Comunicaciones Inalámbricas

Trabajo de Simulación

Llorente, Juan Fermín
N^o de alumno: 69875/0
Correo electrónico: fermin.llorente@ing.unlp.edu.ar

Comunicaciones Móviles y Satelitales (E1605), 2022.

Resumen—*En el presente trabajo se caracteriza un modelo de canal* Rayleigh *con correlación* Jakes. *Además, se utiliza este modelo de canal para simular la transmisión y demodulación de datos en un esquema de CSI parcial a tasa fija con distintos esquemas de modulación y códificaciones a nivel de símbolo. Por último, se simula una comunicación a tasa variable con CSI completa para obtener una caracterización de la tasa media obtenida con este enfoque. Se corrobora el desempeño obtenido mediante simulación con las cotas teóricas desarrolladas para los distintos escenarios.*

1. Introducción

El curso natural de estudio de las comunicaciones presenta como primer modelo de canal el canal *AWGN*. Que introduce ruido blanco gaussiano aditivo a su salida y presenta una ganancia constante. Al trabajar con comunicaciones móviles, donde los receptores se encuentran inmersos en ambientes con variedad de obstáculos, en movimiento casi constante, es necesario reformular el modelo de canal a un canal que presenta desvanecimiento. Debido a los obstáculos podemos encontrar perdidas debido al multicamino, la señal es reflejada en los distintos obstáculos y por esto no se recibe solo el rayo directo. Se recibe la suma de todos los rayos rebotados, cada uno con fases distintas por lo que puede generar degradación en la señal según como sean las condiciones de los rebotes y las características de los rayos recibidos. El modelo que se puede plantear para analizar la incidencia de los múltiples rayos se basa en las asunciones de que suponemos un numero finito de reflectores con características eléctricas conocidas. Se aproxima a la onda que llega a cada reflector por un rayo (por facilidad frente a la resolución de las ecuaciones de *Maxwell*) y se resuelve de forma geométrica el problema. A partir de esto se puede suponer a la señal que llega al receptor móvil en un ambiente urbano como

$$r(t) = \sqrt{2}\Re\left\{\sum_{n=0}^{N(t)} \alpha_n(t)u(t-\tau_n(t))e^{j(2\pi f_c(t-\tau_n(t))+\phi_n(t))}\right\} . \tag{1}$$

Donde los retardos τ_n son distintos para cada camino en el caso general y dependen del tiempo absoluto en el que nos encontremos. Por otro lado la reflexión de los rayos produce perdidas de potencia α_n también distinta para cada camino y dependiente del tiempo. En este trabajo se trata con un canal plano por lo que

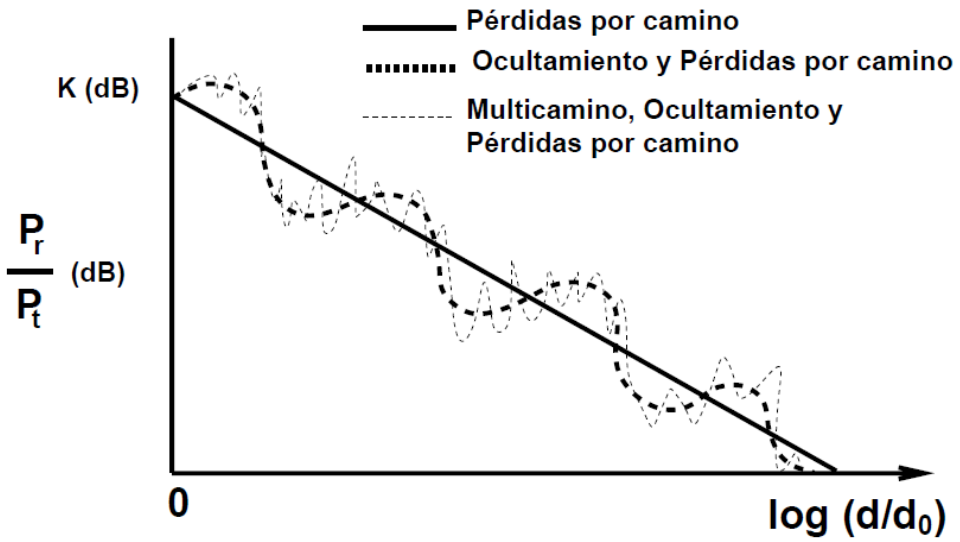


Figura 2: Perdidas combinadas por camino, ocultamiento y fading de banda angosta.

la máxima diferencia de retardos entre dos rayos (dispersión de retardos o T_d a partir de aquí) es mucho menor que la dinámica de la señal $u(t)$. En ese caso los distintos caminos no son separables de manera que podemos considerar todos los retardos nulos pero que generan fading o desvanecimiento (se hará referencia a uno u otro sin distinciones en este trabajo). Esto se debe a que los distintos rayos se suman en fase o contrafase generando atenuación.

Por otro lado se encuentran las perdidas por camino, debidas al camino que recurre la señal. En cursos elementales de magnetismo se estudió que el campo electrico es proporcional a la inversa del cuadrado de la distancia de la fuente. Aquí las perdidas estarán dadas por una expresión similar, pero como consideramos distintos ambientes (que nada tienen de similar al espacio libre) tendremos que considerar perdidas adicionales. Estas perdidas se caracterizan por una expresión como

$$P_r = P_t K \left(\frac{d_0}{d}\right)^{\gamma} \quad , \tag{2}$$

con K una constante de proporcionalidad dada por la ecuación de propagación en el espacio libre de *Friis*. La misma se ajusta empíricamente, por ello surgen los parámetros d_0 y γ. El valor de γ (la característica exponencial de las perdidas por camino) rondará entre 2 y 6 aproximadamente en el caso general, aunque se puede acotar hasta un valor de 4. Cabe aclarar que el modelo planteado en la Ecuación (1) se puede utilizar para caracterización de enlaces en el caso de dos rayos ya que presenta una condición muy pesimista con $\gamma = 4$.

Cómo última contribución tenemos el ocultamiento o *shadowing*, que se debe a que el receptor móvil se encuentra oculto por distintos obstáculos en el camino directo de propagación. El rayo directo o LOS (por *line-of-sight* del inglés) es el que mayor potencia tiene, por lo que una obstrucción al mismo generaría una atenuación considerable. Este fenómeno aleatorio es caracterizado por una variable aleatoria con distribución log-normal de media nula y varianza σ^2_{dB}. De esta manera si reescribimos la Ecuación (2) en escala logarítmica, podemos restar la variable aleatoria de la atenuación a las perdidas por camino. Es de utilidad pensar esto como la suma de muchas contribuciones aleatorias de perdidas en la expresión del efecto combinado. Al tomar todas las perdidas en dB las mismas pasan a ser aditivas (con signo negativo para ser estrictos) y por el teorema del límite central (TCL) esta contribución de *shadowing* resulta normal al tratarla en escala logarítmica. Es decir, la variable aleatoria en veces tiene distribución log-normal.

Este canal con *fading* combina el efecto de las perdidas por el camino recorrido por la señal, ocultamiento, multicamino y *fading* debido a los retardos no separables. En la Fig. 2 se puede ver una gráfica de las perdidas en escala logarítmica y sus distintas contribuciones según aumenta la distancia de propagación.

Como ya se mencionó que en el desarrollo de este trabajo se modela un canal plano, es decir que presenta *flat fading*, podemos considerar un modelo de banda angosta donde la dinámica del mensaje es mucho mas

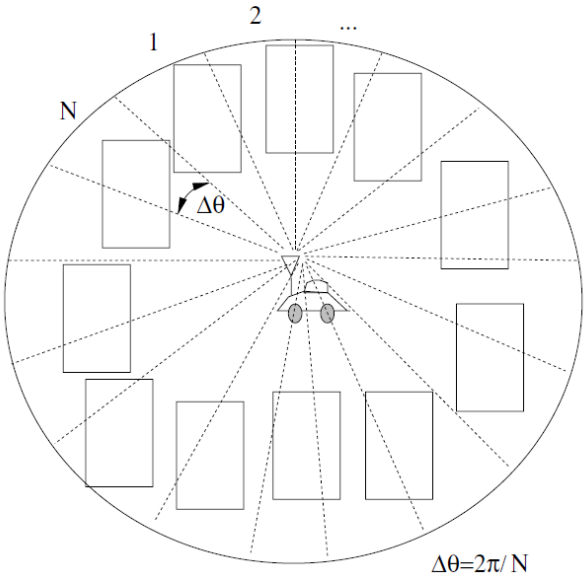


Figura 3: Reflexiones uniformes arribando al receptor móvil.

lenta que la del canal. Esto significa que no podremos resolver los distintos rayos que llegan al receptor móvil debido a las reflexiones. Considerando que existen múltiples reflexiones, o hay dispersión considerablemente densa, se puede plantear un modelo con un entorno con reflexiones uniformes. El mismo fue introducido por *Clarke* y luego desarrollado por *Jakes*. El mismo plantea que el receptor móvil se desplaza a una velocidad v fija por lo que la desviación en frecuencia debido al *doppler* será $f_D = v/\lambda$. Las reflexiones arriban al receptor con ángulos de incidencia θ por lo que se puede obtener una expresión para la señal recibida de la forma

$$r(t) = \sum_{n=0}^{N} \alpha_n e^{-j\phi_n^{\tau}(t)} \ . \tag{3}$$

Un esquema geométrico de la situación planteada se puede observar en la Fig. 3.

Es posible calcular la función de autocorrelación de la señal recibida como

$$R_a(\Delta t) = \mathbb{E}\{r(t+\Delta t)r^*(t)\} = \sum_{n=0}^{N} \mathbb{E}\{\alpha_n^2\}\mathbb{E}\left\{e^{j\phi_n^{\tau}(t)}e^{-j\phi_n^{\tau}(t+\Delta t)}\right\} = P_r J_0(2\pi f_D \Delta t) \ , \tag{4}$$

donde $\tau_n(t) = \tau_n^0 + v\cos(\theta)t/c$ y $\phi_n^{\tau}(t) = 2\pi f_c \tau_n(t) - \phi_n = 2\pi f_c \tau_n^0 - \phi_n + 2\pi v\cos(\theta)t/\lambda$. Este resultado se obtiene de pensar todos los caminos o rayos recibidos como independientes e uniformemente distribuidos en el ángulo de arribo θ. Por esto $\mathbb{E}\{r(t)\} = 0$.

Calculando la transformada de *fourier* de esta autocorrelación se puede hallar la densidad espectral de potencia *doppler* (*Doppler Power Spectrum* en inglés). Es necesario transformar la función de *Bessel* de primera especie y de orden cero, la expresión que se obtiene es

$$S_r(f) = \begin{cases} \dfrac{P_r}{\pi f_D} \dfrac{1}{\sqrt{1-(f/f_D)^2}} & |f| < f_D \\ 0 & |f| > f_D \end{cases} \ . \tag{5}$$

La forma obtenida para el espectro *doppler* con correlación *Jakes* o con el modelo de *Clarke* se puede observar en la Fig. 4.

1.1. Modelo de sistema de comunicación digital en un canal inalámbrico

Es necesario introducir el sistema de comunicación digital pasabanda formado por transmisor, canal y receptor para el mejor entendimiento de este trabajo. Aunque de igual manera para los fines de la simulación

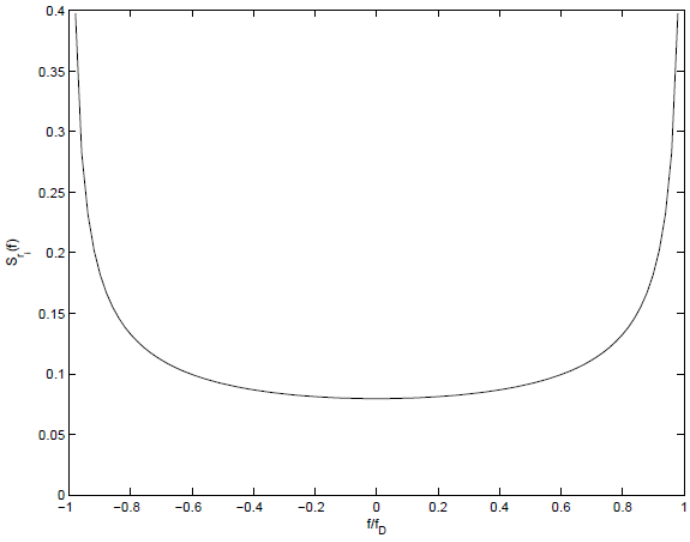


Figura 4: Densidad espectral de potencia *doppler*.

que realizaremos la cuestión se simplifica ya que simplemente queremos poder implementar el modulador, el modelo de canal y luego junto con el receptor óptimo el demodulador. No es necesario obtener la señal banda base, llevarla a pasabanda para su paso por el canal y luego realizar el proceso inverso. En cambio, aquí se trabajará a nivel de símbolo obteniendo resultados equivalentes a generar las señales moduladas pero con costos de simulación abismalmente menores.

Como trabajaremos con un sistema de comunicaciones pasabanda se puede hacer uso del modelo de envolvente compleja. De esta manera es suficiente modelarlo siempre en una base de una dimensión en el espacio de señal pero con cuerpo complejo, es decir con coordenadas complejas. Las mismas se pueden desglozar como coordenadas con componente en fase y en cuadratura. Resultando cada símbolo como

$$a_k = \alpha_k + j\beta_k \ . \tag{6}$$

Esta señal será escalada por la ganancia compleja del canal, que en principio tendrá un valor distinto para cada símbolo. A esto se le agregará ruido blanco gaussiano complejo aditivo para que la secuencia recibida tenga la forma

$$y[k] = c \cdot x[k] + w[k] \quad , \quad w \sim CN(0, N_0) \ . \tag{7}$$

Esto ingresará al receptor óptimo que compara la distancia euclídea del símbolo recibido con todos los símbolos y decide en base de cual es la menor distancia. En caso de introducir un código de repetición a nivel de símbolo se emplea un algoritmo de diversidad en la recepción *MRC* (por *maximal ratio combining*) antes de ingresar al receptor óptimo que ya se conoce para un canal AWGN sin fading. Este algoritmo realiza la siguiente operación

$$z[k] = \frac{\mathbf{c}^H[k]}{\|\mathbf{c}\|}\mathbf{y}[k] = \|\mathbf{c}\|\, x[k] + \frac{\mathbf{c}^H[k]}{\|\mathbf{c}\|}\mathbf{w}[k] \tag{8}$$

Una vez se tiene el símbolo estimado es posible obtener los bits que le corresponden mediante un decodificador de símbolos a bits.

