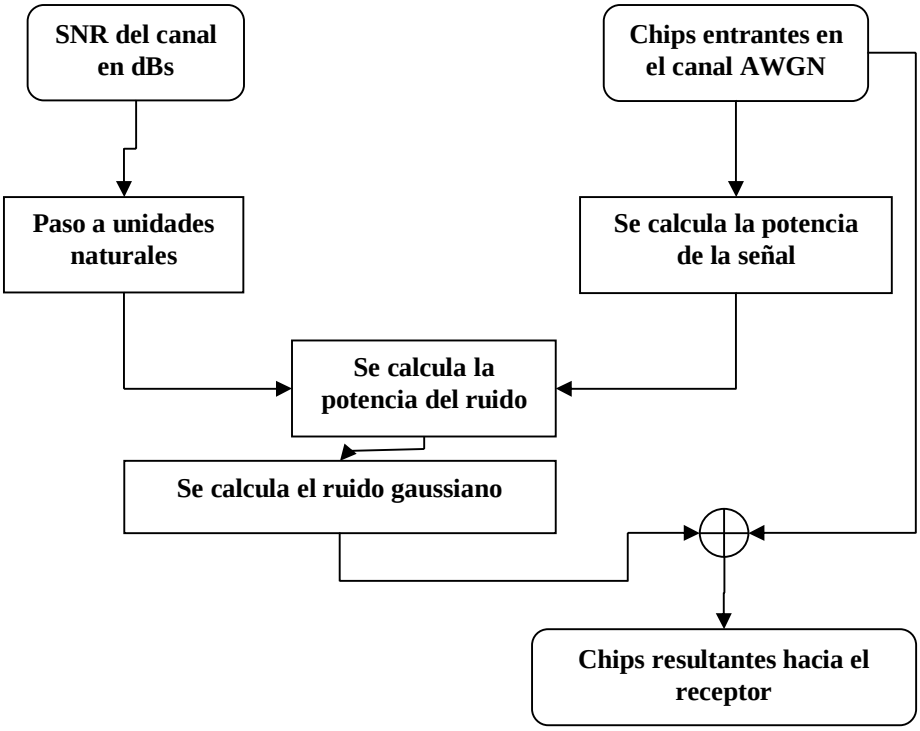


Figura 29. Diagrama de un canal AWGN.

Para el caso en el que queramos simular un canal *Rayleigh* utilizaremos las siguientes aplicaciones: *doppler_effect.m y doppler_impulse.m*.

En estas dos aplicaciones recorremos los pasos necesarios para simular un canal Rayleigh.

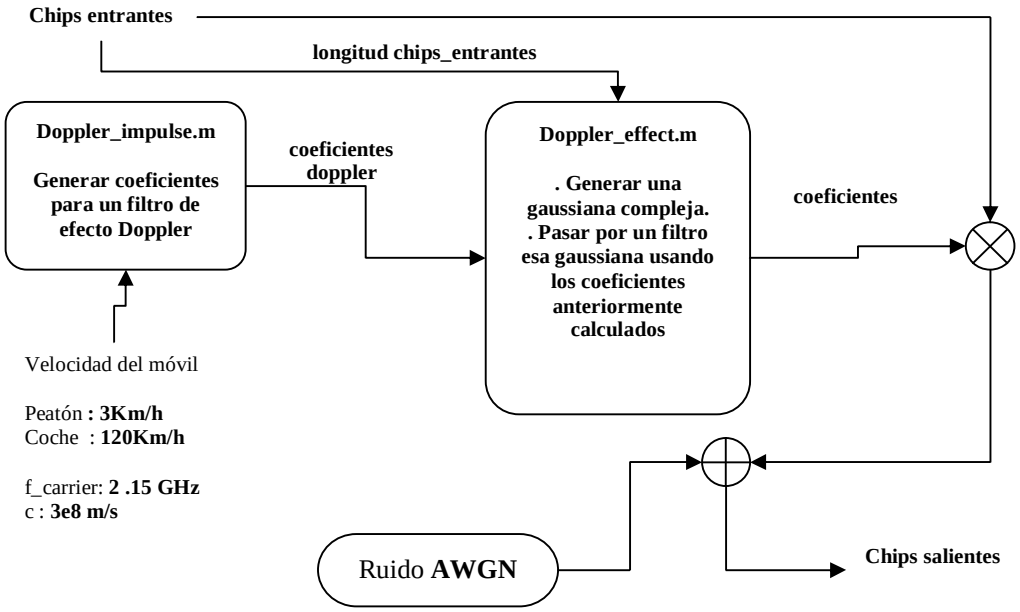


Figura 30. Esquema de un canal Rayleigh

En la aplicación *doppler_impulse.m*, generamos unos coeficientes para simular el efecto Doppler. Para ello vamos a tener en cuenta algunos datos como la *frecuencia Doppler*, la cual la vamos a calcular a partir de la velocidad del móvil y de la frecuencia de portadora. La *frecuencia de muestreo,* la cual es el producto de la frecuencia Doppler por un factor variable. La frecuencia de muestreo junto con la frecuencia de portadora, o lo que equivaldría al tiempo de chip, nos va a dar un factor de interpolación 'R':

$$R = \frac{fm}{fm1}$$

La frecuencia de portadora utilizada es 2.15 GHz, que es la frecuencia de portadora usada en UMTS [15].

Los coeficientes Doppler resultantes los podemos representar, y de esta forma observar la forma del filtro en tiempo y en frecuencia:

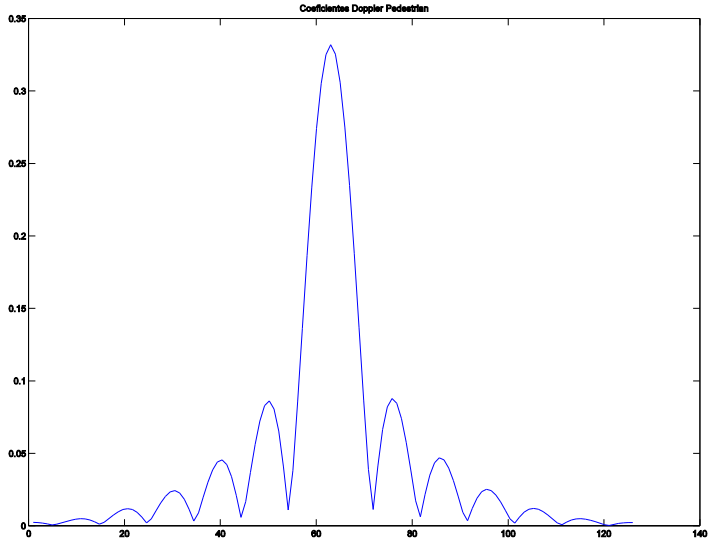


Figura 31. Coeficientes Doppler en el caso de un peatón y un factor variable de 40.

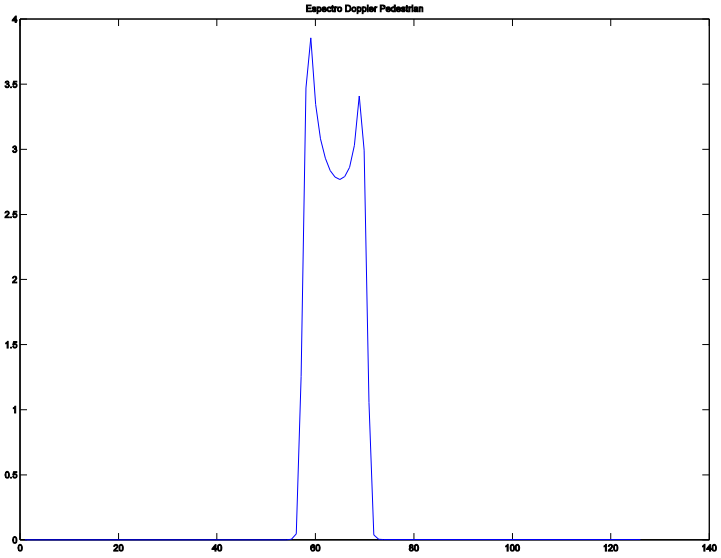


Figura 32. Espectro Doppler para el caso de un peatón y un factor variable de 40.

En la aplicación *doppler_effect.m,* vamos a crear un filtro con los coeficientes generados anteriormente. Generaremos de forma aleatoria una gaussiana compleja y la introduciremos en ese filtro que hemos creado.

n_ped=randn(1,nmuest)+j*randn(1,nmuest); *Generación de una gaussiana compleja*

coeff_ped(k,:)=filter(b_ped,1,n_ped); *Coeficientes a la salida del filtro*

La salida de ese filtro serán los coeficientes por los que tendremos que multiplicar la señal que es transmitida. Por el momento vamos a considerar un solo rayo, es decir, no incluimos multitrayecto. Después, como podemos observar en la Figura 30, añadimos el ruido del canal AWGN que ya explicamos anteriormente.

De esta forma llegaría la señal de cada usuario al receptor. En el caso de que hubiéramos simulado un canal Rayleigh, se multiplicaría la señal que entra al receptor por la respuesta al impulso del canal, la cual suponemos que conoce el receptor. Es decir, multiplicaremos por el conjugado de los coeficientes resultantes del filtro del canal.

Por último sumamos todas las cadenas de chips de los usuarios que llegan al receptor, formando una única señal.

A continuación explicaremos todos los pasos que realizará el receptor para recuperar la información del usuario deseado.

5.1.1.3 Receptor

El primer paso que tomaremos independientemente ya del canal usado, será multiplicar la secuencia total, por el código usado para codificar y expandir la señal del usuario deseado.

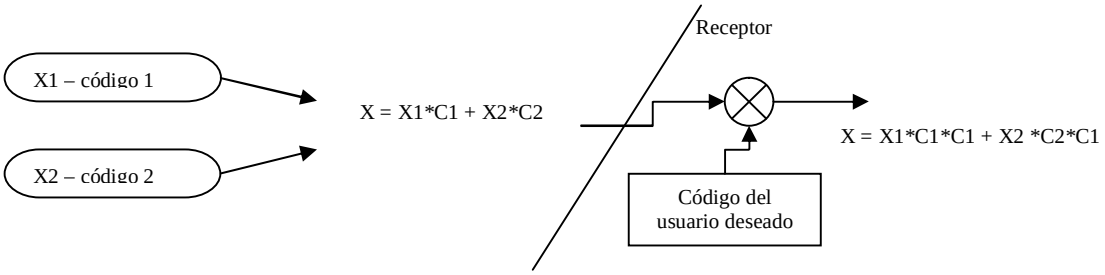


Figura 33. Se multiplica la secuencia recibida por el código deseado.

El siguiente paso es decidir a que bits corresponden la señal que recibo. Para ello voy a agrupar los chips entrantes en grupos equivalentes a la longitud del código utilizado para codificar un bit. De este modo, en cada grupo tendré los chips equivalentes a un símbolo enviado, o a 2 si la modulación utilizada fue QPSK. Con los chips agrupados puedo promediar y obtener un valor el cual me permitirá decidir si el bit original de información era un 1 o un 0.

Es necesario aclarar que si la modulación usada es QPSK, entonces tenemos que hacer una separación entre la parte real y la parte imaginaria de los chips recibidos.

Separamos la parte real y la parte imaginaria, con los comandos de MATLAB® ‘*real*’ y ‘*imag*’ respectivamente.

Ahora tenemos que intercalar las partes para tener los valores ordenados:

```
inf = [p_real p_imag];
rx_total_sum = intercalar_simple(inf);
```

Ahora ya podemos tomar la decisión. El decisor utilizado es un decisor de máxima verosimilitud, por lo que decidiremos si es un 1 o un 0 en función del valor más aproximado al símbolo enviado. Esto es, como los símbolos enviados son 1s o -1s, para los valores mayores que '0', decidiremos '1', pero al estar modulados al revés como comentamos anteriormente, los valores mayores que '0' se considerarán como '-1', y para los valores menores que '0', '1'.

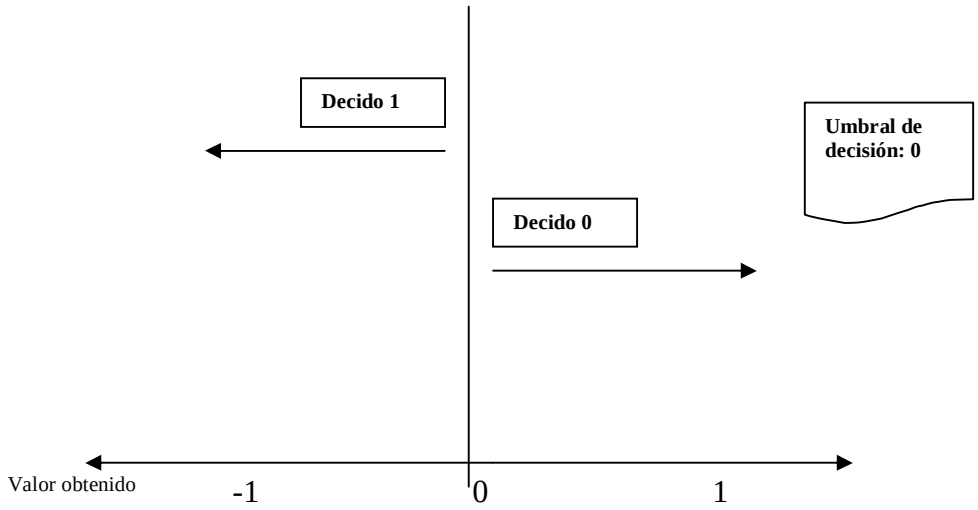


Figura 34. Decisor utilizado en el receptor para una modulación BPSK.

Por ultimo calculamos la probabilidad de error como el número de bits que han sido detectados de forma errónea, entre el número de bits totales. Lo que nos interesa es la probabilidad de error de bit, BER (Bit Error Rate):

$$prob\_error = \frac{num\_errores}{num\_total\_bits\_enviados}$$

En la siguiente figura veremos el diagrama de bloques acerca del funcionamiento de este escenario, de una forma global:

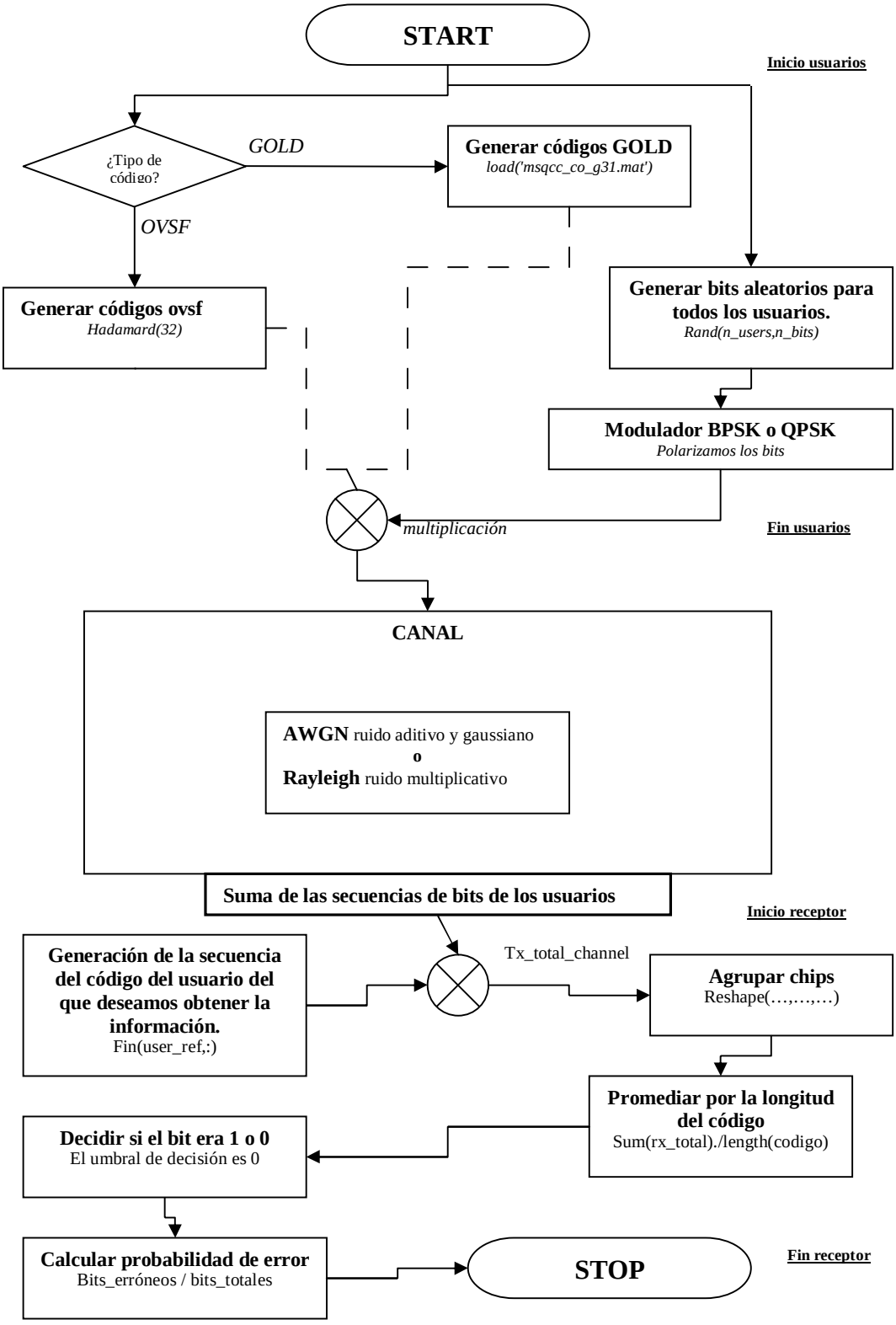


Figura 35. Diagrama de bloques del escenario CDMA de N usuarios y un receptor.

