ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ANÁLISIS CUALITATIVO Y CUANTITATIVO DE MODELOS PARA EL CÁLCULO DEL PER (PACKET ERROR RATE) PARA COMUNICACIONES IEEE802.11P

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

XAVIER ALEJANDRO FLORES CABEZAS
xavier.flores@epn.edu.ec

DIRECTORA: **Ph.D. MARTHA CECILIA PAREDES PAREDES**
cecilia.paredes@epn.edu.ec

Quito, enero 2019

AVAL

Certifico que el presente trabajo fue desarrollado por XAVIER ALEJANDRO FLORES CABEZAS, bajo mi supervisión.

Ph.D. MARTHA CECILIA PAREDES PAREDES
DIRECTORA DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo XAVIER ALEJANDRO FLORES CABEZAS, declaro bajo juramento que el trabajo aquí escrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual, correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normatividad institucional vigente.

XAVIER ALEJANDRO FLORES CABEZAS

DEDICATORIA

Al Alejandro de hace 10 años,

porque la vida siempre puede mejorar

AGRADECIMIENTOS

Hay momentos en la vida que la definen; uno de ellos recuerdo tan claro y tan lejano: mi padre enseñándome las estrellas y diciéndome que yo podría lograr cualquier cosa que me propusiese... y por eso, antes que nada, agradezco a mi padre Diego Flores, por encender la chispa de lo que soy ahora.

Le debo gratitud a muchas personas por estar en este momento de mi vida, pero pocas tan importantes como mi madre Jenny Cabezas, quien logró salvarme en el momento cuando más necesité salvación, y quien me ha mostrado que siempre es posible ser alguien mejor.

Dios me ha puesto al lado de muchas personas maravillosas, siendo una de las más, mi novia y compañera Jhosselyne Ostaiza, una persona llena de luz y potencial quien me ha ayudado a crecer de maneras que no creía posibles, y a conocerme a mí mismo en rincones que no había visitado, a quien agradezco por llenar al mundo de colores más brillantes y por brindarme su apoyo tan necesitado.

Le agradezco a mi hermana Ivi, quien ha sido un hombro en el cual sostenerme cuando he necesitado y por ser la energía opuesta que se vuelve necesaria. A mi hermano Tomás por intentar, ya que en la vida se necesitan contraejemplos y soñadores.

A mi tío Julio que, probablemente sin percatarse, me encaminó en mi carrera y a mi tío Edison, quien siempre intentó brindarme su mano en apoyo. A mis abuelos Sergio y Ali, así como a mi tía Joyce, quienes siempre vieron por nuestro bien. Así también a mi abuela Fani, por ser una roca en su familia.

Un particular agradecimiento a la doctora Martha Paredes y al doctor Luis Urquiza por su completo apoyo a lo largo de este proyecto, por ser ejemplos a seguir y el tipo de tutores que animan a uno a hacer las cosas y hacerlas bien.

Y le agradezco a Dios, por darme una vida llena de bendiciones y oportunidades. Por poner a personas maravillosas en mi camino y permitirme caminar de su lado.

ÍNDICE DE CONTENIDO

AVAL I

DECLARACIÓN DE AUTORÍA II

DEDICATORIA III

AGRADECIMIENTO IV

ÍNDICE DE CONTENIDO V

ÍNDICE DE FIGURAS VIII

ÍNDICE DE TABLAS X

ÍNDICE DE ECUACIONES XI

ÍNDICE DE DIAGRAMAS DE FLUJO XIV

ÍNDICE DE ALGORITMOS XVII

RESUMEN XVIII

ABSTRACT XIX

1 INTRODUCCIÓN **1**

1.1 Objetivos 1

1.2 Alcance 1

1.3 Marco Teórico 3

1.3.1 VANETs 4

1.3.2 WAVE 5

1.3.3 IEEE 802.11p 6

1.3.4 Canal Inalámbrico 26

1.3.5 Parámetros de rendimiento de un sistema inalámbrico 27

1.3.6 Modelos para el cálculo de la PER 29

1.3.7 Selección de modelos 44

2 METODOLOGÍA **46**

2.1 Escenario de simulación 46

2.2 Sistema de comunicación IEEE 802.11p 48

2.2.1 Simulación principal 48

2.2.2 Transmisión 49

2.2.3 Recepción 57

2.3 Modelos teóricos predictivos de la PER 64

2.3.1 Modelo teórico predictivo del BER 1 65

2.3.2 Modelo teórico predictivo del BER 2 66

2.3.3 Modelo teórico predictivo del BER 3 66

2.4 Tratamiento de datos 67

2.4.1 Evaluación de modelos teóricos 67

2.4.2 Optimización de modelos 67

2.4.3 Extensión del dominio para resultados de SNR angosto 69

2.4.4 Minimización del error y evaluación de modelos teóricos 70

2.4.5 Consolidación de datos 70

2.4.6 Presentación de resultados 74

2.4.7 Análisis de resultados 75

2.5 Plan de ejecución de *scripts* 75

2.6 Implementación y ejecución de *scripts* 76

3 RESULTADOS Y DISCUSIÓN **77**

3.1 Resultados de PER de la simulación 77

3.2 Comparación de modelos teóricos 78

3.2.1 Análisis cualitativo 88

3.3 Comparación de modelos teóricos minimizados 96

3.3.1 Análisis cuantitativo 105

3.4 Análisis de resultados 106

3.5 Producto demostrable 107

3.5.1 Funcionalidad 108

3.5.2 Desempeño 108

3.5.3 Diseño robusto e implementación 109
3.5.4 Facilidad de uso 110

4 CONCLUSIONES Y RECOMENDACIONES **111**
4.1 Conclusiones 111
4.2 Recomendaciones 113

5 REFERENCIAS BIBLIOGRÁFICAS **114**

6 ANEXOS **117**

ANEXO I **118**

ANEXO II **160**

ANEXO III **187**

ÍNDICE DE FIGURAS

Figura 1.1 (a) Comunicación V2I (b) Comunicación V2V……4
Figura 1.2 Aplicaciones ITS……5
Figura 1.3 Arquitectura WAVE ……6
Figura 1.4 Alcance IEEE 802.11p ……..7
Figura 1.5 Formato de PPDU……..8
Figura 1.6 Preámbulo corto del PPDU ……..9
Figura 1.7 Valores generadores del preámbulo corto en tiempo……9
Figura 1.8 Valores generadores del preámbulo largo en tiempo ……..10
Figura 1.9 Preámbulo PPDU……10
Figura 1.10 Procesamiento del PPDU……11
Figura 1.11 Aleatorizador ……13
Figura 1.12 Codificador convolucional utilizado en IEEE 802.11p ……15
Figura 1.13 Puntura de un código convolucional ……16
Figura 1.14 Índices en proceso de entrelazado ……17
Figura 1.15 Ejemplo de entrelazado para BPSK ……18
Figura 1.16 Diagramas de constelación para BPSK, QPSK, 16QAM y 64QAM ……..19
Figura 1.17 Ortogonalidad OFDM …… 21
Figura 1.18 Asignación de subportadoras a la IFFT ……23
Figura 1.19 Distribución de las subportadoras OFDM …… 25
Figura 1.20 Proceso de preparación del prefijo cíclico ……26
Figura 1.21 Recepción de señales desfasadas con prefijo cíclico …… 26
Figura 1.22 Canal AWGN ..…… 27
Figura 2.1 Estructura de las funciones y *scripts* de la simulación ……47
Figura 3.1 Resultados simulados de la PER para todas las configuraciones ……. 77
Figura 3.2 Resultados simulados de la PER para todas las configuraciones (semilogarítmico) …… 78
Figura 3.3 Comparación de modelos teóricos con resultados simulados para BPSK r=$1/2$ ……... 80
Figura 3.4 Comparación de modelos teóricos con resultados simulados para BPSK r=$3/4$ ..…… 81
Figura 3.5 Comparación de modelos teóricos con resultados simulados para QPSK r=$1/2$ ……... 82
Figura 3.6 Comparación de modelos teóricos con resultados simulados para

QPSK r=$3/4$ 83
Figura 3.7 Comparación de modelos teóricos con resultados simulados para 16QAM r=$1/2$ 84
Figura 3.8 Comparación de modelos teóricos con resultados simulados para 16QAM r=$3/4$ 85
Figura 3.9 Comparación de modelos teóricos con resultados simulados para 64QAM r=$2/3$ 86
Figura 3.10 Comparación de modelos teóricos con resultados simulados para 64QAM r=$3/4$ 87
Figura 3.11 Comparación de modelos teóricos minimizados con resultados simulados para BPSK r=$1/2$ 97
Figura 3.12 Comparación de modelos teóricos minimizados con resultados simulados para BPSK r=$3/4$ 98
Figura 3.13 Comparación de modelos teóricos minimizados con resultados simulados para QPSK r=$1/2$ 99
Figura 3.14 Comparación de modelos teóricos minimizados con resultados simulados para QPSK r=$3/4$ 100
Figura 3.15 Comparación de modelos teóricos minimizados con resultados simulados para 16QAM r=$1/2$ 101
Figura 3.16 Comparación de modelos teóricos minimizados con resultados simulados para 16QAM r=$3/4$ 102
Figura 3.17 Comparación de modelos teóricos minimizados con resultados simulados para 64QAM r=$2/3$ 103
Figura 3.18 Comparación de modelos teóricos minimizados con resultados simulados para 64QAM r=$3/4$ 104

ÍNDICE DE TABLAS

Tabla 1.1 Familia de estándares WAVE 6

Tabla 1.2 Configuraciones permitidas para IEEE 802.11p 12

Tabla 1.3 Tabla de verdad de operación XOR binaria 13

Tabla 1.4 Esquemas de modulación para IEEE 802.11p 18

Tabla 1.5 Factores de normalización para modulación en IEEE802.11p 20

Tabla 1.6 Asignación de subportadoras a la IFFT 22

Tabla 1.7 Tabla de constantes para ecuación del PER 31

Tabla 1.8 Valores de las constantes c_m y k_m para cada esquema de modulación 33

Tabla 1.9 d_{free} y a_d para codificador convolucional $K = 7$ de la capa física OFDM IEEE 802.11p 37

Tabla 2.1 Resumen de diagramas de flujo y algoritmos por modelo de PER 65

Tabla 3.1 Figuras de la comparación de modelos teóricos de la PER 79

Tabla 3.2 Figuras de la comparación de modelos teóricos minimizados de la PER 96

Tabla 3.3 *Offsets* minimizados de modelos teóricos 104

Tabla 3.4 Errores minimizados de modelos teóricos 104

Tabla 3.5 Resumen de resultados cuantitativos de los modelos de la PER 106

Tabla 4.1 Modelos escogidos por configuración con su offset respectivo 111

Tabla 4.2 Modelos más cercanos por configuración 112

ÍNDICE DE ECUACIONES

Ecuación 1.1 Número de bits por símbolo modulado 12
Ecuación 1.2 Polinomio generador del aleatorizador 12
Ecuación 1.3 Tasa de codificación de código convolucional 13
Ecuación 1.4 Polinomios generadores del codificador convolucional (base 8) 14
Ecuación 1.5 Polinomios generadores del codificador convolucional (base 2) 14
Ecuación 1.6 Patrones de puntura para $r = {}^{3}/_{4}$ y $r = {}^{2}/_{3}$ 15
Ecuación 1.7 Primera permutación del entrelazador 17
Ecuación 1.8 Segunda permutación del entrelazador 17
Ecuación 1.9 Parámetro s para el entrelazado 18
Ecuación 1.10 Transformada inversa de Fourier 21
Ecuación 1.11 Transformada inversa de Fourier discreta 21
Ecuación 1.12 Número de subportadoras totales 23
Ecuación 1.13 Espacio entre subportadoras 23
Ecuación 1.14 Periodo de un símbolo OFDM sin prefijo cíclico 23
Ecuación 1.15 Periodo de un intervalo de guarda 24
Ecuación 1.16 Periodo de un símbolo OFDM con prefijo cíclico 24
Ecuación 1.17 Número de bits por subportadora 24
Ecuación 1.18 Número de bits codificados por símbolo OFDM 24
Ecuación 1.19 Número de bits de datos (antes de codificación convolucional) por símbolo OFDM 24
Ecuación 1.20 BER 28
Ecuación 1.21 BER en dB 28
Ecuación 1.22 PER 29
Ecuación 1.23 PER en dB 29
Ecuación 1.24 PER teórica de acuerdo a [1] 30
Ecuación 1.25 Parámetro a_R de [1] 30
Ecuación 1.26 Parámetro b_R de [1] 31
Ecuación 1.27 PER teórica clásico mostrado en [2] 31
Ecuación 1.28 BER teórica de acuerdo a [2] 31
Ecuación 1.29 Función Q 32
Ecuación 1.30 BER por esquema de modulación de acuerdo a [3] en función del ${E_b}/{N_0}$ 32

Ecuación 1.31 Relación entre E_b/N_0 y SNR..... 32
Ecuación 1.32 BER por esquema de modulación de acuerdo a [3] en función del SNR 33
Ecuación 1.33 PER teórica mediante EVT de acuerdo a [2] 33
Ecuación 1.34 Parámetro a_N de [2] 34
Ecuación 1.35 Parámetro b_N de [2] 34
Ecuación 1.36 Función de error inversa 34
Ecuación 1.37 Parámetros de la función de error inversa 34
Ecuación 1.38 PER teórica para canales con *fading* de acuerdo a [2] 35
Ecuación 1.39 Función Gamma 35
Ecuación 1.40 Relación entre E_b/N_0 y SNR para símbolos OFDM 35
Ecuación 1.41 BER por esquema de modulación de acuerdo a [4] 36
Ecuación 1.42 Cota superior teórica de la PER de acuerdo a [4] 36
Ecuación 1.43 Cota superior teórica de la PER de acuerdo a [4], para los 10 primeros valores de d 37
Ecuación 1.44 PER teórica por esquema de modulación de acuerdo a [4] 37
Ecuación 1.45 BER teórica por esquema de modulación con *fading* de acuerdo a [4] 38
Ecuación 1.46 Parámetro μ de [4] 38
Ecuación 1.47 Tasa de codificación de código convolucional 38
Ecuación 1.48 Longitud de código de bloque equivalente 39
Ecuación 1.49 Tasa de codificación equivalente 39
Ecuación 1.50 *Event Error Rate* 39
Ecuación 1.51 Parámetro λ de [5] 39
Ecuación 1.52 PER teórica de acuerdo a [5] 40
Ecuación 1.53 PER promedio 40
Ecuación 1.54 PER teórica en función de la SNR con *fading* de acuerdo a [6] 40
Ecuación 1.55 PER promedio contemplando PDF 40
Ecuación 1.56 Parámetro ω_0 en [6] 41
Ecuación 1.57 Cota superior de la PER promedio de acuerdo a [6] 41
Ecuación 1.58 PER promedio cuando la SNR tiende al infinito de acuerdo a [6] 41
Ecuación 1.59 Característica empinada de la SNR 41
Ecuación 1.60 PER promedio para SNR empinada de acuerdo a [6] 41
Ecuación 1.61 PDF de la PER en función de la SNR para canales con *fading* de acuerdo a [6] 41

Ecuación 1.62 PER promedio para canales con *fading* de acuerdo a [6] 42
Ecuación 1.63 Probabilidad de que un paquete de L bits no se encuentre errado 42
Ecuación 1.64 Probabilidad de que un paquete de L bits se encuentre errado (PER clásica) 42
Ecuación 1.65 BER teórica de acuerdo a las librerías de MATLAB para modulación BPSK 43
Ecuación 1.66 BER teórica de acuerdo a las librerías de MATLAB para modulación QPSK 43
Ecuación 1.67 BER teórica de acuerdo a las librerías de MATLAB para modulación 16QAM 43
Ecuación 1.68 BER teórica de acuerdo a las librerías de MATLAB para modulación 64QAM 44

ÍNDICE DE DIAGRAMAS DE FLUJO (ANEXO I)

Figura I.1 Nomenclatura para los diagramas de flujo ……118
Figura I.2 Diagrama de flujo de la simulación principal (SNR homogéneo) …..... 119
Figura I.3 Diagrama de flujo de la simulación principal (SNR angosto) ……..120
Figura I.4 Diagrama de flujo del módulo de Tx ……..121
Figura I.5 Diagrama de flujo de la generación del SYNC ……121
Figura I.6 Diagrama de flujo de la generación del preámbulo corto …….. 122
Figura I.7 Diagrama de flujo de la generación del preámbulo corto …….. 122
Figura I.8 Diagrama de flujo de la generación de SIGNAL …… 123
Figura I.9 Diagrama de flujo de la generación de DATA …… 124
Figura I.10 Diagrama de flujo de la aleatorización …… 125
Figura I.11 Diagrama de flujo de la codificación ……..126
Figura I.12 Diagrama de flujo del entrelazado ……127
Figura I.13 Diagrama de flujo del modulado …… 128
Figura I.14 Diagrama de flujo de la modulación OFDM ……129
Figura I.15 Diagrama de flujo de la formación de símbolo OFDM.……130
Figura I.16 Diagrama de flujo del módulo de Rx ……130
Figura I.17 Diagrama de flujo de la extracción del DATA ……..130
Figura I.18 Diagrama de flujo del procesamiento de DATA ……..131
Figura I.19 Diagrama de flujo de la demodulación OFDM ……132
Figura I.20 Diagrama de flujo del mapeo de símbolo OFDM …… 133
Figura I.21 Diagrama de flujo de la demodulación …… 134
Figura I.22 Diagrama de flujo del deentrelazado …… 135
Figura I.23 Diagrama de flujo de la decodificación ……136
Figura I.24 Diagrama de flujo de modulación BPSK …….. 137
Figura I.25 Diagrama de flujo de modulación QPSK ……..138
Figura I.26 Diagrama de flujo de modulación 16QAM ……... 139
Figura I.27 Diagrama de flujo de modulación 64QAM ……... 140
Figura I.28 Diagrama de flujo de demodulación BPSK ……..141
Figura I.29 Diagrama de flujo de demodulación QPSK ……..142
Figura I.30 Diagrama de flujo de demodulación 16QAM ……...143
Figura I.31 Diagrama de flujo de demodulación 64QAM ……144
Figura I.32 Diagrama de flujo del modelo teórico 1 …… 145

Figura I.33 Diagrama de flujo del modelo teórico 2 146
Figura I.34 Diagrama de flujo del modelo teórico 3 147
Figura I.35 Diagrama de flujo del modelo teórico i, con i = 4 , i = 5 o i = 6.....148
Figura I.36 Diagrama de flujo del modelo teórico i, con i = 7, i = 8 o i = 9.....148
Figura I.37 Diagrama de flujo del modelo teórico del BER 1.....149
Figura I.38 Diagrama de flujo del modelo teórico del BER 2.....150
Figura I.39 Diagrama de flujo del modelo teórico del BER 3.....151
Figura I.40 Diagrama de flujo del script de evaluación de modelos 152
Figura I.41 Diagrama de flujo de la minimización del error 153
Figura I.42 Diagrama de flujo del análisis del PER teórico 154
Figura I.43 Diagrama de flujo de extensión de PER 155
Figura I.44 Diagrama de flujo de comparación de valores de PER 156
Figura I.45 Diagrama de flujo de consolidación de datos 157
Figura I.46 Diagrama de flujo de la graficación de resultados 158
Figura I.47 Diagrama de flujo del análisis de resultados 159

ÍNDICE DE ALGORITMOS (ANEXO II)

Algoritmo 1 Simulacion (SNR homogéneo) 160
Algoritmo 2 Simulacion (SNR angosto) 161
Algoritmo 3 txOFDM() 162
Algoritmo 4 CrearSYNC() 162
Algoritmo 5 CrearPreambuloCorto() 162
Algoritmo 6 CrearPreambuloLargo() 162
Algoritmo 7 CrearSIGNAL() 163
Algoritmo 8 CrearDATA() 163
Algoritmo 9 scrambleString() 164
Algoritmo 10 codificarString() 164
Algoritmo 11 entrelazarString() 165
Algoritmo 12 modularString() 166
Algoritmo 13 modulacionOFDM() 166
Algoritmo 14 armarSimboloOFDM() 166
Algoritmo 15 rxOFDM() 167
Algoritmo 16 extraerData() 167
Algoritmo 17 procesarDATA() 167
Algoritmo 18 demodulacionOFDM() 167
Algoritmo 19 extraerSimboloOFDM() 168
Algoritmo 20 demodularString() 168
Algoritmo 21 deentrelazarString() 169
Algoritmo 22 decodificarString() 170
Algoritmo 23 modularBPSK() 171
Algoritmo 24 modularQPSK() 171
Algoritmo 25 modular16QAM() 172
Algoritmo 26 modular64QAM() 173
Algoritmo 27 demodularBPSK() 174
Algoritmo 28 demodularQPSK() 174
Algoritmo 29 demodular16QAM() 175
Algoritmo 30 demodular64QAM() 176

Algoritmo 31 modelo1() 177
Algoritmo 32 modelo2() 178
Algoritmo 33 modelo3() 178
Algoritmo 34 modeloX() (X=4,5,6) 179
Algoritmo 35 modeloX() (X=7,8,9) 179
Algoritmo 36 modeloBER1() 180
Algoritmo 37 modeloBER2() 180
Algoritmo 38 modeloBER3() 181
Algoritmo 39 evaluarModelo() 181
Algoritmo 40 minimizarErrorEQX() 182
Algoritmo 41 ANALISIS_PER 182
Algoritmo 42 EXTENSION_PER 183
Algoritmo 43 COMPARACION_PER 184
Algoritmo 44 CONSOLIDACION_DATOS 184
Algoritmo 45 GRAFICACION_RESULTADOS 185
Algoritmo 46 ANALISIS_RESULTADOS 186

RESUMEN

El presente proyecto de titulación busca comparar los modelos teóricos de la tasa de paquetes errados (*Packet Error Rate*, PER) en función de la relación señal a ruido (*Signal-to-Noise Ratio*, SNR) con respecto a los resultados de la PER obtenidos a través de la simulación de un sistema de comunicación IEEE 802.11p [7]. El objetivo es encontrar el modelo que más se aproxime al simulado, considerando parámetros de transmisión como el esquema de modulación (M) y la tasa de codificación (r). Posteriormente se obtiene el error cuadrático entre la PER de cada modelo y el simulado, para luego buscar un *offset* de SNR, tal que el error sea el mínimo. Los modelos que se implementaron fueron escogidos de un conjunto de modelos predictivos de PER analizados en el capítulo uno de este trabajo de titulación.