Los bits a transmitir serán generados de manera aleatoria y equiprobable. Un modulador los convertirá en los símbolos correspondientes con sus coordenadas complejas. El resultado que obtenemos luego de realizar todos estos pasos son los bits estimados luego de la recepción y resulta muy sencillo calcular tasa de error de bit a partir de los mismos de la siguiente manera

$$BER = \frac{\#\ bits\ errados}{\#\ bits\ transmitidos} \ . \tag{9}$$

Cabe aclarar que no podremos calcular una probabilidad pero podemos estimar una tasa de error teniendo en cuenta que la cantidad de bits es lo suficientemente grande. Esto es valido debido a que el estimador de la P_{eb} es insesgado y consistente. Una regla práctica para calcular cuantos bits necesitamos transmitir para una dada P_{eb} que queremos estimar es tener un número de bits aproximadamente igual a $100/P_{eb}$. Se puede pensar de la siguiente manera, si tenemos 1 error cada 100.000 bits ($P_{eb} = 10^{-5}$), entonces si generamos $100 \times 100{,}000$ bits tenemos 100 realizaciones de chorros de 100.000 bits que tendrán en promedio un error cada uno. Promediando en 100 realizaciones se obtiene un valor estimado que es representativo de lo que buscamos.

Si bien aquí hay que intentar mezclar de manera suficiente como para que haya suficientes combinaciones posibles de realizaciones de canal y de ruido, lo que haría que sea necesario simular un numero un tanto mayor de bits, a gran escala con este criterio se obtienen resultados razonables. En caso de obtener curvas de P_{eb} con poca suavidad, la regla para solucionarlo consiste en simular conjuntos mas grandes.

1.2. Objetivos

- Caracterizar un modelo de canal *Rayleigh* con correlación *Jakes*.
- Simular y corroborar el desempeño que presenta una comunicación a tasa fija con CSI parcial.
- Simular y corroborar el desempeño que presenta una comunicación a tasa variable y CSI completa.

2. Desarrollo y discusiones

2.1. Caracterización del canal con fading

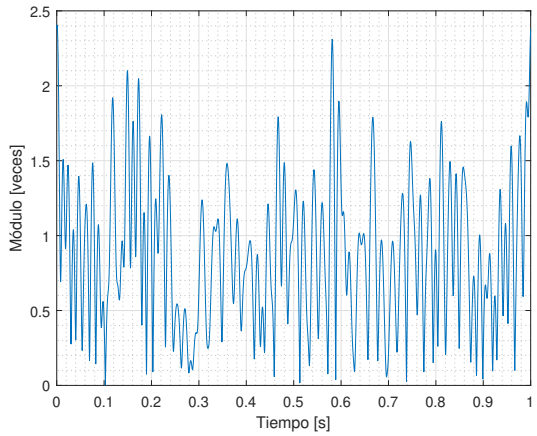
La cátedra entrega un script de *Matlab* que genera una secuencia de ganancias complejas correspondiente a un canal plano con desvanecimiento *Rayleigh* y espectro *Doppler* de acuerdo al modelo de *Clarke*. Se ha modificado a `'CanalFlat.m'` para poder llamarlo como función con los siguientes parámetros:

- T: largo de la realización de canal deseada en segundos.
- t_s: tiempo de muestreo con el que se muestrea el canal.
- h: salida de la función. Secuencia de ganancias complejas del canal.

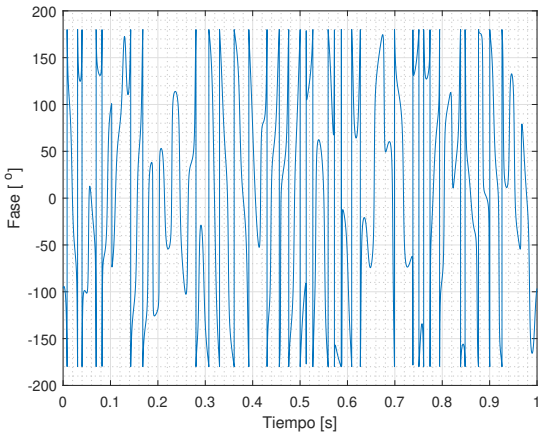
Devuelve en la variable h la ganancia compleja del canal con correlación *Jakes* normalizada para que la ganancia media sea unitaria. Una vez que se corrió el *script* provisto, se puede graficar la magnitud de la ganancia de canal para los distintos retardos, siendo la misma una realización en un t dado. Manteniendo los parámetros originales se obtiene 1 segundo de canal en 900 MHz muestreado a $t_s = 5\,\mu s$. Son valores similares a los de un subcanal GSM de 200 kHz de ancho de banda (sin tener en cuenta multiplexado en tiempo con otros usuarios y suponiendo dispersión de retardos menor al tiempo de muestreo).

Habiendo dado las especificaciones del canal a utilizar, se puede comenzar con la caracterización experimental del mismo. Para esto se considera que el receptor móvil tiene una velocidad media v_m de $60\,km/h$. Es necesario correr el *script* provisto y obtener una realización de canal. Si graficamos la secuencia de amplitudes del canal en el tiempo se puede ver las variaciones que presenta el módulo tanto como la fase, esto se puede observar en la Fig. 5. Con esto es evidente que podemos tener momentos en los que el canal está "de buenas" y no atenúa nada a la señal en ese instante, pero por momentos atenuará de manera notable ya que el módulo de la ganancia es prácticamente nulo (Fig.5a). En cuanto a la fase es evidente como cada símbolo será desfasado de manera distinta, la variación de la misma se observa únicamente entre -180^o y 180^o y por eso hay presencia de saltos en la fase (Fig.5b).

También es posible calcular el espectro *doppler* del canal, obteniendo la Densidad Espectral de Potencia de la secuencia. Hay que notar que al estar trabajando con una señal que no tiene unidades esto no nos daría una idea de cómo se distribuye la potencia (como potencia en unidades de *Watts*) de la señal en el espectro. Es una DEP pero que tendrá unidades de $1/Hz$ ya que la señal es adimensional por ser una ganancia. Se obtiene un *doppler* máximo $f_D = v_m/\lambda = 50\,Hz$, con $\lambda = 33{,}3\,cm$ (definido por la frecuencia de $900\,MHz$). La Fig. 6a muestra cómo el espectro *doppler* de la realización de canal se pega a la curva teórica. El ancho



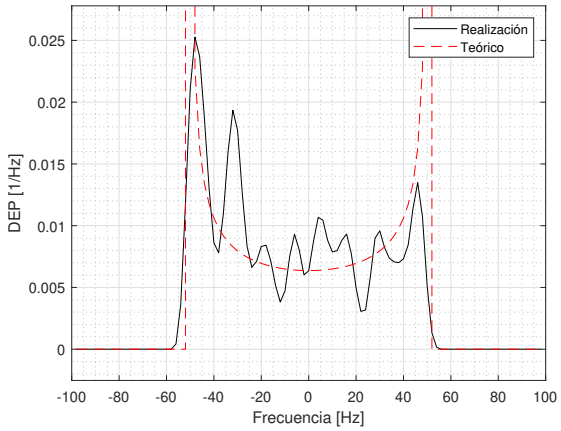
(a) Módulo.



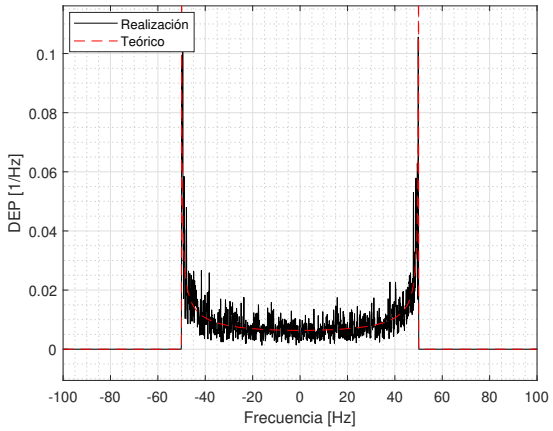
(b) Fase en grados.

Figura 5: Realización de 1 segundo de la secuencia de ganancias complejas del canal.

de banda *doppler* B_D para esta realización de canal no es exactamente el teórico pero no se aleja en gran medida del mismo. El valor teórico es de $100\,Hz$ ya que es el doble del *doppler* máximo, en cambio, el valor experimental de esta realización es cercano a $114\,Hz$. Es posible obtener una realización más larga de canal y ver cómo cada vez se parece más a la curva teórica, lo mismo se observa en la Fig. 6b. En ese caso el ancho de banda *doppler* es prácticamente igual al teórico.



(a) Realización de canal de 1 segundo de largo.



(b) Realización de canal de 50 segundos de largo.

Figura 6: Espectro *doppler* de la realización de canal comparado con el teórico bajo los mismos parámetros.

En principio con este análisis no tenemos información de ningún tipo respecto a parámetros tales como ancho de banda de coherencia B_C o el tiempo de coherencia T_C. Pero podemos utilizar información que ya hemos dado respecto al canal, sabemos que la dispersión de retardos T_D (máxima diferencia entre dos retardos) es menor a $5\,\mu s$. No existe manera de corroborar esto experimentalmente ya que estamos trabajando con un modelo de canal *flat*, es decir, los distintos retardos no son separables. De manera que si queremos obtener el parámetro reciproco de T_D en el dominio de la frecuencia solo podemos hacer uso de la información que se acaba de mencionar. El ancho de banda de coherencia se puede obtener como

$$B_C = 1/(2T_C) > 100\,kHz\ .$$

El mismo indica de alguna manear cuál es la desviación en frecuencia que se debe producir para poder

considerar que el canal esta descorrelacionado. O en otras palabras, el ancho de banda en el cuál el canal se comporta sin cambios significativos en cuanto a correlación. **REVISAR esto**.

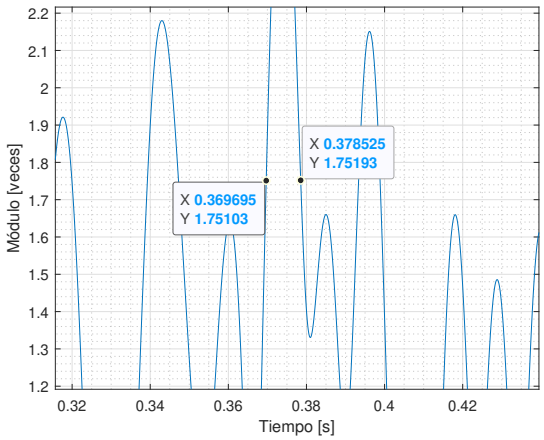


Figura 7: Ancho del pico más angosto presente en la respuesta temporal de la realización de canal.

Para obtener el tiempo de coherencia, es necesario tener una medida de por cuánto tiempo el canal se mantiene correlacionado. Se considerará correlacionado a los fines prácticos ya que en principio la función de autocorrelación nunca llegará a ser completamente nula, pero hay un momento en el cual se puede considerar que esto sucede. De una manera muy grosera se puede observar alguno de los picos que presenta la secuencia de ganancias del canal (Fig. 5a) y medir el ancho del pico más angosto. Podemos considerar que una vez que nos alejamos de ese pico la señal ya no tiene ningún parecido con su versión desplazada (el mismo pico) y por lo tanto está descorrelacionada. Haciendo esto se puede ver que la diferencia de tiempos entre estos instantes es cercana a 0.0088, tomando como ejemplo la Fig. 7. Se puede corroborar esto mismo obteniendo la función de autocorrelación de la secuencia de ganancias $R_{hh}(\Delta t)$, el resultado de la misma se puede observar en la Fig. 8. Se corrobora por otro camino que la secuencia tiene energía unitaria comparando con el valor máximo de la función de autocorrelación. Y al hacer una amplificación en el resultado se puede medir el ancho del lóbulo principal de la función de autocorrelación (ver Fig. 8b) y obtener el soporte de la misma como

$$T_C = 0{,}00864\,s\ .$$

Esto no es otra cosa que el tiempo de coherencia y resulta bastante similar a lo obtenido mediante el primer método. Que incluso si lo comparamos con su reciproco en el dominio de la frecuencia, el ancho de banda *doppler* se puede ver que todo coincide,

$$T_C = 1/B_C = 1/(114\,Hz) = 0{,}0088\ . \tag{10}$$

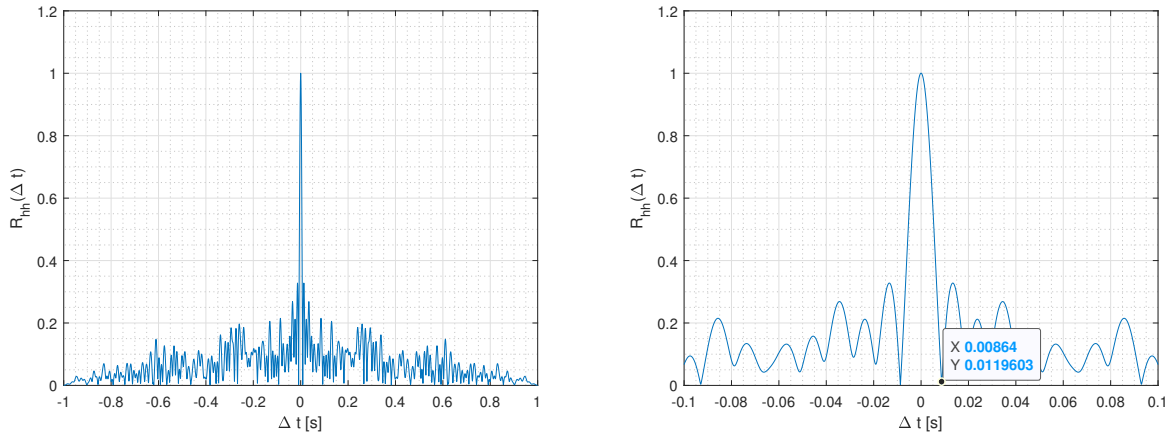
Todos los valores obtenidos están muy cerca de lo que se obtiene si consideramos el caso teórico donde $T_C = 1/B_C = 1/100 = 0{,}01$. Cabe destacar que aquí se considera la Ecuación (10) sin ningún factor en el numerador ni denominador ya que es arbitrario ponerlo o no. Según la bibliografía que se consulte se consideran distintos factores de proporcionalidad, en particular, está convención tomada corresponde con [1].