5.2 Escenarios con diversidad cooperativa.

En los siguientes apartados analizaremos las características de las técnicas de diversidad cooperativa, y las diferencias entre todas ellas.

5.2.1 Escenario 2: Análisis de un sistema CDMA con el método de diversidad cooperativa *Amplify and Forward.*

En este escenario vamos a introducir el concepto de diversidad cooperativa. Dos usuarios cooperan entre sí usando *Amplify and Forward*:

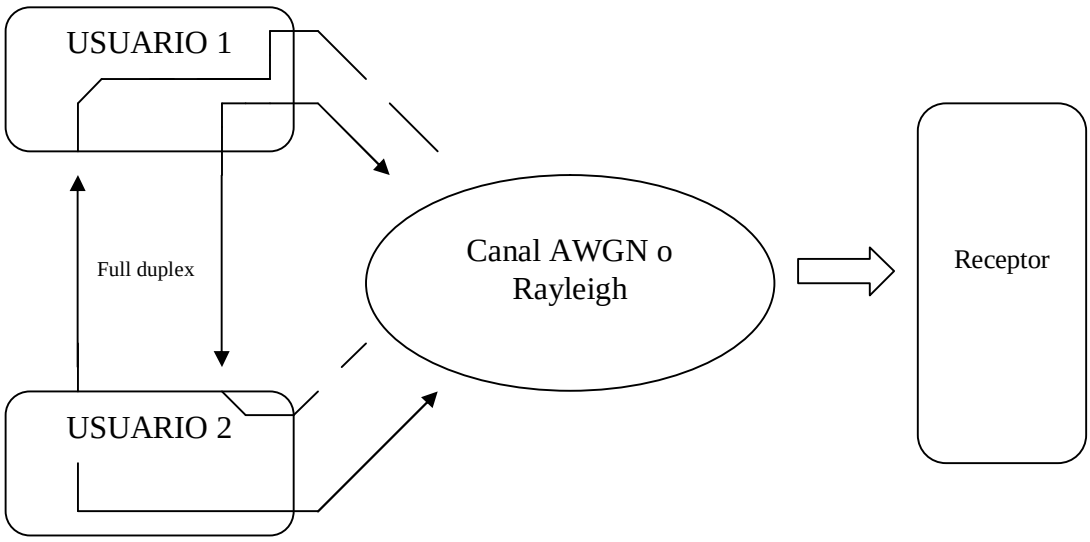


Figura 36. Escenario CDMA con diversidad cooperativa usando AAF.

Como se observa en la Figura 36, entre los dos usuarios tenemos una comunicación *full duplex* de forma que pueden enviar y recibir datos simultáneamente. De esta forma los dos usuarios van a cooperar usando la técnica conocida como *Amplify and Forward.*

Cuando el usuario recibe la señal del "compañero", este la amplifica con un factor 'A', el cual lo explicaremos detalladamente más adelante, y la reenvía hacia el receptor.

Por lo tanto el usuario 1 amplifica los chips recibidos del usuario 2, y el usuario 2 los de usuario 1.

El canal existente entre los dos usuarios solo vamos a contemplar la posibilidad de usar AWGN, el canal usado después ya será como hemos explicado en el escenario anterior.

Partiendo del 'Escenario 1', el siguiente diagrama de bloques explica la comunicación que se produce entre los dos usuarios:

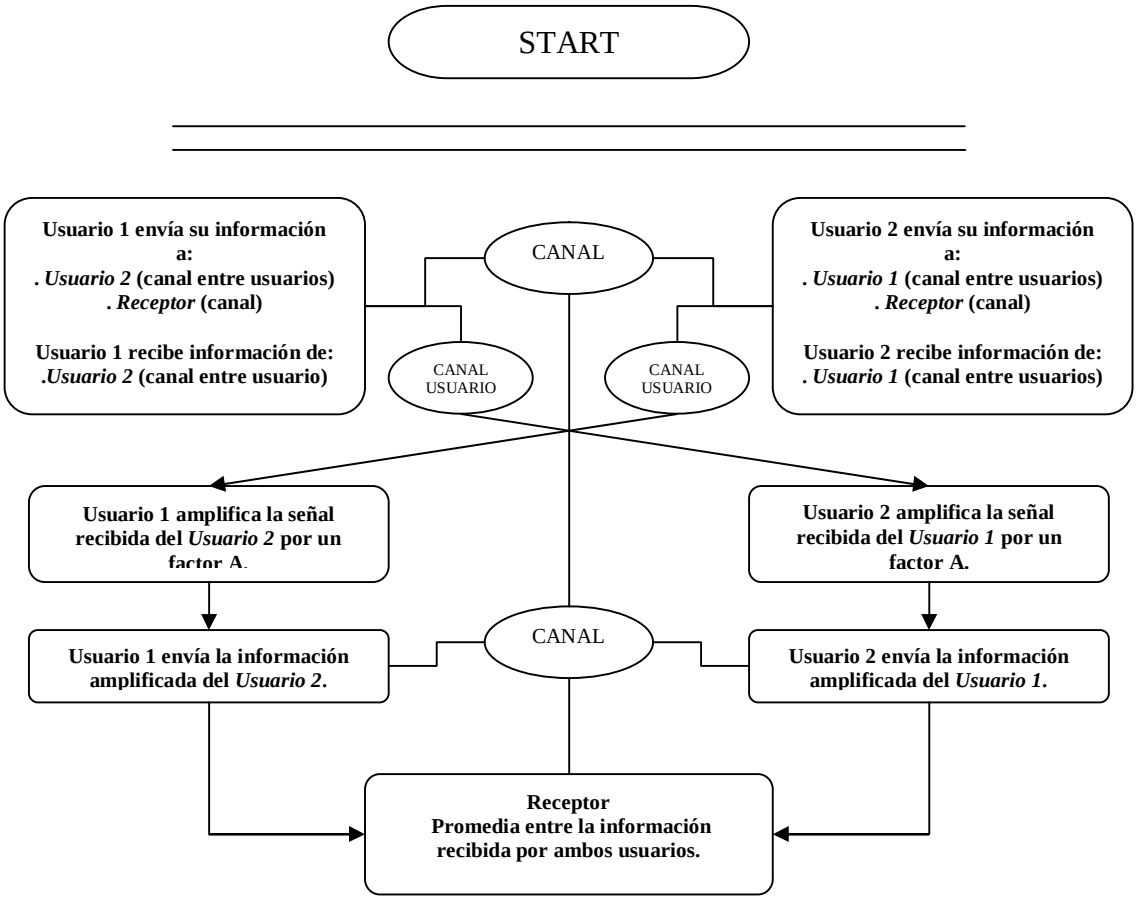


Figura 37. Diagrama de bloques de la cooperación entre usuarios del Escenario 2.

Como observamos en la Figura 37 y en la aplicación *cdma_aaf.m*, en el diagrama se hace referencia al factor de amplificación 'A'. Este factor de amplificación puede ser variable o fijo. En nuestras simulaciones vamos a estudiar dos casos. Por un lado que el factor 'A' sea 1, es decir, los usuarios reenvían la señal con el mismo nivel que les llega; por otro lado variaremos ese valor 'A' y analizaremos como evoluciona la BER para distintos casos.

En este caso de diversidad cooperativa, el orden en el que se envían los chips es el siguiente:

----------	**TS 1**	**TS 2**	**TS 3**	**TS 4**
USUARIO 1	bit_1_cod_1_u1	bit_1_cod_2_u2	bit_2_cod_1_u1	bit_2_cod_2_u2
USUARIO 2	bit_1_cod_2_u2	bit_1_cod_1_u1	bit_2_cod_2_u2	bit_2_cod_1_u1

Señal total	bit_1_cod_1_u1 + bit_1_cod_2_u2	bit_1_cod_2_u2 + bit_1_cod_1_u1	bit_2_cod_1_u1 + bit_2_cod_2_u2	bit_2_cod_2_u2 + bit_2_cod_1_u1

Tabla 2. Orden de los bits en función del usuario y del *time slot* en Amplify and Forward.

Como vemos en la Tabla 2, cada columna hace referencia a un *time slot* y en las filas de 'USUARIO 1' y 'USUARIO 2' vemos qué bit se envía y con qué código ha utilizado. En la última fila vemos en resultado de la suma de los dos usuarios.

Dos cosas importantes acerca de cómo se envían los bits. Primero, como podemos ver en la Tabla 2, la información de un usuario, por ejemplo el 1, siempre se envía codificada con el código suyo propio. Esto significa que el receptor multiplicará por la misma secuencia de código del usuario deseado, tal y como vimos en el escenario anterior. Y segundo, los bits se envían alternando los que se envían directamente y los que son primero recibidos por el compañero y posteriormente reenviados.

A efectos de la programación, para alternar los bits se ha creado la aplicación *intercalar.m*. Esta aplicación me sirve para intercalar la cadena de chips entrantes de la siguiente forma:

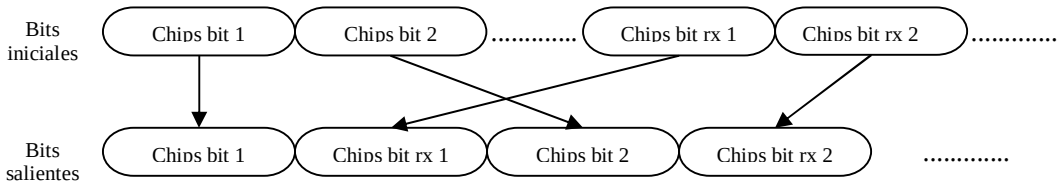


Figura 38. Resultado de la función *intercalar.m*

Para este método, en cuanto al receptor es muy parecido con el apartado sin diversidad. La diferencia está en que ahora la longitud de la señal recibida es el doble ya que en el primer *time slot* recibirá el primer bit de cada usuario, y en el segundo *time slot*, recibirá la retransmisión del primer bit de cada usuario. El resto del proceso es igual que en el caso del apartado anterior.

A continuación veremos otra técnica de diversidad cooperativa y las diferencias con la vista en este apartado.

5.2.2 Escenario 3: Análisis de un sistema CDMA con el método de diversidad cooperativa *Decode and Forward*.

En este escenario entran en juego prácticamente los mismos conceptos vistos en el escenario anterior donde ya introdujimos la diversidad cooperativa. En esta ocasión, el 'compañero' cuando recibe la información, no la amplifica, sino que la decodifica y vuelve a codificarla para enviarla posteriormente. Un aspecto importante de este escenario, es que el usuario cuando vuelve a codificar la información, lo hace con el código de ensanchado que el tiene asignado. Esto significa que la información de un usuario va a llegar al receptor codificada con dos códigos diferentes, lo cual veremos más adelante.

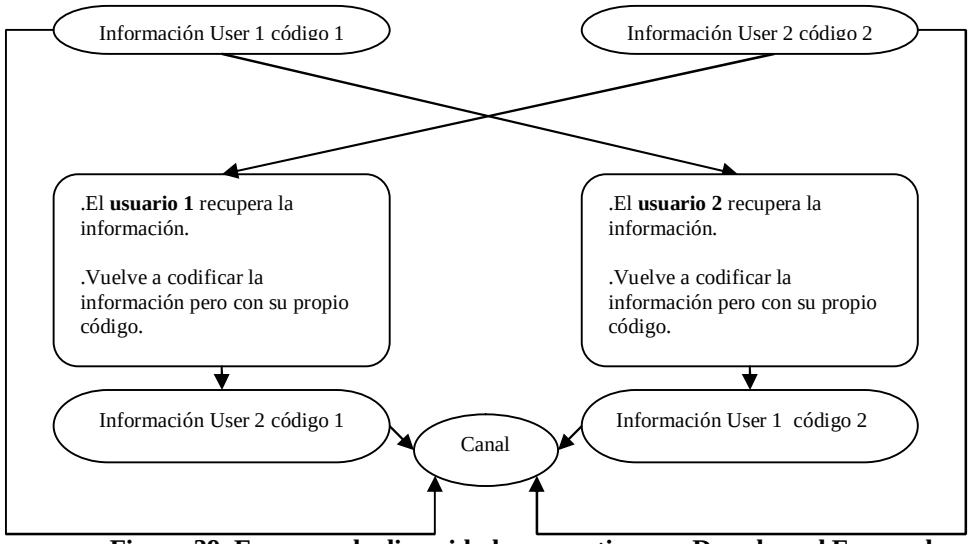


Figura 39. Esquema de diversidad cooperativa con Decode and Forward.

Como se observa en la Figura 39, en ambos usuarios se decodifica la señal del otro, lo que significa que el compañero tiene que conocer el código de ensanchado usado por el otro. El orden en el que se envían los bits hacia el receptor es similar a como vimos en la Tabla 3, pero con una diferencia:

----------	**TS 1**	**TS 2**	**TS 3**	**TS 4**
USUARIO 1	bit_1_cod_1_u1	bit_1_cod_1_u2	bit_2_cod_1_u1	bit_2_cod_1_u2
USUARIO 2	bit_1_cod_2_u2	bit_1_cod_2_u1	bit_2_cod_2_u2	bit_2_cod_2_u1

Señal total	bit_1_cod_1_u1 + bit_1_cod_2_u2	bit_1_cod_1_u2 + bit_1_cod_2_u1	bit_2_cod_1_u1 + bit_2_cod_2_u2	bit_2_cod_1_u2 + bit_2_cod_2_u1

Tabla 3. Orden de los bits en función del *time slot* y el usuario en Decode and Forward.

De igual forma que vimos en la Tabla 2, en la Tabla 3 vemos qué bit y con qué código envían los usuarios 1 y 2 en cada *time slot*, viendo en la última fila el resultado de la suma de los dos usuarios.

En cuanto al receptor, encontramos alguna diferencia con el apartado anterior. Ahora, cuando multipliquemos por el código del usuario del que se desea recuperar la información, debemos tener en cuenta que los chips retransmitidos por el compañero fueron codificados con el código del compañero. Supongamos entonces que queremos decodificar la información del usuario 1. En escenarios anteriores multiplicaríamos toda la señal entrante por el código asignado al usuario '1'. En este caso debemos alternar para cada *slot* de tiempo, siendo los *time slot* impares para el código asignado al usuario deseado, y los time slot pares para el código contrario.

Entonces tendremos que juntar las dos secuencias de códigos, repetidas tantas veces como símbolos enviados, y usar la función *intercalar.m* para alternar los códigos repetidos una y otra vez.

5.2.3 Escenario 4: Análisis de un sistema CDMA con diversidad cooperativa y usando Alamouti.

Para este escenario, la única novedad que vamos a presentar es la que ya señalamos en la Tabla 1. Basándonos en el escenario anterior, el único cambio será ahora reenviar los bits de acuerdo a la tabla mencionada anteriormente.

También, en el receptor para recuperar la información, desharemos el cambio realizado anteriormente. Para ello, en el receptor se vuelve a hacer el conjugado de los chips que no llegaron directamente del usuario, sino los que fueron reenviados por el 'compañero'.

El resto de los pasos seguidos es similar al utilizado en el apartado de *Decode and Forward*.

6 Resultados y discusión

A continuación vamos a analizar los resultados obtenidos de las simulaciones realizadas en el bloque anterior. Compararemos los diferentes escenarios que hemos usado: CDMA sin cooperación, *AAF*, *DAF* o *Alamouti*.

El dato clave para la comparación de los escenarios será la probabilidad de error obtenida en cada una de ellos. La probabilidad de error que trataremos será la BER, es decir que tendremos en cuenta el número de bits totales enviados.

6.1 Método de Monte Carlo

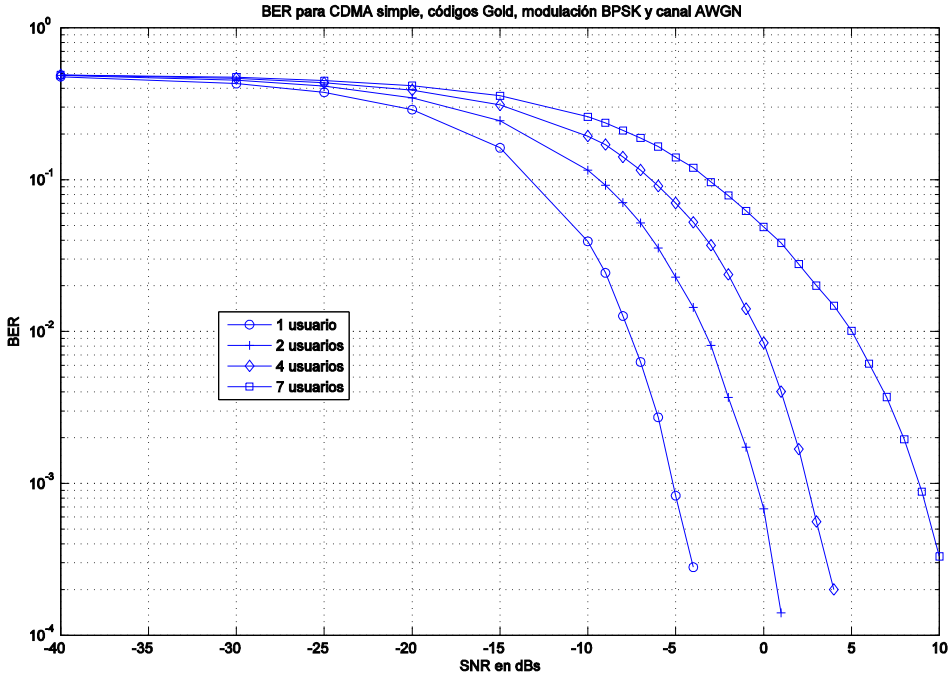
Según el método de Monte Carlo [12] para el cálculo de probabilidades de error de bit (BER), en simulaciones de sistemas de comunicaciones digitales, podemos obtener una P_e fiable del orden de 10/*N*, donde *N* es el número de bits usados en la simulación. Por lo tanto, dicho de otra forma, *N* debe ser del orden de $10/P_e$ con una fiabilidad de 10/*N*. El número de bits que envía cada usuario en las simulaciones realizadas es de 100.000 por lo que teniendo en cuenta el método de Monte Carlo podremos obtener probabilidades de error fiables del orden de 10^{-4}.