Este trabajo presenta como resultados: (1) un resumen de las redes vehiculares y sus estándares, (2) un modelo de comunicación a nivel de capa física IEEE 802.11p sobre un canal AWGN (*Additive White Gaussian Noise*) en MATLAB, (3) un análisis teórico de los modelos predictivos del cálculo de la PER, (4) la implementación paramétrica en MATLAB de los modelos matemáticos para el cálculo de la PER de un conjunto de modelos seleccionados y (5) la determinación del modelo que mejor se adecue a escenarios ad-hoc vehiculares en base a los resultados de las simulaciones.

PALABRAS CLAVE: PER, IEEE 802.11p, OFDM, comparación, modelos teóricos, simulación

ABSTRACT

This final career project looks to compare PER theoretical models as functions of SNR, regarding PER results from an IEEE 802.11p system simulation. The objective is to find the model that most closely approximates the simulated results considering transmission parameters such as the modulation scheme (M) and the coding rate (r). Afterwards, the quadratic error between the PER of each model and the simulated results is obtained to then look for an SNR offset, so that the error be minimal. The chosen models are taken from a set of predictive PER models analyzed in chapter one of this final career project.

This work presents as results the following: (1) a summary of vehicular networks and their standards, (2) a physical layer level IEEE 802.11p communication model over an AWGN channel in MATLAB, (3) a theoretical analysis of the predictive models for PER calculation, (4) the parametric implementation of the mathematical PER predictive models in MATLAB out of a set of chosen models, (5) the model that best fits vehicular ad-hoc scenarios regarding the simulated results, is determined.

KEYWORDS: PER, IEEE 802.11p, OFDM, comparison, theoretical models, simulation

1. INTRODUCCIÓN

En este capítulo se tratarán los aspectos teóricos asociados al proyecto de titulación, además de sus objetivos y el alcance del mismo.

Dentro de los aspectos teóricos se revisarán conceptos básicos de redes vehiculares, en particular la familia de estándares WAVE (*Wireless Access in Vehicular Environments*). De esta, se estudiará el estándar IEEE 802.11p y los procesos físicos comprendidos en el mismo. Finalmente se revisarán modelos teóricos para el cálculo de la PER y se escogerán los apropiados para su análisis.

1.1. Objetivos

El objetivo general de este estudio técnico es analizar cualitativa y cuantitativamente modelos para el cálculo de la PER para comunicaciones IEEE 802.11p.

Los objetivos específicos de este estudio técnico son:

- Describir brevemente la capa física OFDM (*Orthogonal Frequency Division Multi plex*) del estándar IEEE 802.11p y los modelos de la PER existentes.
- Diseñar los *scripts* y funciones para la implementación de la capa física del sistema de comunicación IEEE 802.11p así como los correspondientes a los modelos de la PER.
- Implementar en *scripts* de MATLAB el sistema de comunicaciones IEEE 802.11p y los modelos de cálculos de la PER escogidos.
- Analizar los modelos implementados con respecto a los resultados de la simulación de manera gráfica y matemática basada en errores.

1.2. Alcance

El proyecto de titulación entregará: (1) un modelo realizado en MATLAB mediante *scripts* (archivos .m) de un sistema de comunicación regido por el estándar IEEE 802.11p sobre un canal AWGN (*Additive White Gaussian Noise*) y (2) la implementación paramétrica de los modelos matemáticos para el cálculo de la PER seleccionados y determinará el modelo que mejor se adecue a escenarios ad-hoc vehiculares.

Modelo de comunicación: El modelo de la capa física OFDM seguirá el proceso de formación y recepción descrito en el estándar IEEE 802.11 [8] y se utilizará un canal AWGN, es decir, se implementarán todos los elementos de la capa física desde el procesamiento de la señal hasta la creación y concatenación de preámbulo, cabecera y el rellenado de bits.

Se implementarán las ocho posibles combinaciones de tasa de codificación y esquema de modulación disponibles en el estándar IEEE 802.11 [8] que permiten obtener las diferentes tasas de transmisión de datos.

Se simulará una transmisión de una cadena de bits generados aleatoriamente, agrupados en paquetes de igual tamaño, es decir, para cada paquete se generará su preámbulo y cabecera, se extenderán los datos mediante bits de relleno, estos datos se pasarán en orden a través de:

- Un aleatorizador (*scrambler*) que baraja los datos con respecto a una semilla inicial.
- Un codificador convolucional que otorga la capacidad de corrección de errores a cambio de ingresar bits de redundancia.
- Un entrelazador (*interleaver*) que permuta los bits dentro del paquete.
- Un modulador que asigna a grupos de bits dentro del paquete símbolos de acuerdo al diagrama de constelaciones utilizado.
- Ensamblado de símbolos OFDM para la asignación de cada símbolo modulado a la subportadora del símbolo OFDM respectiva.
- Transformada inversa de Fourier rápida (*Inverse Fast Fourier Transform*, IFFT) para pasar el símbolo al dominio del tiempo.
- La agregación del prefijo cíclico que añade una porción del mismo símbolo OFDM en tiempo al principio del mismo para combatir los efectos adversos del multitrayecto.

Finalmente se transmiten los datos por el canal inalámbrico y en recepción se sigue el proceso inverso para la recuperación de los bits iniciales.

Modelos matemáticos para el cálculo de la PER: Se implementarán *scripts* y funciones en MATLAB correspondientes a los modelos matemáticos descritos escogidos, los cuales retornarán el valor de la PER para los argumentos que se listan a continuación:

- Longitud del paquete en bits.
- Relación señal a ruido (*Signal to Noise Ratio*, SNR).

- Esquema de modulación y tasa de codificación.

Cualquier otro parámetro que sea necesario para implementar los modelos matemáticos se tomará de las especificaciones del estándar IEEE 802.11p. Haciendo uso del modelo del sistema de comunicación IEEE 802.11p, se obtendrán datos de la PER para un rango de SNR adecuado, sobre el cual se simulará el envío de un número fijo de paquetes y se comprobará la existencia de errores en recepción uno a uno a la salida de la capa física por cada valor de SNR del rango. Esto se realizará para cada una de las 8 combinaciones (o configuraciones) posibles de esquemas de modulación y tasas de codificación permitidas.

Con los mismos rangos de SNR y los parámetros para cada una de las 8 configuraciones posibles, se obtendrá la misma información de los modelos matemáticos escogidos, que es, el valor de la PER por cada valor de SNR analizado. Estos datos serán comparados con los datos simulados de manera gráfica mediante gráficas de MATLAB, y de manera matemática mediante cálculo de errores y minimización del *offset* de SNR. Se determinará cuál ecuación presenta una mejor aproximación para cada uno de los casos en base a la comparación matemática.

Finalmente se realizarán *scripts* y funciones de MATLAB, los cuales encontrarán para cada una de las 8 configuraciones posibles y para cada modelo matemático utilizado, el *offset* aplicado a la SNR con el cual se minimiza el error con respecto a los datos simulados obteniendo el error mínimo. Una vez obtenido el error mínimo y el *offset* de SNR correspondiente para cada modelo matemático utilizado en cada una de las 8 configuraciones posibles con respecto a los resultados de la simulación, los *scripts* compararán, para cada configuración, los errores mínimos de cada modelo y darán como resultado la comparación gráfica de errores mínimos por modelo por cada configuración. Con esto podrá elegirse para cada configuración el modelo matemático más favorable con respecto a los resultados simulados y su *offset* en SNR correspondiente. Como resultado de este trabajo de titulación, se presentará un producto demostrable.

1.3. Marco Teórico

Esta sección empezará con un repaso de las tecnologías de redes vehiculares, de las cuales se tomará particularmente la tecnología WAVE (*Wireless Access in Vehicular Environment*), que utiliza el estándar IEEE 802.11p a nivel de capa física y subcapa MAC. Además se realizará un breve estudio de los fundamentos de OFDM (*Orthogonal Frequency Division Multiplexing*) y los procesos involucrados en el mismo. Finalmente se revisarán los modelos teóricos para el cálculo de la PER.

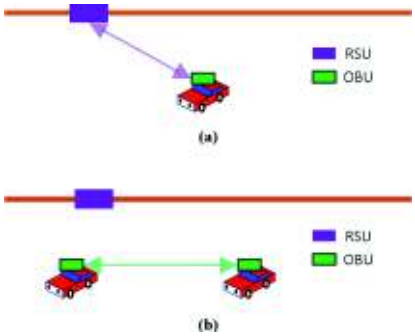


Figura 1.1: (a) Comunicación V2I (b) Comunicación V2V

1.3.1. VANETs

Los sistemas de transporte inteligente (*Intelligent Transport Systems*, ITS) comprenden una amplia gama de tecnologías aplicadas a medios de transporte para poder brindar varios servicios a los usuarios de automóviles [9].

Las redes ad-hoc vehiculares (*Vehicular Ad-Hoc NETworks*, VANETs) son una subclase de las redes ad-hoc móviles (*Mobile Ad-Hoc NETworks*, MANETs), las cuales permiten la comunicación en un ambiente vehicular y son una parte esencial de los ITSs. Existen dos tipos de comunicaciones: entre vehículos (*Vehicle-to-Vehicle*, V2V) o bien, entre un vehículo y equipos de infraestructura fijos colocados a lo largo de la carretera (*Vehicle-to-Infrastructure*, V2I). En su conjunto, a estas comunicaciones se les abrevia como V2x. Un ejemplo que permite aclarar estas comunicaciones se observa en la Figura 1.1. Por su naturaleza, las VANETs son redes de amplia cobertura, bajo consumo energético y que exigen baja latencia [10]. La comunicación dentro de una VANET se la realiza mediante dos tipos de dispositivos [11]:

- **OBU (*On Board Unit*):** interfaz inalámbrica ubicada en el vehículo que le permite comunicarse con otros vehículos o con las unidades fijas en la carretera.
- **RSU (*Road Side Unit*):** unidades inalámbricas de infraestructura colocadas a lo largo de la carretera. Pueden actuar como puntos de acceso (*Access Points*, APs) inalámbricos.

Mediante las redes vehiculares es posible proveer a los usuarios de los vehículos, servicios en forma de aplicaciones, las cuales se pueden dividir en aplicaciones de seguridad y aplicaciones de usuario. Las aplicaciones de seguridad, como su nombre lo sugiere, se encuentran orientadas principalmente a la disminución de los accidentes en la carretera, pero

también se interesan por el control de flujo de tráfico en las intersecciones y por los estados de la congestión vehicular. Por otro lado, las aplicaciones de usuario están orientadas al confort de los usuarios del vehículo (tanto conductor como pasajeros) proveyendo servicios agregados como acceso a internet y aplicaciones P2P (*Peer-to-Peer*) [12]. Una categorización alternativa de aplicaciones para ITS puede evidenciarse en la Figura 1.2 [12].

Figura 1.2: Aplicaciones ITS [12]

Al ser estas, tecnologías emergentes, existen propuestas de varias organizaciones para arquitecturas de ITSs. La ISO propone la arquitectura CALM (*Continuos Air Interface for Long to Medium range*), el C2C-CC propone la arquitectura C2CNet complementada con GeoNet, la IETF incorpora el concepto de VANETs a NEMO (*Network Mobility*) en MANETs, y la IEEE propone la arquitectura WAVE [13]. La banda de frecuencia aprobada para WAVE, CALM y C2CNet es la de 5.9 GHz [14] [15].

1.3.2. WAVE

WAVE, propuesta por IEEE para ITS, es el nombre que denota la familia de estándares que define la arquitectura y estandariza el conjunto de servicios e interfaces para comunicaciones inalámbricas seguras de hasta 1000 m de distancia, para velocidades de hasta 27 Mbps y de baja latencia en ambientes vehiculares [16]. Los estándares WAVE son un conjunto de normas que forman la familia IEEE 1609, expuestas en la Tabla 1.1 en donde se observa a detalle el uso de cada norma de esta familia [17].

Figura 1.3: Arquitectura WAVE [17]

La arquitectura WAVE cuenta con dos planos: uno de datos y otro de administración, cada cual regido bajo sus propios estándares [17]. En la Figura 1.3 se muestra un modelo gráfico de la arquitectura WAVE donde se observan las diferentes normas de la familia IEEE 1609, a nivel de capa física (PHY) y subcapa MAC (*Medium Access Control*) el estándar IEEE 802.11p y a nivel de subcapa LLC (*Logical Link Control*) el estándar IEEE 802.2 [17].

Tabla 1.1: Familia de estándares WAVE [17]

Estándar	Uso
IEEE P1609.0	Arquitectura
IEEE 1609.1-2006	Administración de recursos
IEEE 1609.2	Servicios de seguridad para aplicaciones y mensajes de administración
IEEE 1609.3-2010	Servicios de red
IEEE 1609.4-2010	Operación multi-canal (Capa MAC)
IEEE 1609.5	Administración de comunicaciones
IEEE 1609.6	Administración de instalaciones de aplicaciones
IEEE 1609.11-2010	Protocolo de intercambio de datos de pagos *Over-the-Air*
IEEE 1609.12	Localizaciones de identificadores

1.3.3. IEEE 802.11p

El estándar IEEE 802.11p cubre la capa física (PHY) y subcapa MAC de la arquitectura WAVE, y es destinado para su aplicación en ambientes vehiculares. Su funcionamiento se

Figura 1.4: Alcance IEEE 802.11p [8]

basa en comunicaciones OFDM en el rango de frecuencias de 5.850 GHz a 5.925 GHz, con un ancho de banda total de 75 MHz con 7 canales de 10 MHz cada uno [16] [7]. Las características de la subcapa MAC y de la capa física IEEE 802.11p se fundamentan en IEEE 802.11 [8]. IEEE 802.11p modifica ciertas características de IEEE 802.11 debido al entorno operativo de las redes vehiculares, que es un entorno rápidamente cambiante.

a. Capa PHY

La capa física se divide en 3 subcapas: PLME (*Physical Layer Management Entity*), PLCP (*Physical Layer Convergence Procedure*) y PMD (*Physical Medium Dependent*). El alcance y la relación de IEEE 802.11p con el modelo ISO/OSI se observa en la Figura 1.4 [7].

b. Subcapa PLME

Entidad de administración de la capa física, posee interfaces para comunicarse con una entidad de administración de la estación y para comunicarse con la entidad de administración a nivel MAC llamada MLME (*MAC Sublayer Management Entity*) a través de los puntos de acceso al servicio (*Service Access Points*, SAPs) PLME_SAP y MLME-PLME_SAP respectivamente [8] [18].

La PLME contiene la base de datos de administración (*Management Information Base*, MIB) de la capa física, a la cual se puede acceder desde una entidad de administración de la estación para obtener información de ella, o editarla para configurarla [18].

c. Subcapa PLCP

Subcapa de datos de la capa física. PLCP sirve como una interfaz entre la subcapa MAC que entrega la trama lógica llamada MPDU (*MAC Protocol Data Unit*), que en nivel PLCP se llama PSDU (*PLCP Service Data Unit*) y entrega a la subcapa PMD para que coloque la información en el medio [18].

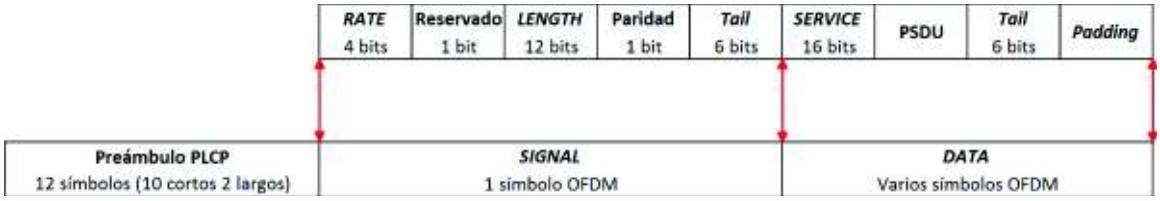


Figura 1.5: Formato de PPDU [8]

c. Subcapa PLCP

Subcapa de datos de la capa física. PLCP sirve como una interfaz entre la subcapa MAC que entrega la trama lógica llamada MPDU (*MAC Protocol Data Unit*), que en nivel PLCP se llama PSDU (*PLCP Service Data Unit*) y entrega a la subcapa PMD para que coloque la información en el medio [18].

La subcapa PLCP pone al PSDU en un formato adecuado, realizando un encapsulamiento a nivel de capa física tal que contenga información acerca de los parámetros de transmisión y de la propia trama (este encapsulamiento consiste en añadir un preámbulo, una cabecera y un *trailer* al PSDU, formando así una trama) [18] [8]. Esta trama de capa física es llamada PPDU (*PLCP Protocol Data Unit*) la cual es enviada a la subcapa PMD para su tratamiento y envío a través del medio físico.

c.1. Trama PLCP (PPDU): El encapsulamiento del PSDU en la subcapa PLCP se muestra en la Figura 1.5. Los campos RATE, LENGTH, Paridad (*Parity*), *Tail* y los bits reservados de la cabecera constituyen el campo SIGNAL, el cual es transmitido como un símbolo OFDM con modulación BPSK y tasa de codificación $r = {}^{1}/_{2}$ (la velocidad más baja). El PSDU junto al campo SERVICE, los bits de cola y los bits de relleno (*Padding*) son llamados el DATA. A esto le precede el preámbulo que consta de 12 símbolos de entrenamiento de estructura fija. La concatenación del preámbulo, el campo SIGNAL y el DATA conforman el PPDU PLCP que será procesado por la subcapa PMD y posteriormente enviado por el medio inalámbrico [8].

c.2. Construcción del preámbulo: El preámbulo se divide en dos partes: un preámbulo largo y un preámbulo corto. El preámbulo corto se crea mediante la concatenación de 10 secuencias cortas, mientras que el largo se crea mediante la concatenación de 2 secuencias largas y su respectivo intervalo de guarda.

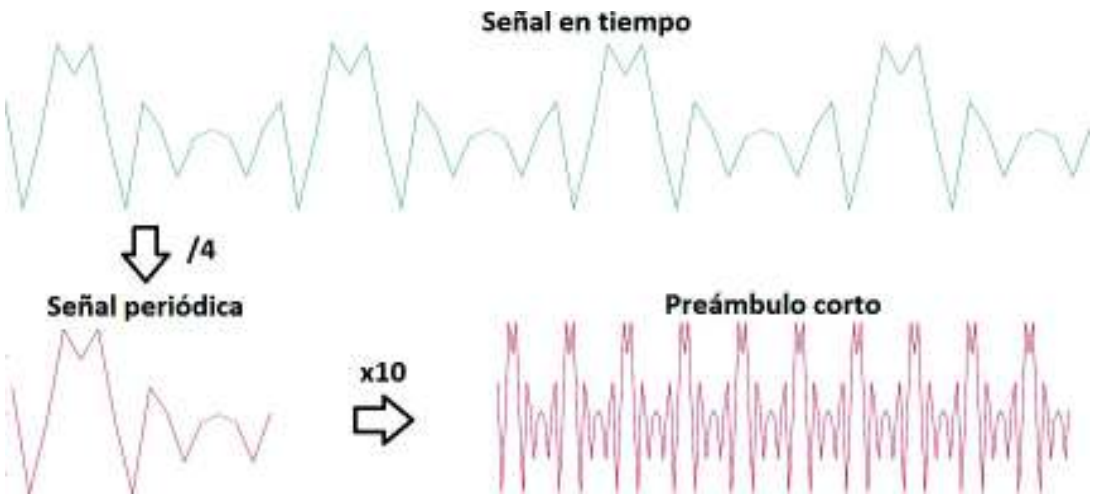


Figura 1.6: Preámbulo corto del PPDU

Índice del arreglo	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Subportadora	-26	-25	-24	-23	-22	-21	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	DC
Valor	0	0	1+i	0	0	0	-1-i	0	0	0	1+i	0	0	0	-1-i	0	0	0	-1-i	0	0	0	1+i	0	0	0	0

Índice del arreglo	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52
Subportadora	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Valor	0	0	0	-1-i	0	0	0	-1-i	0	0	0	1+i	0	0	0	1+i	0	0	0	1+i	0	0	0	1+i	0	0

Figura 1.7: Valores generadores del preámbulo corto en tiempo [8]

El preámbulo corto se genera mediante los siguientes pasos y se resumen en la Figura 1.6.

1. Generar un arreglo de 53 elementos (52 subportadoras + DC).
2. Asignar a este arreglo los valores mostrados en la Figura 1.7.
3. Multiplicar el arreglo por $\sqrt{13/6}$ y pasar por el módulo IFFT.
4. Cortar el primer 1/4 del arreglo resultante (16 muestras) para formar una secuencia corta.
5. Concatenar 10 secuencias cortas una después de la otra, este es el preámbulo corto.

La razón por la que se recorta 1/4 del arreglo después del IFFT es porque el arreglo en frecuencia contiene solo 12 elementos distintos de cero ubicados cada uno a 4 subportadoras de distancia del anterior. Esto resulta en una señal en tiempo con periodicidad de $^1/_4 T_{IFFT}$ por lo que 1/4 de la señal caracteriza a toda la señal por completo [8].

Se obtienen 16 muestras en total porque la salida de la IFFT son símbolos de 64 muestras

en tiempo. Esto se explica con mayor detalle en la sección 1.3.3.

Índice del arreglo	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Subportadora	-26	-25	-24	-23	-22	-21	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	DC
Valor	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	0

Índice del arreglo	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52
Subportadora	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Valor	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

Figura 1.8: Valores generadores del preámbulo largo en tiempo [8]

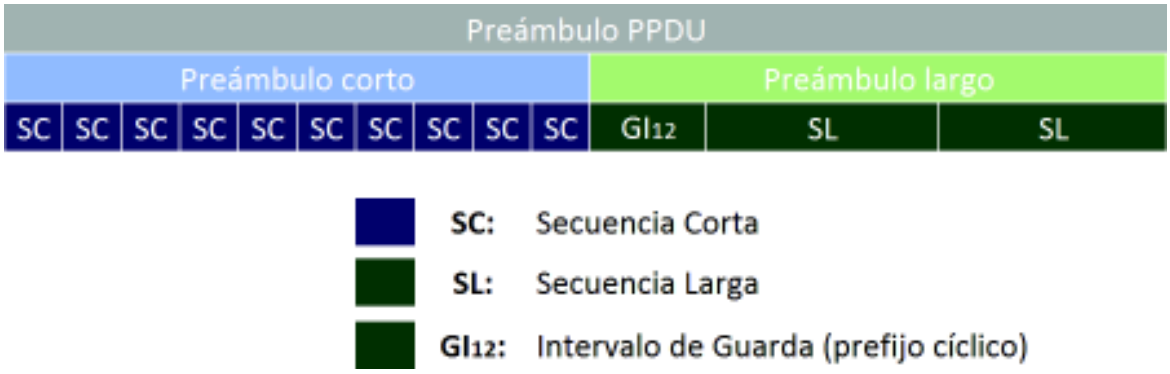


Figura 1.9: Preámbulo PPDU [8]

El preámbulo largo se genera mediante los siguientes pasos.