Se puede corroborar si el canal es *overspread* ($T_D \gg T_C$) o *underspread* ($T_D \ll T_C$). Que es lo mismo que comparar que el producto $T_D \cdot B_C$ sea menor o mayor que uno. En este caso se obtiene

$$T_D \cdot B_C < 5\,\mu s \cdot 114\,Hz = 5{,}7 \cdot 10^{-4} \ll 1\ .$$

Por lo que se puede concluir que el canal es *underspread*, que es lo habitual.

A fin de resumir, los parámetros obtenidos son:



(a) Función de autocorrelación completa.

(b) Amplificación en el origen para observar T_c.

Figura 8: Función de autocorrelación de la secuencia de ganancias complejas del canal.

- Ancho de banda de coherencia: $B_C > 100kHz$.
- Tiempo de coherencia: $T_C = 0{,}0088 \approx 0{,}01$.
- Ancho de banda *doppler*: $B_D = 114\,\text{Hz}$.
- Canal es *underspread*.

2.2. Desempeño con CSI parcial y tasa fija

En esta etapa se considera que el receptor conoce el valor de la ganancia compleja de canal para cada símbolo transmitido. De esta manera el mismo puede emplear un algoritmo de diversidad MRC en caso de agregar repetición de símbolos o incluso entrelazado de los mismos. En principio se pide simular para tres escenarios distintos:

- BPSK.
- 16-QAM con código de repetición a nivel de símbolos de 4 veces, sin entrelazado.
- 16-QAM con código de repetición a nivel de símbolos de 4 veces, con entrelazado.

La velocidad de transmisión que se obtiene en los tres casos es igual, ya que en los enfoques en los que se codifica cada símbolo con cuatro bits (16-QAM) se agrega un código de repetición de cuatro veces. Es decir, lo que podemos ganar en tasa de bit utilizando modulación 16-QAM se pierde al realizar la repetición. Esto es lógico ya que si queremos realizar una comparación a nivel de la probabilidad de error de bit obtenida, siempre operamos a la misma tasa de datos y se puede hacer una comparación justa. La tasa resultante es de 1 bit por símbolo, y como la tasa de muestreo es de $5\,\mu s$ se obtiene una tasa de símbolos $R_s = 200\,kHz$. Esto es el equivalente al ancho de banda del subcanal GSM del que se consideran que son equivalentes los parámetros del escenario planteado en este trabajo. De aquí resulta que la tasa de bit es $R_b = 200\,\text{kbps}$.

Lo que de alguna manera se menciona como pérdida (en cuanto a velocidad de transmisión) al agregar un código de repetición, en realidad no es pérdida ya que va a traer sus ventajas a la hora de analizar la probabilidad de error de bit que es el paso que sigue. La P_{eb} para la modulación BPSK sin código de repetición resulta

$$P_{eb} = P_e = \frac{1}{2}\left(1 - \sqrt{\frac{SNR}{1+SNR}}\right) \approx \frac{1}{4SNR}\ , \tag{11}$$

donde la aproximación es válida para alta SNR. Además se considera detección coherente, es decir, el receptor tiene información de la ganancia compleja que agrega el canal (su amplitud y fase).

Se puede obtener una cota para P_{eb} en la comunicación, ahora utilizando un código de repetición de 4 veces pero sin entrelazado de símbolos. Este caso tiene interés de ser analizado para igualar las tasas de transmisión al utilizar BPSK, pero repetir sin realizar entrelazado no resulta en otra cosa que perdidas. Esto lo podemos explicar ya que al repetir 4 veces cada símbolo, estaremos pasando a cada palabra de 4 símbolos por un canal completamente correlacionado en sus 4 componentes. Se debería contar con un tiempo de coherencia T_C muy bajo comparado con el tiempo de muestreo para conseguir una ganancia al repetir, para ser específicos se debería cumplir $T_C \ll 4t_s$, con t_s el tiempo de muestreo. La cota para la P_{eb} con un sistema M-ario y código de repetición de n veces resulta

$$P_{eb} \leq \frac{(M-1)}{N} P\left(\mathbf{x}_i \to \mathbf{x}_j\right)_{max} \ , \tag{12}$$

donde se considera la peor condición para todos los símbolos. Se utilizó la cota de la unión (para la probabilidad de la unión de eventos). Cada símbolo tiene la misma probabilidad de errar con el resto de los $(M-1)$ símbolos, y esta probabilidad está dada por el símbolo con el que se encuentra más cercano ya que es con el cual hay mayor probabilidad de equivocarse. Por ello aparece el término de $(M-1)$ y se divide por N para llevar la probabilidad de error de símbolo a P_{eb} (considerando que se utilizó código de *Gray* en la asignación de bits a los símbolos). Ahora se puede hallar

$$\begin{aligned} P\left(\mathbf{x}_i \to \mathbf{x}_j\right)_{max} &= \mathrm{E}\left\{P\left(\left[\mathbf{x}_i \to \mathbf{x}_j\right]_{min} | \left\|\mathbf{c}\right\|^2\right)\right\} = \mathrm{E}\left\{Q\left(\sqrt{\frac{|x_i - x_j|^2_{min}| \left\|\mathbf{c}\right\|^2}{2N_0}}\right)\right\} \\ &\leq \mathrm{E}\left\{exp\left(-\frac{SNR|\tilde{x}_i - \tilde{x}_j|^2_{min}| \left\|\mathbf{c}\right\|^2}{4}\right)\right\} = \frac{1}{1 + \frac{SNR|\tilde{x}_i - \tilde{x}_j|^2_{min}}{4}} \ . \end{aligned} \tag{13}$$

Para esto se considera $\tilde{x}_i(k) = x_i(k)/\sqrt{\overline{E}_X}$, y $\overline{E}_X$ la energía de símbolo promedio. Este resultado se puede obtener al considerar que $|\left\|\mathbf{c}\right\|^2 \approx n|\left\|c\right\|^2 \sim \varepsilon(n)$ y utilizando la función generadora de momentos de una VA exponencial para el último paso. Es evidente como el resultado es independiente de la cantidad de veces que se repita el símbolo, siempre que las consideraciones sigan siendo válidas por supuesto. Si repetimos una cantidad suficiente de veces como para poder considera que hay desvanecimientos que se vuelven independientes ya habrá que reformular el resultado para que siga siendo válido, pero en principio para $n = 4$ tiene validez. Si especificamos el resultado obtenido para 16-QAM se obtiene que

$$|\tilde{x}_i - \tilde{x}_j|^2_{min} = \frac{|x_i - x_j|^2_{min}}{\overline{E}_x} = \frac{2}{5} \ .$$

Donde

$$\overline{E}_x = \frac{1}{16}\sum_{i=1}^{16} E_i \ ,$$

ya que se consideran símbolos equiprobables. Teniendo en cuenta esto se puede unificar las Ecuaciones (12) y (13) para obtener

$$P_{eb} \leq \frac{15}{4}\frac{1}{1 + \frac{SNR}{10}} \ . \tag{14}$$

Cabe destacar que esta es una cota bien holgada ya que se esta considerando un escenario bastante pesimista.

2.2.1. Entrelazado

Al aplicar un código de repetición a nivel de símbolo y entrelazando se logra promediar el efecto del desvanecimiento. Siempre que se tenga en cuenta que la ganancia de canal que le corresponde a cada símbolo de la palabra que se forma al repetir, es independiente de la ganancia del resto de la ganancia que corresponde al resto de los símbolos. Para conseguir esto es necesario esperar un tiempo dado por el tiempo de coherencia del canal, es decir, esperar a que el canal se descorrelacione. A partir de aquí se puede definir la profundidad

del entrelazador, la misma es una medida de cuánto tengo que esperar entre dos símbolos sucesivos de una misma palabra.

Si consideramos al entrelazador de a bloques como que forma una matriz, se ingresan las filas y se leen las columnas, en este caso la profundidad es la cantidad de columnas que tiene el bloque formado. La implementación del entrelazador en *Matlab* puede consultarse en la Sección 4.5. El desentrelazador consiste una rutina que vuelve los símbolos al orden natural en que fueron generados los datos, el mismo se puede consultar en la Sección 4.6.

Ahora se puede realizar el mismo procedimiento para obtener la P_{eb} de la Ecuación (14) pero al considerar entrelazado. En este caso se puede considerar que se tiene desvanecimientos independientes (para esto el entrelazador debe haber sido diseñado correctamente) por eso resulta en la productoria al promediar sobre todas las realizaciones posibles de desvanecimiento. En otras palabras, la esperanza del producto resulta en el producto de las esperanzas. Que al estar tomando la media estadística de n VA con la misma distribución (podemos definirlas *iid*) resulta en un resultado similar al ya obtenido pero elevado a la n. Se obtiene la probabilidad de error de símbolo para un canal *Rayleigh* con varianza σ_c^2 unitaria, realizando demodulación coherente, repetición de n veces y entrelazado como

$$P_{eb} \leq \frac{(M-1)}{N} \prod_{k=1}^{n} \frac{1}{1 + SNR\dfrac{|\tilde{x}_i(k) - \tilde{x}_j(k)|^2_{min}}{4}} = \frac{(M-1)}{N} \frac{1}{\left(1 + SNR\dfrac{|\tilde{x}_i(k) - \tilde{x}_j(k)|^2_{min}}{4}\right)^n} . \tag{15}$$

La Ecuación (15) tiene en cuenta que los desvanecimientos en cada uso de canal (dados por el índice k) son independientes entre sí, por eso resulta en la productoria al promediar sobre todas las realizaciones posibles de desvanecimiento. Aquí hay una dependencia con n, la principal ganancia que obtenemos al repetir y entrelazar. Si queremos una expresión para el caso de modulación 16-QAM, se puede hacer lo mismo que ya se mostró anteriormente ahora para el caso $n = 4$ y obtenemos

$$P_{eb} \leq \frac{15}{4} \frac{1}{\left(1 + \dfrac{SNR}{10}\right)^4} . \tag{16}$$

2.2.2. Resultados para BPSK

Habiendo introducido los conceptos necesarios para comenzar a simular los distintos escenarios, se puede analizar los resultados de los mismos. Comenzamos con un esquema de modulación BPSK sin ningún código de repetición. Al realizar una simulación con $5 \cdot 10^7$ bits, en este caso, la misma cantidad de símbolos, se obtiene un desempeño esperable al comparar con la curva teórica presentada en la Ec. (11). Esto mismo se puede observar en la Fig. 9.

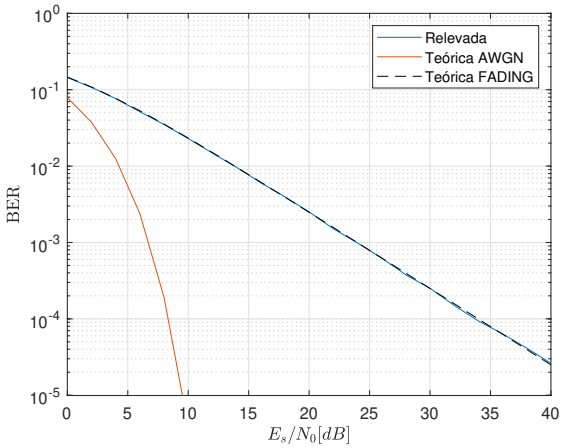


Figura 9: Curva relevada de P_{eb} en función de la E_s/N_0 para $5 \cdot 10^7$ bits, modulación BPSK sin repetición.

Se puede notar como la curva relevada ('Relevada') se pega a la curva punteada ('Teórica FADING') que es la curva teórica para este esquema presentada en la Ecuación (11). Además se puede ver la curva de P_{eb} que obtiene un canal AWGN ('Teórica AWGN') para comparar el desempeño que presentan ambos canales, para conseguir una $P_{eb} = 10^{-5}$ se sabe que con modulación BPSK en un canal AWGN es necesario contar con 9.6dB de E_s/N_0. En cambio, para un canal con desvanecimiento *Rayleigh* se precisa más de 40dB de E_s/N_0 para lograr la misma P_{eb}, un valor que no es prácticamente factible. Más allá de la factibilidad de conseguir ese valor, que en principio se puede alcanzar, tiene sentido analizar si se puede pensar en bajar la P_{eb} aumentando la SNR. Teniendo en cuenta la pendiente de la curva obtenida, no es una solución inteligente. Para atacar esto es que se agrega un enfoque con repetición y entrelazado para conseguir desafectar el efecto del canal pinchado (con muy poca ganancia) que es lo que hace crecer la P_{eb}.

2.2.3. Resultados para 16-QAM con código de repetición de 4 veces

Ya se mostró (Ecuación (14)) que al utilizar un código de repetición a nivel de símbolos de n veces, pero no entrelazar, no se logra obtener diversidad por lo que el desempeño a grandes rasgos (al menos en cuanto a la pendiente de la curva obtenida) no va a distar de los resultados obtenidos al no repetir para un sistema 16-QAM. Si se simula este escenario, el receptor realizando demodulación coherente aplicará un algoritmo MRC (Ecuación (8)) para combinar los n símbolos de la palabra y pesar cada símbolos por la ganancia de canal que le corresponde, con la debida normalización. Si no se entrelazó, la ganancia por la que se escaló cada símbolo de la palabra, es prácticamente la misma pues no llega a estar ni cerca de descorrelacionarse. De esta manera, todos los símbolos de la palabra están pesados por el mismo coeficiente, que si el canal está de malas, la probabilidad de errar incrementa abismalmente.