De esta forma, las probabilidades obtenidas por debajo de 0.0001 no serán fiables por el método visto anteriormente.

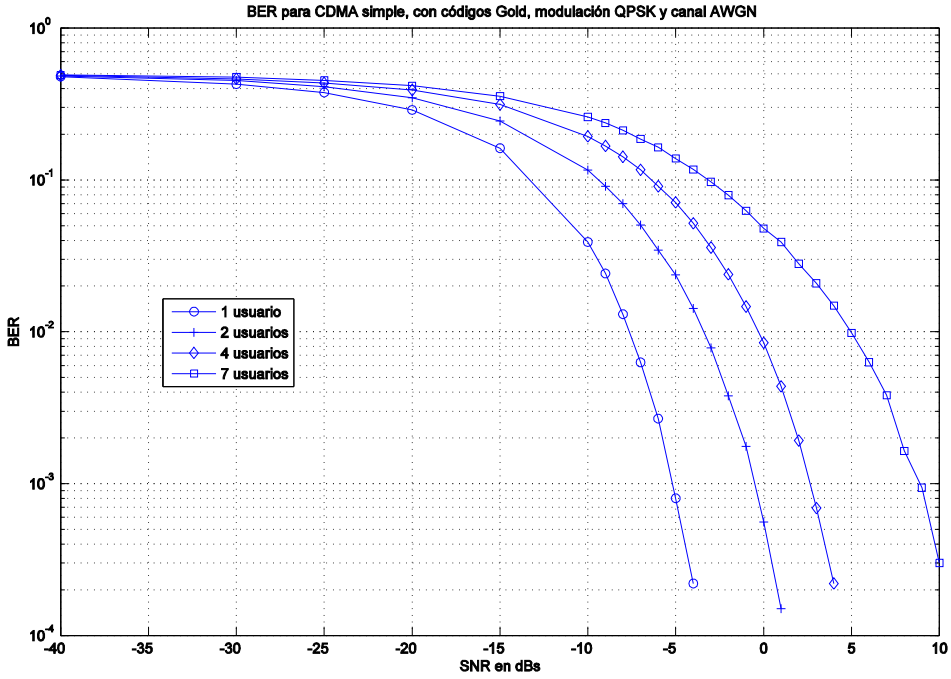
6.2 Resultados en el escenario de CDMA sin cooperación.

Este sistema lo hemos probado para 1, 2, 4 y 7 usuarios transmitiendo al mismo tiempo, probando códigos GOLD o códigos OVSF, con modulaciones BPSK y QPSK y utilizando canal AWGN. Variaremos la SNR entre -40 dBs y 10 dBs.

A continuación podemos observar los primeros resultados obtenidos en las siguientes gráficas:



(a)



(b)

Figura 40. BER para CDMA sin cooperación, con códigos GOLD y canal AWGN. (a) modulación BPSK (b) QPSK

En la Figura 40, observamos el comportamiento de los códigos Gold para dos tipos de modulación distinta. La primera apreciación que podemos hacer es ver como a medida que aumenta el número de usuarios del sistema, la BER tiende a ser mayor; o lo

que es lo mismo, necesitaremos una mayor relación señal a ruido del canal para obtener las mismas prestaciones a medida que aumenta el número de usuarios. También, como es lógico, a mayor SNR del sistema la probabilidad de error disminuye.

Podemos tomar como referencia algunos datos que vemos en la siguiente tabla:

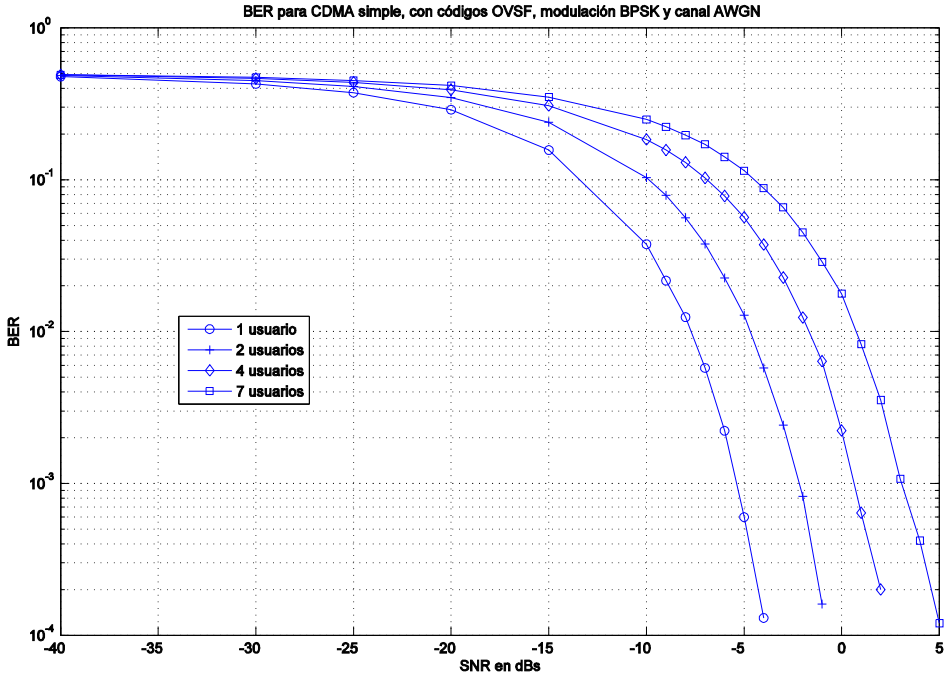
Modulación/ SNR	**-15 dBs**	**-5 dBs**	**-3 dBs**	**0 dBs**
2 usuarios (a)	0.2447	0.0228	0.0081	0.0006
2 usuarios (b)	0.2440	0.0237	0.0078	0.0005

Tabla 4. Comparación entre distintas modulaciones y SNR para CDMA sin cooperación, códigos GOLD.

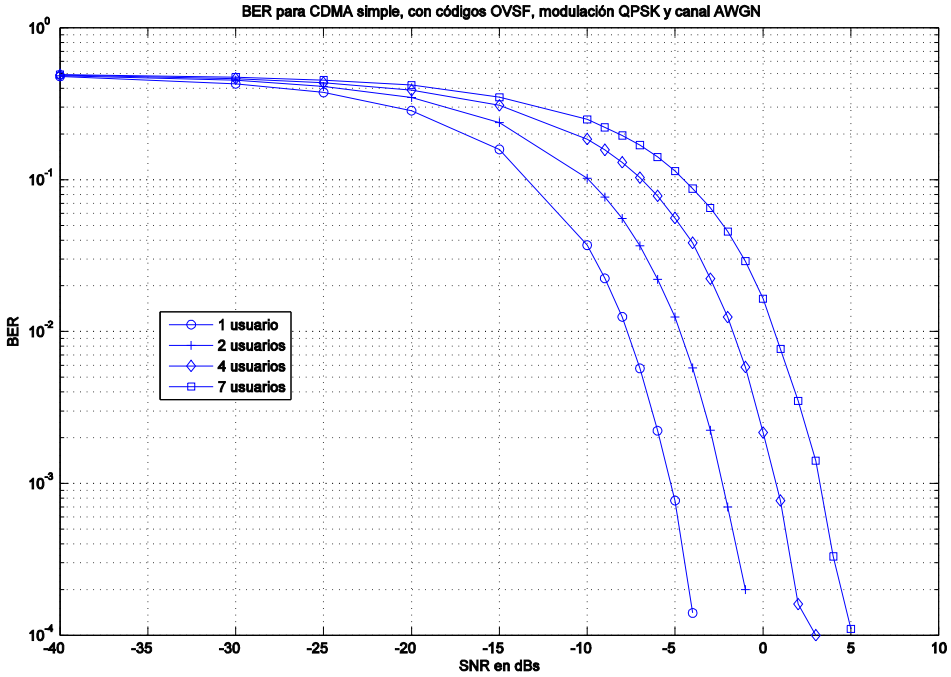
En la Figura 41, también comparamos las dos modulaciones pero para códigos OVSF. La tendencia es muy similar que la vista en la figura anterior por lo que principalmente podremos comprobar como son los códigos utilizados. Si observamos la siguiente tabla podremos detectar alguna diferencia entre los dos códigos:

Modulación/ SNR	**-15 dBs**	**-5 dBs**	**-3 dBs**	**0 dBs**
2 usuarios (a)	0.2390	0.0128	0.0024	< 0.0001
2 usuarios (b)	0.2382	0.0124	0.0022	< 0.0001

Tabla 5. Comparación entre distintas modulaciones y SNR para CDMA sin cooperación, códigos OVSF.



(a)



(b)

Figura 41. BER para CDMA sin cooperación, con códigos OVSF y canal AWGN. (a) modulación BPSK. (b) QPSK

De lo visto hasta ahora en la Tabla 4 y la Tabla 5, podemos ver como la modulación QPSK tiende ligeramente a proporcionar mejores probabilidades de error, aunque muy parecidas a utilizar BPSK.

En la Figura 42, sacamos conclusiones sobre la utilización de los códigos en CDMA sin cooperación sobre un canal AWGN:

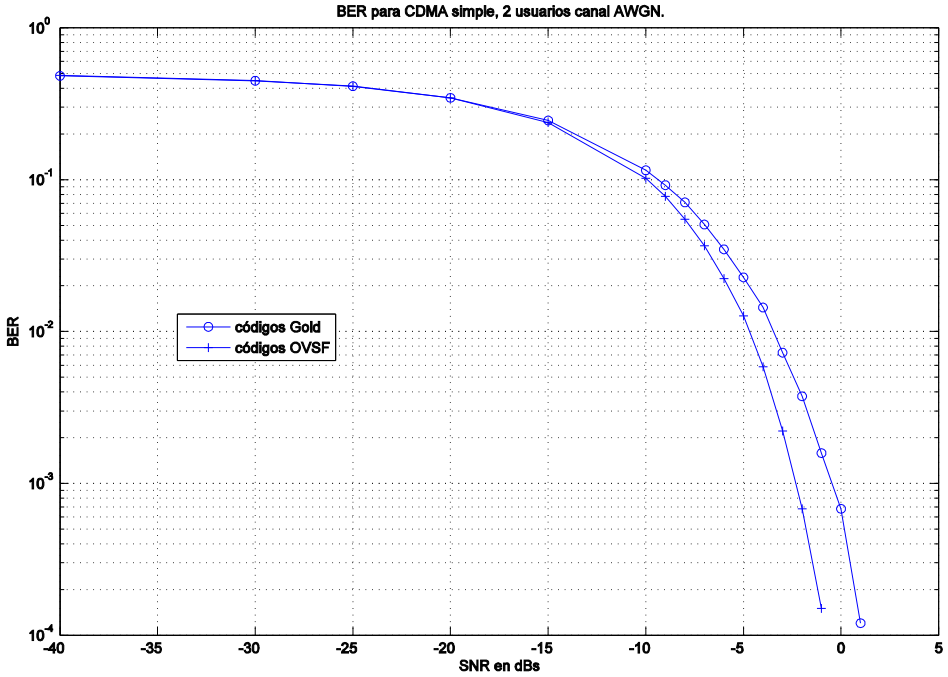


Figura 42. BER comparando los códigos GOLD y OVSF en CDMA sin cooperación, modulación QPSK y canal AWGN.

Podemos entonces comprobar como las probabilidades obtenidas con los códigos OVSF son mejores. Esto es debido a que estamos suponiendo que los usuarios están enviando la información de forma síncrona. En el enlace descendente si ocurriría de esta forma por lo que se suelen usar los códigos OVSF.

6.3 Resultados en el escenario de CDMA con Amplify and Forward.

En este escenario el número de usuarios está fijado a 2 para observar el funcionamiento de la técnica *Amplify and Forward*. Es importante mencionar que el canal utilizado en las comunicaciones entre los usuarios es AWGN con una SNR de 20 dBs, y más tarde hemos probado a cambiar ese valor y analizar las consecuencias. Compararemos esta técnica con el resto de escenarios diseñados y también lo simularemos para un canal Rayleigh.

En la Figura 43 vemos el comportamiento de los códigos utilizados para este escenario. La tendencia es la misma que en el apartado anterior.

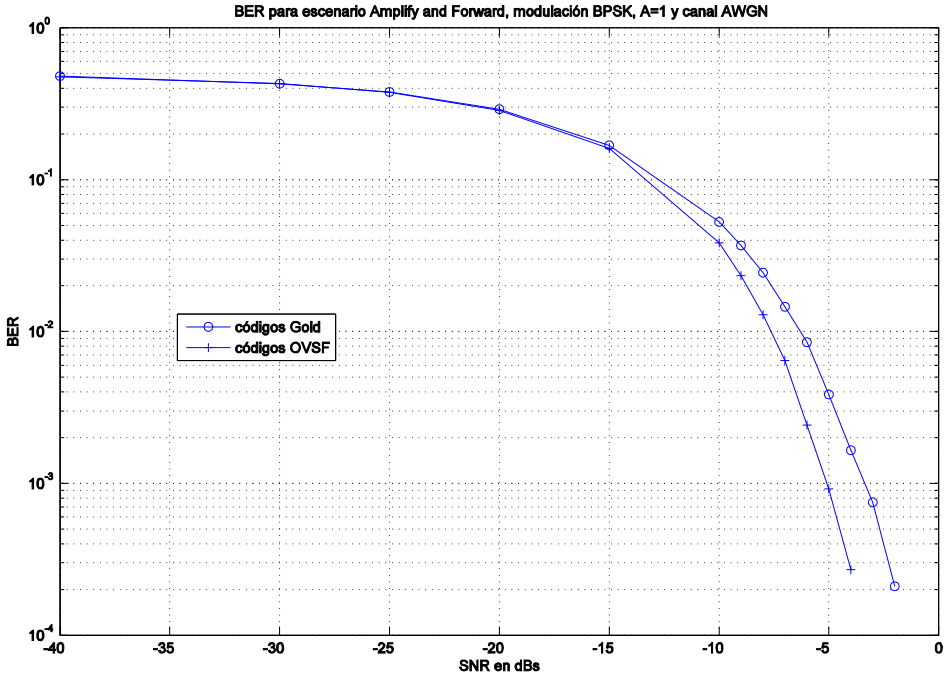


Figura 43. BER en escenario Amplify and Forward, comparando códigos GOLD y OVSF con A=1 y canal AWGN.

Los códigos OVSF vuelven a ofrecer mejores probabilidades de error como mencionamos anteriormente.

En la Figura 44 vamos a comparar las prestaciones obtenidas con la técnica AAF en comparación con el apartado anterior donde no utilizamos ninguna técnica de comunicación cooperativa. Como se observa claramente, hemos mejorado la BER obtenida considerablemente.

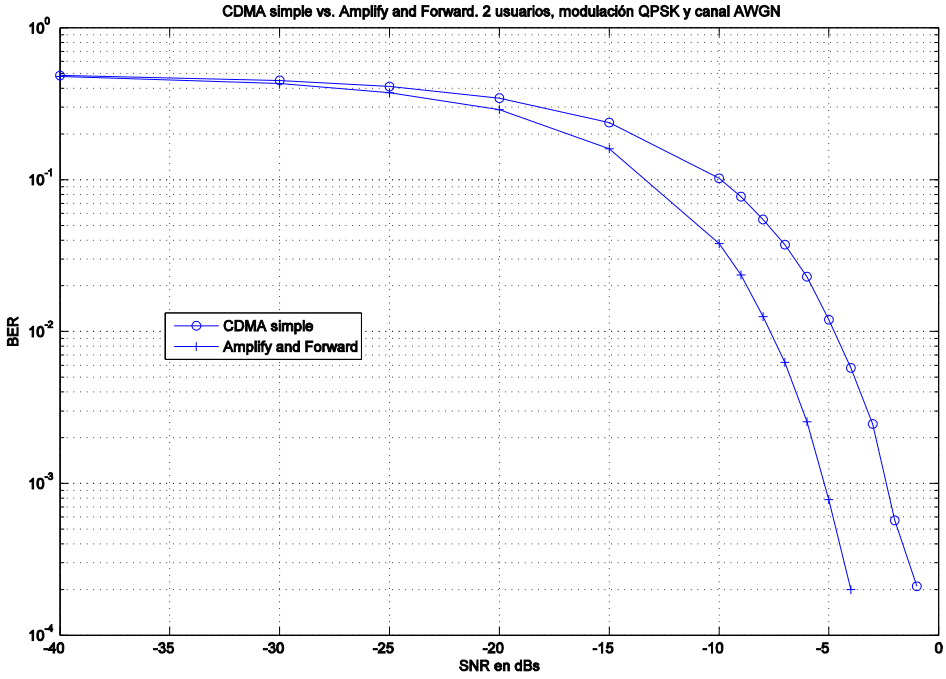


Figura 44. BER comparando CDMA sin cooperación con Amplify and Forward, para A=1 y canal AWGN y códigos OVSF.