1. Generar un arreglo de 53 elementos (52 subportadoras + DC).
2. Asignar a este arreglo los valores mostrados en la Figura 1.8.
3. Pasar este arreglo por el módulo IFFT. El arreglo resultante es una secuencia larga.
4. Obtener la última mitad de la secuencia larga (32 muestras), que corresponde a GI_{12} (prefijo cíclico del preámbulo largo).
5. Concatenar en orden, el GI_{12} y dos secuencias largas para formar el preámbulo largo.

El preámbulo total se genera concatenando el preámbulo corto y el preámbulo largo uno después del otro (Figura 1.9).

d. Subcapa PMD

Subcapa de datos de la capa física. PMD se encuentra en contacto con el medio externo y por debajo de la subcapa PLCP. Esta subcapa se encarga de recibir los datos PLCP con el formato correcto y adecuarlas al medio al que se encuentre conectada [8].

La subcapa PMD ofrece a la PLCP el servicio de envío y recepción de tramas en el medio inalámbrico al convertir la información binaria en señales de radio. Provee, además, los

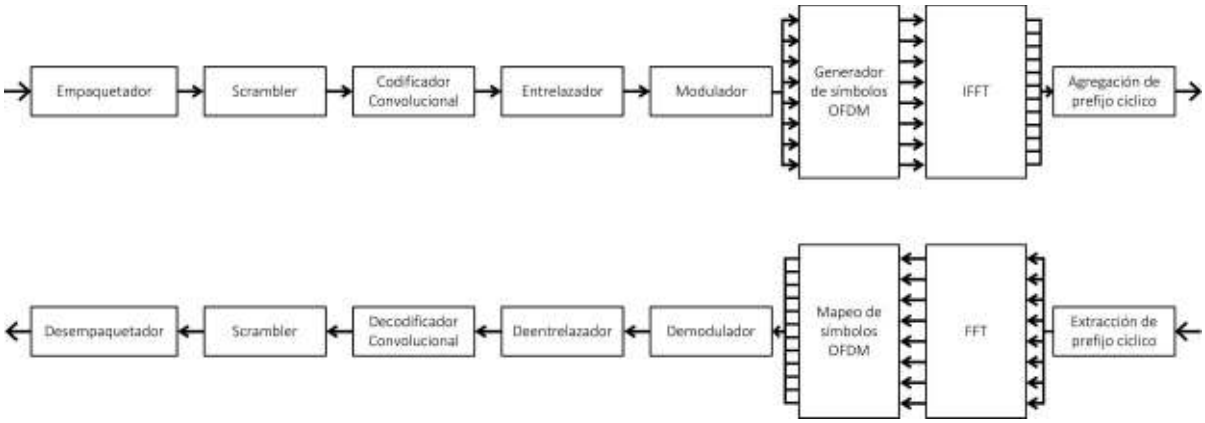


Figura 1.10: Procesamiento del PPDU [8]

procesos para generar un símbolo OFDM a partir de una cadena de bits y todos los procesos que esto implica (aleatorización, codificación, entrelazado, modulación, IFFT, agregación de prefijo cíclico), así como sus análogos en recepción (extracción del prefijo cíclico, FFT, demodulación, deentrelazado, decodificación, aleatorización) como se puede observar en la Figura 1.10 [18]. Este proceso se realiza para el campo DATA. La aleatorización no es llevada a cabo para los bits de cola [8].

Se revisará a detalle cada uno de estos procesos. Cabe mencionar que el orden de estos procesos sigue el diagrama de la Figura 1.10. Dentro de estos procesos existen ciertos parámetros que deben ser definidos previo a su análisis.

d.1. Parámetros de transmisión: Existen varios parámetros concernientes al tratamiento y envío de los datos a nivel de la PMD que se deben definir y explicar.

El estándar puede alcanzar, dependiendo de la combinación de esquema de modulación y tasa de codificación, diferentes velocidades de transmisión. Las configuraciones permitidas se especifican en la Tabla 1.2 [8] [7].

Se tienen los siguientes parámetros dependientes de la modulación y codificación:

- M : tamaño del diccionario de símbolos de modulación (BPSK, QPSK, 16QAM, 64QAM).
- m : número de bits por símbolo modulado (BPSK, QPSK, 16QAM, 64QAM). Viene dado por:

$$m = \log_2(M)$$

Ecuación 1.1: Número de bits por símbolo modulado

Tabla 1.2: Configuraciones permitidas para IEEE 802.11p [7][8]

Esquema de Modulación	Bits por símbolo modulado (m)	Tasa de codificación (r)	Tasa de datos (v_t) [Mbps]
BPSK	1	$1/2$	3
BPSK	1	$3/4$	4,5
QPSK	2	$1/2$	6
QPSK	2	$3/4$	9
16QAM	4	$1/2$	12
16QAM	4	$3/4$	18
64QAM	6	$2/3$	24
64QAM	6	$3/4$	27

- r : tasa de codificación del código convolucional utilizado (incluyendo puntura).

d.2. Aleatorización (*Scrambler*): Mediante el aleatorizador se consigue que no existan secuencias largas de 1s lógicos en los datos binarios. En el proceso de aleatorización se excluyen el campo *Tail* del DATA y la cabecera (campo SIGNAL). El aleatorizador que se explican a continuación es tomado del estándar IEEE 802.11 [8].

Los datos binarios pasan uno a uno a través de un aleatorizador cuyo polinomio generador es $S(x)$, que de acuerdo al estándar IEEE 802.11 [8] es:

$$S(x) = x^7 + x^4 + 1$$

Ecuación 1.2: Polinomio generador del aleatorizador

Para implementar el polinomio, el aleatorizador consiste de dos registros de desplazamiento de 7 posiciones en total que toman el bit de entrada, el bit en la cuarta posición y el bit en la séptima posición, realiza una operación XOR (Tabla 1.3) entre los tres bits de dos en dos y devuelve el resultado como la salida del aleatorizador. Una representación gráfica del aleatorizador se presenta en la Figura 1.11. Inicialmente el aleatorizador debe contener un

valor en cada uno de sus registros, a este valor inicial se le llama semilla, y para esta implementación se toma el valor de 1011101 de acuerdo a lo mostrado en la Tabla L-14 del estándar IEEE 802.11 [8].

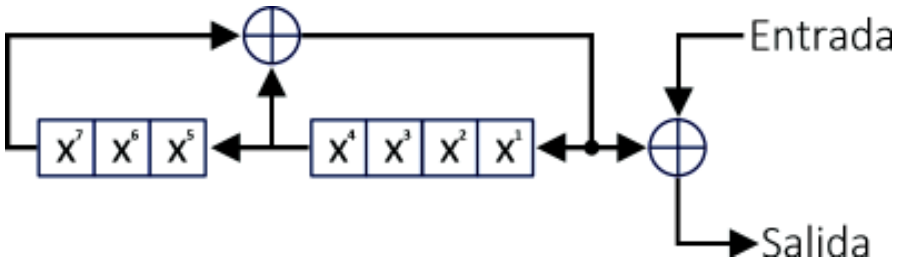


Figura 1.11: Aleatorizador [8]

Tabla 1.3: Tabla de verdad de operación XOR binaria

A	B	$A \oplus B$
1	1	0
1	0	1
0	1	1
0	0	0

d.3. Codificación convolucional: Un codificador convolucional es una máquina de estado finito que por cada k bits de entrada, entrega n bits de salida (con $n > k$). De esta manera, los bits de salida dependen no solamente del bit de entrada, sino que también de los K bits anteriores a la salida que se denominan la longitud del código, en donde K es igual a la memoria del codificador, que es el número de registros en serie que contiene más una unidad [19]. La tasa de codificación (r) viene dada por la expresión:

$$r = \frac{k}{n}$$

Ecuación 1.3: Tasa de codificación de código convolucional

La codificación convolucional brinda la capacidad para corrección de errores en recepción tan robusta como el inverso de la tasa de codificación, a cambio de introducir redundancia en los datos. El codificador convolucional y el proceso de codificación que se explican a continuación son tomados del estándar IEEE 802.11 [8].

Los datos binarios aleatorizados pasan uno a uno a través de un codificador convolucional con los siguientes polinomios generadores expresados de manera octal (base 8):

$$g_0 = 133_8$$
$$g_1 = 171_8$$

Ecuación 1.4: Polinomios generadores del codificador convolucional (base 8)

O bien, de manera binaria (base 2):

$$g_0 = 1011011$$
$$g_1 = 1111001$$

Ecuación 1.5: Polinomios generadores del codificador convolucional (base 2)

El comportamiento de un codificador convolucional se caracteriza por la afectación o no de cada bit de la longitud del código sobre cada bit de salida, en tanto que participen o no en una operación XOR entre todos ellos o no para dar como resultado un determinado bit de salida. Este comportamiento se expresa mediante los polinomios generadores.

Debido a que el comportamiento de los bits de la longitud del código puede afectar a los n bits de salida de manera independiente, se tiene un polinomio generador por cada bit de salida, es decir n polinomios generadores. Por otro lado, como K bits pueden afectar o no a un bit de salida, cada polinomio generador tiene una longitud de K. Luego, debido a que los polinomios generadores expresan si un bit de la longitud del código entra o no a la operación XOR que da como resultado cada bit de salida, cada elemento del polinomio generador puede ser un 0 o un 1 (un bit lógico).

Finalmente, como en general se tiene que pueden existir k entradas simultáneas, se debe tener para cada una de los n bits de salida k polinomios generadores de longitud K. Si se conocen los polinomios generadores de un codificador convolucional, el mismo queda caracterizado por completo [19].

Una representación gráfica de este codificador convolucional en particular viene dada por la Figura 1.12 [8].

Las tasas de codificación permitidas son las expuestas en la Tabla 1.2 [8]. Este codificador tiene una tasa de codificación por defecto de $r = 1/2$, sin embargo, para lograr mayores tasas de codificación con el mismo codificador convolucional, se puede utilizar el concepto

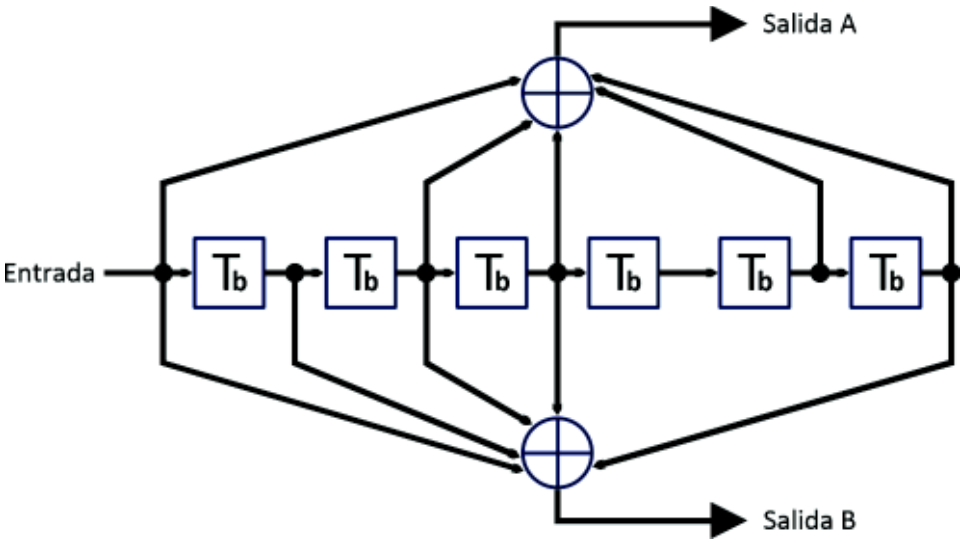


Figura 1.12: Codificador convolucional utilizado en IEEE 802.11p [8]

de **puntura** también conocido como perforación o ponchamiento. Este método consiste en omitir ciertos bits a la salida del codificador para la codificación, y para la decodificación ingresar bits 0 en estos sitios antes de realizar el proceso [19]. Para las tasas de $r = {}^{3}/_{4}$ y $r = {}^{2}/_{3}$ se utilizan los siguientes patrones de puntura respectivamente:

$$\begin{aligned} P_{3/4} &= [111001] \\ P_{2/3} &= [1110] \end{aligned}$$

Ecuación 1.6: Patrones de puntura para $r = {}^{3}/_{4}$ y $r = {}^{2}/_{3}$

Se muestra un ejemplo gráfico de puntura en la codificación en la Figura 1.13.

Para el proceso de decodificación se utiliza preferiblemente el algoritmo de Viterbi que se expone en la sección 11.1 de [19].

d.4. Entrelazado (*Interleaver*): El entrelazador ayuda a contrarrestar los efectos negativos del canal inalámbrico al permutar bits adyacentes, de tal manera que los errores aparezcan aislados en el receptor en lugar de aparecer en ráfagas [20]. El proceso de entrelazado que se explica a continuación es tomado del estándar IEEE 802.11 [8].

Los datos binarios codificados se pasan a través de un entrelazador, el cual consiste en dos permutaciones que se realizarán sobre bloques de longitud fija de datos, bloque a bloque de

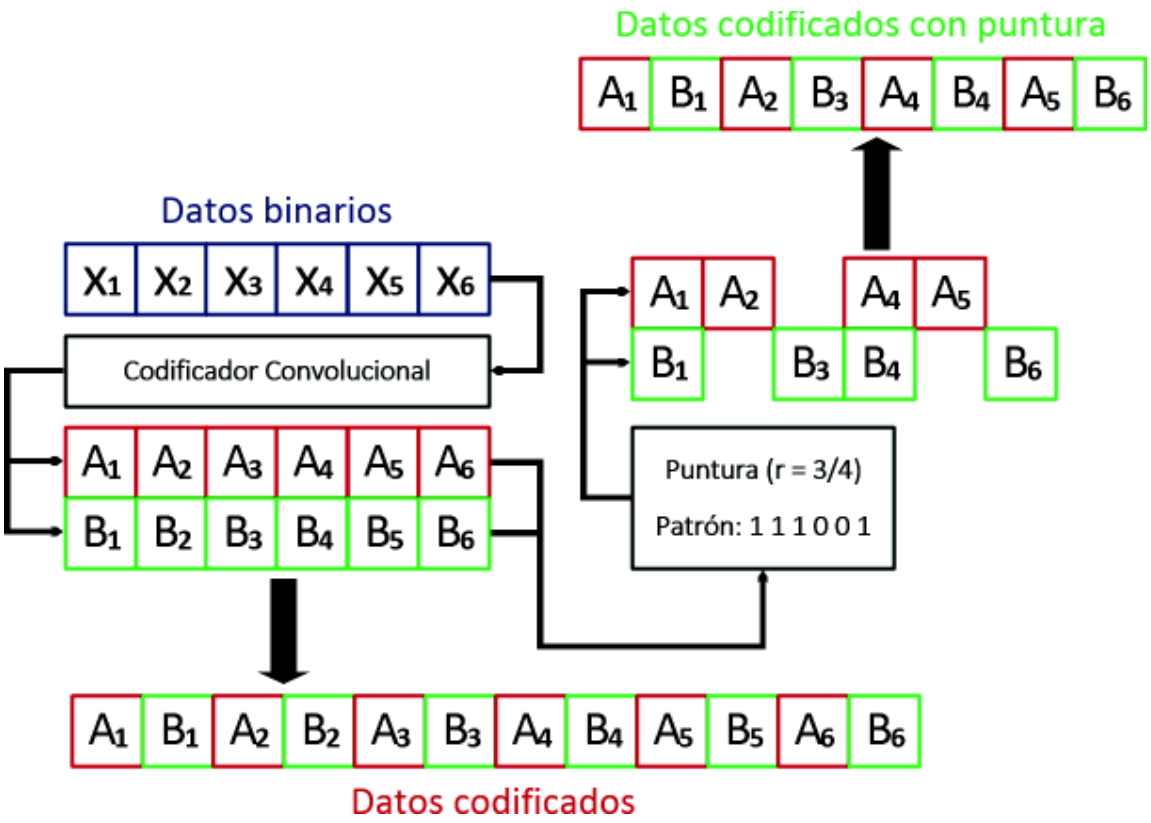


Figura 1.13: Puntura de un código convolucional

manera independiente. El tamaño de los bloques es igual al número de bits codificados por bloque OFDM (N_{CBPS}). La primera permutación asegura que los bits codificados adyacentes no sean colocados en subportadoras adyacentes, mientras que, la segunda permutación asegura que los bits codificados adyacentes sean modulados alternadamente en uno o más bits significativos en una constelación [8].

Las permutaciones se las caracteriza por los índices que indican su posición dentro del bloque, así se tiene que:

- k es el índice de cada elemento antes de la primera permutación.
- i es el índice de cada elemento después de la primera permutación y antes de la segunda.
- j es el índice de cada elemento después de la segunda permutación.

Entonces, se tiene que $k, i, j = 0, 1, \ldots, N_{CBPS} - 1$, como se ilustra en la Figura 1.14.

Las permutaciones se realizan mediante operaciones sobre los índices para determinar los siguientes índices en el proceso. Primero se toma el índice k de cada elemento del bloque y

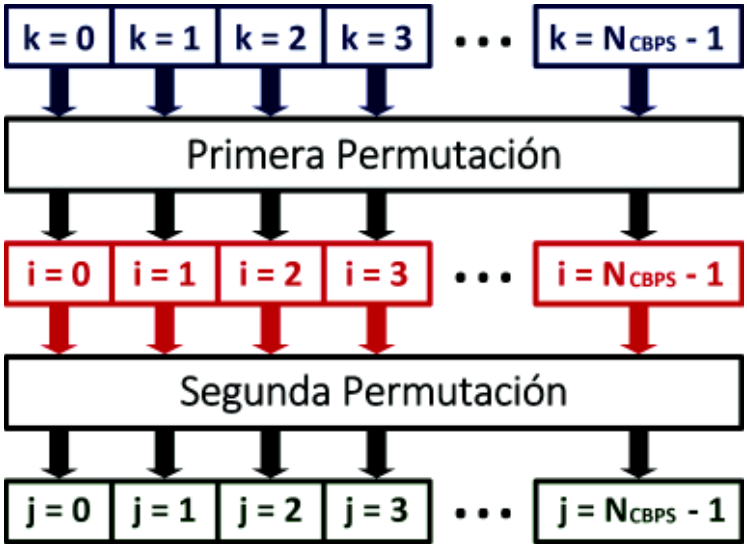


Figura 1.14: Índices en proceso de entrelazado [8]

al realizar la primera permutación se obtiene el índice correspondiente i, y luego se realiza la segunda permutación con los índices i para obtener los índices correspondientes j que indican la posición que deberán tomar los elementos dentro del bloque al final del proceso de entrelazado. La primera permutación viene dada por [8]:

$$i = \left(\frac{N_{CBPS}}{16}\right)(k \bmod 16) + \text{Floor}\left(\frac{k}{16}\right)$$

Ecuación 1.7: Primera permutación del entrelazador

en donde Floor($\cdot$) es la función piso o redondeo hacia abajo (toma el entero más próximo hacia abajo) y mod es la operación módulo [1].

La segunda permutación viene dada por [8]:

$$j = s\text{Floor}\left(\frac{i}{s}\right) + \left(i + N_{CBPS} - \text{Floor}\left(16\frac{i}{N_{CBPS}}\right)\right) \bmod s$$

Ecuación 1.8: Segunda permutación del entrelazador

en donde s viene dado por la Ecuación (1.9) y N_{BPSC} es el número de bits (codificados) por subportadora.

[1]Para detalles de la operación módulo y álgebra modular, revisar [21]

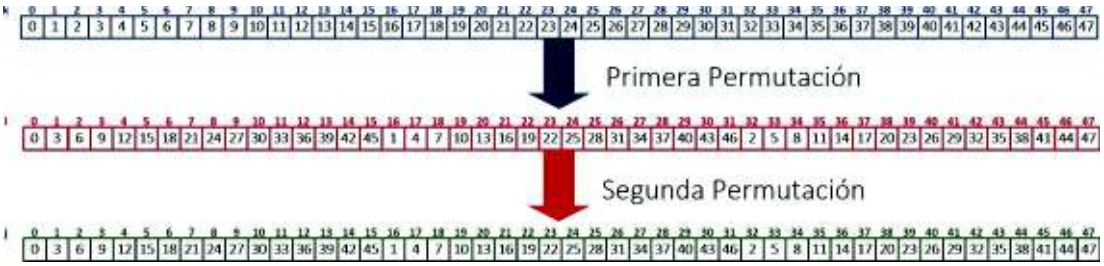


Figura 1.15: Ejemplo de entrelazado para $BPSK$

$$s = \max\left(\frac{N_{BPSC}}{2}, 1\right)$$

Ecuación 1.9: Parámetro s para el entrelazado

Un ejemplo del proceso de entrelazado se presenta en la Figura 1.15.

d.5. Modulación: Los datos binarios entrelazados se pasan en grupos de N_{BPSC} bits (con $N_{BPSC} = m$) a través de un modulador. El modulador toma grupos de m bits y, de acuerdo al esquema de modulación seleccionado, les asigna un solo valor en el dominio de la frecuencia para todo el grupo. Este valor es representado como un punto en el espacio complejo que forma parte del diagrama de constelaciones. Cada esquema de modulación tiene un alfabeto complejo de tamaño M (con $M = 2^m$) [22].

El modulador que se explica a continuación es tomado del estándar IEEE 802.11 [8].

Un resumen de los esquemas de modulación permitidos por el estándar IEEE 802.11p [8] [7] se presenta en la Tabla 1.4.

Tabla 1.4: Esquemas de modulación para IEEE 802.11p

Esquema	M **(tamaño del alfabeto)**	m **(bits por símbolo)**
BPSK	2	1
QPSK	4	2
16QAM	16	4
64QAM	64	6

La asignación de los símbolos por cada grupo de m bits se la realiza de acuerdo a los diagramas de constelación de la Figura 1.16.

Por cuestiones de normalización, la señal modulada deberá multiplicarse por un factor correspondiente a su esquema de modulación, como se especifica en la Tabla 1.5.