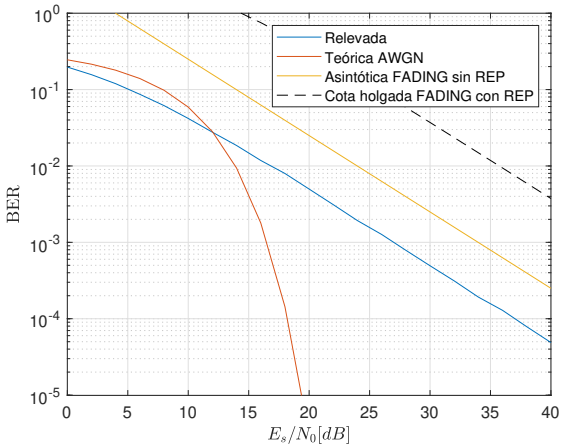


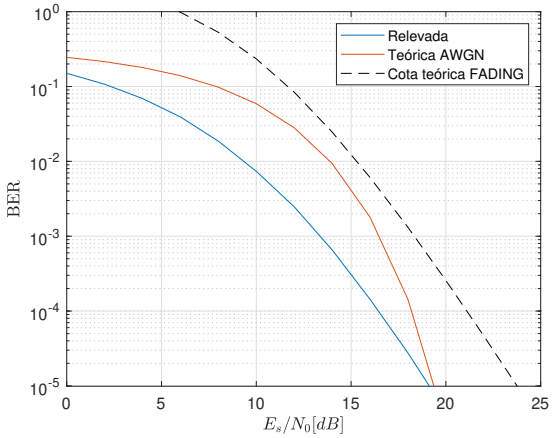
Figura 10: Curva relevada de P_{eb} en función de la E_s/N_0 para $5{,}04 \cdot 10^7$ bits, modulación 16-QAM con código de repetición de 4 veces.

Los resultados de la simulación se pueden observar en la Fig. 10, donde se compara la curva relevada con la cota holgada del canal con *fading* para un código de repetición de $n = 4$ veces. La misma es la que se presenta en la Ecuación (14). Además, se compara con la curva asintótica de P_{eb} para 16-QAM sin repetición. Y por último, se sigue manteniendo la curva de P_{eb} para un canal AWGN para observar y comparar cuál es la diferencia de desempeño. La pendiente de las curvas en los casos con *fading* son las mismas, y son proporcionales a la inversa de la SNR. Según con cuál se compare se observa una diferencia constante de 5dB con la curva asintótica de 16-QAM sin repetición, y de aproximadamente 17dB con la cota holgada que si tiene en cuenta la repetición.

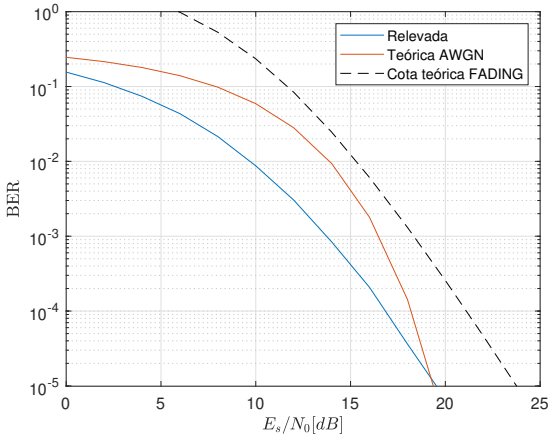
Un enfoque de este estilo sigue siendo inviable ya que hay que poner en juego mucha potencia para conseguir bajar la P_{eb}.

2.2.4. Resultados para 16-QAM con código de repetición de 4 veces y entrelazado

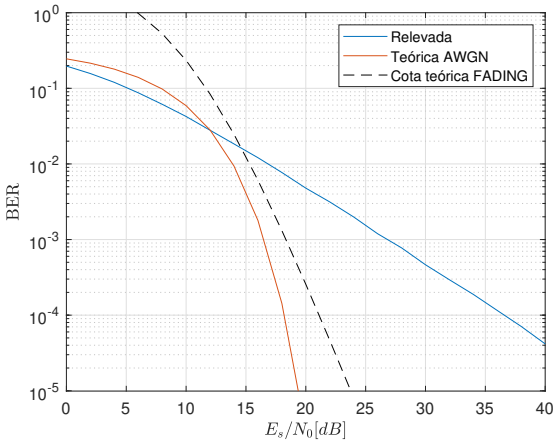
Aquí es donde se intenta combatir los efectos del canal con *fading* para obtener un mejor desempeño. Ya se ha mencionado anteriormente los efectos que lograremos al introducir entrelazado de símbolos, con un correcto diseño del entrelazador esto va a resultar en una diversidad de 4. En la Fig. 11 se puede observar el desempeño obtenido y la corroboración mediante simulación de los resultados desarrollados de manera teórica. Se puede revisar en el repositorio de este trabajo (Sección 4.1) en las entradas más viejas, que existió una implementación donde se generaban intencionalmente canales independientes, cada canal independiente era el que le correspondía a un símbolo de la palabra. De esta manera se conseguía diversidad 4 de manera forzada sin necesidad de un enfoque con entrelazado. Los resultados obtenidos eran iguales a los que muestra la Fig. 11a. En la figura mencionada se presenta una simulación que presenta una profundidad del entrelazador tal como para que entre símbolos sucesivos de una misma palabra haya una separación de $20 \cdot T_C = 200\,ms$. Se consigue una diversidad de 4, y se puede observar cómo el desempeño que presenta un enfoque con *fading* no dista del canal AWGN (y es incluso mejor) para bajas SNR de hasta 20dB aproximadamente. El problema que trae este escenario es que hay que esperar $200\,ms$ entre símbolos sucesivos, y $800\,ms$ para recibir una palabra completa. Una latencia cercana a un segundo no podría ser viable para comunicaciones críticas, ni hablar de una comunicación telefónica de voz.



(a) Profundidad igual a $20 \cdot T_C/t_s$, diversidad 4 con separación de sobra.



(b) Profundidad igual a T_C/t_s, diversidad 4 en el límite teórico.



(c) Profundidad igual a 4, no hay diversidad.

Figura 11: Curva relevada de P_{eb} en función de la E_s/N_0 para $5{,}04 \cdot 10^7$ bits, modulación 16-QAM con código de repetición de 4 veces y entrelazado. Comparación a distintas profundidades de entrelazado.

Como en lo discutido anteriormente se espera más de lo necesario entre símbolos sucesivos, se puede ir aún más lejos y probar el límite teórico para obtener diversidad temporal, para esto se plantea una profundidad que haga que la separación entre símbolos sucesivos de la misma palabra sea de $T_C = 10\,ms$. En este caso si estimamos mal el tiempo de coherencia del canal en la realización que tenemos del mismo, puede que no hayamos conseguido que el mismo se descorrelacione entre símbolos sucesivos. El resultado se puede observar en la Fig. 11b. El desempeño es prácticamente igual y esto presenta un tiempo de latencia mucho menor ya que la matriz del entrelazador de bloques se llena mucho más rápido por lo que se puede demodular cada palabra sin necesidad de esperar tanto tiempo.

Si se tuviese que diseñar un esquema de comunicación inalámbrica se optaría por este enfoque ya que el desempeño es cercano al mejor que se puede obtener en estas condiciones y el tiempo de latencia no resulta elevado. Aquí hay que esperar $40\,ms$ para poder demodular cada palabra, dentro de los rangos aceptables para una comunicación de voz de buena calidad, se considera que hasta $40 - 60\,ms$ es aceptable para una buena comunicación. Esto se obtiene mediante pruebas con usuarios, a distintos valores de latencia, hasta que la comunicación deja de ser posible. Así se obtiene el umbral máximo de latencia aquí mencionado. Considerando que el ancho de banda es suficiente, también se podría considera aceptable para una videollamada, donde el umbral se encuentra en valores cercanos a los aquí mencionados. Incluso este enfoque obtiene resultados similares en cuanto a latencia a los de algunos estándares de telefonía móvil como el IS-95 (estándar interno 95), que presenta $20\,ms$ de latencia.

Por último, es de interés plantear un escenario donde la profundidad es tal que el canal no llegó a descorrelacionarse (aquí una profundidad solo de 4 muestras, que corresponde a un tiempo de espera de $20\,\mu s$ entre símbolos sucesivos), por lo que deja de haber diversidad. Esto se puede observar en la Fig. 11c, y es evidente como la curva relevada vuelve a ser proporcional a la inversa de la SNR por lo que se perdió la diversidad. Aquí no hay latencia prácticamente (obviamente depende de la aplicación, al menos para una comunicación inalámbrica de voz es despreciable) pero no se consigue trabajar con desempeños similares a los del canal AWGN, por lo que es inviable.

2.3. Desempeño con CSI completa y tasa variable

En esta etapa no solo el receptor conoce el valor de la ganancia de canal, si no que el transmisor también la conoce (por esto CSI del inglés de Información del Estado del Canal). Esto se puede lograr de distintas manera pero en principio es con un enlace de comunicación independiente al utilizado para los datos, donde el receptor informa al transmisor de lo que recibe para que se puede calcular cual es la ganancia de canal en ese momento. La idea aquí es plantear una SNR efectiva

$$SNR_{eff} = |c|^2 SNR\ , \tag{17}$$

de manera que según su valor se elija el sistema de modulación y grado de repetición a nivel de símbolos a utilizar. El criterio para elegir es el siguiente:

- No se transmite nada si es menor que $-10dB$.
- BPSK con código de repetición de 4 veces si está entre $-10dB$ y $-5dB$.
- QPSK con código de repetición de 4 veces si está entre $-5dB$ y $0dB$.
- QPSK con código de repetición de 2 veces si está entre $0dB$ y $5dB$.
- QPSK si está entre $5dB$ y $10dB$.
- 16-QAM si es mayor que $10dB$.

Para esto se revisa cada T_C cual es el valor de la SNR efectiva, y se elige el esquema de modulación. El mismo queda fijo por un tiempo T_C donde se vuelve a revisar y si es necesario, se cambia el esquema. Esto es equivalente a pensar en que se divide la secuencia de ganancias del canal en tramos de largo T_C, el código agrega la opción de revisar el valor de la SNR_{eff} en el inicio, centro o final de este intervalo.

2.3.1. Tasa media teórica

Para calcular la tasa media teórica se parte de la siguiente expresión

$$R_m = \frac{1}{t_s}\Big[0 \cdot P(SNR_{eff} < 10^{-10/10}) + \frac{1}{4}P(10^{-10/10} \leq SNR_{eff} < 10^{-5/10}) + \frac{2}{4}P(10^{-5/10} \leq SNR_{eff} < 10^{0/10}) + \frac{2}{2}P(10^{0/10} \leq SNR_{eff} < 10^{5/10}) + 2 \cdot P(10^{5/10} \leq SNR_{eff} < 10^{10/10}) + 4 \cdot P(10^{10/10} \leq SNR_{eff})\Big] . \tag{18}$$

con

$$|c|^2 \sim \text{Exponencial} \quad , \quad f_X(x) = \frac{1}{\overline{P}_r} exp\left(\frac{-x}{\overline{P}_r}\right) . \tag{19}$$

En este caso, como se mencionó en la Sección 2.1, la secuencia de amplitudes del canal tiene potencia media unitaria. Es por esto que podemos considerar $\overline{P}_r = 1$.

Ahora se puede obtener una expresión general para la probabilidad de que la SNR_{eff} se encuentre dentro de un intervalo $[a, b)$. La misma resulta

$$P(a \leq SNR_{eff} < b) = P(a \leq |c|^2 SNR < b) = P(a/SNR \leq |c|^2 < b/SNR) = \int_{a/SNR}^{b/SNR} x \cdot exp\,(-x)\,dx = \tag{20}$$

$$-\,x \cdot exp\,(-x)\Big|_{a/SNR}^{b/SNR} + \int_{a/SNR}^{b/SNR} exp\,(-x)\,dx = exp\,(-a/SNR)\left[\frac{a}{SNR} + 1\right] - exp\,(-b/SNR)\left[\frac{b}{SNR} + 1\right] .$$

Unificando las Ecuaciones (18) y (20) se puede obtener una expresión para la tasa media teórica. Se deja sin expresar explícitamente la misma por ser extensa, pero es la que se utiliza para comparar la tasa media relevada.

2.3.2. Resultados

Al simular para valores de SNR entre 0dB y 40dB, se obtiene una curva de tasa media como la presentada en la Fig. 12. En la misma se utiliza el centro del intervalo para revisar el valor de SNR_{eff}. Se puede ver cómo la curva teórica tiene la misma forma que la relevada, aunque desplazada. Esto se podría atribuir a aspectos prácticos de la simulación en casos de baja SNR, el valor inicial de la tasa no es el mismo pero convergen al mismo lugar a SNR alta. Esto es esperable, al tener una SNR alta deja de tener peso si el canal está de malas o no ya que la atenuación que puede agregar el mismo es insignificante respecto a la alta SNR con la que ingresan los datos. En este escenario solo se transmitiría con 16-QAM por lo que la tasa media queda dominada por la tasa que presenta este sistema, que es igual a 800 $kbps$.

En cuanto a la P_{eb}, este esquema no intenta mitigar de ninguna manera los eventos que causarían mayor perdida de datos. Si no que simplemente aprovecha a mandar mayor cantidad de datos cuando el canal está de buena y cuando el mismo se encuentra pinchado, decide esperar. De esta manera, se espera que el desempeño en términos de P_{eb} no sea distinto al que ya conocemos para un canal inalambrico sin conseguir diversidad. Efectivamente esto se puede corroborar en la Fig. 13, no hay diversidad ya que la curva es proporcional a la inversa de la SNR.