Otro aspecto importante que podemos tener en cuenta en este apartado es el factor de amplificación elegido. Hasta ahora hemos utilizado 1, pero probemos con otros factores de amplificación y veamos que ocurre.

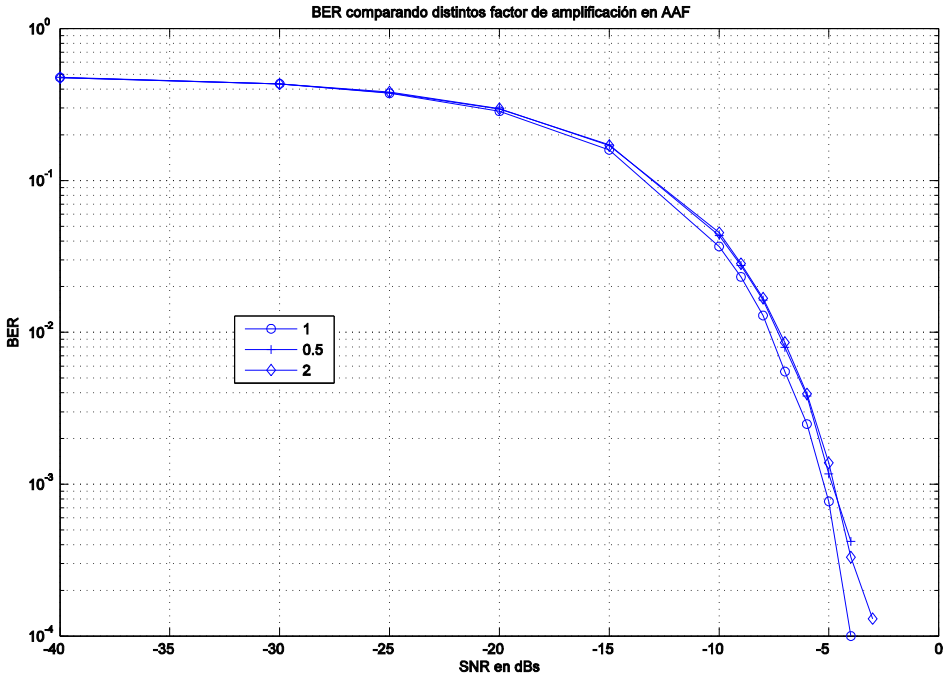


Figura 45. BER comparando distintos factores de amplificación. Canal AWGN y códigos OVSF.

En la Figura 45 hemos probado para factores de amplificación de 0.5 1 y 2, es decir, con el doble y la mitad del valor utilizado inicialmente.

Podemos ver como obtenemos mejores prestaciones para un factor de amplificación de 1, lo que sería que el usuario con el que coopera actúa básicamente de repetidor. Por lo tanto las simulaciones para AAF serán realizadas con el factor de amplificación a 1.

A continuación vamos a introducir en las simulaciones el canal Rayleigh, que como ya vimos en apartados anteriores, es un canal que simula unas condiciones más realistas.

En la Figura 46 vemos las prestaciones para el canal Rayleigh en Amplify and Forward con dos velocidades del móvil diferentes. Como se puede apreciar, las gráficas ofrecen datos muy similares.

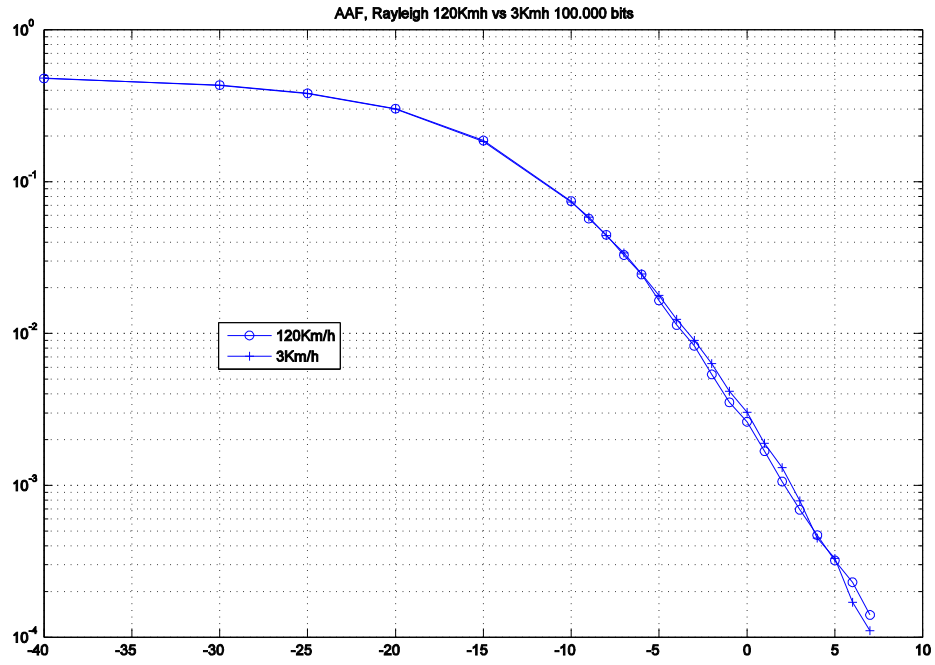


Figura 46. BER para Amplify and Forward y canal Rayleigh, comparando 3Kmh y 120Kmh

Si comparamos lo obtenido para el caso AWGN con lo visto en el caso Rayleigh en estas dos últimas gráficas, obtenemos las gráficas que vemos a continuación:

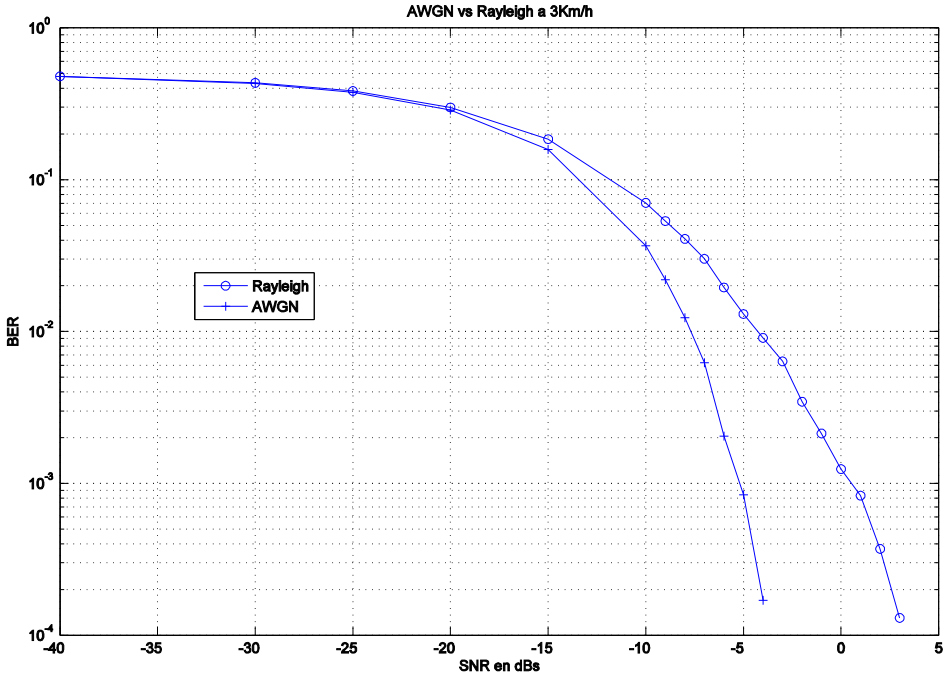


Figura 47. BER para AWGN vs. Rayleigh a 3Km/h en AAF.

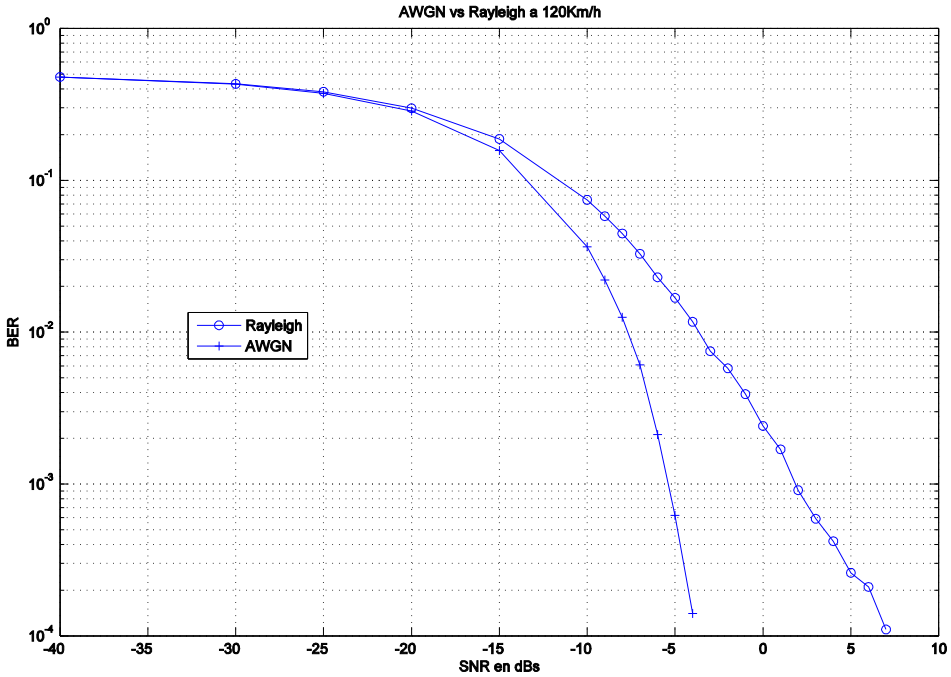


Figura 48. BER para AWGN vs. Rayleigh a 120Km/h en AAF.

Como podemos observar con el canal Rayleigh obtenemos peores prestaciones como ya intuíamos de forma teórica.

En la siguiente gráfica podemos observar las consecuencias de cambiar el valor de la SNR del canal entre los usuarios:

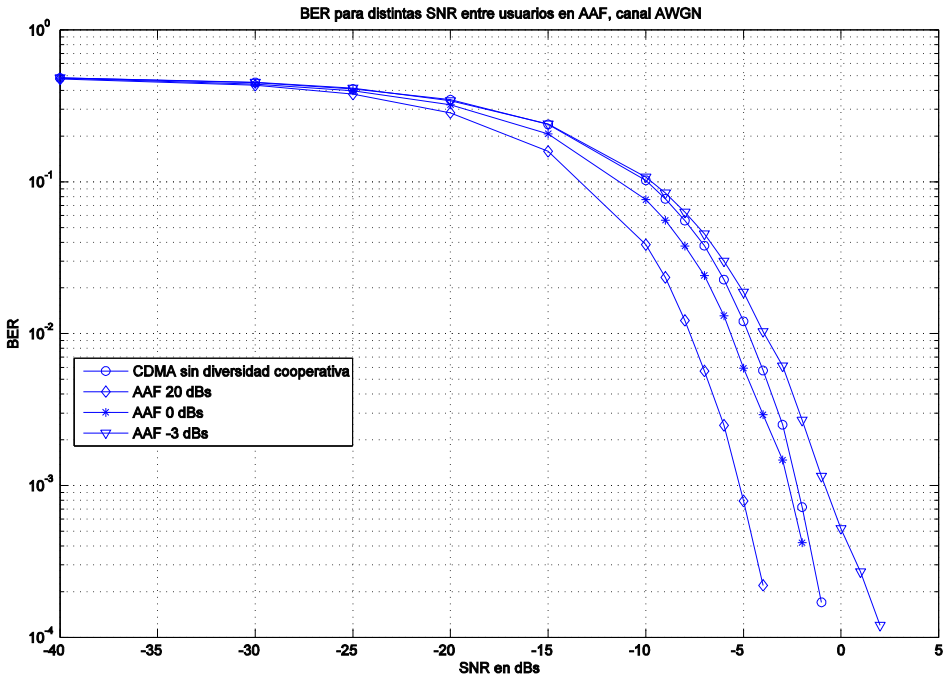


Figura 49. BER para distintos valores de SNR del canal entre usuarios en AAF y canal AWGN.

Como vemos, a medida que el ruido aumenta en el canal entre los usuarios, las probabilidades de error se asemejan a CDMA sin diversidad cooperativa. En concreto vemos como para un valor de -3 dBs ya no compensa la utilización de Amplify and Forward con respecto a no utilizar diversidad cooperativa.

6.4 Resultados en el escenario de CDMA con Decode and Forward.

En este apartado también utilizaremos AWGN como canal entre los dos usuarios con una SNR de 20 dBs, aunque después variaremos ese valor para ver también que consecuencias tiene en esta técnica de diversidad cooperativa.

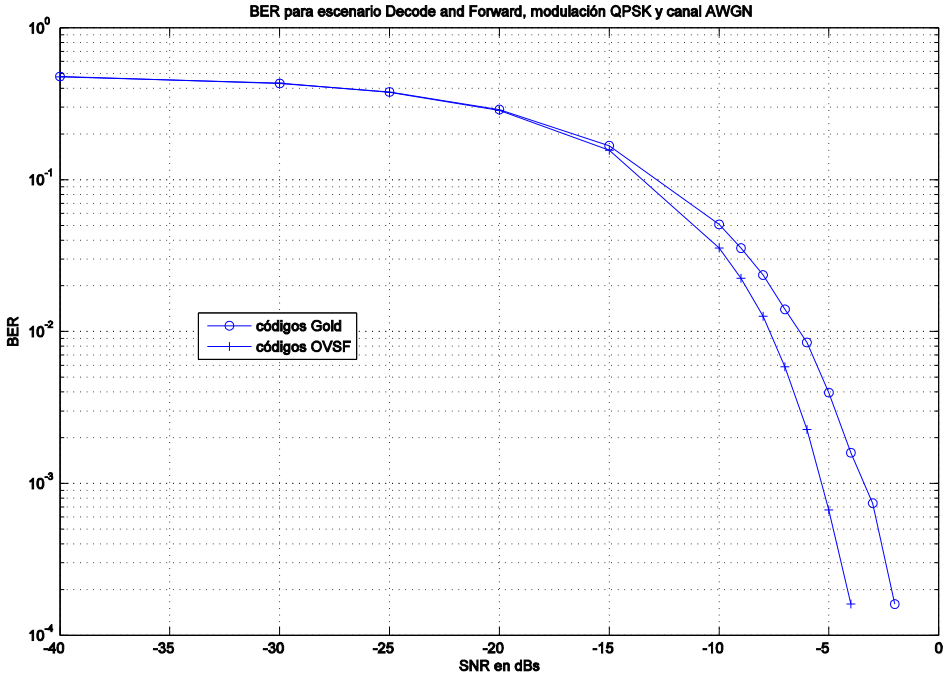


Figura 50. BER para escenario Decode and Forward, con modulación QPSK y canal AWGN.

Una vez más, como vemos en la Figura 50, obtenemos probabilidades de error más bajas para los códigos OVSF, esta vez en el escenario DAF en un canal AWGN. Y como también observamos en la Figura 51 para un canal Rayleigh, los códigos OVSF siguen ofreciendo mejores prestaciones. Aunque en este caso, la diferencia es menor y el comportamiento de los códigos en este tipo de canales se va aproximando.

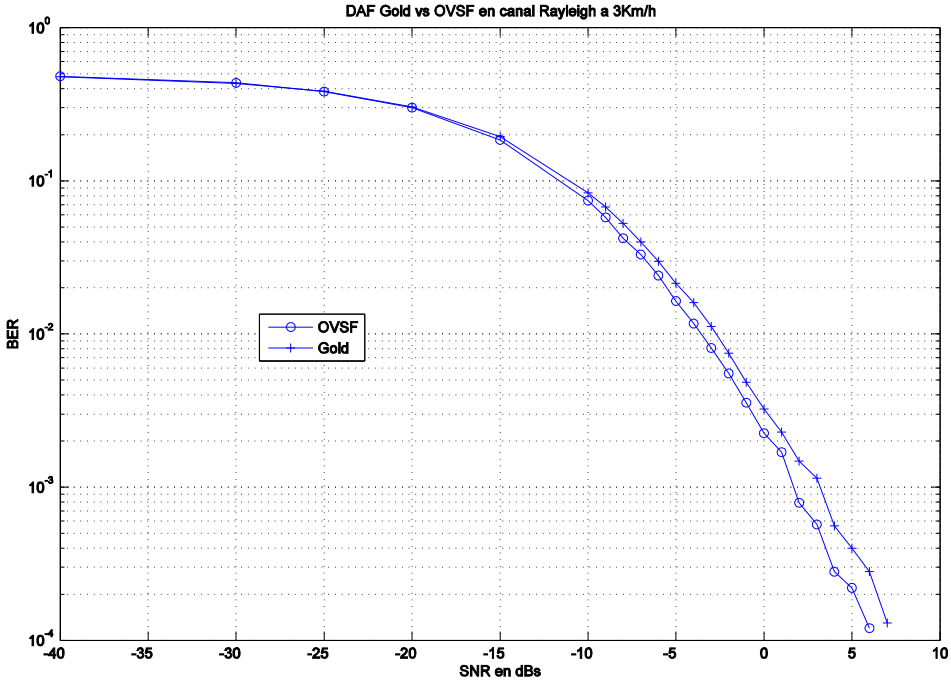


Figura 51. BER en escenario DAF comparando códigos en canal Rayleigh a 3Km/h.