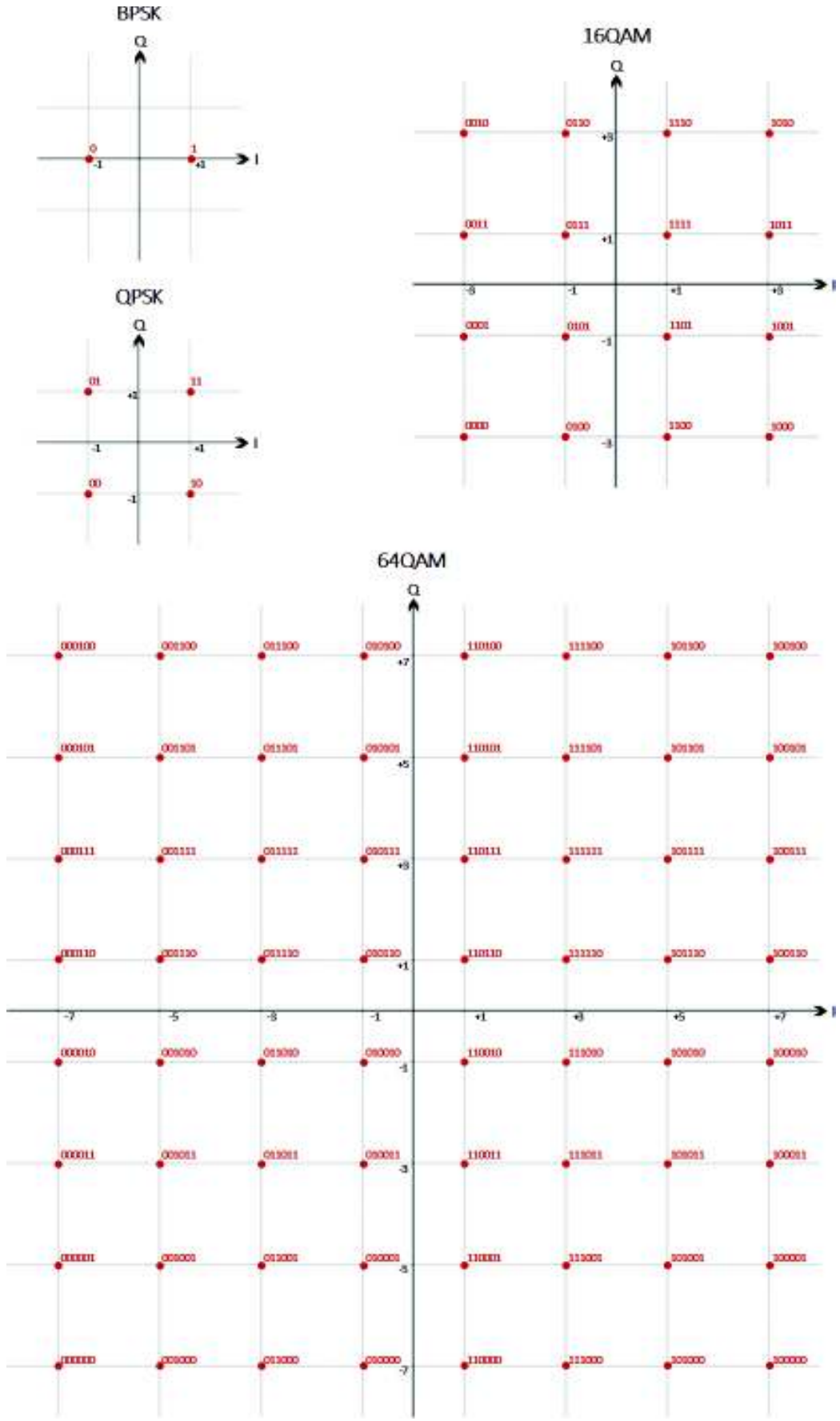


Figura 1.16: Diagramas de constelación para BPSK, QPSK, 16QAM y 64QAM [8]

Tabla 1.5: Factores de normalización para modulación en IEEE802.11p [8]

Esquema	Factor de normalización
BPSK	1
QPSK	$1/\sqrt{2}$
16QAM	$1/\sqrt{10}$
64QAM	$1/\sqrt{42}$

d.6. Transmisión OFDM (*Orthogonal Frequency Division Multiplexing*): La capa física del estándar IEEE 802.11p se fundamenta principalmente en la modulación OFDM. Para entender el proceso de modulación OFDM se inicia con un breve fundamento teórico, seguido por la transformada inversa de Fourier del símbolo OFDM, la especificación de la construcción del símbolo y finalmente el prefijo cíclico.

d.6.1. Fundamento de OFDM: OFDM es un método de transmisión y recepción de datos basado en multiplexación por frecuencia, esto quiere decir que, en OFDM se divide el ancho de banda total (AB) en N porciones pequeñas equiespaciadas una cantidad Δf (con $\Delta f = AB/N$) llamadas subportadoras, por las cuales se envía información de manera paralela e independiente unas de otras. La diferencia de OFDM con respecto a multiplexación por división de frecuencia (*Frequency Division Multiplex*, FDM) convencional, es que las porciones en las que se divide el ancho de banda se sobreponen, pudiendo incrementar la eficiencia espectral de la transmisión. Esta superposición es posible debido a la ortogonalidad entre las subportadoras [22].

La ortogonalidad de subportadoras se basa en la idea de ubicar las subportadoras una a una a lo largo del rango de frecuencias total disponible, de tal manera que cada subportadora se ubica exactamente en el primer cero en frecuencia de la subportadora anterior (como se presenta en la Figura 1.17) [22].

d.6.2. Transformada inversa de Fourier: Para implementar de forma eficiente la modulación OFDM se utiliza la IFFT/FFT (tiempo/frecuencia) [22].

La transformada inversa de Fourier lleva a los datos del dominio de la frecuencia al dominio del tiempo. De manera general la trasformada inversa de Fourier de una función en frecuencia $\hat{f}(\omega)$ viene dada por la Ecuación (1.10) [23].

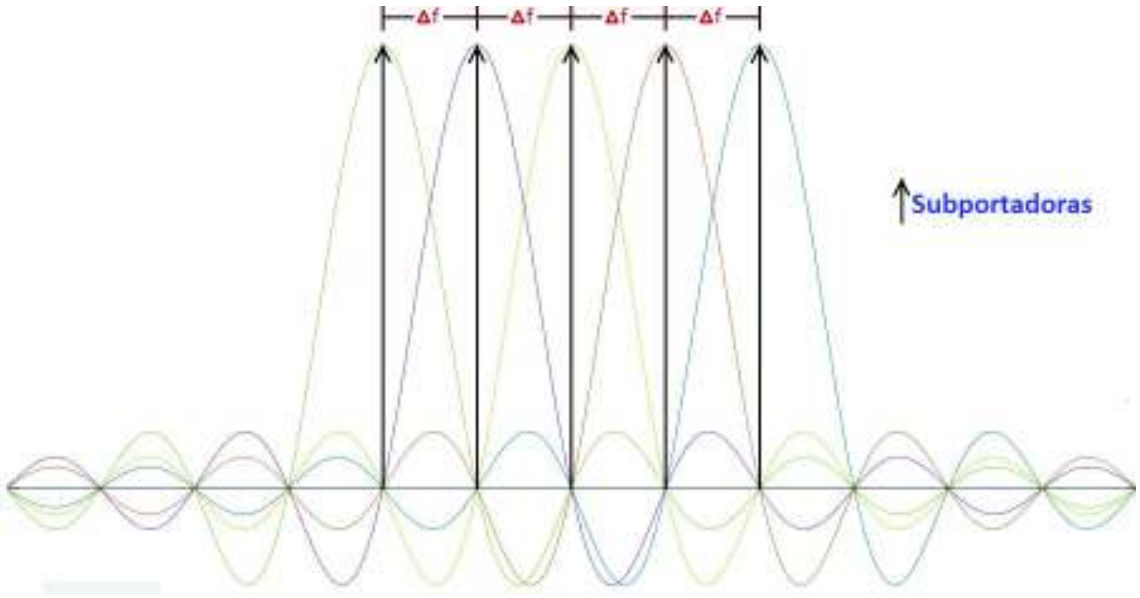


Figura 1.17: Ortogonalidad OFDM

$$\mathcal{F}^{-1}\{\hat{f}(\omega)\} = f(t) = \int_{-\infty}^{\infty} \hat{f}(\omega) e^{2\pi i \omega t}\, d\omega$$

Ecuación 1.10: Transformada inversa de Fourier

Esta señal en el dominio del tiempo ya puede ser enviada a través del canal inalámbrico para su propagación. Sin embargo esta caracterización es válida para una función de datos continuos. Para datos discretos, no obstante, se utiliza la transformada inversa de Fourier discreta [23] que toma las muestras en frecuencia X_k y las lleva a muestras en tiempo x_n. Cada muestra en tiempo de manera general se la obtiene con la siguiente fórmula [23]:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N} kn} \qquad n = 0, \ldots, N-1$$

Ecuación 1.11: Transformada inversa de Fourier discreta

En donde N es el número de muestras en tiempo y en frecuencia, k y n son los índices en frecuencia y en tiempo respectivamente.

Con el uso de un circuito integrado o una función en algún lenguaje de programación (eg. MATLAB) es posible la implementación de una transformada inversa rápida de Fourier discreta (*Inverse Fast Fourier Transform*, IFFT) que cuenta con 64 entradas para las muestras en frecuencia numeradas de 0 hasta 63 y 64 salidas en tiempo numeradas de igual manera.

Se asignarán las subportadoras a las entradas del IFFT tal como indica la Figura 1.18 o la Tabla 1.6.

Tabla 1.6: Asignación de subportadoras a la IFFT [8]

Subportadora	**Entrada** $IFFT$
N/A	0
1	1
2	2
3	3
...	...
25	25
26	26
N/A	27
N/A	...
N/A	37
-26	38
-25	39
...	...
-3	61
-2	62
-1	63

d.6.3. Parámetros OFDM en el estándar: Se tienen los siguientes parámetros dependientes de la modulación OFDM:

- N: número de subportadoras totales utilizadas contando las de guarda y la DC. Este número de subportadoras es necesario para el uso del algoritmo IFFT. Es una constante igual a 64 ($N = 64$).
- N_{ST}: número de subportadoras totales utilizadas sin contar las de guarda y la DC. Es una constante igual a 52 ($N_{ST} = 52$).
- N_{SD}: número de subportadoras de datos utilizadas. Es una constante igual a 48 ($N_{SD} = 48$).
- N_{SP}: número de subportadoras piloto utilizadas. Es una constante igual a 4 ($N_{SP} =$

4). Se tiene:

$$N_{ST} = N_{SD} + N_{SP} = 48 + 4 = 52$$

Ecuación 1.12: Número de subportadoras totales

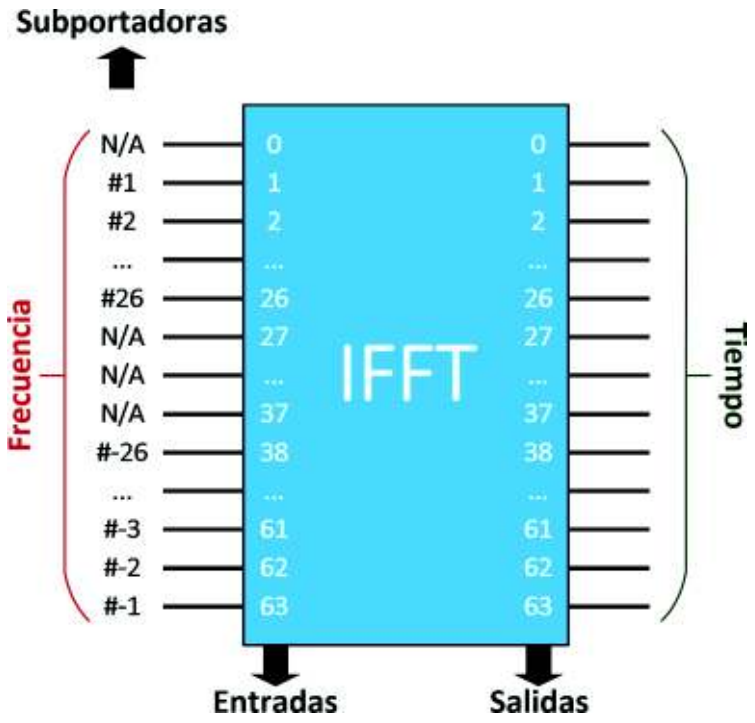


Figura 1.18: Asignación de subportadoras a la IFFT [8]

- AB: ancho de banda utilizado para la transmisión de un símbolo OFDM. Se tiene que AB = 10 MHz.
- Δf: espacio en frecuencia entre subportadoras. Se obtiene al dividir el ancho de banda disponible para el número total de subportadoras (incluyendo subportadoras de guarda y DC). Se tiene:

$$\Delta f = \frac{AB}{N} = \frac{10\ MHz}{64} = 0.15625\ MHz$$

Ecuación 1.13: Espacio entre subportadoras

- T_{IFFT}: periodo de un símbolo OFDM en tiempo (sin tener en cuenta el prefijo cíclico). Se tiene:

$$T_{IFFT} = \frac{1}{\Delta f} = \frac{1}{0.15625\ MHz} = 6.4\ \mu s$$

Ecuación 1.14: Periodo de un símbolo OFDM sin prefijo cíclico

- T_{GI}: periodo del intervalo de guarda, el cual ocupa el prefijo cíclico de un símbolo OFDM en tiempo. Se tiene:

$$T_{GI} = \frac{T_{IFFT}}{4} = \frac{6.4\ \mu s}{4} = 1.6\ \mu s$$

Ecuación 1.15: Periodo de un intervalo de guarda

- T_{SYM}: periodo de un símbolo OFDM en tiempo incluyendo el prefijo cíclico. Se tiene:

$$T_{SYM} = T_{IFFT} + T_{GI} = 6.4\ \mu s + 1.6\ \mu s = 8\ \mu s$$

Ecuación 1.16: Periodo de un símbolo OFDM con prefijo cíclico

- N_{BPSC}: número de bits por subportadora (*Number of Bits Per Sub-Carrier*). Viene dado por:

$$N_{BPSC} = m$$

Ecuación 1.17: Número de bits por subportadora

- N_{CBPS}: número de bits codificados (después de codificación convolucional) por símbolo OFDM (*Number of Coded Bits Per Symbol*). Viene dado por:

$$N_{CBPS} = N_{BPSC} * N_{SD}$$

Ecuación 1.18: Número de bits codificados por símbolo OFDM

- N_{DBPS}: número de bits de datos (antes de codificación convolucional) por símbolo OFDM (*Number of Data Bits Per Symbol*). Viene dado por:

$$N_{DBPS} = N_{CBPS} * r$$

Ecuación 1.19: Número de bits de datos (antes de codificación convolucional) por símbolo OFDM

d.6.4. Construcción del símbolo OFDM: Los datos modulados de acuerdo al diagrama de constelaciones correspondiente se separan en grupos de N_{SD} símbolos modulados y con cada grupo se forman símbolos OFDM. El proceso de formación de un símbolo OFDM que se explica a continuación es tomado del estándar IEEE 802.11 [8].

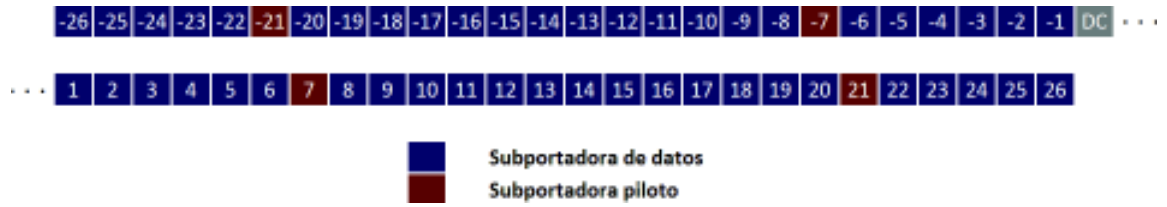


Figura 1.19: Distribución de las subportadoras OFDM [8]

Un símbolo OFDM está compuesto por N_{ST} subportadoras (además de 1 subportadora 0 o DC), de estas, N_{SD} subportadoras corresponden a datos y N_{SP} son subportadoras piloto como se indica en la Figura 1.19. Las subportadoras son numeradas $-26, -25, -24, ..., -2, -1, 0, 1, 2, ..., 24, 25, 26$

Se ubican los datos modulados uno en cada subportadora comenzando en la -26 hasta la 26 saltándose la subportadora 0, en la cual se ubica un valor de 0 que corresponde a la DC y las subportadoras $-21, -7, 7, 21$ las cuales corresponden a las subportadoras piloto. En estas posiciones se ubicarán los valores $1, 1, 1, -1$ multiplicados por un factor de polaridad p_n en donde n es el índice del n-ésimo símbolo OFDM formado. Existen 127 valores de p_n que vienen dados por: $p_{0..,126} = \{1, 1, 1, 1, -1, -1, -1, 1, -1, -1, -1, -1, 1, 1, -1, 1, -1, -1, 1, 1, -1, 1, 1, -1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1, -1, 1, 1, -1, -1, 1, 1, 1, -1, 1, -1, -1, -1, 1, -1, 1, -1, -1, 1, -1, -1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1, -1, 1, -1, 1, -1, 1, 1, -1, -1, -1, 1, 1, -1, -1, -1, -1, 1, -1, -1, 1, -1, 1, 1, 1, 1, -1, 1, -1, 1, -1, 1, -1, -1, -1, -1, -1, 1, -1, 1, 1, -1, 1, -1, 1, 1, 1, -1, -1, 1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1\}$. Para símbolos más allá del 127avo, se realiza la operación $n \mod 127$ para determinar el índice de p_n.

Esta secuencia es generada al pasar una secuencia de todos unos a través del aleatorizador en la Figura 1.11 y cambiando los números 0 por 1 y los números 1 por -1.

d.6.5. Prefijo Cíclico: Debido al multitrayecto, es posible que en recepción lleguen dos señales desfasadas en tiempo, es decir, una con un retardo con respecto a la otra. Esto puede ocasionar interferencias destructivas en algunas subportadoras, lo cual deteriora la señal recibida.

Para combatir efectos de multitrayecto, al símbolo OFDM en tiempo se le anexa una porción del mismo al principio de este. La porción que se toma es la última cuarta parte del símbolo

en tiempo [24]. En el caso discreto de 64 muestras, esto se traduce en tomar las últimas 16 muestras y copiarlas antes de la primera, obteniendo finalmente un arreglo de 80 muestras.

Figura 1.20: Proceso de preparación del prefijo cíclico [8]

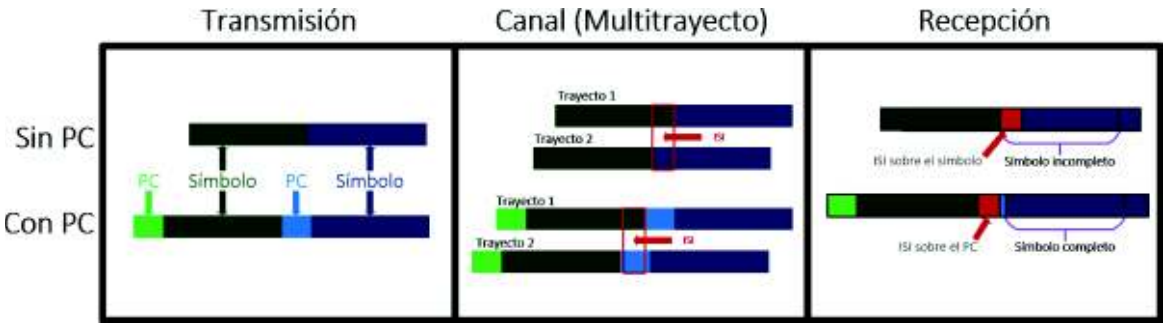


Figura 1.21: Recepción de señales desfasadas con prefijo cíclico en tiempo.

La Figura 1.20 ilustra este proceso [24]. Esta señal que consiste de 80 muestras en tiempo pasa al canal inalámbrico para su propagación.

Al añadir el prefijo cíclico a la señal, se tiene un tiempo de guarda en el cual se tolera esta interferencia entre señales desfasadas [24] (La Figura 1.21 muestra una comparación entre paquetes consecutivos transmitidos con y sin prefijo cíclico, y cómo éste permite la recuperación de los mismos, aún con efectos de ISI (*Inter-Symbol Interference*).

Además, ya que el prefijo cíclico es una extensión de la misma señal, toda esta es periódica y por lo tanto es posible recuperar la señal original en frecuencia a través de un FFT sin importar cuales muestras del símbolo completo son las que se toman [24].

1.3.4. Canal Inalámbrico

Las señales digitales generadas en capa física son transformadas en señales analógicas y emitidas al espacio libre con la ayuda de una antena. La propagación de la señal en el aire

se realiza en forma de ondas electromagnéticas. Estas son captadas por un receptor que también cuenta con una antena que trabaje en la misma frecuencia que el transmisor [22].

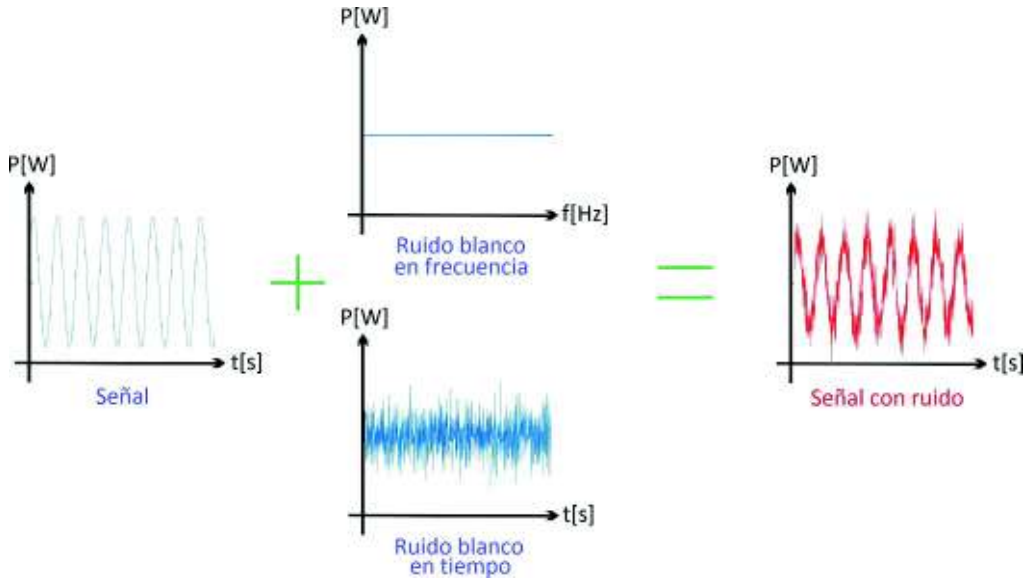


Figura 1.22: Canal AWGN

El medio de propagación (canal inalámbrico) es el aire y debido a su naturaleza puede presentar varios fenómenos físicos que afecten a la señal electromagnética como son: el ruido, el desvanecimiento, el multitrayecto, el efecto *doppler* o la interferencia [22].