3. Conclusiones

Las discusiones y comparaciones pertinentes fueron realizadas durante el desarrollo del trabajo. Se consigue validar mediante simulación los resultados obtenidos teóricamente para la P_{eb} que presentan distintos esquemas de modulación y codificación en un canal inalámbrico con *fading*. Además, se realiza una caracterización de un modelo de canal con desvanecimiento *Rayleigh* y correlación *Jakes*.

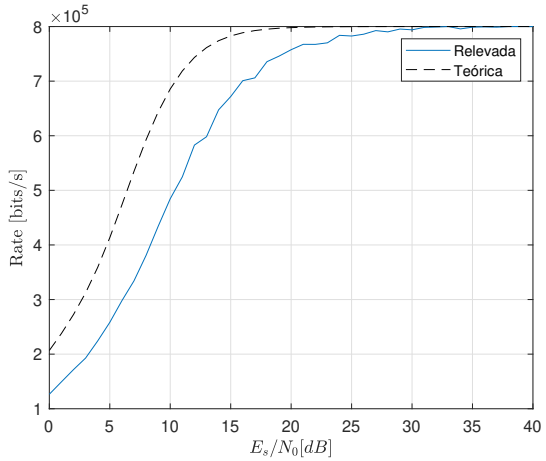


Figura 12: Tasa media obtenida en un esquema de CSI completo.

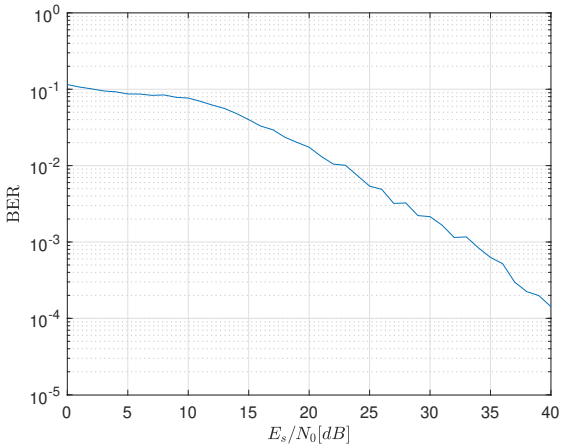


Figura 13: Probabilidad de error de bit obtenida al trabajar a tasa variable y CSI completo.

Se puede concluir que es de vital importancia aprovechar y hacer uso de las características que presenta un canal inalámbrico para poder conseguir un enlace de comunicaciones que tenga una implementación con sentido práctico. Se pudo corroborar cómo resulta inviable la utilización de un canal con *fading* sin utilizar técnicas que logren diversidad temporal ya que se precisa aumentar un orden de magnitud la SNR para lograr bajar en un orden de magnitud la P_{eb}.

En este caso se trabajó a una velocidad promedio del móvil fija, pero podría considerarse que la misma sea mayor que la aquí utilizada. En ese caso resultaría en un *doppler* máximo de mayor valor y como esta dispersión caracteriza la tasa de variación del canal, haría que las variaciones del mismo sean más rápidas. Resultando en una menor latencia a la hora de realizar entrelazado. El caso contrario implicaría trabajar con una velocidad promedio del móvil más chica, que resulta en un fading más lento y de aquí en latencias mayores para conseguir los mismos resultados.

3.1. Trabajo a futuro

En este caso el trabajo a futuro será considerado pendiente. Pero es natural proponer la comparación de los resultados con cotas para la P_{eb} menos holgadas, que deberían pegarse más a las curvas relevadas. Además sería de interés poder observar la desviación estándar que presenta la simulación de *Monte Carlo* efectuada en este trabajo, para poder analizar cuál es el rango de curvas que se pueden obtener para una

dada cantidad de realizaciones que se simula. De esta manera se podría descartar cuando los resultados no se comportan como lo esperado por falta de datos en la simulación, aquí no es completamente necesario ya que se obtienen resultados acordes a lo esperado y cuando las curvas dejan de tener suavidad se debe a falta de datos simulados. Pero como las imágenes generadas para este informe siempre son con gran cantidad de realizaciones, este último no es un problema observable aquí.

Por otro lado, se podrían agregar otros modelos de canal. No implica un desafío mayor ya que solo bastaría con cambiar el *script* que genera la secuencia de ganancias del canal y el resto se mantiene igual, se podrían analizar resultados y desempeño para modelos de canal estandarizados en las comunicaciones inalámbricas.

4. Anexo scripts de simulación MATLAB

Esta sección hace las veces de un índice de archivos para poder consultar las simulaciones y gráficos generados en este informe. El archivo se encuentra adjunto a la entrega del informe y lleva por nombre `'ProyectoWireless_Fermin.zip'`. Además, las siguientes secciones contienen el código de las distintas funciones y *scripts* de mayor importancia para el trabajo.

4.1. Repositorio con el proyecto

Se puede acceder al repositorio utilizado para el control de cambios del proyecto, con las distintas etapas del mismo. La rama *master* contiene el proyecto final con los resultados aquí mostrados.

4.2. Caracterización del canal con fading

```
clc;    clear variables;    close all;
%==========================================================================
%                            TRABAJO FINAL
%                       Curso Wireless - CoMyS 2022
%                           Llorente, J. F.
%                    Caracterizaci n del modelo de canal.
%==========================================================================

%% Introducci n
T = 1; %Tiempo de simulaci n en segundos.
ts = 5e-6;
h = CanalFlat(T,ts);
fc=900e6;                % Frecuencia de portadora (Hz)
V=60/3.6;
vel_ms = V;
lambda=3e8/fc;           % Longitud de onda
fDm=V/lambda;            % Doppler m ximo
N=round(T/ts);           % # muestras (divisible por 4D)
fs=1/ts;                 % Frecuencia de muestreo (Hz)
D=round(fDm*N/fs);       % # de puntos para filtro Jakes (el total es 4D)
fD=D*fs/N;               % Doppler m ximo redondeado
Df=fD/D;                % Paso de frecuencia para la generaci n del espectro
fr=-fD+Df:Df:fD-Df;     % Eje de frecuencias para filtro Jakes

D_s = 2*V/lambda; %Doppler spread.
T_c = 1/(4*D_s);     %Tiempo de coherencia.

%% Gr fica de la ganancia de canal
figure;
plot((0:N-1)*ts, abs(h))
grid on, grid minor;
title('Ganancia del canal')
ylabel('Valor absoluto (veces)')
xlabel('Tiempo (s)')

%% Gr fica del espectro Doppler (puede tardar un rato...)
[DOP,ff]=pwelch(h,[],[],2*fr,fs);
```

```
S_a_neg = (ff>-fDm & ff<fDm).*1./(pi*fDm*sqrt(1-(ff/fDm).^2));
figure;
plot(ff,DOP,'k'),grid on, grid minor,hold on;
title('Espectro de la ganancia de canal'),ylabel('Densidad Espectral de potencia [W/Hz]'),xlabel(
    'Frecuencia(Hz)');
ylim([-0.1*max(DOP) max(DOP)*1.1]);
plot(ff,S_a_neg,'--r');
legend('Realizaci n','Te rico');

%% Verificar Tiempo de Coherencia de manera empirica.
[AC_empirica,deltat] = xcorr(h,h);
AC_empirica = AC_empirica*ts;
figure;
plot(deltat*ts,abs(AC_empirica)),grid on,grid minor;

% El Tc da 0.00812, si tomamos el Tc = 1/Bc = 0.01 y es aprox
% igual. Es arbitrario elegri el dividido 4 o no, en Goldsmith no tiene
% div 4.
```

4.3. Desempeño con CSI parcial y tasa fija

```
%=========================================================================
%                            TRABAJO FINAL
%                      Curso Wireless - CoMyS 2022
%                          Llorente, J. F.
%                 Desempe o con CSI parcial y tasa fija
%=========================================================================
addpath('./Functions');
clc;    clear variables;
close all;
%============================CONFIGURACION=================================
theta = 0;      REP_CODE_FLAG = 1;  INT_CODE_FLAG = 1;
LW = 2;       ts = 5e-6;           M = 16 ;
%=========================================================================
N=log2(M);
%% Estimaci n de la PEB
Tc = 0.0162;
samps_inTc = floor(Tc/ts);
NumB=5e7;
Rs = 200e3;
T = 1;
n = 1;
if (REP_CODE_FLAG == 1)
    n = 4;
end
samps_toDecorr = 0.5*samps_inTc;    %Cuantos T_C espero entre s mbolos consecutivos.
samps_toDecorr = samps_toDecorr - mod(samps_toDecorr,n);

Ns_xloop = Rs;
Nb_xloop = Ns_xloop*N;
loop = ceil(NumB/Nb_xloop);
NumS = loop*Ns_xloop;
NumB = loop*Nb_xloop;

paso = 2;     limite=40; %Parametros para la relevaci n de la curva.
EsN0_dB = 0:paso:limite;
Peb = 0.*EsN0_dB;

Es = 1;
A = SymbEnergy2Amp(M,Es);
[Asignacion_bits, Asignacion_coords]=AsignacionBITSyCOORD(M,A); %Por si quiero modificar el
    c digo entre bits y simbs.
jj=1;
for EsN0db=0:paso:limite
    p = (1:loop)*0;
    %Se al. Cada loop tiene una realizaci n de bits distinta.
```

```
45      bits_t = randi([0 1],1,Nb_xloop); %Bits transmitidos.
46      [aki,akq] = generarSimbolos(bits_t,A,M);
47      ak_t = aki + 1i*akq;
48      if (REP_CODE_FLAG == 1)
49          ak_rep = repCod(ak_t,n);
50          ak = ak_rep;    %Para debug.
51      else
52          ak = ak_t;
53      end
54
55      if (INT_CODE_FLAG == 1)
56          [ak_int,ceros] = Interleaver(ak,samps_toDecorr);
57          ak = ak_int;    %Para debug.
58      else
59          ceros = 0;
60      end
61      %Ruido. Se decide generar uno distinto en cada cambio de SNR y no en
62      %cada loop porque eleva bastante el costo computacional (tiempo de
63      %simulaci n).
64      N0_veces = var(ak)/(10^(EsN0db/10));
65      if (INT_CODE_FLAG ~=1)
66          WGNi = sqrt(N0_veces/2)*randn(1,Ns_xloop*n); %RBG con varianza N0/2 = 1/2.
67          WGNq = sqrt(N0_veces/2)*randn(1,Ns_xloop*n); % En modelo son ind entonces genero dos
                veces.
68          c_noise = (WGNi + 1i*WGNq);
69      else
70          WGNi = sqrt(N0_veces/2)*randn(1,n*Ns_xloop + ceros);
71          WGNq = sqrt(N0_veces/2)*randn(1,n*Ns_xloop + ceros);
72          c_noise = (WGNi + 1i*WGNq);
73      end
74
75      for iteracion=1:loop
76          %Canal. Cada loop tiene una realizaci n de canal distinta.
77          if (REP_CODE_FLAG == 0)
78              h = CanalFlat(n*T,ts);
79              h_mat = reshape(h,n,[]);
80          else
81              h = CanalFlat(n*T + ceros*ts,ts);
82              if (INT_CODE_FLAG == 1)
83                  h_mat = reshape(deInterleaver(h,samps_toDecorr),n,[]);
84              else
85                  h_mat = reshape(h,n,[]);
86              end
87              norm_hmat = abs(h_mat).^2;
88              norm_hmat = sqrt(sum(norm_hmat));
89              %A esta altura ya tengo la norma de cada c = [c0 c1 c2 c3] para cada simbolo
                    repetido.
90          end
91  %         h = 0*h + 1;  %Para probar canal AWGN.
92          y_n = ak.*h + c_noise;
93
94  %     Dividiendo por Channel se cancelan los modulos y las fases se restan
95  %     por lo que deja de estar presente la secuencia de ganancias de canal.
96
97          if (REP_CODE_FLAG == 1)
98              if (INT_CODE_FLAG == 1)
99                  y_n = deInterleaver(y_n,samps_toDecorr);
100             end
101             y_mrl = reshape(y_n,n,[]).*conj(h_mat);
102             y_mrl = sum(y_mrl)./norm_hmat;
103             Simbolos_r_1=ReceptorOptimo(real(y_mrl),imag(y_mrl),A*norm_hmat,M,Asignacion_coords);
104         else
105             Simbolos_r_1=ReceptorOptimo(real(y_n./h),imag(y_n./h),A,M,Asignacion_coords);
106         end
107
108         bits_r_1=ConvaBits(Simbolos_r_1,Asignacion_coords,M);
109         p(iteracion)=sum(bits_r_1(1:(end-ceros))~=bits_t)/Nb_xloop;
110
```

```
111     end
112     Peb(jj) = mean(p);
113     jj=jj+1;
114 end
115 %% Gr ficos
116 fprintf("Simulaci n Wireless por canal con desvanecimiento Rayleigh.\nEsquema de modulaci n: %s
        ",ModScheme(M));
117 if(REP_CODE_FLAG == 1)
118     fprintf(" + c digo de repetici n de %d veces",n);
119 end
120 if(INT_CODE_FLAG == 1)
121     fprintf(" + entrelazado que consigue diversidad %d.\n",n);
122 else
123     fprintf(".\n");
124 end
125 fprintf("%g bits simulados.\n",NumB);
126
127 EsN0_veces = 10.^(EsN0_dB/10);
128 Peb_BPSK=qfunc(sqrt(2*EsN0_veces));%Igual a Pes
129 Peb_QPSK=qfunc(sqrt(2*EsN0_veces/2));  %Dos BPSK ind en fase y cuadratura.
130 Pes_16QAM_holgada=3*qfunc(sqrt(4/5*EsN0_veces/4));
131 Peb_16QAM_holgada=Pes_16QAM_holgada/4;
132 Peb_BPSK_fading = 0.5*(1-sqrt(EsN0_veces./(1+EsN0_veces)));
133 Peb_QPSK_fading = 0.5*(1-sqrt(EsN0_veces./(2+EsN0_veces)));
134 Peb_16QAM_fading_sinrep = 5/2./EsN0_veces;
135 Peb_16QAM_fading_cotaholgada = 15/4./(1+2/5*EsN0_veces/4);
136 Pendiente = 1./(4*EsN0_veces).^n;
137 Peb_BPSK_fading_REPyINT = nchoosek(2*n-1,n).*Pendiente;
138 Peb_16QAM_fading_int_cotaholgada = 15/4./(1+2/5*EsN0_veces/4).^n;
139 figure;
140 switch M
141     case 2
142         if(REP_CODE_FLAG && INT_CODE_FLAG)
143             semilogy(EsN0_dB,Peb,EsN0_dB,Peb_BPSK,EsN0_dB,Peb_BPSK_fading_REPyINT,'--k','
                    LineWidth',LW/4);
144         else
145             semilogy(EsN0_dB,Peb,EsN0_dB,Peb_BPSK,EsN0_dB,Peb_BPSK_fading,'--k','LineWidth',LW/4)
                    ;
146         end
147         legend('Relevada','Te rica AWGN','Te rica FADING');
148     case 4
149         if(REP_CODE_FLAG && INT_CODE_FLAG)
150             semilogy(EsN0_dB,Peb,EsN0_dB,Peb_QPSK,EsN0_dB,Pendiente,'--k','LineWidth',LW/4);
151             legend('Relevada','Te rica AWGN','Pendiente te rica FADING');
152         else
153             semilogy(EsN0_dB,Peb,EsN0_dB,Peb_QPSK,EsN0_dB,Peb_QPSK_fading,'--k','LineWidth',LW/4)
                    ;
154             legend('Relevada','Te rica AWGN','Te rica FADING');
155         end
156
157     otherwise
158         if(REP_CODE_FLAG && INT_CODE_FLAG)
159             semilogy(EsN0_dB,Peb,EsN0_dB,Peb_16QAM_holgada,EsN0_dB,
                    Peb_16QAM_fading_int_cotaholgada,'--k','LineWidth',LW/4);
160             legend('Relevada','Te rica AWGN','Cota te rica FADING');
161         elseif(REP_CODE_FLAG && ~INT_CODE_FLAG)
162             semilogy(EsN0_dB,Peb,EsN0_dB,Peb_16QAM_holgada,EsN0_dB,Peb_16QAM_fading_sinrep,
                    EsN0_dB,Peb_16QAM_fading_cotaholgada,'--k','LineWidth',LW/4);
163             legend('Relevada','Te rica AWGN','Asint tica FADING sin REP','Cota holgada FADING
                    con REP');
164         else
165             semilogy(EsN0_dB,Peb,EsN0_dB,Peb_16QAM_holgada,EsN0_dB,Peb_16QAM_fading_sinrep,'--k',
                    'LineWidth',LW/4);
166             legend('Relevada','Te rica AWGN','Asint tica FADING');
167         end
168 end
169 set(gca,'FontSize',11);
170 % title(sprintf("Curva de probabilidad de error de bit %s (%g bits)",ModScheme(M),NumB));
```

```
grid on, ylabel('BER','Interpreter','Latex'),xlabel('$$E_s/N_0 [dB]$$','Interpreter','Latex');
ylim([9.9e-6 1]);
```