A continuación comparamos las dos técnicas de comunicación cooperativa vistas hasta ahora, Amplify and Forward y Decode and Forward, utilizando un canal Rayleigh. Como vemos en la Figura 52 DAF ofrece mejores prestaciones que la técnica AAF con factor de amplificación 1.

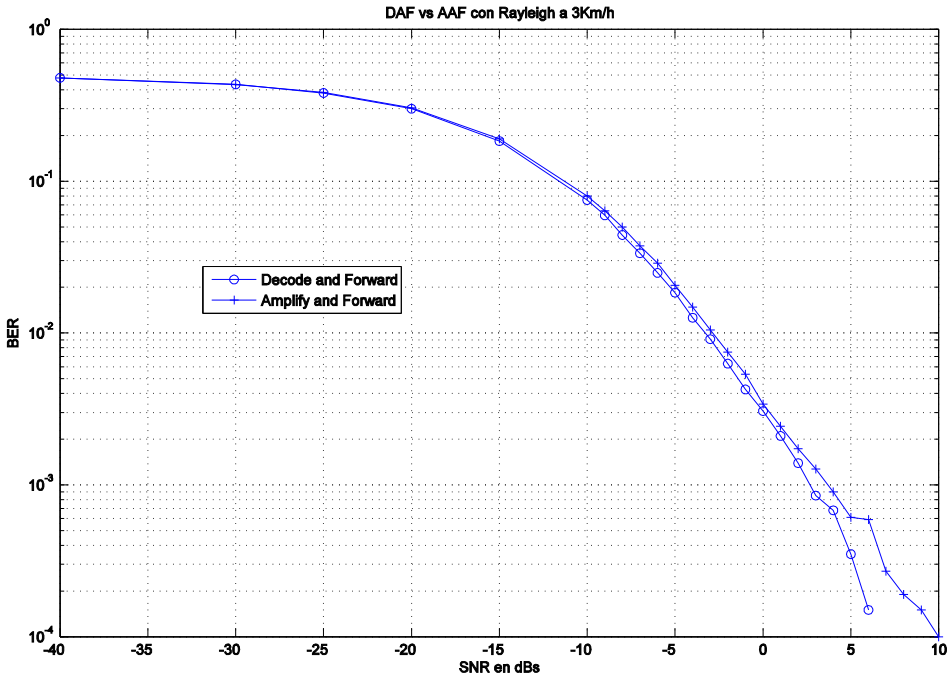


Figura 52. BER comparando escenario AAF y DAF con canal Rayleigh a 3Km/h.

Si además hacemos la comparación, véase la Figura 53, en un canal AWGN con códigos OVSF y una modulación QPSK, añadiendo el escenario de CDMA sin cooperación, vemos como *DAF* ofrece los mejores resultados aunque por muy poca diferencia junto con *AAF*.

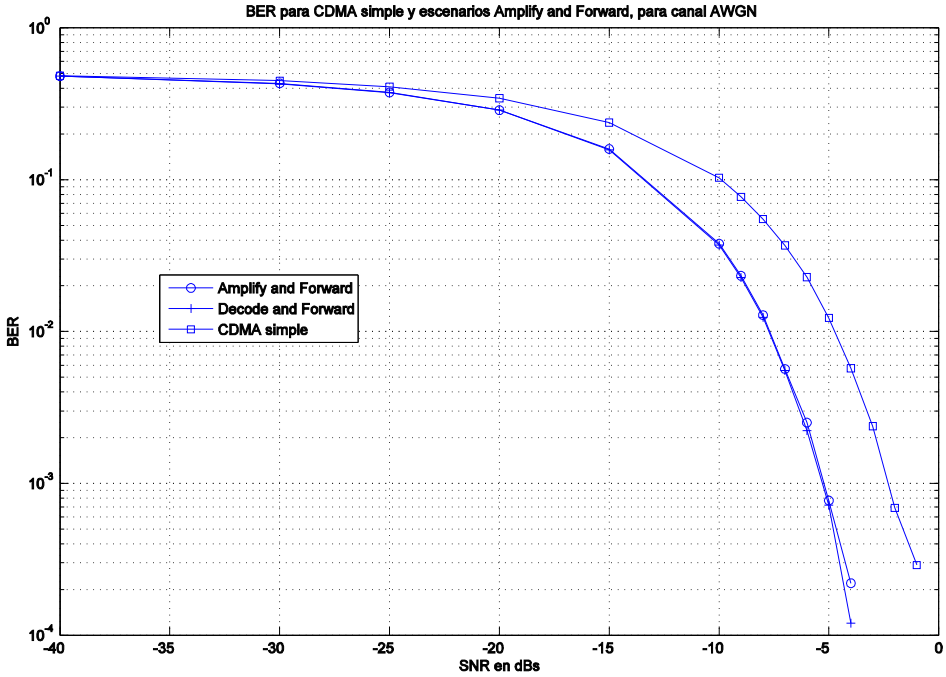


Figura 53. BER comparando escenarios CDMA sin cooperación, Amplify and Forward y Decode and Forward, en canal AWGN.

A continuación, en la Figura 54 vemos como variando la SNR del canal entre los usuarios, llega un momento en que las prestaciones ya no compensan con respecto a no utilizar diversidad cooperativa. En concreto vemos como a -3 dBs todavía sigue siendo efectiva esta técnica, pero a -10 dBs la probabilidad de error ya no desciende por mucho que aumente la SNR del canal hacia el receptor de los dos usuarios.

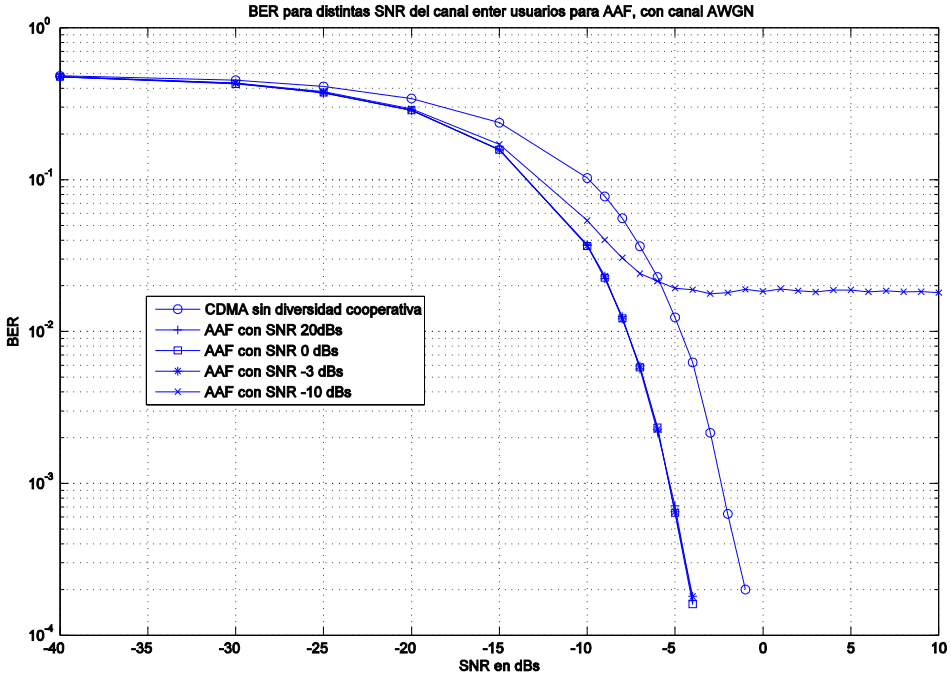


Figura 54. BER para distintos valores de SNR del canal entre usuarios para DAF, con canal AWGN

6.5 Resultados en el escenario de CDMA con *Alamouti*.

Incluimos dentro de este apartado, la técnica de *Alamouti* y la compararemos con la de *Decode and Forward* sin utilizar *Alamouti*.

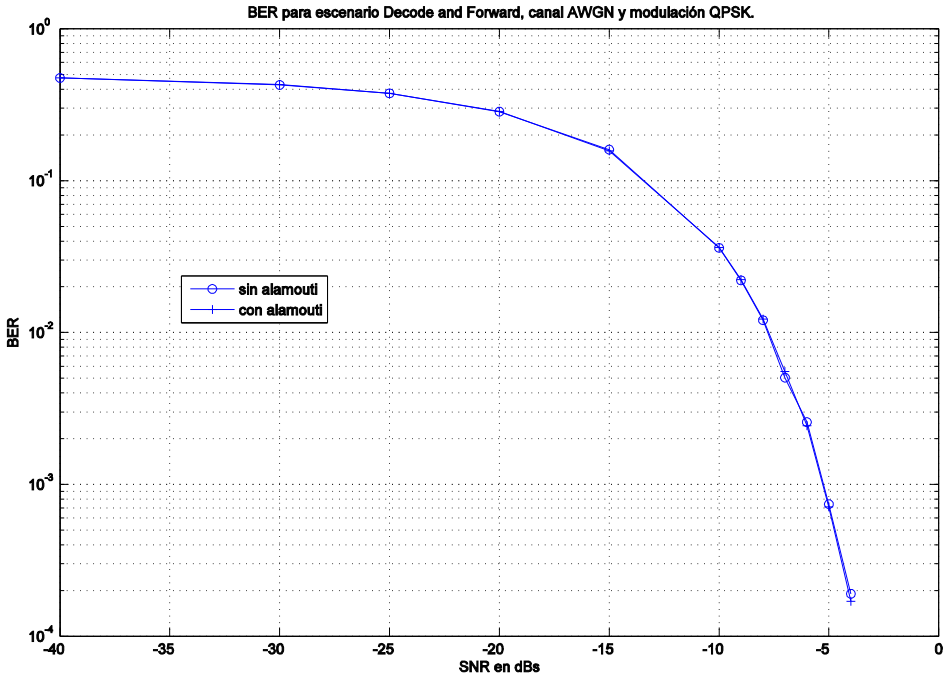


Figura 55. BER comparando la técnica de Alamouti en un canal AWGN.

Como vemos en la Figura 55 las prestaciones en un escenario con canal AWGN son casi idénticas.

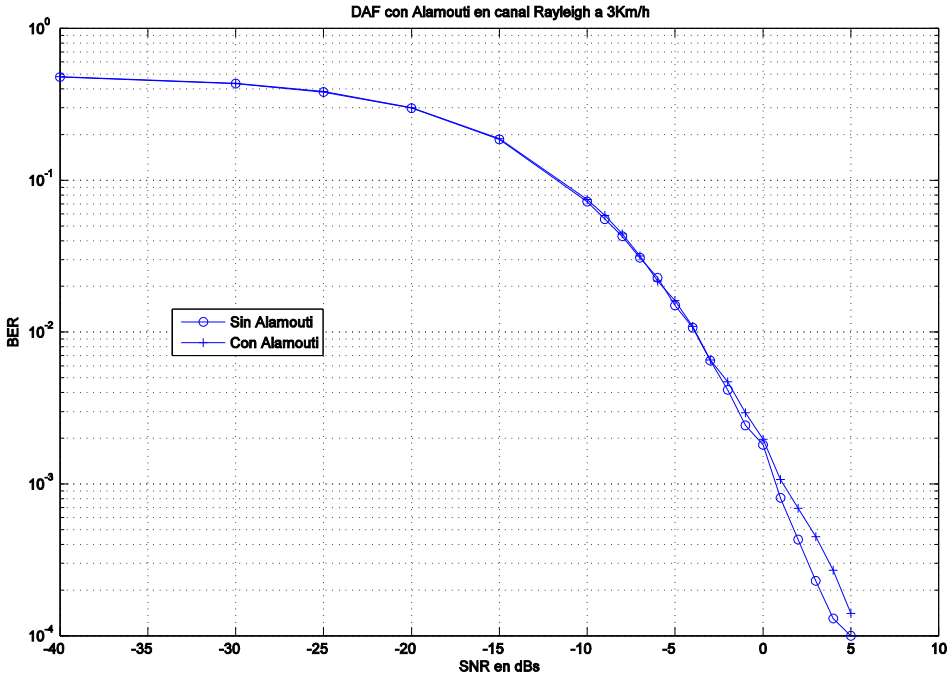


Figura 56. BER comparando la técnica de Alamouti en canal Rayleigh a 3Km/h.

Viendo la Figura 56 vemos como en el caso de utilizar un canal Rayleigh la técnica de *Alamouti* no sale favorecida con respecto a la técnica de *Decode and Forward*.

6.6 Análisis de los resultados.

En este punto ya hemos obtenido resultados de todos los escenarios diseñados con las distintas posibilidades de las que se ha hablado. Por lo tanto podemos hacer un análisis de lo obtenido en las gráficas.

En primer lugar hemos comprobado como los códigos OVSF han obtenido mejores prestaciones en todos los escenarios simulados aunque no debemos olvidar las condiciones en la que se ha simulado, ya que los usuarios envían la información de forma totalmente síncrona. Cabe destacar que en los casos en los que hemos simulado utilizando un canal Rayleigh la diferencia entre los dos tipos de códigos ha sido bastante menor, lo que proporcionaba un comportamiento más aproximado en ambos códigos.

También hemos observado las diferencias entre usar técnicas de comunicación cooperativa o no usarlas. Tanto en la Figura 52 como en la Figura 53, podemos ver como el uso de *Amplify and Forward* o *Decode and Forward* ha mejorado considerablemente las prestaciones. Es importante mencionar que entre estas dos técnicas la diferencia no es tan significativa tanto en un canal AWGN como en un canal Rayleigh. Sin embargo si encontramos más diferencia en el tiempo de procesado, ya que

en el caso de *Decode and Forward*, el tiempo de procesado del escenario es notablemente mayor que en el caso de *Amplify and Forward*.

Es necesario apuntar, en los escenarios de diversidad cooperativa, que el canal utilizado entre los dos usuarios tenía una SNR alta. Debido a esto las prestaciones han mejorado considerablemente. En el momento en que hemos disminuido la SNR del canal entre los usuarios, las prestaciones de estas técnicas han ido decreciendo. En la técnica *Amplify and Forward* en cuanto ese canal llegó al valor de -3 dBs ya no compensaba utilizarla con respecto a no utilizar diversidad cooperativa. En el caso de *Decode and Forward*, ese valor permitía más margen; en concreto a -10 dBs el sistema ya no se comportaba correctamente.

Esto último tiene sentido, ya que cuanto mayor sea el ruido entre los usuarios, la señal que le llegue al receptor, estará mas distorsionada por lo que hará peores estimaciones. También vemos como *Amplify and Forward* es más sensible a este cambio, lo que también es lógico ya que esta técnica amplifica la señal tal y como le llega, por lo que la señal retransmitida conservará el ruido del canal entre usuarios. En el caso de *Decode and Forward,* como el usuario que coopera recupera la información y luego la vuelve a codificar, no arrastra el ruido del canal entre los usuarios, por lo que no es tan sensible a ese cambio en el ruido del canal entre los usuarios.

Hemos comprobado también como el canal AWGN tiene un mejor comportamiento que el canal Rayleigh. Ya comentamos en apartados anteriores que el canal AWGN podría describirse como el canal ideal en comunicaciones, mientras que el canal Rayleigh sería ponernos en un caso más realista. Esto último ha quedado plasmado y hemos visto la diferencia de prestaciones en un caso y otro.

En el caso de las simulaciones realizadas con canal Rayleigh, se ha observado que las prestaciones son más estables a velocidades de 3 Km/h, la de un peatón por ejemplo, que a velocidades como 120 Km/h, la velocidad de un coche en autopista.

Además también comprobamos que en el escenario *Amplify and Forward* el mejor factor de amplificación es ‘1’ para las condiciones en las que estaba diseñado el sistema.

7 Conclusión

Ya hemos visto que CDMA es una tecnología que juega un papel importante en la telefonía móvil 3G. En este proyecto hemos incluido la comunicación cooperativa a esta tecnología con el fin de ver que prestaciones nos aportan.

Tal y como ha resultado de este proyecto podemos ver que la comunicación cooperativa puede acercar grandes mejoras a la tecnología CDMA existente. La aplicación de técnicas como *Amplify and Forward* o *Decode and Forward* pueden mejorar las probabilidades de error de los sistemas, lo cual puede mejorar la velocidad de comunicación y así la calidad del servicio ofrecido.

Observamos como los códigos OVSF son los que ofrecen mejor tasa de error en determinadas circunstancias de los diferentes escenarios, aunque los códigos GOLD son una alternativa interesante en los escenarios con canal Rayleigh ya que el comportamiento tiende a ser más parecido.