Existen diferentes modelos que ayudan a describir un canal inalámbrico, algunos de ellos incluyen los efectos de desvanecimiento. El modelo más básico es el canal AWGN. Un canal AWGN es aquel que añade ruido blanco a la señal, este canal tiene una respuesta plana en frecuencia, es decir, el ruido tiene una misma potencia para todas las frecuencias del espectro, es aditivo ya que se suma a la señal en amplitud y es gaussiano porque presenta una distribución normal en tiempo con un valor medio de cero (ver Figura 1.22) [25].

1.3.5. Parámetros de rendimiento de un sistema inalámbrico

Para determinar la calidad de una comunicación es necesario tener parámetros objetivos que puedan ser medidos y expresen de una manera cuantificable aspectos relacionados con la calidad y el rendimiento de la comunicación.

La calidad de una comunicación inalámbrica se ve reflejada, por ejemplo, en la integridad de los mensajes enviados o en el tiempo en que demora en llegar un mensaje del transmisor al receptor. La integridad de un mensaje se refiere a, en qué medida el mensaje interpretado en recepción corresponde al generado por el transmisor, el cual puede ser comprometido

por factores externos como interferencias o ataques intencionados de terceros; por factores inherentes del canal de comunicación como el ruido térmico o blanco, las pérdidas por propagación, el multitrayecto; o bien por factores propios del sistema de comunicación [25].

Una medida de la integridad de un mensaje consiste en comparar el obtenido en recepción con el generado en transmisión. Se busca realizar esta comparación a la salida de la capa física, así que se debe realizar comparaciones a nivel de bit.

Esta comparación contempla varios aspectos, por ejemplo si se desea realizar una comparación a nivel de los bits recibidos con respecto a los transmitidos se usa la tasa de error de bit (*Bit Error Rate*, BER). Si la comparación se realiza a nivel de trama o paquete en capa física recibida con respecto a una enviada se emplea la tasa de error de paquetes (*Packet Error Rate*, PER). En cambio si se desea hacer una comprobación a nivel de los símbolos (de modulación) recibidos con respecto a los enviados se puede usar la tasa de símbolos errados (*Symbol Error Rate*, SER) [3] [22] [6].

A continuación se revisan brevemente estos parámetros.

a. BER

La BER mide la cantidad de bits errados con respecto al total de recibidos, es decir, es la relación entre bits errados ($b_{errados}$) y los bits totales recibidos ($b_{totales}$). Esta relación puede ser expresada de forma adimensional (veces) mediante la Ecuación (1.20) o en decibelios (dB) mediante la Ecuación (1.21) [26].

$$BER = \frac{b_{errados}}{b_{totales}}$$

Ecuación 1.20: BER

$$BER[dB] = 10\log_{10}\left(\frac{b_{errados}}{b_{totales}}\right)$$

Ecuación 1.21: BER en dB

La BER da una idea de la integridad de los bits recibidos, al saber la proporción del mensaje que se ha recibido correctamente, y la proporción que no es confiable. Un sistema de comunicaciones es diseñado tomando en cuenta la minimización de la BER esperada.

b. PER

Cuando la unidad de comparación es un paquete de bits, entonces un mejor parámetro para medir la confiabilidad de las transmisiones es la PER, es decir, la PER mide en recepción la

relación entre paquetes (o tramas) erradas con respecto a los paquetes totales recibidos como se indica en la Ecuación (1.22) [6].

$$PER = \frac{p_{errados}}{p_{totales}}$$

Ecuación 1.22: PER

en donde $p_{errados}$ es el número de paquetes errados y $p_{totales}$ es el número total de paquetes recibidos. Esta relación también puede ser expresada en decibelios (dB) como se indica en la Ecuación (1.23).

$$PER[dB] = 10 \log_{10} \left(\frac{p_{errados}}{p_{totales}} \right)$$

Ecuación 1.23: PER en dB

Un paquete se considera errado cuando uno o más de los bits que lo conforman tiene error [5]. También cabe mencionar que cuando se habla de paquetes en este contexto, se refiere a paquetes a nivel de capa física.

1.3.6. Modelos para el cálculo de la PER

La PER resultante de una comunicación es un valor empírico que se obtiene mediante la comparación de los paquetes enviados y los recibidos. Es posible realizar experimentos sobre ciertas condiciones para cierto tipo de comunicación y obtener valores reales de la PER, sin embargo esto implica que el sistema de comunicaciones se encuentra ya diseñado y construido.

Una alternativa para la obtención de la PER es mediante el uso de simulaciones que reflejen lo suficientemente bien el sistema de comunicaciones a analizar.

Otra alternativa para su cálculo es el uso de modelos teóricos. Estos son modelos matemáticos paramétricos que permiten la obtención de la PER en función de las configuraciones específicas de la comunicación.

Sea cual sea la alternativa para el cálculo de la PER, es necesario validarla, así, a nivel de simulaciones se tienen que validar con mediciones reales; a nivel de modelos teóricos se puede validar tanto con mediciones reales como con resultados de simulaciones.

Existen varios modelos teóricos para la PER, cada uno destinado para cierto tipo de sistemas de comunicaciones con ciertos elementos o parámetros. Para la determinación del modelo

que más se ajuste a las redes VANET es deseable realizar una comparación de los modelos teóricos existentes y determinar cuál de ellos presenta resultados más similares a los de una simulación lo más real posible. Para esto, en primera instancia, se revisa la literatura sobre los modelos teóricos predictivos para el cálculo de la PER.

a. Estado del arte

Se ha realizado una revisión de la literatura existente relacionada con cálculo de la PER. A continuación se presenta un resumen de cada modelo encontrado y al final se justifican los modelos que se escogen para su implementación en este trabajo de titulación.

Al revisar estos modelos se enfatiza en los parámetros que considera cada modelo y en el tipo de comunicación al que se aplica.

b. Modelo para redes vehiculares por interpolación

Abrate et al. en su trabajo dado en [1] proponen un modelo analítico para el cálculo de la PER, el cual contempla los distintos esquemas de modulación y codificación convolucional aceptados en el estándar IEEE 802.11p. El modelo es obtenido mediante la interpolación de los datos de la PER tomados de una simulación realista[2]. Esta simulación es de la capa física OFDM IEEE 802.11a en un canal AWGN con multitrayecto realizada en *simulink* (MATLAB).

Este modelo matemático se describe en función de un valor de SNR y de la longitud de los paquetes; y da como resultado la PER esperada. La función está dada por la Ecuación (1.24) [1].

$$PER(\gamma, L) = \frac{1 - \tanh\left(a_R(L) - b_R(L)(\gamma + c)\right)}{2}$$

Ecuación 1.24: PER teórica de acuerdo a [1]

en donde γ es la SNR en dB, L es la longitud del paquete en bits, $a_R(\cdot)$ y $b_R(\cdot)$ son parámetros obtenidos en función del esquema de modulación y tasa de codificación, los cuales son descritos en las ecuaciones (1.25) y (1.26) respectivamente y la constante c es un parámetro de ajuste debido a la precisión de la simulación, que los autores ubicaron en 10 dB [1].

$$a_R(L) = c_1 e^{d_1 L} + c_2 e^{d_2 L}$$

Ecuación 1.25: Parámetro a_R de [1]

[2] http://www.mathworks.com/matlabcentral/fileexchange/3540-ieee-802-11a-wlan-model

$$b_R(L) = c_3 e^{d_3 L} + c_4 e^{d_4 L}$$

Ecuación 1.26: Parámetro b_R de [1]

las constantes c_i y d_i ($i = 1, 2, 3, 4$) dependen de la modulación y codificación utilizadas. Sabiendo que la modulación y la codificación determinan la velocidad de transmisión, los valores de estas constantes se encuentran en la Tabla 1.7.

Tabla 1.7: Tabla de constantes para ecuación del PER [1]

v_{Tx}	c_1	c_2	c_3	c_4	d_1	d_2	d_3	d_4
3 *Mbps*	-13.64	8.939	-1.3	0.6895	1.286e-4	-4.565e-3	1.003e-4	-4.927e-3
4.5 *Mbps*	-15.29	9.228	-1.103	0.5428	1.216e-4	-4.469e-3	9.348e-5	-4.772e-3
6 *Mbps*	-19.85	10.31	-1.313	0.5674	1.027e-4	-4.712e-3	8.23e-5	-5.05e-3
9 *Mbps*	-22.12	10.93	-1.174	0.4898	9.079e-5	-3.987e-3	7.308e-5	-4.162e-3
12 *Mbps*	-25.97	12.83	-1.258	0.5459	1.124e-4	-6.063e-3	8.505e-5	-6.508e-3
18 *Mbps*	-29.14	12.89	-1.164	0.4501	1.017e-4	-4.607e-3	8.58e-5	-4.811e-3
24 *Mbps*	-39.27	18.17	-1.354	0.5715	8.056e-5	-4.3134e-3	6.918e-5	-4.409e-3
27 *Mbps*	-37.24	16.17	-1.219	0.4811	8.734e-5	-5.424e-3	7.552e-5	-5.779e-3

c. Modelo sin codificación sobre canal AWGN

Mahmood y Janti en su trabajo dado en [2] describen un modelo general para la obtención de la PER sin considerar esquemas con codificación sobre canales AWGN, y desarrollan un modelo aproximado considerando modulación, usando la teoría de valor extremo (*Extreme Value Theory*, EVT) [2].

El modelo general para la obtención de la PER sobre canales AWGN bajo un esquema sin codificación viene dado por la siguiente ecuación:

$$PER(\gamma) = 1 - (1 - b_e(\gamma))^L$$

Ecuación 1.27: PER teórica clásico mostrado en [2]

en donde γ representa el SNR, $b_e(\gamma)$ la BER de cada esquema de modulación utilizado en IEEE 802.11p (BPSQ, QPSK, 16QAM, 64QAM) y L el tamaño del paquete en bits. El BER se obtiene a partir de la Ecuación (1.28).

$$b_e(\gamma) = c_m Q\left(\sqrt{k_m \gamma}\right)$$

Ecuación 1.28: BER teórica de acuerdo a [2]

en donde c_m y k_m son constantes dependientes del esquema de modulación y $Q(\cdot)$ es la función Q definida por la Ecuación (1.29) [3]

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} \exp\left(-\frac{u^2}{2}\right) du$$

Ecuación 1.29: Función Q

El valor de c_m y k_m se obtiene a partir de (1.30) para cada tipo de modulación [3]. Cabe indicar que se toma el 1º elemento de la sumatoria correspondiente para poder aislar las constantes c_m y k_m.

$$BPSK: \; b_e\left(\frac{E_b}{N_0}\right) = Q\left(\sqrt{2\frac{E_b}{N_0}}\right)$$

$$QPSK: \; b_e\left(\frac{E_b}{N_0}\right) = Q\left(\sqrt{2\frac{E_b}{N_0}}\right)$$

$$16QAM: \; b_e\left(\frac{E_b}{N_0}\right) = \frac{3}{4}Q\left(\sqrt{\frac{4}{5}\frac{E_b}{N_0}}\right)$$

$$64QAM: \; b_e\left(\frac{E_b}{N_0}\right) = \frac{7}{12}Q\left(\sqrt{\frac{2}{7}\frac{E_b}{N_0}}\right)$$

Ecuación 1.30: BER por esquema de modulación de acuerdo a [3] en función del E_b/N_0

El parámetro E_b/N_0 es la relación de la energía de bit por densidad espectral de ruido [27]. Como el parámetro de interés es el SNR, se lo relaciona con el E_b/N_0 mediante la fórmula (1.31) [4].

$$\frac{E_b}{N_0} = \gamma \frac{AB}{v_t}$$

Ecuación 1.31: Relación entre E_b/N_0 y SNR

donde AB es el ancho de banda y v_t es la velocidad de transmisión. Sabiendo que el ancho de banda para comunicaciones IEEE 802.11p es de 10 MHz, se tiene entonces que:

$$BPSK:\ b_e(\gamma) = Q\left(\sqrt{2\left(\frac{10}{v_t}\right)\gamma}\right)$$
$$QPSK:\ b_e(\gamma) = Q\left(\sqrt{2\left(\frac{10}{v_t}\right)\gamma}\right)$$
$$16QAM:\ b_e(\gamma) = \frac{3}{4}Q\left(\sqrt{\frac{4}{5}\left(\frac{10}{v_t}\right)\gamma}\right)$$
$$64QAM:\ b_e(\gamma) = \frac{7}{12}Q\left(\sqrt{\frac{2}{7}\left(\frac{10}{v_t}\right)\gamma}\right)$$

Ecuación 1.32: BER por esquema de modulación de acuerdo a [3] en función del SNR

siempre que v_t esté en Mbps, entonces los valores de las constantes c_m y k_m vienen dados por los valores expuestos en la Tabla 1.8.

Tabla 1.8: Valores de las constantes c_m y k_m para cada esquema de modulación [2]

Modulación	c_m	k_m
BPSK	1	$2\ (10/v_t)$
QPSK	1	$2\ (10/v_t)$
16QAM	$3/4$	$(4/5)\ (10/v_t)$
64QAM	$7/12$	$(2/7)\ (10/v_t)$

La Ecuación (1.27) junto a las ecuaciones (1.32) representan el modelo general para el cálculo de la PER bajo un esquema sin codificación sobre un canal AWGN. Las constantes en la Tabla 1.8 serán utilizadas para el modelo aproximado obtenido mediante EVT que se revisará a continuación.

d. Modelo sin codificación sobre canal AWGN mediante EVT

Continuando con el modelo anterior, mediante una manipulación matemática utilizando EVT en [2] se llega a un modelo aproximado para el cálculo de la PER dado por la siguiente ecuación [2]:

$$PER(\gamma) \approx 1 - \exp\left(-\exp\left(-\frac{\gamma - a_N}{b_N}\right)\right)$$

Ecuación 1.33: PER teórica mediante EVT de acuerdo a [2]

en donde γ es el SNR y a_N y b_N son constantes dependientes del esquema de modulación, cuyos valores vienen dados por las ecuaciones (1.34) y (1.35) respectivamente [2].

$$a_N = \frac{2}{k_m}\left(erf^{-1}\left(1-\frac{2}{Lc_m}\right)\right)^2$$

Ecuación 1.34: Parámetro a_N de [2]

$$b_N = \frac{2}{k_m}\left(erf^{-1}\left(1-\frac{2}{Lc_m e}\right)\right)^2 - a_N$$

Ecuación 1.35: Parámetro b_N de [2]

En donde e es el número de Euler ($e \approx 2{,}71828$), L es el tamaño del paquete en bits, las constantes c_m y k_m se toman de la Tabla 1.8 y $\mathrm{erf}^{-1}(\cdot)$ es la función de error inversa dada por la Ecuación (1.36) [28]:

$$\mathrm{erf}^{-1}(x) = \sum_{k=0}^{\infty} \frac{c_k}{2k+1}\left(\frac{\sqrt{\pi}}{2}x\right)^{2k+1}$$

Ecuación 1.36: Función de error inversa

donde las constantes c_k se obtienen por las siguientes ecuaciones [28]:

$$c_0 = 1$$
$$c_k = \sum_{m=0}^{k-1} \frac{c_m c_{k-1-m}}{(m+1)(2m+1)}$$

Ecuación 1.37: Parámetros de la función de error inversa

La Ecuación (1.33) representa una aproximación de las ecuaciones (1.32) y (1.28) y por lo tanto aplica al mismo ámbito (esquema sin codificación, canal AWGN).

Los autores de [2], además proponen una aproximación de la PER promedio con respecto a la SNR que parte del anterior análisis, para canales con *fading* tipo Nakagami-m. Esta PER promedio viene dada por la Ecuación (1.38).

$$\overline{PER}(\bar{\gamma}) \approx 1 - \frac{m^m(-1)^{m-1}}{\bar{\gamma}^m\Gamma(m)}\frac{d^{m-1}}{ds^{m-1}}\left(\frac{e^{-sa_N}\Gamma(1+sb_N)}{s}\right)$$

Ecuación 1.38: PER teórica para canales con *fading* de acuerdo a [2]

En donde $\Gamma(\cdot)$ es la función Gamma definida como:

$$\Gamma(z) = \int_0^\infty x^{z-1}e^{-x}dx$$

Ecuación 1.39: Función Gamma

que es una generalización del factorial [29], y $s = {}^{m}/_{\bar{\gamma}}$ [2].

e. Modelo de cota superior para sistema OFDM

Song y Choi en su trabajo dado en [4] describen un modelo para el cálculo de una cota superior de la PER específicamente en sistemas IEEE 802.11p, sobre canales AWGN y Rayleigh realizando consideraciones más precisas con respecto a la tasas de codificación, la distancia libre (d_{free}) y el número de caminos a distancias dadas (a_d) del diagrama de *Trellis* del codificador convolucional con los patrones de puntura correspondientes.

Toma también en cuenta un valor de SNR por símbolo OFDM considerando la aportación del prefijo cíclico, aproximando expresiones para la BER con la función $Q(\cdot)$ descrita en la Ecuación (1.29). La PER por configuración o modo (Tabla 1.2) se encuentra parametrizada en última instancia con respecto a la SNR y a la longitud del paquete (L).

Para transformar la ${}^{E_b}/_{N_0}$ a SNR se extiende la idea expuesta en la Ecuación (1.31) considerando además la modulación y la porción del símbolo OFDM correspondiente al prefijo cíclico. La SNR por símbolo OFDM denotado por γ_s para el 80 % del símbolo (correspondiente a la porción del símbolo sin el prefijo cíclico) viene definido por la Ecuación (1.40):

$$\gamma_s = 0.8\log_2(M)\frac{AB}{v_t}\gamma = \frac{8m}{v_t[Mbps]}\gamma$$

Ecuación 1.40: Relación entre ${}^{E_b}/_{N_0}$ y SNR para símbolos OFDM

siendo γ la SNR, v_t la velocidad de transmisión en Mbps y m el número de bits por símbolo de modulación (BPSK, QPSK, 16QAM, 64QAM).

El artículo presenta las siguientes aproximaciones de la BER (b_e) para cada tipo de modulación, considerando la SNR por símbolo OFDM (γ_s).

$$BPSK:\ b_e^{BPSK}(\gamma_s) = Q\left(\sqrt{2\gamma_s}\right)$$
$$QPSK:\ b_e^{QPSK}(\gamma_s) = Q\left(\sqrt{\gamma_s}\right)$$
$$16QAM:\ b_e^{16QAM}(\gamma_s) = \frac{3}{4}Q\left(\sqrt{\frac{1}{5}\gamma_s}\right)$$
$$64QAM:\ b_e^{64QAM}(\gamma_s) = \frac{7}{12}Q\left(\sqrt{\frac{1}{21}\gamma_s}\right)$$

Ecuación 1.41: BER por esquema de modulación de acuerdo a [4]

Finalmente, se llega a la Ecuación (1.43) que define la cota de la PER por cada modo. Con "conf", se refiere a la combinación de esquema de modulación y tasa de codificación permitidas en el estándar. El esquema de modulación se considera con el parámetro P_d^{mod} cuyo valor se obtiene a partir de (1.44). La tasa de codificación se considera al elegir los d_{free} correspondientes, de acuerdo a la tasa de codificación deseada, expuestos en la Tabla 1.9. En esta misma tabla también se encuentran los primero 10 valores de a_d para valores de $d = d_{free}, d_{free+1}, ..., d_{free+9}$.

$$PER_{conf} \leq 1 - \left(1 - \sum_{d=d_{free}}^{\infty} a_d P_d^{mod}\right)^L$$

Ecuación 1.42: Cota superior teórica de la PER de acuerdo a [4]

La Tabla 1.9 indica los valores de d_{free} para cada tasa de codificación del código convolucional utilizado en la capa física OFDM de IEEE 802.11p, así como los primeros 10 valores de a_d, partiendo de $d = d_{free}$.

Como se tienen solo los 10 primeros valores de a_d y cada siguiente valor es menos representativo en cuanto el aporte a la cota, la sumatoria infinita se corta en el décimo término, es decir:

$$PER_{conf} \leq 1 - \left(1 - \sum_{d=d_{free}}^{d_{free+9}} a_d P_d^{mod}\right)^L$$

Ecuación 1.43: Cota superior teórica de la PER de acuerdo a [4], para los 10 primeros valores de d

Tabla 1.9: $d_{f\,ree}$ y a_d para codificador convolucional $K = 7$ de la capa física OFDM IEEE 802.11p [4]

Tasa de codificación	$d_{f\,ree}$	$a_d \quad d = d_{free}, d_{free+1}, \ldots$
1/2	10	11, 0, 38, 0, 193, 0, 1331, 0, 7275, 0, ...
3/4	5	8, 31, 160, 892, 4512, 23307, 121077, 625059, 3234886, 16753077, ...
2/3	6	1, 16, 48, 158, 642, 2435, 9174, 34705, 131585, 499608, ...

El término d_{free} se refiere a la distancia mínima del código convolucional, la cual representa la mínima distancia de una palabra código que se aleje del camino de todo 0 en el diagrama de *Trellis*, mientras que el término a_d representa el número de caminos en el diagrama de *Trellis* de longitud d [19]. L representa la longitud del paquete.

Las siguientes ecuaciones describen el término P_d^{mod} utilizadas en (1.43) para un d dado. El término b_e^{mod} es tomado de las ecuaciones (1.41) y "mod" corresponde al esquema de modulación (BPSK, QPSK, 16QAM, 64QAM).

$$P_d^{mod} = \begin{cases} \sum_{k=(d+1)/2}^{d} \binom{d}{k} \left(b_e^{mod}\right)^k \left(1 - b_e^{mod}\right)^{d-k} & dimpar \\ \sum_{k=d/2+1}^{d} \binom{d}{k} \left(b_e^{mod}\right)^k \left(1 - b_e^{mod}\right)^{d-k} + \frac{1}{2}\binom{d}{d/2} \left(b_e^{mod}\right)^{\frac{d}{2}} \left(1 - b_e^{mod}\right)^{\frac{d}{2}} & dpar \end{cases}$$

Ecuación 1.44: PER teórica por esquema de modulación de acuerdo a [4]

Las ecuaciones (1.40), (1.41), (1.43), (1.44) y la Tabla 1.9 representan un modelo matemático para el cálculo del PER sobre un canal AWGN, que toma en cuenta los esquemas de modulación y codificación de la capa física del estándar IEEE 802.11p.