4.4. Desempeño con CSI completa y tasa variable

```
%=========================================================================
%                           TRABAJO FINAL
%                     Curso Wireless - CoMyS 2022
%                         Llorente, J. F.
%                    Simulaci n a tasa variable
%=========================================================================
addpath('./Functions');
clc;    clear variables; close all;
%% %============================CONFIGURACION==============================
LW = 2;       ts = 5e-6;
INTERVAL_SET = 1;   INTERVAL_CENTER = 2;    INTERVAL_END = 3;
Interval_OP = [1 1 0; 1 2 1; 1 0 -1;0 1 0];    OP = INTERVAL_SET;

NONE  = 0;  %   - No se transmite nada.                    (0)
BPSK4 = 1;  %   - BPSK4 : BPSK con c digo de repetici n 4. (1)
QPSK4 = 2;  %   - QPSK4 : QPSK con c digo de repetici n 4. (2)
QPSK2 = 3;  %   - QPSK2 : QPSK con c digo de repetici n 2. (3)
QPSK  = 4;  %   - QPSK  : QPSK sin codigo de repetici n.   (4)
QAM16 = 5;  %   - QAM16 : 16QAM sin c digo de repetici n.  (5)
%=========================================================================
%% Estimaci n de la PEB
NumB=1e7;
Rs = 200e3;
T = 50;

T_c = 0.018;
samples_in_Tc = round(T_c/ts);
loop = floor(NumB/samples_in_Tc);

EsN0dB_vect = 0:40;
p = EsN0dB_vect*0;
R = EsN0dB_vect*0;
for jj = 1:length(EsN0dB_vect)
    times0_NONE = 0;    times1_BPSK4 = 0;    times2_QPSK4 = 0;   %Para debug.
    times3_QPSK2 = 0;    times4_QPSK = 0;     times5_QAM16 = 0;

    h = CanalFlat(2*T,ts);
    EsN0dB = EsN0dB_vect(jj);
    EsN0veces = 10^(EsN0dB/10);
    bits_t=randi([0 1],1,NumB);
    Bindx = 1;  ii=1;
    bits_r = [];
    while (Bindx<=(NumB-samples_in_Tc*4) && ii<=loop)
        i = ii - times0_NONE;   %Para evitar que no se transmitan los datos a los que les toca un
             canal pinchado.

        indx_c = floor( (Interval_OP(2,OP)*(ii-Interval_OP(1,OP)) + ...
            Interval_OP(4,OP)) *samples_in_Tc/Interval_OP(2,OP)) + ...
            Interval_OP(3,OP);   %Indice para tomar el valor en el inicio, medio o final del
                intervalo de largo T_c.
        SNReff = 20*log10(abs(h(indx_c))) + EsN0dB;
        SNRrange = (SNReff<-10)*NONE + (SNReff>=-10 && SNReff<-5)*BPSK4 + ...
            (SNReff>=-5 && SNReff<0)*QPSK4 + (SNReff>=0 && SNReff<5)*QPSK2 + ...
            (SNReff>=5 && SNReff<10)*QPSK + (SNReff>=10)*QAM16 ;
        switch SNRrange
            case NONE  % No se transmite nada si es menor que -10dB.
                times0_NONE = times0_NONE + 1;
                aux_bits = [];
            case BPSK4  % BPSK con c digo de repetici n 4 entre -10 y -5dB.
                [aux_bits,Bindx] = EtEwirelessComm(bits_t,h((ii-1)*samples_in_Tc+1:ii*
                    samples_in_Tc),Bindx,BPSK4,EsN0dB);
```

```
                times1_BPSK4 = times1_BPSK4 + 1;
            case QPSK4  % QPSK con c digo de repetici n 4 entre -5 y 0dB.
                times2_QPSK4 = times2_QPSK4 + 1;
                [aux_bits,Bindx] = EtEwirelessComm(bits_t,h((ii-1)*samples_in_Tc+1:ii*
                    samples_in_Tc),Bindx,QPSK4,EsN0dB);
            case QPSK2  % QPSK con c digo de repetici n 2 entre 0 y 5dB.
                times3_QPSK2 = times3_QPSK2 + 1;
                [aux_bits,Bindx] = EtEwirelessComm(bits_t,h((ii-1)*samples_in_Tc+1:ii*
                    samples_in_Tc),Bindx,QPSK2,EsN0dB);
            case QPSK  % QPSK entre 5 y 10dB.
                times4_QPSK = times4_QPSK + 1;
                [aux_bits,Bindx] = EtEwirelessComm(bits_t,h((ii-1)*samples_in_Tc+1:ii*
                    samples_in_Tc),Bindx,QPSK,EsN0dB);
            otherwise % 16-QAM si es mayor que 10dB.
                times5_QAM16 = times5_QAM16 + 1;
                [aux_bits,Bindx] = EtEwirelessComm(bits_t,h((ii-1)*samples_in_Tc+1:ii*
                    samples_in_Tc),Bindx,QAM16,EsN0dB);
        end
        ii = ii + 1;
        bits_r = [bits_r aux_bits];
    end
    % Calculo de tasa media relevada.
    time_sim = (ii-1)*samples_in_Tc*ts; %Ultimo indice tomado del canal, multiplicado por Ts para
         tener el tiempo.
    R(jj) = length(bits_r)/time_sim;    %Tasa media para este valor de SNR.
    bits_tt = bits_t(1:length(bits_r));  %Se descartan los bits que no se transmitieron...
    p(jj) = sum(bits_r~=bits_tt)/length(bits_tt);

end
EsN0_veces = 10.^(EsN0dB_vect/10);
% Calculo de tasa media te rica.
P_ab = @(a,b,SNR) exp(-a./SNR).*(a./SNR + 1) - exp(-b./SNR).*(b./SNR + 1);
R_m = 1/ts*(1/4*P_ab(10^(-1),10^(-0.5),EsN0_veces) + 1/2*P_ab(10^(-0.5),10^(0),EsN0_veces) ...
    + P_ab(10^(0),10^(0.5),EsN0_veces) + 2*P_ab(10^(0.5),10^(1),EsN0_veces) + 4*P_ab(10^(1)
        ,10^(100),EsN0_veces));

%% Gr ficos
figure;
semilogy(EsN0dB_vect,p,'LineWidth',LW/4);
set(gca,'FontSize',11);
% title("Curva de probabilidad de error de bit tasa variable");
grid on, ylabel('BER','Interpreter','Latex'),xlabel('$$E_s/N_0 [dB]$$','Interpreter','Latex');
ylim([9.9e-6 1]);
figure;
plot(EsN0dB_vect,R,EsN0dB_vect,R_m,'--k','LineWidth',LW/4);
set(gca,'FontSize',11);
% title("Tasa media obtenida");
grid on, ylabel('Rate [bits/s]','Interpreter','Latex'),xlabel('$$E_s/N_0 [dB]$$','Interpreter','
    Latex');
legend('Relevada','Te rica');
```

4.5. Entrelazador

```
%=========================================================================
%                          Block Interleaver
%                                JFL
%   Se elige profundidad n de manera tal que se obtenga diversidad 4 ya que
%   es como tener 4 canales en tiempo, lo que logra que la pendiente de la
%   curva de Peb vs. SNR caiga con SNR^4.
%=========================================================================
%           y = Interleaver(x,n)
%   x   --> Secuencia de datos a entrelazar.
%   n   --> Profundidad del interleaver. mmm
%
%   y   --> Secuencia entrelazada (en formato fila).
%  ceros--> La cantidad de ceros con los que complet .
```

```
%=========================================================================
function [y,ceros] = Interleaver(x,n)
    dim = size(x);
    ceros = 0;
    if(dim(1)>1)     % Se revisa que ingrese vector fila.
        x = x.';
    end
    if(mod(length(x),n)~=0)
        ceros = (n-mod(length(x),n));
        x = [x (1:(n-mod(length(x),n)))*0];     %Completo con ceros.
    end
    Block = reshape(x,[length(x)/n n]).';
    y = reshape(Block,1,[]);
end
```

4.6. Desentrelazador

```
%=========================================================================
%                         Block Deinterleaver
%                                 JFL
%=========================================================================
%           y = deInterleaver(x,n)
%   x   --> Secuencia de datos entrelazada.
%   n   --> Profundidad con la que se entrelazaron los datos.
%
%   y   --> Secuencia desentrelazada (en formato fila).
%=========================================================================
function y = deInterleaver(x,n)
    dim = size(x);
    if(dim(1)>1)     % Se revisa que ingrese vector fila.
        x = x.';
    end
    Block = reshape(x,[n length(x)/n]);
    y = reshape(Block.',1,[]);  %Se transpone aca para dejar bien formateada la matriz Block.
end
```

4.7. Modelo de canal

```
% Simulador de canal con Rayleigh Flat Fading
% Septiembre de 2012 - PAR
% ========================================================================
% h = CanalFlat(T,ts)
% T   --- Tiempo de simulaci n.
% ts  --- Tiempo de muestreo.
function h = CanalFlat(T,ts)
    fc=900e6;               % Frecuencia de portadora (Hz)
    V=60/3.6;               % velocidad en m/s
    % ====================================================================
    lambda=3e8/fc;          % Longitud de onda
    fDm=V/lambda;           % Doppler m ximo
    N=round(T/ts);          % # muestras (divisible por 4D)
    fs=1/ts;                % Frecuencia de muestreo (Hz)
    D=round(fDm*N/fs);      % # de puntos para filtro Jakes (el total es 4D)
    fD=D*fs/N;              % Doppler m ximo redondeado
    % ====================================================================
    Df=fD/D;               % Paso de frecuencia para la generaci n del espectro
    fr=-fD+Df:Df:fD-Df;    % Eje de frecuencias para filtro Jakes
    jk=sqrt(1./sqrt(1-(fr/fDm).^2));% Filtro en frecuencia!
    JK=fftshift([jk 0]);   % Filtro acomodado para usar FFT

    HC=sqrt(D)*(randn(1,2*D)+1i*randn(1,2*D)); % Espectro gaussiano BLANCO
    HJ2=HC.*JK;             % Espectro con forma JAKES
    HJN=[HC(1:D).*JK(1:D) zeros(1,N-2*D) HC(D+1:2*D).*JK(D+1:2*D)];
    %Completamos la parte nula del espectro
    %(luego al hacer IFFT se interpolan los valores de ganancia del canal)
```

```
    hj=sqrt(N)*ifft(HJN);    % Ganancia del canal compleja CON CORRELACI N JAKES
    h=hj./std(hj);           % Normalizamos para tener ganancia media 1
end
```