Hemos visto que cuando la SNR del canal existente entre los dos usuarios que cooperan en las técnicas de diversidad cooperativa es alta, resultan efectivas las técnicas de *Amplify and Forward*, *Decode and Forward* y *Alamouti*. Bajo las condiciones en las que se han hecho las simulaciones, hemos obtenido los mejores resultados con la técnica de *Decode and Forward*. Esta técnica proporciona mejor calidad en la comunicación aunque con el inconveniente de que el tiempo de procesado es algo mayor que en el caso de *Amplify and Forward.*

Después del análisis de todos los resultados, podemos sacar conclusiones acerca de la utilización de las técnicas de diversidad cooperativa en sistemas CDMA. La técnica de *Decode and Forward*, al ser más tolerante con el ruido del canal entre los usuarios, podría ser más efectiva en escenarios en los que exista más ruido como por ejemplo en zonas urbanas. Por otro lado, la técnica de *Amplify and Forward*, al ser más sensible al ruido del canal entre los usuarios, pero por otro lado obtener buenos resultados y un tiempo de procesado menor que *Decode and Forward*, sería interesante su uso en zonas rurales o suburbanas en las que no existe tanta interferencia.

Por lo tanto, las ventajas que nos aporta *Amplify and Forward* en sistemas CDMA sería que ofrece mejoras en la velocidad de las comunicaciones y ofrece un tiempo de procesado bajo en comparación con otras técnicas de diversidad cooperativa. Las ventajas de *Decode and Forward* son que ofrece unas prestaciones todavía mejores que la anterior técnica y más tolerancia con el ruido en el canal entre los usuarios.

7.1 Trabajos futuros

Tomando como base los trabajos realizados en este proyecto, podrían partir de aquí una serie de trabajos para ahondar más en el tema tratado o añadir algunas mejoras. Veamos algunas de ellas:

- Crear nuevos escenarios con nuevas técnicas de comunicación cooperativa.
- Analizar distintos mecanismos para ver cuando es efectivo usar comunicación cooperativa y cuando no.

- Investigar como se podría controlar el uso de la comunicación cooperativa desde el receptor. Es decir, el receptor con la información que recibe, que sea capaz de ordenar a los usuarios el uso de comunicación cooperativa en el caso de que sea posible.
- Aplicar estos avances en el marco de la telefonía móvil de nueva generación 4G.
- Introducir diferentes rayos en el canal Rayleigh, creando así un canal multitrayecto.

Referencias

[1] **CDMA Development Group: http://www.cdg.org**

[2] **Robert A. Scholtz, IEEE "The Origins of Spread-Spectrum Communications".**

[3] **Prasad, R. "*CDMA for wireless personal communications*" 1996.**

[4] **Flavio Muratore, "UMTS Mobile Communications for the Future" 2001.**

[5] **Aria Nosratinia, University of Texas, Dallas, Todd E. Hunter, Nortel Networks, Ahmadreza Hedayat, University of Texas, Dallas. Artículo de la revista "*IEEE Communications Magazine*" "*Cooperative Communications in Wíreless Networks*". Octubre 2004**

[6] **Izabas M. Alamouti " *A Simple Transmit Diversity Technique for Wireless Communications*" 1998.**

[7] **José M. Hernando Rábanos "Comunicaciones móviles de tercera generación UMTS" 2000**

[8] **The 3rd Generation Partnership Project: Technical Specification Group Radio Access Network; Spreading and Modulation (FDD); http://www.3gpp.org/**

[9] **M.P.M. Hall. "Propagation of radiowaves" 1996**

[10] **Jose María Hernando Rábanos. "Comunicaciones móviles" Segunda edición, 2004.**

[11] **Roger L. Freeman "Radio System Design for Telecommunications" Second Edition, 1997**

[12] **Michel C. Jeruchim, senior member, IEEE "Techniques for Estimating the Bit Error Rate in the Simulation of Digital Communication Systems".**

[13] **Ayuda de MATLAB®.**

[14] **Andrew Richardson, "WCDMA Design Handbook" 2005.**

[15] **Harri Holma, Antti Toskala, "WCDMA for UMTS" 2001.**

[16] **Evelio Martínez, "La evolución de la telefonía móvil", articulo publicado en la revista RED, Mayo 2001.**

ANEXO i

Códigos MATLAB®

A continuación se adjuntan los códigos programados para la simulación de todos los escenarios [13].

autocorr_code.m

```
%PFC Ignacio Mingote Marina - I.T.T.: SISTEMAS DE TELECOMUNICACION
%
%Programa en el que observaremos la autocorrelación que existe entre los
%distintos códigos.

%parametros:
%-----------
%----------- cod: código utilizado
%-----------        'gold' para Gold
%-----------        'ovsf' para Ovsf
%-----------
%----------- cor: matriz de autocorrelaciones de cada uno de los códigos
%                 de la matriz

function [autocor] = autocorr_code(cod);

%Selección de códigos
if (cod == 'gold')
%Cargamos códigos gold
    load('msqcc_co_g31.mat');
    codigos = msqcc_co_g31;
end

if(cod == 'ovsf')
%Matriz de hadamard para los códigos OVSF
    codigos_old = hadamard(32);
%Eliminamos la primera fila ya que no es un código utilizado.
    codigos = codigos_old(2:32,:);
end

%Calculamos la autocorrelación de cada uno de los códigos
for i=1:length(codigos(:,1))
    autocor = xcorr(codigos(i,:));
%Representamos la autocorrelación del código
    stem((-length(autocor)/2)+0.5:1:(length(autocor)/2)-0.5,abs(autocor))
    pause
    close all
end
```

corr_codes.m

```
1   %PFC Ignacio Mingote Marina - I.T.T.: SISTEMAS DE TELECOMUNICACION
2   %
3   %Programa en el que observaremos la correlación que existe entre los
4   %distintos códigos.
5
6   %Parámetros:
7   %-----------
8   %----------- cod: código utilizado
9   %-----------        'gold' para Gold
10  %-----------        'ovsf' para Ovsf
11  %-----------
12  %----------- media_tot_cod: media total de las correlaciones de los códigos
13  %----------- cor: matriz de correlaciones entre todos los códigos
14  %                 de la matriz
15
16  function [cor,media_cor,media_tot_cor] = corr_codes(cod);
17
18  if (cod == 'gold')
19      load('msqcc_co_g31.mat');
20      codigos = msqcc_co_g31;
21  end
22
23  if(cod == 'ovsf')
24      codigos_old = hadamard(32);
25      codigos = codigos_old(2:32,:);
26  end
27
28  %Con esta función mostraremos los coeficientes de correlacion entre los
29  %distintos códigos
30  cor = corr(codigos');
31
32  %También calculamos los coeficientes de correlación medios para
33  %cada uno de los códigos.
34
35  %Para ello primero eliminamos la diagonal que hace referencia a la
36  %autocorrelación de los códigos.
37  cor_diag = abs(cor) - diag(diag(cor));
38
39  media_cor = sum(cor_diag,1)./(length(cor_diag(:,1))-1);
40
41  %Y la media total de las correlaciones de todos los códigos.
42  media_tot_cor = mean(media_cor)
```

cdma.m

```
%PFC Ignacio Mingote Marina - I.T.T.: SISTEMAS DE TELECOMUNICACION
%
%Programa que simula las prestaciones de los distintos códigos utilizados
%en CDMA en un escenario que consistirá en un número de usuarios de 1 a N,
%un canal AWGN o Rayleigh y un receptor.
%
%Analizaremos las prestaciones analizando la probabilidad de error del
%sistema.
%
%Parámetros: cdma(cod, mod, n_users, n_bits, user_ref, snr, channel)
%-----------
%----------- cod: código usado por los usuarios.
%-----------         'g' si es Gold.
%-----------         'w' si es Walsh.
%----------- mod: tipo de modulación usada.
%-----------         'bpsk' para modulación BPSK
%-----------         'qpsk' para modulación QPSK
%----------- n_users: número de usuarios que transmiten.
%----------- n_bits: número de bits que transmiten.
%----------- user_ref: usuario del que el receptor quiere obtener su
%-----------            información
%----------- snr: relación señal a ruido del canal.
%----------- channel: tipo de canal.
%-----------         'awgn' si es AWGN.
%-----------         'rale' si es rayleigh.
%-----------
%----------- p_error: probabilidad de error,BER.

function [p_error] = cdma(cod, mod, n_users, n_bits, user_ref, snr, channel);

%-----------------------------------------------------------%
%----------------PARTE DEL TRANSMISOR---------------------%
%-----------------------------------------------------------%
%Cargo los códigos que voy a utilizar.
if (cod == 'g')
    load('msqcc_co_g31.mat');
    codigos = msqcc_co_g31;
end
if (cod == 'w')
    codigos_walsh = hadamard(32);
%Elimino la primera fila de la matriz ya que no se trata de ningún código
%que vayan a utilizar los usuarios.
    codigos = codigos_walsh(2:32,:);
end

%Se generan bits aleatorios para todos los usuarios.
bits = round(rand(n_users, n_bits));

%Ahora creo la matriz de codigos repetida tantas veces como bits haya.
%Pongo todos los códigos en columna.
a = codigos(1:n_users,:)';
%De cada chip de codigo lo repito como bits se envien.
b = kron(a, ones(1,n_bits));
```

```
%lo coloco todo en una columna los codigos repetidos uno encima del otro.
c = b(:);
%Le doy la vuelta para ponerlo en fila.
d = c';
%Esa matriz la ordeno para que coja todas las muestras que
%equivalen al primer usuario para que esten en una sola fila, así para cada
%usuario. Y le doy la vuelta para que coincidan las dimensiones.
fin = reshape(d,length(codigos(1,:))*n_bits,n_users);
fin = fin';

%--------------------MODULADOR-----------------------------%
if (mod == 'bpsk')
    %---MODULADOR BPSK---%
    bits_mod = bpsk(bits);
    %---MODULADOR BPSK---%
    %El número de símbolos es igual al número de bits
    n_simb = n_bits;
end

if (mod == 'qpsk')
    %---MODULADOR QPSK---%
    bits_mod = qpsk(bits);
    %---MODULADOR QPSK---%

    %Entonces al haber la mitad de símbolos, fin tiene la mitad de
    %longitud.
    fin = fin(:,1:length(fin)/2);

    %En este caso el número de símbolos es la mitad del número de bits.
    n_simb = n_bits/2;
end
%--------------------FIN MODULADOR-----------------------%

%Expando cada bit por el número de chips del código.
bits_expand = zeros(n_users, n_simb*length(codigos(1,:)));

%Por motivos de memoria cuando son 7 usuarios lo hacemos en dos partes.
if (n_users == 7)
    bits_expand(1:4,:) = kron(bits_mod(1:4,:),ones(1,length(codigos(1,:))));
    bits_expand(5:7,:) = kron(bits_mod(5:7,:),ones(1,length(codigos(1,:))));
else
    bits_expand = kron(bits_mod,ones(1,length(codigos(1,:))));
end

%Entonces fin tiene la misma dimension que bits_expand y la puedo
%multiplicar punto a punto.
%Matriz de chips que transmitiran los usuarios. Cada fila un usuario.
tx = bits_expand.*fin;

%-----------------------------------------------------------%
%----------------FIN PARTE DEL TRANSMISOR-----------------%
%-----------------------------------------------------------%
```

```
106
107     %-------------------------------------------------------%
108     %----------------PARTE DEL CANAL ----------------------%
109     %-------------------------------------------------------%
110
111 -   if (channel == 'rale')
112 -       for u=1:n_users,
113 -           [tx_total_after_rl(u,:),coeficientes_filtro(u,:)] = rl(tx(u,:));
114 -       end
115 -   end
116
117 -   for s=1:length(snr),
118 -       if (channel == 'awgn')
119 -           tx_total_channel = awgn(tx,snr(s),mod);
120 -       end
121 -       if (channel == 'rale')
122             %Despues le añado el ruido aditivo.
123 -           tx_total_channel = awgn(tx_total_after_rl,snr(s),mod);
124 -       end
125
126         %-------------------------------------------------------%
127         %----------------FIN PARTE DEL CANAL --------------------%
128         %-------------------------------------------------------%
129
130         %-------------------------------------------------------%
131         %----------------PARTE DEL RECEPTOR----------------------%
132         %-------------------------------------------------------%
133
134       %Si el canal era Rayleigh, multiplico en el receptor por el conjugado de
135       %los coeficientes.
136 -     if (channel == 'rale')
137 -         tx_total_channel = tx_total_channel.*conj(coeficientes_filtro);
138 -     end
139       %Ahora sumo las cadenas de chips de los usuarios como si juntara la señal
140       %de todos ellos.
141 -     tx_total_channel = sum(tx_total_channel,1);
142
143       %Tengo que multiplicar el vector que recibo por el codigo del usuario que
144       %quiero recuperar repetido una vez tras otra para que tenga la misma
145       %dimension.
146 -     rx = tx_total_channel.*fin(user_ref,:);
147
148       %Ahora tengo que dividir la fila de tanto en tanto como número de chips
149       %tenga el código utilizado.
150 -     rx_total = reshape(rx, length(codigos(1,:)),n_simb);
151
152       %Hago un sum para sumar cada columna y luego promediar y decidir.
153 -     rx_total_sum = sum(rx_total)./length(codigos(1,:));
154
155       %Tengo que decidir si el bit era un '1' o un '0' para ello tengo que poner
156       %un umbral.
157       %En el caso de que la señal hubiera sido modulada con QPSK, tenemos que
158       %demodularla:
```

```
159 -       if (mod == 'qpsk')
160             %Separamos la parte real y la parte imaginaria.
161 -           p_real = real(rx_total_sum);
162 -           p_imag = imag(rx_total_sum);
163             %Ahora tenemos que intercalar las partes para tener los bits ordenados
164 -           inf = [p_real p_imag];
165 -           rx_total_sum = intercalar_simple(inf);
166 -       end
167 -       informacion_recuperada = rx_total_sum<0;
168         %Calculo la probabilidad de error como el número de errores cometidos
169         %entre el número de bits totales. BER
170 -       p_error(s) = sum(abs(informacion_recuperada - bits(user_ref,:)))/n_bits
171         %Por Monte Carlo
172 -       if (p_error(s) < 0.0001)
173 -           p_error(s) = 0;
174 -       end
175 -   end
```

bpsk.m

```
1   %PFC Ignacio Mingote Marina - I.T.T.: SISTEMAS DE TELECOMUNICACION
2   %
3   %--------- MODULADOR BPSK --------
4   %Modulo los bits de forma que los 1 son -1 y los 0 son 1.
5   %    1 ---> -1
6   %    0 ---> 1
7   %Parámetros:
8   %----------- bits: bits entrantes al modulador.
9   %-----------
10  %----------- bits_modulados: bits modulados.
11
12  function [bits_modulados] = bpsk(bits);
13
14  %Busco en que posiciones están los bits que son 0.
15  bits_zero = find(bits==0);
16  %Busco en que posiciones están los bits que son 1
17  bits_uno = find(bits==1);
18  %Convierto esos bits en tanto en 1 como en -1 respectivamente,
19  %inicializo la matriz de bits modulados
20  bits_modulados = zeros(length(bits(:,1)),length(bits(1,:)));
21  bits_modulados(bits_zero) = 1;
22  bits_modulados(bits_uno) = -1;
```

qpsk.m

```
1   %PFC Ignacio Mingote Marina - I.T.T.: SISTEMAS DE TELECOMUNICACION
2   %
3   %--------- MODULADOR QPSK --------
4   %Parámetros:
5   %----------- bits: bits entrantes al modulador.
6   %-----------
7   %----------- bits_modulados: bits modulados.
8
9   function [bits_modulados] = qpsk(bits);
10
11  %Busco en que posiciones están los bits que son 0.
12  bits_zero = find(bits==0);
13  %Busco en que posiciones están los bits que son 1
14  bits_uno = find(bits==1);
15  %Convierto esos bits en tanto en 1 como en -1 respectivamente,
16  %inicializo la matriz de bits modulados.
17  bits_modulados = zeros(length(bits(:,1)),length(bits(1,:)));
18  bits_modulados(bits_zero) = 1;
19  bits_modulados(bits_uno) = -1;
20
21  jota = [1 j]
22  jota = kron(jota,ones(1,length(bits(1,:))/2));
23  %Ahora intercambio el vector para que me queden las 'j' en las
24  %posiciones pares
25  jota_d = intercalar_simple(jota);
26
27  %Ahora repito el vector tantas veces como usuarios haya.
28  jota_new = kron(jota_d,ones(length(bits(:,1)),1));
29  %Multiplico las j por los bits correspondientes.
30  bits_modulados = bits_modulados.*jota_new;
31  %Sumo los bits de dos en dos formando símbolos con parte real e
32  %imaginaria
33  bits_modulados = dyaddown(bits_modulados,1) + dyaddown(bits_modulados,2);
```

intercalar_simple.m

```
1   %PFC Ignacio Mingote Marina - I.T.T.: SISTEMAS DE TELECOMUNICACION
2   %
3   %Función que intercala los elementos de un vector
4
5   function [vector_new] = intercalar_simple(vector_old);
6
7 - vector_aa = reshape(vector_old, length(vector_old)/2, 2);
8 - vector_bb = vector_aa';
9 - vector_cc = vector_bb(:);
10 - vector_new = vector_cc';
```

intercalar.m

```
%PFC Ignacio Mingote Marina - I.T.T.: SISTEMAS DE TELECOMUNICACION
%
%Función que intercala los elementos de un vector
%Igual que intercalar_simple.m pero cuando la señal está todavía ensanchada

function [vector_new] = intercalar(vector_old, n_simb, long_codes);

aa = [vector_old(1,1:n_simb*long_codes);vector_old(1,(n_simb*long_codes)+1:length(vector_old))];
bb = aa(:);
cc = bb';
dd = reshape(cc,long_codes*2,n_simb);
ee = dd';
ff = [dyaddown(ee,1) dyaddown(ee,2)];
gg = ff';
hh = gg(:);
vector_new = hh';
```