El artículo además extiende la noción para canales con *fading* tipo Rayleigh. Para el cálculo de la PER sobre estos canales se utilizan las ecuaciones (1.40), (1.43) y (1.44) cambiando las definiciones de la BER (b_e) por modo por las siguientes ecuaciones:

$$BPSK:\ b_e^{BPSK}(\gamma_s) = \frac{1}{2}\left(1 - \sqrt{\frac{\gamma_s}{1+\gamma_s}}\right)$$

$$MQAM:\ b_e^{MQAM}(\gamma_s) = 2\left(\frac{\sqrt{M}-1}{\sqrt{M}}\right)\left(\frac{1}{\log_2 M}\right)\sum_{i=1}^{\sqrt{M}/2}\left(1 - \mu_i^{MQAM}\right)$$

Ecuación 1.45: BER teórica por esquema de modulación con *fading* de acuerdo a [4]

En donde $MQAM$ puede representar modulación tipo $QPSK$ ($M = 4$), $16QAM$ ($M = 16$) o $64QAM$ ($M = 64$). El término μ^{MQAM} se define mediante la siguiente ecuación:

$$\mu_i^{MQAM} = \sqrt{\frac{1.5(2i-1)^2\gamma_s}{M - 1 + 1.5(2i-1)^2\gamma_s}}$$

Ecuación 1.46: Parámetro μ de [4]

f. Modelo de errores no homogéneos en capa física

Khalili y Salamatian en su trabajo dado en [5] proponen un nuevo modelo teórico predictivo de la PER que considera la no uniformidad de los errores a la salida de la capa física debido al proceso de decodificación convolucional (comúnmente bajo el algoritmo de Viterbi). Como no se asume que los errores están distribuidos de una manera uniforme, se necesita determinar una distribución de los eventos de error. Para simplificar la obtención de esta distribución, se utiliza la técnica de terminación del código para relacionar la probabilidad de error del código convolucional al de un código de bloque, el cual tiene una probabilidad de error más simple.

Un código convolucional toma k bits como entrada y da n bits como salida, teniendo que la tasa de codificación r es dada por:

$$r = \frac{k}{n}$$

Ecuación 1.47: Tasa de codificación de código convolucional

Los bits de salida dependen no solo de los bits de entrada, sino también de la longitud del

código o *constraint lenght* que se denota como K, y es el número de bits anteriores que influyen en una salida en particular. Para el codificador de IEEE 802.11p, se tiene que $K = 7$.

Se entiende como un evento de error cuando el camino seguido en el diagrama de *Trellis* en el proceso de decodificación se separa del camino seguido en el proceso de codificación respectivo. La longitud de un evento de error es igual a la cantidad de bits errados más K bits no errados.

La técnica de terminación de código finaliza un código convolucional después de cierto número τ de entradas. Este número τ de símbolos debe contener al menos K ceros al final, los cuales hacen volver al codificador a su estado nulo. Con este corte, es posible interpretar el sistema entre las salidas y entradas del codificador como un código de bloque de longitud:

$$N = n\tau$$

Ecuación 1.48: Longitud de código de bloque equivalente

Este código de bloque equivalente tendrá una tasa de codificación (R) igual a:

$$R = (1 - \theta)r$$
$$\theta = \frac{K}{\tau}$$

Ecuación 1.49: Tasa de codificación equivalente

Se define la tasa de eventos de error (*Error Event Rate*, EER) de acuerdo a:

$$EER(\gamma) = a_{d_{free}} \exp\left(-R\gamma d_{free}\right)$$

Ecuación 1.50: *Event Error Rate*

en donde γ es la SNR, d_{free} es la longitud libre del código y $a_{d_{free}}$ es el número de caminos cerrados en el diagrama de *Trellis* de longitud d_{free}. Estos valores se pueden obtener a partir de la Tabla 1.9.

Se obtiene un parámetro λ que se deriva del exponente de error sobre códigos de bloque y la longitud promedio de eventos de error:

$$\lambda(\gamma) = \frac{EER(\gamma)}{1 - \left((K+1) + \frac{1}{n\left(\gamma/2 - \sqrt{2\gamma r} + r\right)}\right) EER(\gamma)}$$

Ecuación 1.51: Parámetro λ de [5]

Con este parámetro, se obtiene una expresión para una cota superior del PER, que está definida por la siguiente ecuación:

$$PER(\gamma, L) = 1 - (1 - \lambda(\gamma))^L$$

Ecuación 1.52: PER teórica de acuerdo a [5]

en donde L es la longitud del paquete. Esta ecuación es supuesta para modulación BPSK.

g. Modelo de PER promedio bajo distribución de SNR

Xi y Burr en su trabajo dado en [6] buscan una cota superior para la PER promedio para transmisiones sobre canales con ruido AWGN, *fading* tipo Rayleigh, y *fading* tipo Nakagami-m.

La PER promedio se toma con respecto a la respuesta para cada valor de SNR, tomando en cuenta la probabilidad de cada SNR en el canal. Se obtiene de manera general con la siguiente relación:

$$P_{prom} = \int_0^\infty f(\gamma) p(\gamma) d\gamma$$

Ecuación 1.53: PER promedio

en donde P_{prom} es el PER promedio, $f(\gamma)$ es la respuesta de la PER para cada SNR en particular sobre el canal AWGN, $p(\gamma)$ es la función de distribución de probabilidad (*Probability Distribution Function*, PDF) para cada valor de SNR sobre el canal con *fading* y γ es la SNR.

Considerando un canal con *fading* Rayleigh, la PDF viene dada por:

$$p(\gamma) = \frac{1}{\bar{\gamma}} \exp\left(-\gamma/\bar{\gamma}\right)$$

Ecuación 1.54: PER teórica en función de la SNR con *fading* de acuerdo a [6]

en donde $\bar{\gamma}$ es la SNR promedio, entonces la PER promedio para un canal Rayleigh viene dado por:

$$P_{prom}(\bar{\gamma}) = \frac{1}{\bar{\gamma}} \int_0^\infty f(\gamma) \exp\left(-\gamma/\bar{\gamma}\right) d\gamma$$

Ecuación 1.55: PER promedio contemplando PDF

Luego, teniendo en cuenta que:

$$\omega_0 = \int_0^\infty f(\gamma) d\gamma$$

Ecuación 1.56: Parámetro ω_0 en [6]

la PER promedio sobre un canal Rayleigh cuasi-estático viene acotado por:

$$P_{prom}(\bar{\gamma}) \leq 1 - \exp\left(-\omega_0/\bar{\gamma}\right)$$

Ecuación 1.57: Cota superior de la PER promedio de acuerdo a [6]

que, a medida que la SNR promedio tiende al infinito, esta expresión se obtiene por:

$$P_{prom}(\bar{\gamma}) \approx 1 - \exp\left(-\omega_0/\bar{\gamma}\right) \approx \omega_0/\bar{\gamma}$$

Ecuación 1.58: PER promedio cuando la SNR tiende al infinito de acuerdo a [6]

Si se considera que la PER presenta una característica empinada sobre la SNR, es decir que existe un γ_0 y un $\alpha < 0$ tal que:

$$f(\gamma \mid \gamma \leq \alpha\gamma_0) \approx 1$$
$$f(\gamma \mid \gamma > \gamma_0/\alpha) \approx 0,$$

Ecuación 1.59: Característica empinada de la SNR

entonces, a medida que α se aproxima a 1, es decir, a medida que el rango de SNR para el cual la PER esté entre 0 y 1 sin ser 0 o 1 se reduzca, se tiene que:

$$P_{prom}(\bar{\gamma}) \to 1 - \exp\left(-\gamma_0/\bar{\gamma}\right)$$

Ecuación 1.60: PER promedio para SNR empinada de acuerdo a [6]

Este concepto se extiende para canales Nakagami-m, donde, en este caso, m es el factor de forma de la distribución Nakagami. La PDF en este tipo de canal viene dada por:

$$p(\gamma) = \frac{m^m \gamma^{m-1}}{(\bar{\gamma})^m \Gamma(m)} \exp\left(-\frac{m\gamma}{\bar{\gamma}}\right)$$
$$\omega_m = \int_0^\infty \gamma^{m-1} f(\gamma) d\gamma.$$

Ecuación 1.61: PDF de la PER en función de la SNR para canales con *fading* de acuerdo a [6]

Con un proceso similar al anterior se llega a la siguiente cota:

$$P_{prom}(\bar{\gamma}) \leq \frac{m^m \omega_m}{\Gamma(m)} (\bar{\gamma})^{-m}$$

Ecuación 1.62: PER promedio para canales con *fading* de acuerdo a [6]

Para estos desarrollos se considera que se cuenta ya con un $f(\gamma)$, que es la función que describe la relación de la PER con la SNR. Estos desarrollos proveen una cota superior para la PER promedio en canales Rayleigh o Nakagami-m cuando la SNR es lo suficientemente alta.

h. Otros modelos

La Ecuación (1.27) es utilizada de manera general para el cálculo de la PER. La base de esta ecuación es calcular la probabilidad de que cada bit dentro del paquete resulte no estar errado y tomar el complemento probabilístico de esta cantidad.

Sabiendo que b_e es la BER, o bien, la probabilidad de que un bit se encuentre errado, entonces $(1 - b_e)$ es la probabilidad de que un bit no se encuentre errado. Dentro de un paquete se tienen L bits mutuamente independientes (en principio), por lo que la probabilidad de que ninguno de los bits se encuentre errado es una multiplicación de la probabilidad de que un bit no se encuentre errado L veces, sea: $(1-b_e)(1-b_e)(1-b_e)$...$(1-b_e)$ L veces, o bien:

$$(1 - b_e)^L$$

Ecuación 1.63: Probabilidad de que un paquete de L bits no se encuentre errado

Siendo esta la probabilidad de que ningún bit se encuentre errado dentro del paquete, el complemento de esta probabilidad viene dado por:

$$1 - (1 - b_e)^L$$

Ecuación 1.64: Probabilidad de que un paquete de L bits se encuentre errado (PER clásica)

Esta expresión es, entonces, la probabilidad de que al menos un bit dentro del paquete se encuentre errado. Esto es, la probabilidad de que el paquete se encuentre errado: la PER

esperada.

Esta es una expresión clásica para el cálculo de la PER, ya que asume que la existencia de un error en un bit es independiente de los otros bits en el paquete, lo cual, de hecho, no se cumple para sistemas con codificadores convolucionales como se ve en [5].

Sin embargo, esta es una aproximación que se puede tomar en cuenta para la comparación teórica de este trabajo de titulación. Si bien con [2] se utiliza esta relación con b_e obtenido de [3], existen otras expresiones para el término b_e que se pueden utilizar con la Ecuación (1.27) y que servirán de comparación.

MATLAB cuenta con librerías para el cálculo de la BER bajo ciertas condiciones. Se utilizará la función $berawgn(\cdot)$ de MATLAB, la cual utiliza las siguientes ecuaciones para el cálculo de la BER:

$BPSK$:

$$b_e = Q\left(\sqrt{2\frac{E_b}{N_0}}\right)$$

Ecuación 1.65: BER teórica de acuerdo a las librerías de MATLAB para modulación BPSK

$QPSK$:

$$b_e = Q\left(\sqrt{2\frac{E_b}{N_0}}\right)$$

Ecuación 1.66: BER teórica de acuerdo a las librerías de MATLAB para modulación QPSK

$16QAM$:

$$b_e = \frac{1}{4}\sum_{k=1}^{2}\sum_{i=0}^{4(1-2^{-k})-1}\left\{(-1)^{\left\lfloor\frac{i2^{k-1}}{4}\right\rfloor}\left(2^{k-1}-\left\lfloor\frac{i2^{k-1}}{4}\right\rfloor+\frac{1}{2}\right)Q\left((2i+1)\sqrt{\frac{12}{15}\frac{E_b}{N_0}}\right)\right\}$$

Ecuación 1.67: BER teórica de acuerdo a las librerías de MATLAB para modulación 16QAM

$64QAM$:

$$b_e = \frac{1}{12}\sum_{k=1}^{3}\sum_{i=0}^{8(1-2^{-k})-1}\left\{(-1)^{\left\lfloor \frac{i2^{k-1}}{8}\right\rfloor}\left(2^{k-1}-\left\lfloor \frac{i2^{k-1}}{8}\right\rfloor+\frac{1}{2}\right)Q\left((2i+1)\sqrt{\frac{18}{63}\frac{E_b}{N_0}}\right)\right\}$$

Ecuación 1.68: BER teórica de acuerdo a las librerías de MATLAB para modulación 64QAM

Para la relación entre SNR y E_b/N_0 se utilizará la Ecuación (1.31). Si bien las ecuaciones para BPSK y QPSK son las mismas que las correspondientes en las ecuaciones (1.30), cambian para 16QAM y 64QAM.

Estas ecuaciones no serán implementadas en código, ya que se encuentran definidas en la función $berawgn(\cdot)$ de MATLAB.

1.3.7. Selección de modelos

Luego de haber revisado varios modelos para el cálculo de la PER, se escogerán los listados a continuación para su implementación y comparación con respecto a resultados simulados. Se mencionarán las ecuaciones principales, los parámetros necesarios para ellas se tomarán de la subsección respectiva.

- **Modelo 1:** Modelo para redes vehiculares por interpolación (Ecuación (1.24)).
- **Modelo 2:** Modelo sin codificación sobre canal AWGN mediante EVT (Ecuación (1.33)).
- **Modelo 3:** Modelo de errores no homogéneos en capa física (Ecuación (1.52)).
- **Modelo 4:** Modelo de cota superior para sistema OFDM (Ecuación (1.43)), aplicando ecuaciones (1.32) para la BER.
- **Modelo 5:** Modelo de cota superior para sistema OFDM (Ecuación (1.43)), aplicando ecuaciones (1.41) para la BER.
- **Modelo 6:** Modelo de cota superior para sistema OFDM (Ecuación (1.43)), aplicando ecuaciones (1.65), (1.66), (1.67) y (1.68) para la BER.
- **Modelo 7:** Modelo sin codificación sobre canal AWGN (Ecuación (1.27)), aplicando ecuaciones (1.32) para la BER.
- **Modelo 8:** Modelo sin codificación sobre canal AWGN (Ecuación (1.27)), aplicando ecuaciones (1.41) para la BER.

- **Modelo 9:** Modelo sin codificación sobre canal AWGN (Ecuación (1.27)), aplicando ecuaciones (1.65), (1.66), (1.67) y (1.68) para la BER.

Se han escogido estos modelos debido a que cumplen total o parcialmente los requerimientos en tanto la parametrización necesaria (esquema de modulación, tasa de codificación y longitud del paquete) para su implementación en funciones. Además, son dependientes de la SNR y retornan valores de PER.

Sus ámbitos de aplicación pueden ser interpretados como sistemas inalámbricos de comunicación de manera general, siendo algunos de ellos más específicos para comunicaciones vehiculares. El modelo 3 en particular es válido para configuraciones con modulación BPSK, pero se tomará en cuenta para todas las configuraciones con el fin compararlas en tanto la forma que tiene la curva de PER en función de la SNR.

2. METODOLOGÍA

Tras haber realizado un estudio de los modelos teóricos a comparar y de los bloques funcionales involucrados en el tratamiento de datos para una comunicación IEEE 802.11p, se procede a realizar el diseño de los *scripts* y funciones para la implementación de:

1. El transmisor y receptor IEEE 802.11p con todos los bloques involucrados a nivel de PHY.
2. El canal AWGN.
3. Los modelos teóricos para el cálculo de la PER.
4. *Scripts* auxiliares:
   a) Obtención de datos de la simulación y de cada modelo teórico propuesto.
   b) Tratamiento de datos, obtención y muestra de resultados.
5. *Script* de la simulación principal para la integración de todos los bloques y la obtención de datos simulados.

Este diseño contempla el desarrollo de los algoritmos y esquematización de los diagramas de flujo de cada *script* a implementar. Además se propondrá un plan de ejecución de *scripts* para una correcta obtención y tratamiento de datos.

Posteriormente se realizará dicha implementación de *scripts* y funciones en el software MATLAB, y se ejecutarán en el orden establecido, obteniendo los *scripts* y funciones en archivos .m y los resultados en archivos .mat.

El desarrollo del código en MATLAB para el sistema de comunicaciones y cálculo de la PER se explicará a través de diagramas de flujo y algoritmos (pseudocódigo). Se tendrá un diagrama/algoritmo para cada uno de los módulos que serán implementados por medio de *scripts* y funciones. Todos los diagramas de flujo se encuentran en el Anexo I, mientras que todos los algoritmos se encuentran en el Anexo II.

2.1. Escenario de simulación

La estructura de los *scripts* y las funciones de la simulación principal se observa en la Figura 2.1.

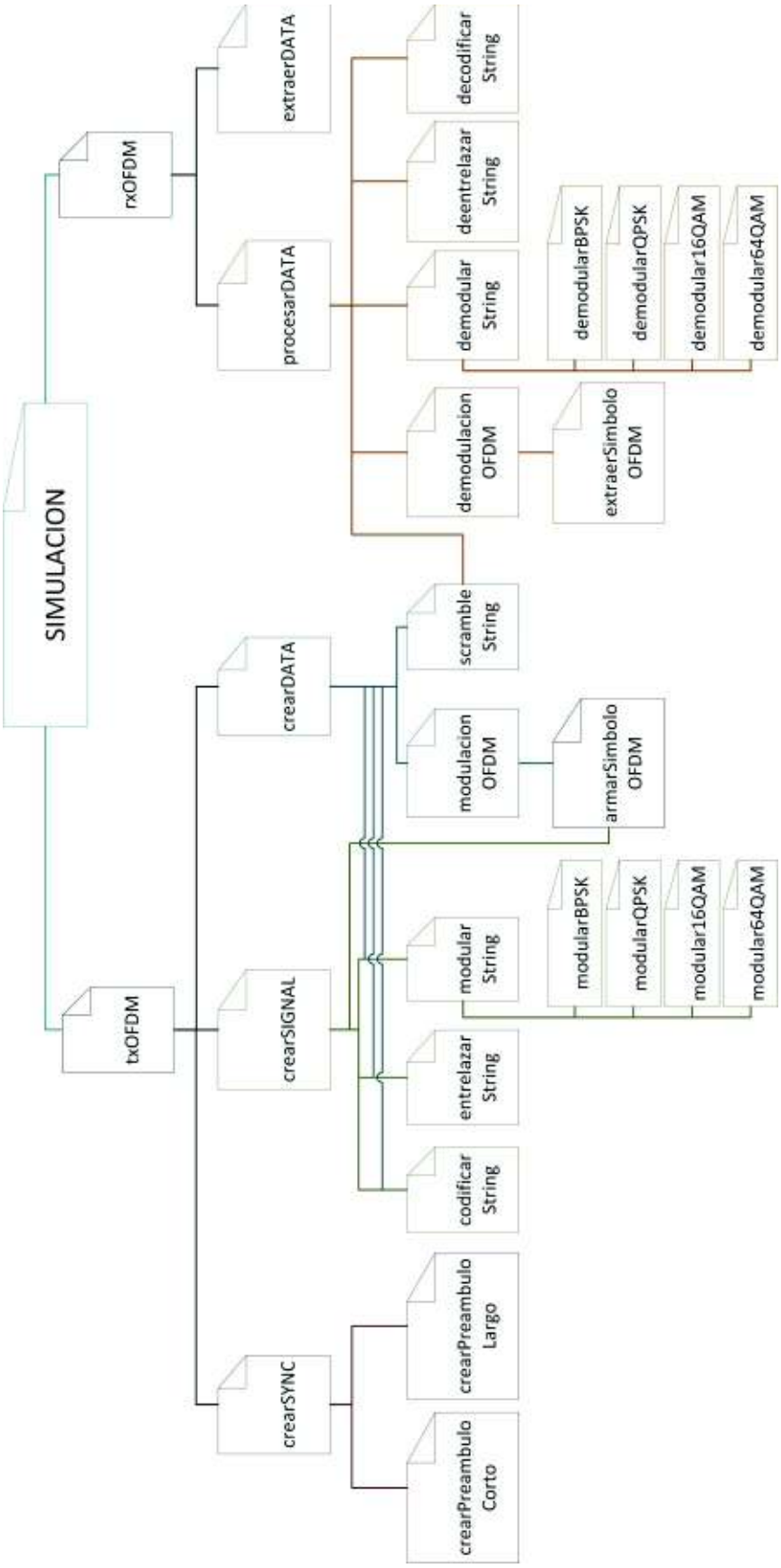


Figura 2.1: Estructura de las funciones y *scripts* de la simulación

2.2. Sistema de comunicación IEEE 802.11p

A continuación se mostrarán los diagramas de flujo correspondientes a cada uno de los módulos para la implementación del sistema de comunicación IEEE 802.11p

2.2.1. Simulación principal

Este es el *script* de la simulación principal; desde aquí se realizan las siguientes tareas:

- Ingresar parámetros de transmisión (número de paquetes, número de bits por paquete, rango de SNR).
- Llamar a los *scripts* de transmisión y recepción.
- Simular la transmisión, envío y recepción de paquetes para los parámetros de transmisión ingresados y cada una de las configuraciones en la Tabla 1.2 para un mismo rango de SNR.
- Calcular los parámetros de rendimiento para cada configuración (BER, PER, número de errores).
- Guardar resultados en archivos .mat para su posterior análisis.

Para la simulación del *script* principal se presentan 2 modos de ejecución:

- **SNR homogéneo:** el diagrama de flujo se puede apreciar en la Figura I.2 (Anexo I), mientras que el algoritmo se presenta en el Algoritmo 1 (Anexo II).
- **SNR angosto:** se toma en cuenta un rango de SNR personalizado para cada configuración. El diagrama de flujo se puede apreciar en la Figura I.3 (Anexo I), mientras que el algoritmo se presenta en el Algoritmo 2 (Anexo II).

2.2.2. Transmisión

Para implementar la transmisión de un paquete según el diagrama presentado en la Figura 2.1, se ha creado la función txOFDM() que realiza las siguientes tareas:

- Verificar que el paquete tenga una longitud válida de acuerdo al estándar.
- Llamar a funciones para la creación del preámbulo (CrearSYNC()), cabecera (CrearSIGNAL()) y *payload* (CrearDATA()).

- Concatenar los elementos del paquete en tiempo.
- Enviar PSDU y el paquete en tiempo como parámetros de salida.

La llamada de la función es la siguiente:

```
[Paquete, PSDU] = txOFDM(numBit, M, r)
```

Entrada: número de bits por paquete (numBit), esquema de modulación (M), tasa de codificación (r)

Salida: trama capa física en tiempo (Paquete), payload en bits (PSDU)

El diagrama de flujo correspondiente a esta función se puede apreciar en la Figura I.4 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 3 (Anexo II).

a. Creación del preámbulo

Para crear el preámbulo se ha desarrollado la función CrearSYNC(), la cual permite crear el preámbulo corto y el preámbulo largo; concatenar estos 2 elementos y devolver como salida el campo SYNC del estándar (PPDU PLCP). Las tareas de esta función son:

- Llamar a funciones para creación del preámbulo corto (CrearPreambuloCorto()) y el preámbulo largo (CrearPreambuloLargo()).
- Concatenar ambos preámbulos y enviar como parámetro de salida.