4.8. Comunicación inalámbrica punto a punto

```
%=========================================================================
%                    End to End Wireless Communication
%                                 JFL
%=========================================================================
%      [bits_r,Bindx_out] = EtEwirelessComm(bits_t,h,Bindx,SCHEME,EsN0dB)
%   bits_t   --> Secuencia de bits a enviar.
%   h        --> Secuencia de amplitudes del canal.
%   Bindx    --> Indice de bit de entrada.
%   SCHEME   --> Esquema de modulaci n + codificaci n.
%   - BPSK4 : BPSK con c digo de repetici n 4. (1)
%   - QPSK4 : QPSK con c digo de repetici n 4. (2)
%   - QPSK2 : QPSK con c digo de repetici n 2. (3)
%   - QPSK  : QPSK sin codigo de repetici n.   (4)
%   - QAM16 : 16QAM sin c digo de repetici n.  (5)
%   EsN0dB   --> SNR en dB de los simbolos transmitidos.
%
%   bits_r   --> Secuencia de bits recibidos.
%   Bindx_out--> Indice de bit de salida.
%=========================================================================
function [bits_r,Bindx_out] = EtEwirelessComm(bits_t,h,Bindx,SCHEME,EsN0dB)
Es = 1;

BPSK4 = 1;  %   - BPSK4 : BPSK con c digo de repetici n 4. (1)
QPSK4 = 2;  %   - QPSK4 : QPSK con c digo de repetici n 4. (2)
QPSK2 = 3;  %   - QPSK2 : QPSK con c digo de repetici n 2. (3)
QPSK  = 4;  %   - QPSK  : QPSK sin codigo de repetici n.   (4)
QAM16 = 5;  %   - QAM16 : 16QAM sin c digo de repetici n.  (5)
switch SCHEME
    case BPSK4
        repN = 4;
        M = 2;  N = log2(M);
        A = SymbEnergy2Amp(M,Es);
        [~, Asignacion_coords]=AsignacionBITSyCOORD(M,A);
        NumS = length(h);
        bits_actuales = bits_t(Bindx:(Bindx + floor(NumS*N/repN) - 1));
        if (mod(length(bits_actuales),N))
            bits_actuales = bits_actuales(1:end-mod(length(bits_actuales),N));
        end
        Bindx_out = Bindx + length(bits_actuales);
        [aki,akq] = generarSimbolos(bits_actuales,A,M);
        ak = aki + 1i*akq;
        ak = repCod(ak,repN);
        if length(ak) ~= length(h)
            h = h(1:length(ak));    %Matcheo largos si NumS/N/rep no fuese un entero...
        end
        NumS = length(ak);
        N0_veces = var(ak)/(10^(EsN0dB/10));
        WGNi = sqrt(N0_veces/2)*randn(1,NumS); %RBG con varianza N0/2 = 1/2.
        WGNq = sqrt(N0_veces/2)*randn(1,NumS); % En modelo son ind entonces genero dos veces.
        c_noise = (WGNi + 1i*WGNq);
        y_n = ak.*h + c_noise;

        h_mat = reshape(h,repN,[]); % Se puede ya que h fue llevado al largo de ak que es
            multiplo de repN.
        norm_hmat = abs(h_mat).^2;
        norm_hmat = sqrt(sum(norm_hmat));

        y_mrl = reshape(y_n,repN,[]).*conj(h_mat);
        y_mrl = sum(y_mrl)./norm_hmat;

```

```
60          Simbolos_r = ReceptorOptimo(real(y_mrl),imag(y_mrl),A*norm_hmat,M,Asignacion_coords);
61          bits_r = ConvaBits(Simbolos_r,Asignacion_coords,M);
62
63      case QPSK4
64          repN = 4;
65          M = 4;  N = log2(M);
66          A = SymbEnergy2Amp(M,Es);
67          [~, Asignacion_coords]=AsignacionBITSyCOORD(M,A);
68          NumS = length(h);
69          bits_actuales = bits_t(Bindx:(Bindx + floor(NumS*N/repN) - 1));
70          if (mod(length(bits_actuales),N))
71              bits_actuales = bits_actuales(1:end-mod(length(bits_actuales),N));
72          end
73          Bindx_out = Bindx + length(bits_actuales);
74          [aki,akq] = generarSimbolos(bits_actuales,A,M);
75          ak = aki + 1i*akq;
76          ak = repCod(ak,repN);
77          if length(ak) ~= length(h)
78              h = h(1:length(ak));    %Matcheo largos si NumS/N/rep no fuese un entero...
79          end
80          NumS = length(ak);
81          N0_veces = var(ak)/(10^(EsN0dB/10));
82          WGNi = sqrt(N0_veces/2)*randn(1,NumS); %RBG con varianza N0/2 = 1/2.
83          WGNq = sqrt(N0_veces/2)*randn(1,NumS); % En modelo son ind entonces genero dos veces.
84          c_noise = (WGNi + 1i*WGNq);
85          y_n = ak.*h + c_noise;
86
87          h_mat = reshape(h,repN,[]); % Se puede ya que h fue llevado al largo de ak que es
                multiplo de repN.
88          norm_hmat = abs(h_mat).^2;
89          norm_hmat = sqrt(sum(norm_hmat));
90
91          y_mrl = reshape(y_n,repN,[]).*conj(h_mat);
92          y_mrl = sum(y_mrl)./norm_hmat;
93
94          Simbolos_r = ReceptorOptimo(real(y_mrl),imag(y_mrl),A*norm_hmat,M,Asignacion_coords);
95          bits_r = ConvaBits(Simbolos_r,Asignacion_coords,M);
96      case QPSK2
97          repN = 2;
98          M = 4;  N = log2(M);
99          A = SymbEnergy2Amp(M,Es);
100         [~, Asignacion_coords]=AsignacionBITSyCOORD(M,A);
101         NumS = length(h);
102         bits_actuales = bits_t(Bindx:(Bindx + floor(NumS*N/repN) - 1));
103         if (mod(length(bits_actuales),N))
104             bits_actuales = bits_actuales(1:end-mod(length(bits_actuales),N));
105         end
106         Bindx_out = Bindx + length(bits_actuales);
107         [aki,akq] = generarSimbolos(bits_actuales,A,M);
108         ak = aki + 1i*akq;
109         ak = repCod(ak,repN);
110         if length(ak) ~= length(h)
111             h = h(1:length(ak));    %Matcheo largos si NumS/N/rep no fuese un entero...
112         end
113         NumS = length(ak);
114         N0_veces = var(ak)/(10^(EsN0dB/10));
115         WGNi = sqrt(N0_veces/2)*randn(1,NumS); %RBG con varianza N0/2 = 1/2.
116         WGNq = sqrt(N0_veces/2)*randn(1,NumS); % En modelo son ind entonces genero dos veces.
117         c_noise = (WGNi + 1i*WGNq);
118         y_n = ak.*h + c_noise;
119
120         h_mat = reshape(h,repN,[]); % Se puede ya que h fue llevado al largo de ak que es
                multiplo de repN.
121         norm_hmat = abs(h_mat).^2;
122         norm_hmat = sqrt(sum(norm_hmat));
123
124         y_mrl = reshape(y_n,repN,[]).*conj(h_mat);
125         y_mrl = sum(y_mrl)./norm_hmat;
```

```
126
127         Simbolos_r = ReceptorOptimo(real(y_mrl),imag(y_mrl),A*norm_hmat,M,Asignacion_coords);
128         bits_r = ConvaBits(Simbolos_r,Asignacion_coords,M);
129     case QPSK
130         repN = 1;
131         M = 4;  N = log2(M);
132         A = SymbEnergy2Amp(M,Es);
133         [~, Asignacion_coords]=AsignacionBITSyCOORD(M,A);
134         NumS = length(h);
135         bits_actuales = bits_t(Bindx:(Bindx + floor(NumS*N/repN) - 1));
136         if (mod(length(bits_actuales),N))
137             bits_actuales = bits_actuales(1:end-mod(length(bits_actuales),N));
138         end
139         Bindx_out = Bindx + length(bits_actuales);
140         [aki,akq] = generarSimbolos(bits_actuales,A,M);
141         ak = aki + 1i*akq;
142         if length(ak) ~= length(h)
143             h = h(1:length(ak));    %Matcheo largos si NumS/N/rep no fuese un entero...
144         end
145         NumS = length(ak);
146         N0_veces = var(ak)/(10^(EsN0dB/10));
147         WGNi = sqrt(N0_veces/2)*randn(1,NumS); %RBG con varianza N0/2 = 1/2.
148         WGNq = sqrt(N0_veces/2)*randn(1,NumS); % En modelo son ind entonces genero dos veces.
149         c_noise = (WGNi + 1i*WGNq);
150         y_n = ak.*h + c_noise;
151         Simbolos_r = ReceptorOptimo(real(y_n./h),imag(y_n./h),A,M,Asignacion_coords);
152         bits_r = ConvaBits(Simbolos_r,Asignacion_coords,M);
153     case QAM16
154         repN = 1;
155         M = 16;  N = log2(M);
156         A = SymbEnergy2Amp(M,Es);
157         [~, Asignacion_coords]=AsignacionBITSyCOORD(M,A);
158         NumS = length(h);
159         bits_actuales = bits_t(Bindx:(Bindx + floor(NumS*N/repN) - 1));
160         if (mod(length(bits_actuales),N))
161             bits_actuales = bits_actuales(1:end-mod(length(bits_actuales),N));
162         end
163         Bindx_out = Bindx + length(bits_actuales);
164         [aki,akq] = generarSimbolos(bits_actuales,A,M);
165         ak = aki + 1i*akq;
166         if length(ak) ~= length(h)
167             h = h(1:length(ak));    %Matcheo largos si NumS/N/rep no fuese un entero...
168         end
169         NumS = length(ak);
170         N0_veces = var(ak)/(10^(EsN0dB/10));
171         WGNi = sqrt(N0_veces/2)*randn(1,NumS); %RBG con varianza N0/2 = 1/2.
172         WGNq = sqrt(N0_veces/2)*randn(1,NumS); % En modelo son ind entonces genero dos veces.
173         c_noise = (WGNi + 1i*WGNq);
174         y_n = ak.*h + c_noise;
175         Simbolos_r = ReceptorOptimo(real(y_n./h),imag(y_n./h),A,M,Asignacion_coords);
176         bits_r = ConvaBits(Simbolos_r,Asignacion_coords,M);
177     otherwise
178         disp('El esquema esta mal!');
179 end
180 end
```

4.9. Funciones propias para la simulación independiente del canal

Las funciones aquí presentadas son reutilizadas del trabajo de simulación llevado a cabo para *Comunicaciones Digitales*.

4.9.1. Asignación de bits y coordenadas para cada símbolo

```
1 %========================================================================
2 %                     ASIGNACION de SIMBOLOS y COORDENADAS
```

```
%                          x Ferm n Llorente
%==========================================================================
%
%   M   ---> Cantidad de simbolos del sistema.
%   A   ---> Valor de la amplitud A en la constelaci n.
%
%  [Asignacion_bits , Asignacion_coords]=AsignacionBITSyCOORD(M,A);
%==========================================================================
function [Asignacion_bits , Asignacion_coords]=AsignacionBITSyCOORD(M,A)
switch M
    case 2
        s0=0;   coordS0=-A;
        s1=1;   coordS1=A;
        Asignacion_bits=[s0;s1];
        Asignacion_coords=[coordS0,coordS1];
    case 4
        s0=[0 0];   coordS0=A+1i*A;
        s1=[0 1];   coordS1=-A+1i*A;
        s3=[1 1];   coordS3=-A-1i*A;
        s2=[1 0];   coordS2=A-1i*A;
        Asignacion_bits=[s0;s1;s2;s3];
        Asignacion_coords=[coordS0,coordS1,coordS2,coordS3];
    case 16
        s0=[0 0 0 0];   coordS0=-3+3*1i;
        s1=[0 0 0 1];   coordS1=-3+1i;
        s2=[0 0 1 0];   coordS2=-3-3*1i;
        s3=[0 0 1 1];   coordS3=-3-1i;
        s4=[0 1 0 0];   coordS4=-1+3*1i;
        s5=[0 1 0 1];   coordS5=-1 +1i;
        s6=[0 1 1 0];   coordS6=-1-3*1i;
        s7=[0 1 1 1];   coordS7=-1-1i;
        s8=[1 0 0 0];   coordS8=3+3*1i;
        s9=[1 0 0 1];   coordS9=3+1i;
        s10=[1 0 1 0];  coordS10=3-3*1i;
        s11=[1 0 1 1];  coordS11=3-1i;
        s12=[1 1 0 0];  coordS12=1+3*1i;
        s13=[1 1 0 1];  coordS13=1+1i;
        s14=[1 1 1 0];  coordS14=1-3*1i;
        s15=[1 1 1 1];  coordS15=1-1i;

        Asignacion_bits=[s0;s1;s2;s3;s4;s5;s6;s7;s8;s9;s10;s11;s12;s13;s14;s15];
        Asignacion_coords=A*[coordS0,coordS1,coordS2,coordS3,coordS4,coordS5,coordS6,coordS7,
            coordS8,...
                                coordS9,coordS10,coordS11,coordS12,coordS13,coordS14,coordS15];
end
```