awgn.m

```
%PFC Ignacio Mingote Marina - I.T.T.: SISTEMAS DE TELECOMUNICACION
%
%Programa que simula un CANAL AWGN
%
%Parámetros:
%----------- chips_entrantes: vector de chips que emite el usuario.
%----------- snr: relación señal a ruido del canal.
%----------- mod: modulación utilizada
%-----------
%----------- chips_salientes: vector de chips después de pasar por el
%                             canal.
function [chips_salientes] = awgn(chips_entrantes, snr, mod);

%-------------------------------------------------------%
%-----------------------CANAL AWGN----------------------%
%-------------------------------------------------------%

% Para simular el canal AWGN tengo que generar tantas muestras aleatorias
% como la longitud del vector fila 'chips_entrantes', de una distribución
% gaussiana con media cero y varianza la potencia de ruido del canal.
% Y tantos vectores como usuarios en el sistema.

%número de chips por usuario
lce = length(chips_entrantes);
%número de usuarios
ndu = length(chips_entrantes(:,1));

%Sabiendo que  SNR (dB) = (Potencia de la señal) / (Potencia de ruido)
%Primero voy a calcular la potencia de la señal de cada usuario.

p_tx = sum(abs(chips_entrantes').^2)/lce;

%La snr la tengo que pasar a unidades naturales ya que se me pasa en dB's

snr_nat = (10^(snr/10));

%Ya puedo calcular la potencia del ruido que será la varianza de la
%gaussiana

N = p_tx/snr_nat;

%De esta forma tenemos en cuenta la modulación usada.
n=[ndu,lce];

if (mod == 'bpsk')
n = kron(sqrt(N'), ones(1,lce)).*randn(ndu,lce);
else
```

```
48 -  n = kron(sqrt(N'/2), ones(1,lce)).*randn(ndu,lce) + kron(sqrt(N'/2), ones(1,lce)).*randn(ndu,lce)*i;
49 -  end
50
51    %Ahora sumo punto a punto el vector de ruido con el vector a
52    %transmitir.
53
54 -  chips_salientes = n+chips_entrantes;
55    %-------------------------------------------------------%
56    %--------------------FIN CANAL AWGN---------------------%
57    %-------------------------------------------------------%
```

cdma_aaf.m

```
1   %PFC Ignacio Mingote Marina - I.T.T.: SISTEMAS DE TELECOMUNICACION
2   %
3   %Programa que simula las prestaciones de distintos códigos utilizados en
4   %CDMA en un escenario de diversidad cooperativa, en el que 2 usuarios (Full
5   %Duplex) cooperan entre sí usando Amplify and Forward.
6   %
7   %Parámetros: cdma_aaf(cod, mod, n_bits, snr, snr_users, A, channel)
8   %-----------
9   %----------- cod: código usado por los usuarios.
10  %-----------        'g' si es Gold.
11  %-----------        'w' si es Walsh.
12  %----------- mod: modulación utilizada
13  %-----------        'bpsk' para BPSK
14  %-----------        'qpsk' para QPSK
15  %----------- n_bits: número de bits que transmiten.
16  %----------- snr: relación señal a ruido del canal.
17  %----------- snr_users: relación señal a ruido entre los usuarios.
18  %----------- A: factor de amplificacion.
19  %----------- channel: tipo de canal.
20  %-----------        'awgn' si es AWGN.
21  %-----------        'rale' si es Rayleigh.
22  %-----------
23  %----------- p_error: probabilidad de error,BER.
24
25  function [p_error] = cdma_aaf(cod, mod, n_bits, snr, snr_users, A, channel);
26
27  %-----------------------------------------------------------%
28  %-------PARTE DE LA COMUNICACION ENTRE USUARIOS-----------%
29  %-----------------------------------------------------------%
30
31  % Tenemos fijado el número de usuarios del sistema: 2
32  n_users = 2;
33
34  %Lo primero que voy a hacer es generar bits aleatorios para los dos usuarios.
35  bits = round(rand(n_users, n_bits));
36
37  %Cargo los códigos que voy a utilizar.
38  if (cod == 'g')
39      load('msqcc_co_g31.mat');
40      codigos = msqcc_co_g31;
41  end
42  if (cod == 'w')
43      codigos_walsh = hadamard(32);
44  %Elimino la primera fila de la matriz ya que no se trata de ningún código
45  %que vayan a utilizar los usuarios.
46      codigos = codigos_walsh(2:32,:);
47  end
48
49
```

```
50    %Ahora creo la matriz de codigos repetida tantas veces como bits haya.
51    %Pongo todos los códigos en columna.
52 -  a = codigos(1:n_users,:)';
53    %De cada chip de codigo lo repito como bits se envien.
54 -  b = kron(a, ones(1,n_bits));
55    %lo coloco todo en una columna los codigos repetidos uno encima del otro.
56 -  c = b(:);
57    %Le doy la vuelta para ponerlo en fila.
58 -  d = c';
59    %Esa matriz la ordeno para que coja todas las muestras que
60    %equivalen al primer usuario para que esten en una sola fila, así para cada
61    %usuario. Y le doy la vuelta para que coincidan las dimensiones.
62 -  fin = reshape(d,length(codigos(1,:))*n_bits,n_users);
63 -  fin = fin';
64
65    %-------------------MODULADOR----------------------------%
66 -  if (mod == 'bpsk')
67        %---MODULADOR BPSK---%
68 -      bits_mod = bpsk(bits);
69        %---MODULADOR BPSK---%
70        %El número de símbolos es igual al número de bits
71 -      n_simb = n_bits;
72 -  end
73
74 -  if (mod == 'qpsk')
75        %---MODULADOR QPSK---%
76 -      bits_mod = qpsk(bits);
77        %---MODULADOR QPSK---%
78
79        %Entonces al haber la mitad de símbolos, fin tiene la mitad de
80        %longitud.
81 -      fin = fin(:,1:length(fin)/2);
82
83        %En este caso el número de símbolos es la mitad del número de bits.
84 -      n_simb = n_bits/2;
85 -  end
86    %--------------------FIN MODULADOR-----------------------%
87
88    %Expando cada bit por el número de chips del código.
89 -  bits_expand = kron(bits_mod,ones(1,length(codigos(1,:))));
90
91    %Entonces fin tiene la misma dimension que bits_expand y la puedo
92    %multiplicar punto a punto.
93    %Matriz de chips que transmitiran los usuarios. Cada fila un usuario.
94 -  tx = bits_expand.*fin;
95
96    %Los bits de los dos usuarios pasan por el canal inter-usuario.
97    %Consideraremos este canal inter-usuario del tipo AWGN
98
```

```
 99    %------------------CANAL ENTRE USUARIOS------------------%
100
101    %Los bits que llegan al usuario 2 del usuario 1.
102 -  rx_2 = awgn(tx(1,:),snr_users,mod);
103    %Los bits que llegan al usuario 1 del usuario 2.
104 -  rx_1 = awgn(tx(2,:),snr_users,mod);
105
106    %------------------FIN CANAL ENTRE USUARIOS--------------%
107
108    %------------------AMPLIFICACION DE SEÑAL----------------%
109
110    %Amplifico los chips que recibe el usuario 2 del usuario 1.
111 -  rx_2 = rx_2.*A;
112    %Amplifico los chips que recibe el usuario 1 del usuario 2.
113 -  rx_1 = rx_1.*A;
114
115    %--------------FIN AMPLIFICACION DE SEÑAL----------------%
116
117    %Los chips que enviara el usuario 1 será: sus chips y los chips recibidos del
118    %compañero 2. Y viceversa.
119
120    % -- USER 1 --
121    % Transmitirá sus bits y los bits del usuario 2.
122 -  tx_u1 = [tx(1,:) rx_1];
123
124    % -- USER 2 --
125    % Transmitirá sus bits y los bits del usuario 1.
126 -  tx_u2 = [tx(2,:) rx_2];
127
128    %La matriz de chips que envían los usuarios queda:
129 -  tx_total= [tx_u1;tx_u2];
130
131    %-------------------------------------------------------%
132    %-------FIN PARTE DE LA COMUNICACION ENTRE USUARIOS------%
133    %-------------------------------------------------------%
134
135    %Ahora recoloco la matriz de modo a como se enviarían los bits en el caso
136    %real
137    % User 1 :    b1_u1   b1_u2   b2_u1  b2_u2
138    % User 2 :    b1_u2   b1_u1   b2_u2  b2_u1
139
140    %          TS1                   TS2
141    %     b1_u1 + b1_u2      b1_u1rx + b1_u2rx
142
143 -  tx_total(1,:) = intercalar(tx_total(1,:), n_simb, length(codigos(1,:)));
144 -  tx_total(2,:) = intercalar(tx_total(2,:), n_simb, length(codigos(2,:)));
145
146    %Ahora los datos pasan por al canal hacia el receptor.
147
```

```
148     %-------------------------------------------------------%
149     %----------------PARTE DEL CANAL ------------------------%
150     %-------------------------------------------------------%
151
152 -   if (channel == 'rale')
153 -       for n=1:n_users,
154 -           [tx_total_after_rl(n,:),coeficientes_filtro(n,:)] = rl(tx_total(n,:));
155 -       end
156 -   end
157
158 -   for s=1:length(snr),
159 -       if (channel == 'awgn')
160 -           tx_total_channel = awgn(tx_total,snr(s),mod);
161 -       end
162 -       if (channel == 'rale')
163             %Despues le añado el ruido aditivo.
164 -           tx_total_channel = awgn(tx_total_after_rl,snr(s),mod);
165 -       end
166
167         %-------------------------------------------------------%
168         %----------------FIN PARTE DEL CANAL --------------------%
169         %-------------------------------------------------------%
170
171         %-------------------------------------------------------%
172         %----------------PARTE DEL RECEPTOR----------------------%
173         %-------------------------------------------------------%
174         %Ahora la energía de señal que me llega es mayor, ya que tengo el doble de
175         %información para decidir si es un 0 o un 1.
176
177         %Si el canal era Rayleigh, multiplico en el receptor por el conjugado de
178         %los coeficientes.
179 -       if (channel == 'rale')
180 -           tx_total_channel = tx_total_channel.*conj(coeficientes_filtro);
181 -       end
182
183         %Ahora ya sumo los chips punto a punto.
184 -       tx_total_channel = sum(tx_total_channel,1);
185
186         %Tengo que multiplicar el vector que recibo por el codigo del usuario que
187         %quiero recuperar repetido una vez tras otra para que tenga la misma
188         %dimension.
189 -       fin_div = [fin(1,:) fin(1,:)];
190 -       rx_total_channel = tx_total_channel.*fin_div(1,:);
191
192         %Ahora tengo que dividir la fila de tantos en tantos chips como tenga el
193         %código, multiplicado por dos, ya que los bits retransmitidos vienen a
194         %continuacion.
195 -       rx_recived = reshape(rx_total_channel,length(codigos(1,:))*2,n_simb);
196 -       rx = sum(rx_recived,1)./(length(codigos(1,:))*2);
197
```

```
198         %En el caso de que la señal hubiera sido modulada con QPSK, tenemos que
199         %demodularla:
200 -       if (mod == 'qpsk')
201             %Separamos la parte real y la parte imaginaria.
202 -           p_real = real(rx);
203 -           p_imag = imag(rx);
204             %Ahora tenemos que intercalar las partes para tener los bits ordenados
205 -           inf = [p_real p_imag];
206 -           rx = intercalar_simple(inf);
207 -       end
208
209         %Tengo que decidir si el bit era un '1' o un '0' para ello tengo que poner
210         %un umbral.
211 -       informacion_recuperada = rx<0;
212
213         %Calculo la probabilidad de error.
214 -       p_error(s) = sum(abs(informacion_recuperada - bits(1,:)))/n_bits
215         %Por Monte Carlo
216 -       if (p_error(s) < 0.0001)
217 -           p_error(s) = 0;
218 -       end
219 -   end
```

rl.m

```
%PFC Ignacio Mingote Marina - I.T.T.: SISTEMAS DE TELECOMUNICACION
%
%Programa que simula un CANAL RAYLEIGH.

function [chips_out,coeficientes_filtro] = rl(chips_in);

% Utilizo la función doppler que me genera los coeficientes por lo que
%multiplico mi señal.

[chips_out,coeficientes_filtro] = doppler_effect_v2(chips_in);

```

doppler_effect.m

```
%PFC Ignacio Mingote Marina - I.T.T.: SISTEMAS DE TELECOMUNICACION
%
% Programa que genera los coeficientes necesarios en un canal Rayleigh.
%
% Parámetros:
%----------- chips: chips enviados por el usuario.
%-----------
%----------- coeff_ped_i: coeficientes del canal Rayleigh para ese usuario
%----------- tramatot_concanal: chips resultantes del usuario.

function [tramatot_concanal,coeff_ped_i] = doppler_effect_v2(chips);

% Duracion total de la trama UWB (todos los simbolos)
tramatot_size=length(chips);

%-----------------DATOS-------------------
% Frecuencia de muestreo
fm=3.84e6;
% Frecuencia de portadora UP.
fc=2.15e9;
% Velocidad de la luz
c=3e8;
% Longitud de onda
lambda=c/fc;

ntaps_ped=1;   %Ya que solo consideramos un rayo. (sin multipath)

% Atenuaciones y dispersion en tiempo.
attdB_ped=[0 -9.7 -19.2 -22.8];
att_ped=10.^(attdB_ped/10);
delay_ped=[0 110 190 410]*10^(-9);  % en nseg
delay_muest_ped=round(delay_ped*fm);   % en muestras

%%% Efecto Doppler %%%
%%%%%%%%%%%%%%%%% Pedestrian (3Km/h)% %Vehicle (120Km/h)%%%%%%%%%%%%%%%%
% Velocidad de un peatón o vehículo en m/s
v = 3000/3600;
% frecuencia Doppler
fd_ped=v/lambda;
%Factor variable. Ajustable.
fm1=fd_ped*2000;
% factor de interpolacion
R=floor(fm/fm1);
% Coeficientes Doppler
b_ped=doppler_impulse(fd_ped,fm1,256);

% figure; plot(abs(b_ped))
% title('Coeficientes Doppler Pedestrian')
% pause
% figure; plot(fftshift(abs(fft(b_ped))))
% title('Espectro Doppler Pedestrian')

```

```
53    % Generamos los coef de cada tap
54    % Muestras de ruido para generar los coef. Doppler
55 -  nmuest=ceil(tramatot_size/R)+256;
56 -  coeff_ped=zeros(ntaps_ped,nmuest);
57    %Creamos los vectores donde guardamos los coeff.
58 -  coeff_ped_trunc=[];
59 -  coeff_ped_i=[];
60    %Por cada rayo unos coeficicentes H.
61 -  for k=1:ntaps_ped,
62 -      n_ped=randn(1,nmuest)+j*randn(1,nmuest);
63 -      coeff_ped(k,:)=filter(b_ped,1,n_ped);
64    %       size(coeff_ped(k,:))
65 -      coeff_ped_trunc(k,:)=coeff_ped(k,256:nmuest);
66          %       size(coeff_ped_trunc(k,:))
67 -      num_coeff=length(coeff_ped_trunc(k,:));
68          % eje de tiempos antes de la interpolacion
69 -      t=[0:1/fm:ceil(tramatot_size/R)/fm].';
70           % eje de tiempos despues de la interpolacion
71 -      ti=[0:ceil(tramatot_size/R)/fm/(tramatot_size-1):ceil(tramatot_size/R)/fm].';
72           % eje de tiempos antes de la interpolacion
73           % x=(linspace(1,R*num_coeff,num_coeff)).';
74           % eje de tiempos despues de la interpolacion
75           % xi=(linspace(1,R*num_coeff,R*num_coeff)).';
76 -      coeff_ped_i_real(k,:)=interp1q(t,real(coeff_ped_trunc(k,:)).',ti);
77 -      coeff_ped_i_imag(k,:)=interp1q(t,imag(coeff_ped_trunc(k,:)).',ti);
78 -      coeff_ped_i=coeff_ped_i_real+j*coeff_ped_i_imag;
79 -      pot_coeff=sum(abs(coeff_ped_i(k,:)).^2)/length(coeff_ped_i(k,:));
80 -      coeff_ped_i(k,:)=coeff_ped_i(k,:)/sqrt(pot_coeff);
81 -      pot_coeff=sum(abs(coeff_ped_i(k,:)).^2)/length(coeff_ped_i(k,:));
82 -  end
83    % ------- Pasamos la señal por el canal MP ------- %
84 -  long=min(length(coeff_ped_i(1,:)),length(chips));
85
86 -  trama1=att_ped(1)*[zeros(1,delay_muest_ped(1)) chips(1:long-delay_muest_ped(1))].*coeff_ped_i(1,1:long);
87    % trama2=att_ped(2)*[zeros(1,delay_muest_ped(2)) chips(1:long-delay_muest_ped(2))].*coeff_ped_i(2,1:long);
88    % trama3=att_ped(3)*[zeros(1,delay_muest_ped(3)) chips(1:long-delay_muest_ped(3))].*coeff_ped_i(3,1:long);
89    % trama4=att_ped(4)*[zeros(1,delay_muest_ped(4)) chips(1:long-delay_muest_ped(4))].*coeff_ped_i(4,1:long);
90    % trama5=att_ped(5)*[zeros(1,delay_muest_ped(5)) chips(1:long-delay_muest_ped(5))].*coeff_ped_i(5,1:long);
91    % trama6=att_ped(6)*[zeros(1,delay_muest_ped(6)) chips(1:long-delay_muest_ped(6))].*coeff_ped_i(6,1:long);
92
93 -  tramatot_concanal=trama1;   % +trama2+trama3+trama4+trama5+trama6;
```