La llamada de la función es la siguiente:

```
SYNC = CrearSYNC()
```

Salida: campo SYNC del PPDU PLCP (SYNC)

El diagrama de flujo se puede apreciar en la Figura I.5 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 4 (Anexo II).

b. Creación del preámbulo corto

La función para la creación del preámbulo corto denominada CrearPreambuloCorto() realiza las siguientes tareas:

- Generar el arreglo para el preámbulo corto descrito en el estándar IEEE 802.11

[8].

- Realizar la IFFT del arreglo y generar el preámbulo corto en tiempo al concatenar 10 copias del primer $1/4$ de este.
- Enviar como parámetro de salida el preámbulo corto.

La llamada de la función es la siguiente:

```
ShortP = CrearPreambuloCorto()
```

Salida: preámbulo corto en tiempo del PPDU PLCP (ShortP)

El diagrama de flujo se puede apreciar en la Figura I.6 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 5 (Anexo II).

c. Creación del preámbulo largo

La función para la creación del preámbulo largo denominada CrearPreambuloLargo() realiza las siguientes tareas:

- Generar el arreglo para el preámbulo largo descrito en el estándar IEEE 802.11 [8].
- Realizar la IFFT del arreglo y concatenar una copia de la última mitad del mismo.
- Enviar como parámetro de salida el preámbulo largo.

La llamada de la función es la siguiente:

```
LongP = CrearPreambuloLargo()
```

Salida: preámbulo largo en tiempo del PPDU PLCP (LongP)

El diagrama de flujo se puede apreciar en la Figura I.7 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 6 (Anexo II).

d. Creación de la cabecera

La cabecera de un paquete PLCP corresponde al campo SIGNAL. Para su creación se ha implementado la función CrearSIGNAL() que realiza las siguientes tareas:

- Generar los bits de cabecera de acuerdo a IEEE 802.11 [8] y lo especificado en la

Figura 1.5.

- Llamar a las funciones para la codificación (codificarString()), entrelazado (entrelazarString()), modulado (modularString()), y modulación OFDM (modulacionOFDM()) de la cabecera con tasa $r = 1/2$ y $M = 2$ de acuerdo a lo indicado al estándar IEEE 802.11 [8].
- Enviar como parámetro de salida el campo SIGNAL.

La llamada de la función es la siguiente:

```
SIGNAL = CrearSIGNAL()
```

Salida: campo SIGNAL en tiempo del PPDU PLCP (SIGNAL)

El diagrama de flujo correspondiente se presenta en la Figura I.8 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 7 (Anexo II).

e. Creación del campo DATA

La creación del campo DATA del PPDU PLCP, el cual contiene el *payload* se ha implementado en la función CrearDATA(), la cual realiza las siguientes tareas:

- Generar los bits del campo SERVICE y el campo *Tail* de acuerdo a IEEE 802.11 [8] (Figura 1.5).
- Generar los bits de *padding* necesarios de acuerdo a IEEE 802.11 [8].
- Generar el campo DATA del PPDU PLCP.
- Llamar a funciones para la aleatorización (scrambleString()), codificación (codificarString()), entrelazado (entrelazarString()), modulado (modularString()), y modulación OFDM (modulacionOFDM()) del campo DATA para los diferentes valores de r y M de acuerdo al estándar IEEE 802.11 [8] que se describen en la sección siguiente.
- Enviar DATA como parámetro de salida.

La llamada de la función es la siguiente:

```
DATA = CrearDATA(PSDU, M, r)
```

Entrada: *payload* (PSDU), esquema de modulación (M), tasa de codificación (r)
Salida: campo DATA en tiempo del PPDU PLCP (DATA)

El diagrama de flujo se puede apreciar en la Figura I.9 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 8 (Anexo II).

f. Aleatorización de datos

El proceso de aleatorización de los datos se ha implementado en la función scrambleString() que realiza las siguientes tareas:

- Realizar la aleatorización de los datos de entrada bit a bit, teniendo en cuenta el estado inicial, siguiendo al estándar IEEE 802.11 [8].
- Enviar arreglo de bits resultante como parámetro de salida.

La llamada de la función es la siguiente:

```
SDatos = scrambleString(datos, estadoInicial)
```

Entrada: bits de datos (datos), estado inicial de aleatorizador (estadoInicial)
Salida: bits aleatorizados (SDatos)

El diagrama de flujo se puede apreciar en la Figura I.10 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 9 (Anexo II).

g. Codificación convolucional de datos

El proceso de codificación convolucional se ha implementado en la función codificarString() que realiza las siguientes tareas:

- Generar el diagrama de *Trellis* del codificador convolucional a partir de los parámetros del codificador del estándar IEEE 802.11 [8].
- Verificar que la tasa de codificación sea válida.
- Definir el patrón de puntura [8] de acuerdo a la tasa de codificación.
- Codificar datos siguiendo el diagrama de *Trellis* generado y el patrón de puntura escogido.

- Enviar arreglo de bits resultante como parámetro de salida.

La llamada de la función es la siguiente:

```
CDatos = codificarString(datos, R)
```

Entrada: bits de datos (datos), tasa de codificación (R)
Salida: bits codificados (CDatos)

El diagrama de flujo se puede apreciar en la Figura I.11 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 10 (Anexo II).

h. Entrelazado de datos

El proceso de entrelazado de datos se ha implementado en la función entrelazarString() que realiza las siguientes tareas:

- Separar el arreglo de bits de entrada en grupos de N_{CBPS} para su procesamiento grupo por grupo.
- Realizar la primera permutación sobre cada grupo según el estándar IEEE 802.11 [8].
- Realizar la segunda permutación sobre cada grupo permutado según el estándar IEEE 802.11 [8].
- Enviar arreglo de bits total como parámetro de salida.

La llamada de la función es la siguiente:

```
IDatos = entrelazarString(datos, Ncbps)
```

Entrada: bits de datos (datos), N_{CBPS} (Ncbps)
Salida: bits entrelazados (IDatos)

El diagrama de flujo se puede apreciar en la Figura I.12 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 11 (Anexo II).

i. Modulación de datos

El proceso de modulación se ha implementado en la función modularString() que realiza las

siguientes tareas:

- Discernir entre el esquema de modulación con respecto al valor de M.
- Llamar al *script* de modulación respectivo, con parámetro el arreglo de bits de entrada multiplicado por el factor de normalización descrito en el estándar IEEE 802.11 [8].
- Enviar arreglo de símbolos resultante como parámetro de salida.

La llamada de la función es la siguiente:

```
MDatos = modularString(datos, M)
```

Entrada: bits de datos (datos), esquema de modulación (M)
Salida: símbolos complejos (MDatos)

El diagrama de flujo se puede apreciar en la Figura I.13 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 12 (Anexo II).

j. Transmisión OFDM

El proceso de modulación OFDM se ha implementado en la función modulacionOFDM() que realiza las siguientes tareas:

- Obtener el arreglo de polaridad de las subportadoras piloto.
- Calcular el número de símbolos OFDM a ser generados de acuerdo al estándar IEEE 802.11 [8].
- Separar el arreglo de entrada en grupos de 48 símbolos para la generación de cada símbolo OFDM.
- Para cada grupo, llamar a la función de generación del símbolo OFDM en tiempo (armarSimboloOFDM()) y concatenar en un arreglo que los contenga.
- Enviar arreglo de símbolos OFDM en tiempo resultante como parámetro de salida.

La llamada de la función es la siguiente:

```
simbolos = modulacionOFDM(datos)
```

Entrada: símbolos complejos de datos (datos)

Salida: símbolos OFDM (simbolos)

El diagrama de flujo se puede apreciar en la Figura I.14 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 13 (Anexo II).

k. Generación de símbolo OFDM

El proceso de generación de un símbolo OFDM se ha implementado en la función armarSimboloOFDM() que realiza las siguientes tareas:

- Asignar valores arreglo de entrada y pilotos con su polaridad, al símbolo OFDM en frecuencia de acuerdo al estándar IEEE 802.11 [8].
- Redistribuir símbolo OFDM para su tratamiento discreto de acuerdo al estándar IEEE 802.11 [8].
- Realizar la IFFT del símbolo OFDM en frecuencia y concatenar el último $1/4$ del mismo.
- Enviar el símbolo OFDM en tiempo resultante como parámetro de salida.

La llamada de la función es la siguiente:

```
OFDMSym = armarSimboloOFDM(datos, polPiloto)
```

Entrada: 48 símbolos complejos (datos), polaridad de pilotos (polPiloto)
Salida: 1 símbolo OFDM (OFDMSym)

El diagrama de flujo se puede apreciar en la Figura I.15 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 14 (Anexo II).

l. Modulación BPSK

La modulación BPSK de datos se ha implementado en la función modularBPSK() que realiza las siguientes tareas:

- Asignar a cada bit de entrada, un símbolo complejo de acuerdo al diagrama de constelaciones correspondiente de la Figura 1.16.
- Enviar arreglo de símbolos complejos resultantes como parámetro de salida.

La llamada de la función es la siguiente:

```
modulado = modularBPSK(datos)
```

Entrada: bits de datos (datos)
Salida: símbolos BPSK en frecuencia (modulado)

El diagrama de flujo se puede apreciar en la Figura I.24 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 23 (Anexo II).

m. Modulación QPSK

La modulación QPSK de datos se ha implementado en la función modularQPSK() que realiza las siguientes tareas:

- Asignar a cada dos bits de entrada, un símbolo complejo de acuerdo al diagrama de constelaciones correspondiente de la Figura 1.16.
- Enviar arreglo de símbolos complejos resultantes como parámetro de salida.

La llamada de la función es la siguiente:

```
modulado = modularQPSK(datos)
```

Entrada: bits de datos (datos)
Salida: símbolos QPSK en frecuencia (modulado)

El diagrama de flujo se puede apreciar en la Figura I.25 (Anexo I), mientras que el algoritmo se lo puede apreciar en el Algoritmo 24 (Anexo II).

n. Modulación 16QAM

La modulación 16QAM de datos se ha implementado en la función modular16QAM() que realiza las siguientes tareas:

- Asignar a cada cuatro bits de entrada, un símbolo complejo de acuerdo al diagrama de constelaciones correspondiente de la Figura 1.16.
- Enviar arreglo de símbolos complejos resultantes como parámetro de salida.

La llamada de la función es la siguiente:

```
modulado = modular16QAM(datos)
```

Entrada: bits de datos (datos)
Salida: símbolos 16QAM en frecuencia (modulado)

El diagrama de flujo se lo puede apreciar en la Figura I.26 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 25 (Anexo II).

o. Modulación 64QAM

La modulación 64QAM de datos se ha implementado en la función modular64QAM() que realiza las siguientes tareas:

- Asignar a cada seis bits de entrada, un símbolo complejo de acuerdo al diagrama de constelaciones correspondiente de la Figura 1.16.
- Enviar arreglo de símbolos complejos resultantes como parámetro de salida.

La llamada de la función es la siguiente:

```
modulado = modular64QAM(datos)
```

Entrada: bits de datos (datos)
Salida: símbolos 64QAM en frecuencia (modulado)

El diagrama de flujo se puede apreciar en la Figura I.27 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 26 (Anexo II).

2.2.3. Recepción

Para implementar la recepción de un paquete según el diagrama presentado en la Figura 2.1, se ha creado la función rxOFDM() que realiza las siguientes tareas:

- Llamar a la función para la extracción del campo DATA del paquete (extraerDATA()).
- Llamar a la función para el procesamiento del campo DATA (procesarData()), para obtener el PSDU correspondiente.

- Enviar PSDU como parámetro de salida.

La llamada de la función es la siguiente:

```
PSDU = rxOFDM(PAQUETE, numBit, M, r)
```

Entrada: número de bits por paquete (numBit), esquema de modulación (M), tasa de codificación (r)

Salida: *payload* del PPDU (PSDU)

El diagrama de flujo se puede apreciar en la Figura I.16 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 15 (Anexo II).

a. Extracción del campo DATA

La extracción del campo DATA de un PPDU se ha implementado en la función extraerData() que realiza las siguientes tareas:

- Extraer las muestras correspondientes al campo DATA del paquete de acuerdo al estándar IEEE 802.11 [8].
- Enviar DATA como parámetro de salida.

La llamada de la función es la siguiente:

```
DATAT = extraerDATA(PAQUETE)
```

Entrada: PSDU en tiempo (PAQUETE)

Salida: símbolos OFDM correspondientes al campo DATA en tiempo (DATAT)

El diagrama de flujo se puede apreciar en la Figura I.17 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 16 (Anexo II).

b. Procesamiento del campo DATA

El procesamiento del campo DATA se ha implementado en la función procesarData() que realiza las siguientes tareas:

- Llamar a las funciones para la demodulación OFDM (demodulacionOFDM()), demodulación (demodularString()), deentrelazado (deentrelazarString()), decodificación

(decodificarString()) y aleatorización del PSDU (scrambleString()) con la tasa de codificación (r) y el esquema de modulación (M), de acuerdo al estándar IEEE 802.11 [8].

- Enviar PSDU como parámetro de salida.

La llamada de la función es la siguiente:

```
PSDU = procesarData(DATA, numBit, M, r)
```

Entrada: campo DATA en símbolos OFDM de datos en tiempo (DATA), número de bits por paquete (numBit), esquema de modulación (M), tasa de codificación (r)

Salida: *payload* en bits (PSDU)

El diagrama de flujo se puede apreciar en la Figura I.18 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 17 (Anexo II).

c. Demodulación OFDM

La demodulación OFDM se ha implementado en la función demodulacionOFDM() que realiza las siguientes tareas:

- Calcular el número de símbolos OFDM recibidos.
- Llamar a la función para el procesamiento del símbolo OFDM (extraerSimboloOFDM()) para cada grupo de 80 muestras.
- Concatenar todos los símbolos en frecuencia extraídos en un arreglo.
- Enviar el arreglo como parámetro de salida.

La llamada de la función es la siguiente:

```
MDATA = demodulacionOFDM(DATA)
```

Entrada: símbolos OFDM en tiempo (DATA)

Salida: símbolos complejos en frecuencia (MDATA)

El diagrama de flujo se puede apreciar en la Figura I.19 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 18 (Anexo II).

d. Mapeo del símbolo OFDM

El mapeo del símbolo OFDM se ha implementado en la función extraerSimboloOFDM() que realiza las siguientes tareas:

- Extraer las muestras que no corresponden al prefijo cíclico.
- Realizar el FFT a las muestras extraídas.
- Redistribuir las muestras resultantes sobre un arreglo de acuerdo al estándar IEEE 802.11 [8].
- Extraer las muestras de datos del arreglo resultante.
- Enviar este arreglo de símbolos como parámetro de salida.

La llamada de la función es la siguiente:

```
datosF = extraerSimboloOFDM(OFDMSym, polPiloto)
```

Entrada: símbolo OFDM en tiempo (OFDMSym), polaridad de los pilotos (polPiloto)
Salida: 48 símbolos complejos en frecuencia (datosF)

El diagrama de flujo se lo puede apreciar en la Figura I.20 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 19 (Anexo II).

e. Demodulación de datos

La demodulación de los datos se ha implementado en la función demodularString() que realiza las siguientes tareas:

- Discernir entre el esquema de modulación con respecto al valor de M.
- Llamar a la función de demodulación respectiva, enviando como parámetro el arreglo de bits.
- Dividir la salida de las funciones para el factor de normalización respectivo descrito en el estándar IEEE 802.11 [8] (Tabla 1.5).
- Enviar arreglo de símbolos resultante como parámetro de salida.

La llamada de la función es la siguiente:

```
IDATA = demodularString(MDATA, M)
```

Entrada: símbolos complejos en frecuencia (MDATA), esquema de modulación (M)

Salida: bits de datos (IDATA)

El diagrama de flujo se lo puede apreciar en la Figura I.21 (Anexo I), mientras que el algoritmo se lo puede apreciar en el Algoritmo 20 (Anexo II).

f. Deentrelazado de datos

El deentrelazado de datos se ha implementado en la función deentrelazarString() que realiza las siguientes tareas:

- Separar el arreglo de bits de entrada en grupos de N_{CBPS} para el procesamiento grupo por grupo.
- Realizar la segunda permutación inversa sobre cada grupo de según el estándar IEEE 802.11 [8].
- Realizar la primera permutación inversa sobre cada grupo permutado según el estándar IEEE 802.11 [8].
- Enviar arreglo de bits total resultante como parámetro de salida.

La llamada de la función es la siguiente:

```
CDATA = deentrelazarString(IDATA, Ncbps)
```

Entrada: bits de datos (IDATA), N_{CBPS} (Ncbps)

Salida: bits deentrelazados (CDATA)

El diagrama de flujo se puede apreciar en la Figura I.22 (Anexo I), mientras que el algoritmo se lo puede apreciar en el Algoritmo 21 (Anexo II).

g. Decodificación convolucional de datos

El proceso de decodificación convolucional mediante el algoritmo de Viterbi se ha implementado en la función decodificarString() que realiza las siguientes tareas:

- Generar el diagrama de *Trellis* del codificador convolucional a partir de los parámetros del codificador del estándar IEEE 802.11 [8].
- Verificar que la tasa de codificación sea válida.
- Definir el patrón de puntura y la profundidad del código [8] de acuerdo a la tasa

de codificación.

- Decodificar bits de acuerdo al algoritmo de Viterbi y los parámetros definidos.
- Enviar arreglo de bits resultante como parámetro de salida.

La llamada de la función es la siguiente:

```
SDATA = decodificarString(CDATA, R, M)
```

Entrada: bits de datos (CDATA), esquema de modulación (M), tasa de codificación (R)

Salida: bits decodificados (SDATA)

El diagrama de flujo se puede apreciar en la Figura I.23 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 22 (Anexo II).

h. Demodulación BPSK

La demodulación BPSK se ha implementado en la función demodularBPSK() que realiza las siguientes tareas:

- Asignar a cada símbolo complejo de entrada, un bit de acuerdo al diagrama de constelaciones correspondiente de la Figura 1.16.
- Enviar arreglo de bits resultantes como parámetro de salida.

La llamada de la función es la siguiente:

```
IDATA = demodularBPSK(MDATA)
```

Entrada: símbolos BPSK en frecuencia (MDATA)

Salida: bits demodulados (IDATA)

El diagrama de flujo se puede apreciar en la Figura I.28 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 27 (Anexo II).

i. Demodulación QPSK

La demodulación QPSK se ha implementado en la función demodularQPSK() que realiza las siguientes tareas:

- Asignar a cada símbolo complejo de entrada, dos bits de acuerdo al diagrama de constelaciones correspondiente de la Figura 1.16.
- Enviar arreglo de bits resultantes como parámetro de salida.

La llamada de la función es la siguiente:

```
IDATA = demodularQPSK(MDATA)
```

Entrada: símbolos QPSK en frecuencia (MDATA)
Salida: bits demodulados (IDATA)

El diagrama de flujo se puede apreciar en la Figura I.29 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 28 (Anexo II).

j. Demodulación 16QAM

La demodulación 16QAM se ha implementado en la función demodular16QAM() que realiza las siguientes tareas:

- Asignar a cada símbolo complejo de entrada, cuatro bits de acuerdo al diagrama de constelaciones correspondiente de la Figura 1.16.
- Enviar arreglo de bits resultantes como parámetro de salida.

La llamada de la función es la siguiente:

```
IDATA = demodular16QAM(MDATA)
```

Entrada: símbolos 16QAM en frecuencia (MDATA)
Salida: bits demodulados (IDATA)

El diagrama de flujo se puede apreciar en la Figura I.30 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 29 (Anexo II).

k. Demodulación 64QAM

La demodulación 64QAM se ha implementado en la función demodular64QAM() que realiza las siguientes tareas:

- Asignar a cada símbolo complejo de entrada, seis bits de acuerdo al diagrama de constelaciones correspondiente de la Figura 1.16.
- Enviar arreglo de bits resultantes como parámetro de salida.

La llamada de la función es la siguiente:

```
IDATA = demodular64QAM(MDATA)
```

Entrada: símbolos 64QAM en frecuencia (MDATA)
Salida: bits demodulados (IDATA)

El diagrama de flujo se puede apreciar en la Figura I.31 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 30 (Anexo II).

2.3. Modelos teóricos predictivos de la PER

En esta sección se detalla cómo se ha implementado en MATLAB los modelos especificados en la sección 1.3.7. Para cada uno de estos modelos se ha desarrollado una función que toma como parámetro de entrada el valor de SNR, el esquema de modulación (M), la tasa de codificación (r) y la longitud de paquete (l); y como salida se tendrá el valor de PER calculado con respecto a las ecuaciones de cada modelo. De manera que estas funciones realizan las siguientes tareas:

- Calcular el valor de la PER de acuerdo a las ecuaciones del modelo respectivo, tomando como parámetros de entrada un valor de SNR (SNR), el esquema de modulación (M), la tasa de codificación (r) y la longitud de paquete en bits (l).
- Enviar el valor de PER como parámetro de salida.

La llamada de la función es la siguiente:

```
PER = modeloX(SNR, M, r, l)
```

El detalle de diagramas de flujo y algoritmos correspondientes a cada modelo (sección 1.3.7) se observa en la Tabla 2.1.

Tabla 2.1: Resumen de diagramas de flujo y algoritmos por modelo de PER

Modelo de PER	Diagrama de flujo	Algoritmo
Modelo 1	Figura I.32 (Anexo I)	Algoritmo 31 (Anexo II)
Modelo 2	Figura I.33 (Anexo I)	Algoritmo 32 (Anexo II)
Modelo 3	Figura I.34 (Anexo I)	Algoritmo 33 (Anexo II)
Modelo 4	Figura I.35 (i=4) (Anexo I)	Algoritmo 34 (i=4) (Anexo II)
Modelo 5	Figura I.35 (i=5) (Anexo I)	Algoritmo 34 (i=5) (Anexo II)
Modelo 6	Figura I.35 (i=6) (Anexo I)	Algoritmo 34 (i=6) (Anexo II)
Modelo 7	Figura I.36 (i=7) (Anexo I)	Algoritmo 35 (i=7) (Anexo II)
Modelo 8	Figura I.36 (i=8) (Anexo I)	Algoritmo 35 (i=8) (Anexo II)
Modelo 9	Figura I.36 (i=9) (Anexo I)	Algoritmo 35 (i=9) (Anexo II)

2.3.1. Modelo teórico predictivo del BER 1

Para el cálculo de la BER de las ecuaciones (1.28), se define la función modeloBER1() que realiza las siguientes tareas:

- Realizar el cálculo del BER de acuerdo a las ecuaciones (1.28), un valor de SNR, el esquema de modulación (M) y la tasa de codificación (r).
- Enviar el valor de BER como parámetro de salida.