4.9.2. Conversión de símbolos a bits según la asignación que presenta la función anterior

```
%==========================================================================
%                      Conversion de simbolos a bits
%                          x Ferm n Llorente
%==========================================================================
%   s   ---> Vector de simbolos
%   Asignacion_coords   ---> Asignaci n de coordenadas para cada simbolo.
%   M   ---> Cantidad de simbolos del sistema.
%
%   b = ConvaBits(s,Asignacion_coords,M);
%==========================================================================
function b=ConvaBits(s,Asignacion_coords,M)
    N=log2(M);
    dim=length(s);
    simbs=zeros(M,dim);

    switch M
        case 2
            b=(s==Asignacion_coords(2));%S1 es un uno y S0 es un cero.
        case 4
```

```
            simbs(1,:)=s.*0;
            simbs(2,:)=(s==Asignacion_coords(2))*1;
            simbs(3,:)=(s==Asignacion_coords(3))*2;
            simbs(4,:)=(s==Asignacion_coords(4))*3;
            simbs_num=sum(simbs);
            bits_array=fliplr(de2bi(simbs_num));
            b=reshape(bits_array.',[1 dim*N]);
        case 16
            simbs(1,:)=s.*0;
            simbs(2,:)=(s==Asignacion_coords(2))*1;
            simbs(3,:)=(s==Asignacion_coords(3))*2;
            simbs(4,:)=(s==Asignacion_coords(4))*3;
            simbs(5,:)=(s==Asignacion_coords(5))*4;
            simbs(6,:)=(s==Asignacion_coords(6))*5;
            simbs(7,:)=(s==Asignacion_coords(7))*6;
            simbs(8,:)=(s==Asignacion_coords(8))*7;
            simbs(9,:)=(s==Asignacion_coords(9))*8;
            simbs(10,:)=(s==Asignacion_coords(10))*9;
            simbs(11,:)=(s==Asignacion_coords(11))*10;
            simbs(12,:)=(s==Asignacion_coords(12))*11;
            simbs(13,:)=(s==Asignacion_coords(13))*12;
            simbs(14,:)=(s==Asignacion_coords(14))*13;
            simbs(15,:)=(s==Asignacion_coords(15))*14;
            simbs(16,:)=(s==Asignacion_coords(16))*15;
            simbs_num=sum(simbs);
            bits_array=fliplr(de2bi(simbs_num));
            b=reshape(bits_array.',[1 dim*N]);
    end
end
```

4.9.3. Generar símbolos en base a un vector de bits

```
%=========================================================================
%                          Generar simbolos
%                          x Ferm n Llorente
%=========================================================================
%   Funci n que recibe los bits a transmitir y le asigna coordenadas a cada
%   combinaci n de bits. Devuelve las coordenadas complejas de cada simbolo
%   transmitido.
%
%   b   ---> Vector de bits
%   A   ---> Valor de la amplitud A en la constelaci n.
%   M   ---> Cantidad de simbolos del sistema de comunicaciones.
%
%   [aki,akq]=generarSimbolos(b,A,M)
%=========================================================================
function [aki,akq]=generarSimbolos(b,A,M)
N=log2(M);
NumS=length(b)/N;
ak=zeros(1,NumS);
switch M
    case 2  %BPSK
        s0=0;   coordS0=-A;
        s1=1;   coordS1=A;
        simbs_0=b==s0;
        simbs_1=b==s1;
        ak=simbs_0*coordS0+simbs_1*coordS1;
    case 4  %QPSK
        % Asigno secuencia de bits y coordenadas a simbolos.
        s0=[0 0];   coordS0=A+1i*A;
        s1=[0 1];   coordS1=-A+1i*A;
        s2=[1 1];   coordS2=-A-1i*A;
        s3=[1 0];   coordS3=A-1i*A;

        simb_bits=reshape(b,[N NumS]);
        aux0=simb_bits(1:N,:)==s0.';
        aux1=simb_bits(1:N,:)==s1.';
```

```
36          aux2=simb_bits(1:N,:)==s2.';
37          aux3=simb_bits(1:N,:)==s3.';
38          simbs_0=aux0(1,:)==1 & aux0(2,:)==1;
39          simbs_1=aux1(1,:)==1 & aux1(2,:)==1;
40          simbs_2=aux2(1,:)==1 & aux2(2,:)==1;
41          simbs_3=aux3(1,:)==1 & aux3(2,:)==1;
42          ak=simbs_0*coordS0+simbs_1*coordS1+simbs_2*coordS2+simbs_3*coordS3;
43
44      case 16  %16QAM
45          s0=[0 0 0 0];   coordS0=-3+3*1i;
46          s1=[0 0 0 1];   coordS1=-3+1i;
47          s2=[0 0 1 0];   coordS2=-3-3*1i;
48          s3=[0 0 1 1];   coordS3=-3-1i;
49          s4=[0 1 0 0];   coordS4=-1+3*1i;
50          s5=[0 1 0 1];   coordS5=-1 +1i;
51          s6=[0 1 1 0];   coordS6=-1-3*1i;
52          s7=[0 1 1 1];   coordS7=-1-1i;
53          s8=[1 0 0 0];   coordS8=3+3*1i;
54          s9=[1 0 0 1];   coordS9=3+1i;
55          s10=[1 0 1 0];  coordS10=3-3*1i;
56          s11=[1 0 1 1];  coordS11=3-1i;
57          s12=[1 1 0 0];  coordS12=1+3*1i;
58          s13=[1 1 0 1];  coordS13=1+1i;
59          s14=[1 1 1 0];  coordS14=1-3*1i;
60          s15=[1 1 1 1];  coordS15=1-1i;
61          simb_bits=reshape(b,[N NumS]);
62          aux0=simb_bits(1:N,:)==s0.';
63          aux1=simb_bits(1:N,:)==s1.';
64          aux2=simb_bits(1:N,:)==s2.';
65          aux3=simb_bits(1:N,:)==s3.';
66          aux4=simb_bits(1:N,:)==s4.';
67          aux5=simb_bits(1:N,:)==s5.';
68          aux6=simb_bits(1:N,:)==s6.';
69          aux7=simb_bits(1:N,:)==s7.';
70          aux8=simb_bits(1:N,:)==s8.';
71          aux9=simb_bits(1:N,:)==s9.';
72          aux10=simb_bits(1:N,:)==s10.';
73          aux11=simb_bits(1:N,:)==s11.';
74          aux12=simb_bits(1:N,:)==s12.';
75          aux13=simb_bits(1:N,:)==s13.';
76          aux14=simb_bits(1:N,:)==s14.';
77          aux15=simb_bits(1:N,:)==s15.';
78          %Algoritmo que sirve para cualquier codigo,  a  n  si no es el de Gray
79          simbs_0=aux0(1,:)==1 & aux0(2,:)==1 & aux0(3,:)==1 & aux0(4,:)==1;
80          simbs_1=aux1(1,:)==1 & aux1(2,:)==1 & aux1(3,:)==1 & aux1(4,:)==1;
81          simbs_2=aux2(1,:)==1 & aux2(2,:)==1 & aux2(3,:)==1 & aux2(4,:)==1;
82          simbs_3=aux3(1,:)==1 & aux3(2,:)==1 & aux3(3,:)==1 & aux3(4,:)==1;
83          simbs_4=aux4(1,:)==1 & aux4(2,:)==1 & aux4(3,:)==1 & aux4(4,:)==1;
84          simbs_5=aux5(1,:)==1 & aux5(2,:)==1 & aux5(3,:)==1 & aux5(4,:)==1;
85          simbs_6=aux6(1,:)==1 & aux6(2,:)==1 & aux6(3,:)==1 & aux6(4,:)==1;
86          simbs_7=aux7(1,:)==1 & aux7(2,:)==1 & aux7(3,:)==1 & aux7(4,:)==1;
87          simbs_8=aux8(1,:)==1 & aux8(2,:)==1 & aux8(3,:)==1 & aux8(4,:)==1;
88          simbs_9=aux9(1,:)==1 & aux9(2,:)==1 & aux9(3,:)==1 & aux9(4,:)==1;
89          simbs_10=aux10(1,:)==1 & aux10(2,:)==1 & aux10(3,:)==1 & aux10(4,:)==1;
90          simbs_11=aux11(1,:)==1 & aux11(2,:)==1 & aux11(3,:)==1 & aux11(4,:)==1;
91          simbs_12=aux12(1,:)==1 & aux12(2,:)==1 & aux12(3,:)==1 & aux12(4,:)==1;
92          simbs_13=aux13(1,:)==1 & aux13(2,:)==1 & aux13(3,:)==1 & aux13(4,:)==1;
93          simbs_14=aux14(1,:)==1 & aux14(2,:)==1 & aux14(3,:)==1 & aux14(4,:)==1;
94          simbs_15=aux15(1,:)==1 & aux15(2,:)==1 & aux15(3,:)==1 & aux15(4,:)==1;
95          ak = simbs_0*coordS0 + simbs_1*coordS1 + simbs_2*coordS2 + simbs_3*coordS3 + simbs_4*
                coordS4 + simbs_5*coordS5 +...
96               simbs_6*coordS6 + simbs_7*coordS7 + simbs_8*coordS8 + simbs_9*coordS9 + simbs_10*
                    coordS10 + simbs_11*coordS11 +...
97               simbs_12*coordS12 + simbs_13*coordS13 + simbs_14*coordS14 + simbs_15*coordS15;
98          ak = ak*A;
99  end
100 aki=real(ak);
101 akq=imag(ak);
```

```
102 end
```

4.9.4. Receptor óptimo

```
1  %=========================================================================
2  %                          Receptor  ptimo
3  %                         x Ferm n Llorente
4  %=========================================================================
5  %
6  %   Devuelve un vector de largo 'cantidad de simbolos' con las coordenadas
7  %   de los simbolos que se recibieron luego de tomar la decisi n con las
8  %   fronteras  ptimas.
9  %
10 %   ak_r_i ---> Coordenadas recibidas en la rama en fase.
11 %   ak_r_q ---> Coordenadas recibidas en la rama en cuadratura.
12 %   A      ---> Amplitud A de las coordenadas de la constelaci n.
13 %   M      ---> Numero de simbolos del sistema que se modela.
14 %   AsignacionCoords ---> Coordenadas de los M simbolos.
15 %
16 %   simb=ReceptorOptimo(ak_r_i,ak_r_q,A,M,AsignacionCoords)
17 %=========================================================================
18 function simb=ReceptorOptimo(ak_r_i,ak_r_q,A,M,AsignacionCoords)
19     if(imag(ak_r_q)~=0)
20         ak_r_q = imag(ak_r_q);
21     end
22     simb_aux=zeros(M,length(ak_r_i));
23     switch M
24         case 2
25             simb_aux(1,:)=(ak_r_i<0).*AsignacionCoords(1);
26             simb_aux(2,:)=(ak_r_i>0).*AsignacionCoords(2);
27             simb=sum(simb_aux);
28         case 4
29             simb_aux(1,:)=((ak_r_i>0)&(ak_r_q>0)).*AsignacionCoords(1);
30             simb_aux(2,:)=((ak_r_i<0)&(ak_r_q>0)).*AsignacionCoords(2);
31             simb_aux(3,:)=((ak_r_i>0)&(ak_r_q<0)).*AsignacionCoords(3);
32             simb_aux(4,:)=((ak_r_i<0)&(ak_r_q<0)).*AsignacionCoords(4);
33             simb=sum(simb_aux);
34         case 16
35             simb_aux(1,:)=((ak_r_i<-2*A)&(ak_r_q>2*A)).*AsignacionCoords(1);
36             simb_aux(2,:)=((ak_r_i<-2*A)&(ak_r_q>0 & ak_r_q<2*A)).*AsignacionCoords(2);
37             simb_aux(3,:)=((ak_r_i<-2*A)&(ak_r_q<-2*A)).*AsignacionCoords(3);
38             simb_aux(4,:)=((ak_r_i<-2*A)&(ak_r_q<0 & ak_r_q>-2*A)).*AsignacionCoords(4);
39             simb_aux(5,:)=((ak_r_i<0 & ak_r_i>-2*A)&(ak_r_q>2*A)).*AsignacionCoords(5);
40             simb_aux(6,:)=((ak_r_i<0 & ak_r_i>-2*A)&(ak_r_q>0 & ak_r_q<2*A)).*AsignacionCoords(6)
                   ;
41             simb_aux(7,:)=((ak_r_i<0 & ak_r_i>-2*A)&(ak_r_q<-2*A)).*AsignacionCoords(7);
42             simb_aux(8,:)=((ak_r_i<0 & ak_r_i>-2*A)&(ak_r_q<0 & ak_r_q>-2*A)).*AsignacionCoords
                   (8);
43             simb_aux(9,:)=((ak_r_i>2*A)&(ak_r_q>2*A)).*AsignacionCoords(9);
44             simb_aux(10,:)=((ak_r_i>2*A)&(ak_r_q>0 & ak_r_q<2*A)).*AsignacionCoords(10);
45             simb_aux(11,:)=((ak_r_i>2*A)&(ak_r_q<-2*A)).*AsignacionCoords(11);
46             simb_aux(12,:)=((ak_r_i>2*A)&(ak_r_q<0 & ak_r_q>-2*A)).*AsignacionCoords(12);
47             simb_aux(13,:)=((ak_r_i>0 & ak_r_i<2*A)&(ak_r_q>2*A)).*AsignacionCoords(13);
48             simb_aux(14,:)=((ak_r_i>0 & ak_r_i<2*A)&(ak_r_q>0 & ak_r_q<2*A)).*AsignacionCoords
                   (14);
49             simb_aux(15,:)=((ak_r_i>0 & ak_r_i<2*A)&(ak_r_q<-2*A)).*AsignacionCoords(15);
50             simb_aux(16,:)=((ak_r_i>0 & ak_r_i<2*A)&(ak_r_q<0 & ak_r_q>-2*A)).*AsignacionCoords
                   (16);
51             simb=sum(simb_aux);
52     end
53 end
```

4.9.5. Obtener amplitud de las coordenadas a partir de la energía de símbolo

```
1  function A = SymbEnergy2Amp(M,Es)
```

```
    switch (M)
        case 2
            A = sqrt(Es);
        case 4
            A = sqrt(Es/2);
        otherwise    %16QAM
            A = sqrt(Es/10);
    end
end
```

4.9.6. Obtener string con el nombre de la modulación usada

```
function System = ModScheme(M)
    switch (M)
        case 2
            System = "BPSK";
        case 4
            System = "QPSK";
        case 16
            System = "16QAM";
        otherwise
            System = "Error";
    end
end
```

Referencias

[1] A. Goldsmith, “Wireless Communications”, Cambridge University Press, 2005.

[2] D. Tse and P. Viswanath, “Fundamentals of Wireless Communications”, Cambridge University Press, 2008.