doppler_impulse.m

```
%PFC Ignacio Mingote Marina - I.T.T.: SISTEMAS DE TELECOMUNICACION
%
%Programa que simula un filtro de efecto Doppler. Calcula los coeficientes
%del filtro de efecto Doppler.
%
%Parámetros:
%----------- fd: Frecuencia de Dopler
%----------- fs: Frecuencia de muestreo
%----------- N: Número de muestras
%-----------
%----------- r: coeficientes del filtro Doppler.

function [r] = doppler_impulse(fd,fs,N);

%----------------FILTRO DE EFECTO DOPPLER-----------------%
sum=0;
half=0.5*(N-1);
double t=0;
w=hamming(N);
k=0;
t=[];
for k=1:N
    bessel = besselj(0.25,2*pi*fd*abs(k-half)/fs;
    r(k)=w(k)*2^(1/4)*(2*pi*fd*abs(k-half)/fs)^(-1/4)*gamma(3/4)*bessel;
end
l=1;
for k=1:(N/2)
    r_real(k)=r(l);
    l=l+1;
    r_imag(k)=r(l);
    l=l+1;
    t=[t;(k)/fs];
    sum=sum+(r_real(k)^2+r_imag(k)^2);
end
r_real=r_real/sqrt(sum);
r_imag=r_imag/sqrt(sum);
r=r_real+i.*r_imag;
%---------------FIN DE FILTRO DOPPLER--------------------%
```

cdma_daf.m

```
%PFC Ignacio Mingote Marina - I.T.T.: SISTEMAS DE TELECOMUNICACION
%
%Programa que simula las prestaciones de distintos códigos utilizados en
%CDMA en un escenario de diversidad cooperativa, en el que 2 usuarios (Full
%Duplex) cooperan entre sí usando Decode and Forward.
%
%Parámetros: cdma_daf(cod, mod, n_bits, snr, snr_users, channel, alamouti)
%-----------
%----------- cod: código usado por los usuarios.
%-----------        'g' si es Gold.
%-----------        'w' si es Walsh.
%----------- mod: modulación utilizada
%-----------        'bpsk' para BPSK
%-----------        'qpsk' para QPSK
%----------- n_bits: número de bits que transmiten.
%----------- snr: relación señal a ruido del canal.
%----------- snr_users: relación señal a ruido entre los usuarios.
%----------- channel: tipo de canal.
%-----------        'awgn' si es AWGN.
%-----------        'rale' si es Rayleigh.
%----------- alamouti: técnica de Alamouti (QPSK).
%-----------        'si' se utiliza Alamouti.
%-----------        'no' no se utiliza Alamouti.
%-----------
%----------- p_error: probabilidad de error,BER.

function [p_error] = cdma_daf(cod, mod, n_bits, snr, snr_users, channel, alamouti);

%-----------------------------------------------------------%
%-------PARTE DE LA COMUNICACION ENTRE USUARIOS-----------%
%-----------------------------------------------------------%

% Tenemos fijado el número de usuarios del sistema: 2
n_users = 2;

%Lo primero que voy a hacer es generar bits aleatorios para los dos usuarios.
bits = round(rand(n_users, n_bits));

%Cargo los códigos que voy a utilizar.
if (cod == 'g')
    load('msqcc_co_g31.mat');
    codigos = msqcc_co_g31;
end
if (cod == 'w')
    codigos_walsh = hadamard(32);
%Elimino la primera fila de la matriz ya que no se trata de ningún código
%que vayan a utilizar los usuarios.
    codigos = codigos_walsh(2:32,:);
end
```

```
50
51    %Ahora creo la matriz de codigos repetida tantas veces como bits haya.
52    %Pongo todos los códigos en columna.
53 -  a = codigos(1:n_users,:)';
54    %De cada chip de codigo lo repito como bits se envien.
55 -  b = kron(a, ones(1,n_bits));
56    %lo coloco todo en una columna los codigos repetidos uno encima del otro.
57 -  c = b(:);
58    %Le doy la vuelta para ponerlo en fila.
59 -  d = c';
60    %Esa matriz la ordeno para que coja todas las muestras que
61    %equivalen al primer usuario para que esten en una sola fila, así para cada
62    %usuario. Y le doy la vuelta para que coincidan las dimensiones.
63 -  fin = reshape(d,length(codigos(1,:))*n_bits,n_users);
64 -  fin = fin';
65
66    %-------------------MODULADOR---------------------------%
67 -  if (mod == 'bpsk')
68        %---MODULADOR BPSK---%
69 -      bits_mod = bpsk(bits);
70        %---MODULADOR BPSK---%
71        %El número de símbolos es igual al número de bits
72 -      n_simb = n_bits;
73 -  end
74
75 -  if (mod == 'qpsk')
76        %---MODULADOR QPSK---%
77 -      bits_mod = qpsk(bits);
78        %---MODULADOR QPSK---%
79
80        %Entonces al haber la mitad de símbolos, fin tiene la mitad de
81        %longitud.
82 -      fin = fin(:,1:length(fin)/2);
83
84        %En este caso el número de símbolos es la mitad del número de bits.
85 -      n_simb = n_bits/2;
86 -  end
87    %--------------------FIN MODULADOR-----------------------%
88
89    %Expando cada bit por el número de chips del código.
90 -  bits_expand = kron(bits_mod,ones(1,length(codigos(1,:))));
91
92    %Entonces fin tiene la misma dimension que bits_expand y la puedo
93    %multiplicar punto a punto.
94    %Matriz de chips que transmitiran los usuarios. Cada fila un usuario.
95 -  tx = bits_expand.*fin;
96
97    %Los bits de los dos usuarios pasan por el canal inter-usuario.
98    %Consideraremos este canal inter-usuario del tipo AWGN
99
```

```
100    %------------------CANAL ENTRE USUARIOS------------------%
101
102    %Los bits que llegan al usuario 2 del usuario 1.
103 -  rx_2 = awgn(tx(1,:),snr_users,mod);
104    %Los bits que llegan al usuario 1 del usuario 2.
105 -  rx_1 = awgn(tx(2,:),snr_users,mod);
106
107    %------------------FIN CANAL ENTRE USUARIOS--------------%
108
109    %-------------DETECCIÓN DE LA INFORMACIÓN----------------%
110
111    %USER 1
112    %Sigue el proceso habitual para decodificar la información recibida y
113    %reenviarla.
114
115 -  rx_1 = rx_1.*fin(2,:);
116
117    %Ahora tengo que dividir la fila de tanto en tanto como número de chips
118    %tenga el código utilizado.
119 -  rx_uno = reshape(rx_1, length(codigos(1,:)),n_simb);
120
121    %Hago un sum para sumar cada columna y luego promediar y decidir.
122 -  rx_total_1 = sum(rx_uno)./length(codigos(1,:));
123
124    %Tengo que decidir si el bit era un '1' o un '0' para ello tengo que poner
125    %un umbral.
126    %En el caso de que la señal hubiera sido modulada con QPSK, tenemos que
127    %demodularla:
128 -  if (mod == 'qpsk')
129        %Separamos la parte real y la parte imaginaria.
130 -      p_real_1 = real(rx_total_1);
131 -      p_imag_1 = imag(rx_total_1);
132        %Ahora tenemos que intercalar las partes para tener los bits ordenados
133 -      inf_1 = [p_real_1 p_imag_1];
134 -      rx_total_1 = intercalar_simple(inf_1);
135 -  end
136
137 -      informacion_recuperada_1 = rx_total_1<0;
138
139
140    %USER 2
141    %Sigue el proceso habitual para decodificar la información recibida y
142    %reenviarla.
143
144 -  rx_2 = rx_2.*fin(1,:);
145
146    %Ahora tengo que dividir la fila de tanto en tanto como número de chips
147    %tenga el código utilizado.
148 -  rx_dos = reshape(rx_2, length(codigos(1,:)),n_simb);
149
150    %Hago un sum para sumar cada columna y luego promediar y decidir.
151 -  rx_total_2 = sum(rx_dos)./length(codigos(1,:));
152
```

```
153     %Tengo que decidir si el bit era un '1' o un '0' para ello tengo que poner
154     %un umbral.
155     %En el caso de que la señal hubiera sido modulada con QPSK, tenemos que
156     %demodularla:
157 -   if (mod == 'qpsk')
158         %Separamos la parte real y la parte imaginaria.
159 -       p_real_2 = real(rx_total_2);
160 -       p_imag_2 = imag(rx_total_2);
161         %Ahora tenemos que intercalar las partes para tener los bits ordenados
162 -       inf_2 = [p_real_2 p_imag_2];
163 -       rx_total_2 = intercalar_simple(inf_2);
164 -   end
165
166     %Por lo tanto la información recuperada por el usuario 2 es la siguiente.
167 -   informacion_recuperada_2 = rx_total_2<0;
168
169     %La información recuperada por los dos usuarios es la siguiente.
170 -   bits_re = [informacion_recuperada_1; informacion_recuperada_2];
171
172     %Calculo la BER parcial
173     % p_error_parcial_2 = sum(abs(informacion_recuperada_2 - bits(1,:)))/n_bits
174     % p_error_parcial_1 = sum(abs(informacion_recuperada_1 - bits(2,:)))/n_bits
175
176     %-----------FIN DETECCIÓN DE LA INFORMACIÓN--------------%
177
178     %Se vuelven por lo tanto a modular y codificar los bits recibidos por los
179     %usuarios.
180
181     %-------------------MODULADOR-----------------------------%
182 -   if (mod == 'bpsk')
183         %---MODULADOR BPSK---%
184 -       bits_re_mod = bpsk(bits_re);
185         %---MODULADOR BPSK---%
186 -   end
187
188 -   if (mod == 'qpsk')
189         %---MODULADOR QPSK---%
190 -       bits_re_mod = qpsk(bits_re);
191         %---MODULADOR QPSK---%
192
193         %Si utilizamos la técnica de Alamouti vamos a enviar los símbolos de
194         %otra forma.
195 -       if (alamouti == 'si')
196 -           bits_re_mod = conj(bits_re_mod);
197 -       end
198 -   end
199     %--------------------FIN MODULADOR-----------------------%
200
201     %----------RECODIFICACIÓN DE LA INFORMACIÓN--------------%
202     %Expando cada bit por el número de chips del código.
203 -   bits_expand_re = kron(bits_re_mod,ones(1,length(codigos(1,:))));
```

```
204
205     %Entonces fin tiene la misma dimension que bits_expand y la puedo
206     %multiplicar punto a punto.
207     %Ahora tengo que tener en cuenta de que el USER 1 está codificando la
208     %información del 2 con el código 1; y viceversa.
209
210 -   tx_re = bits_expand_re.*fin;
211
212     %Los chips que enviara el usuario 1 será: sus chips y los chips recibidos del
213     %compañero 2. Y viceversa.
214
215     % -- USER 1 --
216     % Transmitirá sus bits y los bits del usuario 2.
217 -   tx_u1 = [tx(1,:) tx_re(1,:)];
218     % -- USER 2 --
219     % Transmitirá sus bits y los bits del usuario 1.
220 -   tx_u2 = [tx(2,:) tx_re(2,:)];
221
222     %La matriz de chips que envían los usuarios queda:
223 -   tx_total= [tx_u1;tx_u2];
224
225     %Ahora recoloco la matriz de modo a como se enviarían los bits en el caso
226     %real
227     % User 1 :    b1_u1_c1   b1_u2_c1   b2_u1  b2_u2
228     % User 2 :    b1_u2_c2   b1_u1_c2   b2_u2  b2_u1
229
230 -   tx_total(1,:) = intercalar(tx_total(1,:), n_simb, length(codigos(1,:)));
231 -   tx_total(2,:) = intercalar(tx_total(2,:), n_simb, length(codigos(1,:)));
232
233     %           TS1                    TS2
234     %     b1_u1_c1 + b1_u2_c2      b1_u1rx_c2 + b1_u2rx_c1
235     %-------------------------------------------------------%
236     %-------FIN PARTE DE LA COMUNICACION ENTRE USUARIOS------%
237     %-------------------------------------------------------%
238
239     %Ahora los datos pasan por al canal hacia el receptor.
240
241     %-------------------------------------------------------%
242     %----------------PARTE DEL CANAL ------------------------%
243     %-------------------------------------------------------%
244
245 -   if (channel == 'rale')
246 -       for u=1:n_users,
247 -       [tx_total_after_rl(u,:),coeficientes_filtro(u,:)] = rl(tx_total(u,:));
248 -       end
249 -   end
250
```

```
for s=1:length(snr),
    if (channel == 'awgn')
        tx_total_channel = awgn(tx_total,snr(s),mod);
    end
    if (channel == 'rale')
        %Despues le añado el ruido aditivo.
        tx_total_channel = awgn(tx_total_after_rl,snr(s),mod);
    end

    %-----------------------------------------------------%
    %----------------FIN PARTE DEL CANAL ------------------%
    %-----------------------------------------------------%

    %-----------------------------------------------------%
    %----------------PARTE DEL RECEPTOR---------------------%
    %-----------------------------------------------------%
    %Ahora la energía de señal que me llega es mayor, ya que tengo el doble de
    %información para decidir si es un 0 o un 1.

    %EN EL RECEPTOR MULTIPLICO POR EL CONJUGADO DE LOS COEFICIENTES.
    if(channel == 'rale')
        tx_total_channel = tx_total_channel.*conj(coeficientes_filtro);
    end

    %Ahora ya sumo los chips punto a punto.
    tx_total_channel = sum(tx_total_channel,1);

    %Tengo que multiplicar el vector que recibo por el codigo del usuario que
    %quiero recuperar repetido una vez tras otra para que tenga la misma
    %dimension.

    %En este caso como la información de un usuario me llega con dos códigos
    %distintos, tengo que multiplicar alternando el código de uno y del otro.

    fin_daf = [fin(1,:) fin(2,:)];
    fin_daf = intercalar(fin_daf, n_simb, length(codigos(1,:)));

    rx_total_channel = tx_total_channel.*fin_daf;

    %Ahora tengo que dividir la fila de tantos en tantos chips como tenga el código.
    rx_recived = reshape(rx_total_channel,length(codigos(1,:)),n_simb*2);
    rx = sum(rx_recived,1)./(length(codigos(1,:)));
    %Separo en dos filas los bits enviados directamente y los reenviados.
    rx = reshape(rx, 2, n_simb);
    %Si uso Alamouti vuelvo a hacer el conjugado de los simbolos que cambie
    if (alamouti == 'si')
        rx = [rx(1,:);conj(rx(2,:))];
    end
    %Promedio entre los dos simbolos de cada bit.
    rx = sum(rx,1)./2;
```

```
301
302         %En el caso de que la señal hubiera sido modulada con QPSK, tenemos que
303         %demodularla:
304 -       if (mod == 'qpsk')
305             %Separamos la parte real y la parte imaginaria.
306 -           p_real = real(rx);
307 -           p_imag = imag(rx);
308             %Ahora tenemos que intercalar las partes para tener los bits ordenados
309 -           inf = [p_real p_imag];
310 -           rx = intercalar_simple(inf);
311 -       end
312
313         %Tengo que decidir si el bit era un '1' o un '0' para ello tengo que poner
314         %un umbral.
315 -       informacion_recuperada = rx<0;
316
317         %Calculo la probabilidad de error.
318 -       p_error(s) = sum(abs(informacion_recuperada - bits(1,:)))/n_bits;
319         %Por Monte Carlo
320 -       if (p_error(s) < 0.0001)
321 -           p_error(s) = 0;
322 -       end
323 -   end
```

graficas.m

```
%PFC Ignacio Mingote Marina - I.T.T.: SISTEMAS DE TELECOMUNICACION
%
%Programa con el que representamos la probabilidad de error en función
% del número de usuarios y de la SNR.

%DATOS CON LOS QUE VOY A REALIZAR LA SIMULACIÓN.

users = [1];
snrs = [-40 -30 -25 -20 -15 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10];
n_bits = 100000;
user_ref = 1;
%users: matriz con el número de usuarios que forman parte del escenario.
%snrs: matriz con las distintas snr's del canal de comunicaciones.
%n_bits: número de bits de información con el que se realiza la simulación.


for i=1:length(users);
    p(1,:) = cdma('w', 'qpsk', users(i), n_bits, user_ref, snrs, 'rale')
end
ampl = [1 0.5 2]

% p=ones(2,length(snrs))
% for i=1:length(users);
%     p(1,:) = cdma_daf('w', 'qpsk', n_bits, snrs, 20, 'rale', 'no')
% end
%
% for i=1:length(users);
%     p(2,:) = cdma_daf('w', 'qpsk', n_bits, snrs, 20, 'rale', 'si')
% end

    for j=1:length(snrs);
        p(1,j) = cdma_aaf('w', 'qpsk', n_bits, snrs(j), 20, 1,'rale')
        if (p(1,j) < 0.0001)
            p(1,j) = 0;
        end
    end

%Represento las probabilidades de error de los distintos conjuntos de
%usuarios.
  semilogy(snrs,p(1,:),'o-')
  grid on
  hold on
```