La llamada de la función es la siguiente:

```
BER = modeloBER1(SNR, M, r)
```

Entrada: un valor de SNR (SNR), esquema de modulación (M), tasa de codificación (r)
Salida: valor del BER calculado con respecto a las ecuaciones (1.28) (BER)

El diagrama de flujo se puede apreciar en la Figura I.37 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 36 (Anexo II). La función $Q(\cdot)$ viene definida mediante la Ecuación (1.29).

2.3.2. Modelo teórico predictivo del BER 2

Para el cálculo de la BER de las ecuaciones (1.41), se define la función modeloBER2() que realiza las siguientes tareas:

- Realizar el cálculo del BER de acuerdo a las ecuaciones (1.41), un valor de SNR, el esquema de modulación (M) y la tasa de codificación (r).
- Enviar el valor de BER como parámetro de salida.

La llamada de la función es la siguiente:

```
BER = modeloBER2(SNR, M, r)
```

Entrada: un valor de SNR (SNR), esquema de modulación (M), tasa de codificación (r)
Salida: valor del BER calculado con respecto a las ecuaciones (1.41) (BER)

El diagrama de flujo se puede apreciar en la Figura I.38 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 37 (Anexo II). La función $Q(\cdot)$ viene definida mediante la ecuación 1.29.

2.3.3. Modelo teórico predictivo del BER 3

Para el cálculo de la BER de las ecuaciones (1.65), (1.66), (1.67) y (1.68), se define la función modeloBER3() que realiza las siguientes tareas:

- Realizar el cálculo del BER de acuerdo a las ecuaciones (1.65), (1.66), (1.67) y (1.68), un valor de SNR (SNR), el esquema de modulación (M) y la tasa de codificación (r).
- Enviar el valor de BER como parámetro de salida.

La llamada de la función es la siguiente:

```
BER = modeloBER3(SNR, M, r)
```

Entrada: un valor de SNR (SNR), esquema de modulación (M), tasa de codificación (r)
Salida: valor del BER calculado con respecto a las ecuaciones (1.65), (1.66), (1.67) y (1.68) (BER)

El diagrama de flujo se puede apreciar en la Figura I.39 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 38 (Anexo II).

2.4. Tratamiento de datos

Habiendo desarrollado las funciones de la simulación y los modelos teóricos para la obtención de la PER, es necesario diseñar los *scripts* y funciones para el tratamiento, análisis y presentación de estos datos.

2.4.1. Evaluación de modelos teóricos

Para consolidar la evaluación de los modelos del cálculo de la PER sobre un rango de SNR, se define la función evaluarModelo() que realiza las siguientes tareas:

- Escoger el modelo teórico deseado en base al parámetro **num**.
- Correr el modelo teórico deseado para cada SNR del rango de SNR ingresado y guardar los resultados en un vector de PER.
- Enviar el vector de PER como parámetro de salida.

La llamada de la función es la siguiente:

```
PER = evaluarModelo(SNR, M, r, l, num)
```

Entrada: rango de SNR de evaluación (SNR), esquema de modulación (M), tasa de codificación (r), longitud de los paquetes en bits (l), identificador de modelo teórico a utilizar (num)

Salida: rango de PER calculado de las ecuaciones respectivas del modelo escogido (PER)

El diagrama de flujo se puede apreciar en la Figura I.40 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 39 (Anexo II).

2.4.2. Optimización de modelos

Para obtener el error mínimo y *offset* asociado de los modelos teóricos con respecto a los resultados simulados, se define la función minimizarErrorEQX() que realiza las siguientes tareas:

- Introducir un *offset* en SNR para el modelo **numEq** que minimice el error entre los valores experimentales obtenidos de la simulación y los valores teóricos resultantes del modelo **numEq**, con una precisión de **threshold**.
- Enviar valores de error mínimo obtenido y *offset* de SNR respectivo como parámetros de salida.

La llamada de la función es la siguiente:

```
[c, e] = minimizarErrorEQX(SNR, DatosE, l, M, r, numEq, c0)
```

Entrada: valores experimentales de la PER (DatosE), rango de SNR correspondiente a estos valores (SNR), longitud de los paquetes (l), esquema de modulación (M), tasa de codificación (r), identificador de ecuación de minimización (numEq), *offset* inicial (c0)

Salida: *offset* en SNR del modelo escogido para minimización del error (c), valor del error minimizado (e)

Este algoritmo utiliza los valores teóricos (de un modelo en particular escogido) y los experimentales de la PER para una configuración en particular (Tabla 1.2) y calcula el error cuadrático acumulado entre sus muestras. Luego asigna un *offset* en una dirección y vuelve a calcular este error cuadrático acumulado. Si la diferencia entre estos dos errores es lo suficientemente pequeña, es decir, si este error empieza a converger al mínimo posible, se devuelve estos valores de *offset* y error mínimo.

Si esto no se cumple, se vuelve a asignar un *offset* en la misma dirección y se repite el proceso. Por otro lado, si en algún momento no se ha encontrado el error mínimo y el error en una iteración es mayor que la anterior, se reduce el paso del *offset* a la mitad y se lo asigna en la dirección contraria. Podemos pensar que la gráfica teórica oscila alrededor de la simulada con pasos cada vez más pequeños hasta que se encuentre lo más cercana posible a esta.

El diagrama de flujo se puede apreciar en la Figura I.41 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 40 (Anexo II).

a. Análisis de datos

Para obtener valores teóricos de PER de todos los modelos para todas las configuraciones

sobre un rango fijo de SNR, se crea el *script* ANALISIS_PER que realiza las siguientes tareas:

- Obtener datos de la PER teórica para todos los modelos, cada modelo para todas las configuraciones, todas sobre un mismo rango de SNR.
- Guardar los datos con nombres apropiados, en archivos .mat.

Este *script* toma un rango fijo de SNR y define una matriz de parámetros para evaluar cada modelo teórico en cada configuración (esquema de modulación y tasa de codificación). Luego guarda los resultados consolidados por configuración y por modelo en archivos externos .mat.

El diagrama de flujo se puede apreciar en la Figura I.42 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 41 (Anexo II).

2.4.3. Extensión del dominio para resultados de SNR angosto

Para extender el dominio (en SNR) de los valores de la PER obtenidos mediante la simulación de SNR angosto y obtener los valores teóricos con todos los modelos para todas las configuraciones sobre los mismos valores de SNR extendidos correspondientes, se crea el *script* EXTENSION_PER que realiza las siguientes tareas:

- Extender el dominio (en SNR) de los valores de PER simulados por SNR angosto, una cantidad fija de muestras a la izquierda y a la derecha.
- Evaluar los modelos teóricos de la PER uno a uno sobre el rango de SNR extendido correspondiente por cada caso.
- Guardar los datos con nombres apropiados en archivos .mat.

Debido a que las funciones de la PER toman valores entre 0 y 1 en un rango angosto de SNR, a la izquierda de este rango la función toma valores de 1, y a la derecha de 0, haciendo posible una extensión trivial del dominio hacia ambos lados.

Esta extensión se la realiza para poder representar gráficamente la PER de cada configuración junto a las respuestas de los modelos teóricos, en donde ciertos modelos teóricos pueden presentar valores fuera del rango angosto de simulación.

El diagrama de flujo se puede apreciar en la Figura I.43 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 42 (Anexo II).

2.4.4. Minimización del error y evaluación de modelos teóricos

Para la obtención de los parámetros de minimización de todos los modelos teóricos en todas las configuraciones en comparación a los resultados simulados respectivos, se crea el *script* COMPARACION_PER.

Además, este *script* se encarga de obtener los valores de la PER para los modelos teóricos considerando el *offset* de SNR, teniendo así los valores de la PER minimizados por cada modelo para cada configuración con respecto a los valores simulados. El *script* realiza las siguientes tareas:

- Obtener el error mínimo y el *offset* de SNR asociado a este error de cada modelo teórico, que resultan del proceso de minimización de dichos modelos con respecto a los resultados simulados por SNR angosto.
- Obtener los valores de la PER para cada modelo teórico, considerando el *offset* resultante del proceso de minimización.
- Guardar los datos de *offsets*, errores y valores de PER con nombres apropiados, en archivos .mat.

Los parámetros de minimización son el *offset* de SNR y el error mínimo para cada modelo, por cada configuración.

Para el proceso de optimización (minimización del error) es necesario obtener un *offset* inicial para cada proceso de minimización, es decir, para cada modelo teórico y a su vez para cada configuración. Debido a que varios modelos teóricos pueden presentar comportamientos distintos, se crea una matriz **C0** de *offsets* iniciales, la cual debe ser definida mediante un proceso inicial de experimentación con cada par modelo teórico / configuración.

El diagrama de flujo se puede apreciar en la Figura I.44 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 43 (Anexo II).

2.4.5. Consolidación de datos

Para la recolección y consolidación de datos resultantes de la ejecución de *scripts* anteriores en un solo archivo para una más práctica utilización, se crea el *script* CONSOLIDACION_DATOS que realiza las siguientes tareas:

- Importar los archivos de datos generados por los anteriores *scripts* de análisis de

datos.

- Nombrar los datos y vectores de una manera estructurada, para su fácil manipulación.
- Guardar todos los datos estructurados en un solo archivo .mat.

Debido a que estos son los datos que se utilizarán para el análisis, a continuación se especifican los nombres y contenidos de los datos en el archivo DATOS_ CONSOLIDADOS (se considera como “modo” a: SNR angosto, SNR homogéneo y SNR angosto extendido):

- **NUMBIT_G**: Número (escalar) de bits por paquete para todas las configuraciones y modos.
- **NUMPAQ_G**: Número (escalar) de paquetes transmitidos para todas las configuraciones y modos.
- **PARAMETERS_G**: Matriz de parámetros de cada configuración de transmisión de acuerdo a la Tabla 1.2.
- **C_MIN**: Matriz de *offsets* por errores minimizados por cada configuración, por cada modelo teórico.
- **E_MIN**: Matriz de errores minimizados por cada configuración, por cada modelo teórico.
- **SNUM_FIN**: Número (escalar) de muestras para todas las configuraciones para el modo de SNR angosto.
- **SNRV_FIN**: Matriz de límites inferiores y superiores de la SNR por cada configuración para el modo de SNR angosto.
- **SNRV_FIN_EXT**: Matriz de límites inferiores y superiores de la SNR por cada configuración para el modo de SNR angosto extendido.
- **SNUM_FIN_EXT**: Número (escalar) de muestras para todas las configuraciones para el modo de SNR angosto extendido.
- **SNR_HOM**: Rango (vector) de SNR para todas las configuraciones para el modo de SNR homogéneo.
- **PER_SIM_HOM**: Matriz de valores de la PER obtenidos mediante simulación por configuración para el modo de SNR homogéneo.
- **PER_SIM_FIN**: Matriz de valores de la PER obtenidos mediante simulación por configuración para el modo de SNR angosto.
- **PER_SIM_FIN_EXT**: Matriz de valores de la PER obtenidos mediante simulación

por configuración para el modo de SNR angosto extendido.

- **PER_THEO_FIN_EXT**: Matriz de valores de la PER obtenidos por configuración de todos los modelos teóricos concatenados para el modo de SNR angosto extendido.
- **PER_THEO_FIN_EXT_CONF1**: Matriz de valores de la PER obtenidos para la configuración $BPSK$, $r = 1/2$ para todos los modelos teóricos para el modo de SNR angosto extendido.
- **PER_THEO_FIN_EXT_CONF2**: Matriz de valores de la PER obtenidos para la configuración $BPSK$, $r = 3/4$ para todos los modelos teóricos para el modo de SNR angosto extendido.
- **PER_THEO_FIN_EXT_CONF3**: Matriz de valores de la PER obtenidos para la configuración $QPSK$, $r = 1/2$ para todos los modelos teóricos para el modo de SNR angosto extendido.
- **PER_THEO_FIN_EXT_CONF4**: Matriz de valores de la PER obtenidos para la configuración $QPSK$, $r = 3/4$ para todos los modelos teóricos para el modo de SNR angosto extendido.
- **PER_THEO_FIN_EXT_CONF5**: Matriz de valores de la PER obtenidos para la configuración $16QAM$, $r = 1/2$ para todos los modelos teóricos para el modo de SNR angosto extendido.
- **PER_THEO_FIN_EXT_CONF6**: Matriz de valores de la PER obtenidos para la configuración $16QAM$, $r = 3/4$ para todos los modelos teóricos para el modo de SNR angosto extendido.
- **PER_THEO_FIN_EXT_CONF7**: Matriz de valores de la PER obtenidos para la configuración $64QAM$, $r = 2/3$ para todos los modelos teóricos para el modo de SNR angosto extendido.
- **PER_THEO_FIN_EXT_CONF8**: Matriz de valores de la PER obtenidos para la configuración $64QAM$, $r = 3/4$ para todos los modelos teóricos para el modo de SNR angosto extendido.
- **PER_THEO_HOM**: Matriz de valores de la PER obtenidos por configuración de todos los modelos teóricos concatenados para el modo de SNR homogéneo.
- **PER_THEO_HOM_CONF1**: Matriz de valores de la PER obtenidos para la configuración $BPSK$, $r = 1/2$ para todos los modelos teóricos para el modo de SNR

homogéneo.

- **PER_THEO_HOM_CONF2**: Matriz de valores de la PER obtenidos para la configuración $BPSK$, $r = 3/4$ para todos los modelos teóricos para el modo de SNR homogéneo.
- **PER_THEO_HOM_CONF3**: Matriz de valores de la PER obtenidos para la configuración $QPSK$, $r = 1/2$ para todos los modelos teóricos para el modo de SNR homogéneo.
- **PER_THEO_HOM_CONF4**: Matriz de valores de la PER obtenidos para la configuración $QPSK$, $r = 3/4$ para todos los modelos teóricos para el modo de SNR homogéneo.
- **PER_THEO_HOM_CONF5**: Matriz de valores de la PER obtenidos para la configuración $16QAM$, $r = 1/2$ para todos los modelos teóricos para el modo de SNR homogéneo.
- **PER_THEO_HOM_CONF6**: Matriz de valores de la PER obtenidos para la configuración $16QAM$, $r = 3/4$ para todos los modelos teóricos para el modo de SNR homogéneo.
- **PER_THEO_HOM_CONF7**: Matriz de valores de la PER obtenidos para la configuración $64QAM$, $r = 2/3$ para todos los modelos teóricos para el modo de SNR homogéneo.
- **PER_THEO_HOM_CONF8**: Matriz de valores de la PER obtenidos para la configuración $64QAM$, $r = 3/4$ para todos los modelos teóricos para el modo de SNR homogéneo.
- **PER_THEO_FIN_MIN**: Matriz de valores de la PER obtenidos por configuración de todos los modelos teóricos minimizados concatenados para el modo de SNR angosto.
- **PER_THEO_FIN_MIN_CONF1**: Matriz de valores de la PER obtenidos para la configuración BPSK, $r = 1/2$ para todos los modelos teóricos minimizados para el modo de SNR angosto.
- **PER_THEO_FIN_MIN_CONF2**: Matriz de valores de la PER obtenidos para la configuración BPSK, $r = 3/4$ para todos los modelos teóricos minimizados para el modo de SNR angosto.

- **PER_THEO_FIN_MIN_CONF3**: Matriz de valores de la PER obtenidos para la configuración QPSK, $r = {}^{1}/_{2}$ para todos los modelos teóricos minimizados para el modo de SNR angosto.
- **PER_THEO_FIN_MIN_CONF4**: Matriz de valores de la PER obtenidos para la configuración QPSK, $r = {}^{3}/_{4}$ para todos los modelos teóricos minimizados para el modo de SNR angosto.
- **PER_THEO_FIN_MIN_CONF5**: Matriz de valores de la PER obtenidos para la configuración 16QAM, $r = {}^{1}/_{2}$ para todos los modelos teóricos minimizados para el modo de SNR angosto.
- **PER_THEO_FIN_MIN_CONF6**: Matriz de valores de la PER obtenidos para la configuración 16QAM, $r = {}^{3}/_{4}$ para todos los modelos teóricos minimizados para el modo de SNR angosto.
- **PER_THEO_FIN_MIN_CONF7**: Matriz de valores de la PER obtenidos para la configuración 64QAM, $r = {}^{2}/_{3}$ para todos los modelos teóricos minimizados para el modo de SNR angosto.
- **PER_THEO_FIN_MIN_CONF8**: Matriz de valores de la PER obtenidos para la configuración 64QAM, $r = {}^{3}/_{4}$ para todos los modelos teóricos minimizados para el modo de SNR angosto.

El diagrama de flujo se puede apreciar en la Figura I.45 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 44 (Anexo II).

2.4.6. Presentación de resultados

Para la presentación de los resultados en gráficas de PER vs. SNR se crea el *script* GRAFICACION_DATOS. Cada gráfica de PER vs. SNR será acompañada de una gráfica semilogarítmica correspondiente, siendo el eje logarítmico el eje de las ordenadas que representa la PER. El *script* realiza las siguientes tareas:

- Cargar los archivos de datos generados por los anteriores *scripts* de análisis de datos.
- Presentar los datos de la PER en función de la SNR en distintas gráficas.

El diagrama de flujo se puede apreciar en la Figura I.46 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 45 (Anexo II).

2.4.7. Análisis de resultados

Para la presentación textual del análisis entre modelos teóricos con respecto a los resultados simulados de la PER para cada configuración, se crea el *script* ANALISIS_RESULTADOS que realiza las siguientes tareas:

- Determinar cuál modelo teórico presenta el menor *offset* (en módulo) de SNR para obtener el error mínimo respectivo y guardarlos en variables.
- Determinar cuál modelo teórico presenta el menor error minimizado (en módulo) y guardar este error, el *offset* respectivo y el número de modelo correspondiente en variables.
- Mostrar en pantalla un resumen de los resultados obtenidos.

El diagrama de flujo se puede apreciar en la Figura I.47 (Anexo I), mientras que el algoritmo se observa en el Algoritmo 46 (Anexo II).

2.5. Plan de ejecución de *scripts*

Para la obtención, tratamiento y presentación de resultados se seguirá el siguiente plan de ejecución de *scripts*:

1. SIMULACION
2. SIMULACION_ANGOSTO
3. ANALISIS_PER
4. EXTENSION_PER
5. COMPARACION_PER
6. CONSOLIDACION_DATOS
7. GRAFICACION_RESULTADOS
8. ANALISIS_RESULTADOS

Para poder obtener resultados de una manera ordenada, cada uno de estos *scripts* guarda información en archivos externos que serán adoptados por los subsecuentes. Estos archivos se guardarán en la misma ubicación de los *scripts*.

2.6. Implementación y ejecución de *scripts*

Los *scripts* y funciones serán implementados en MATLAB de acuerdo a los diagramas de flujo y algoritmos expuestos, para guardarse en archivos .m en una misma carpeta y luego ejecutarse de acuerdo al plan de ejecución expuesto en la sección 1.3.7. Los resultados de la ejecución de los *scripts* se guardan en archivos .mat que pueden ser importados en MATLAB.

Los resultados, junto a las funciones y *scripts* se encuentran adjuntos en el CD Anexo III.

3. RESULTADOS Y DISCUSIÓN

Se presentan los resultados obtenidos de las simulaciones del sistema de comunicación a nivel de capa física IEEE 802.11p y de los modelos del cálculo de la PER (se considera una longitud de paquete fija de 4000 bits). Los resultados se presentan por medio de curvas de la PER vs. SNR. Se presentan inicialmente los resultados de PER vs. SNR para cada combinación de tasa de codificación y tipo de modulación; y, luego se presentan los resultados teóricos de cada modelo de la PER analizado por cada configuración.

3.1. Resultados de PER de la simulación

En esta sección se presentan los resultados de PER vs. SNR para cada combinación de tasa de codificación (r) y esquema de modulación (M) permitidos en el estándar (ver Tabla 1.2). Estos resultados sirven como punto de comparación para los modelos teóricos del cálculo de la PER. Así, en la Figura 3.1 se presentan los resultados en una escala lineal, mientras que en la Figura 3.2 se observan en escala semilogarítmica.

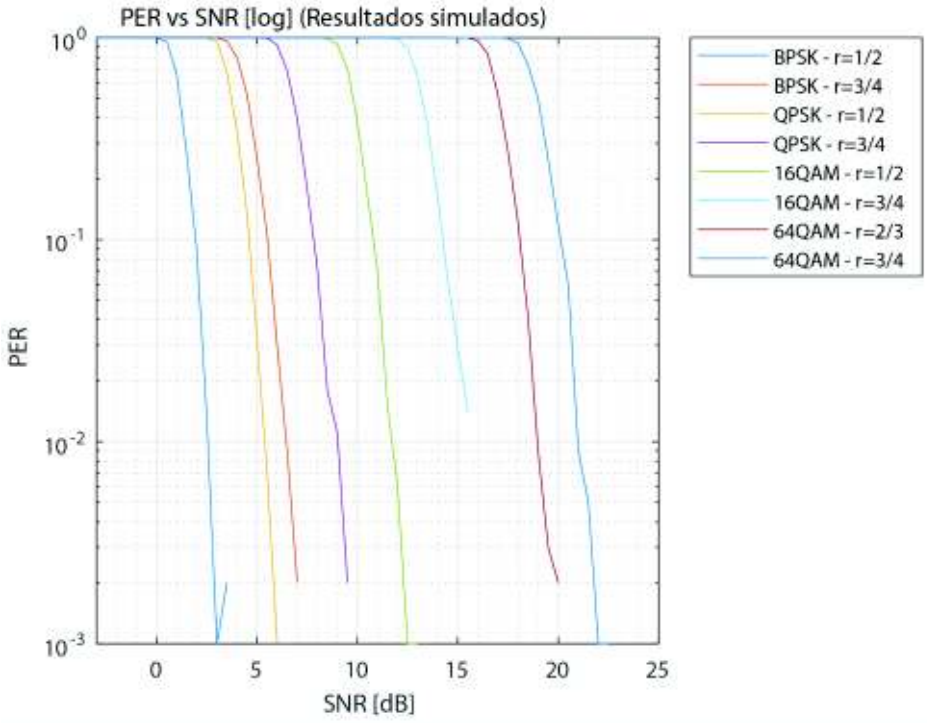


Figura 3.1: Resultados simulados de la PER para todas las configuraciones

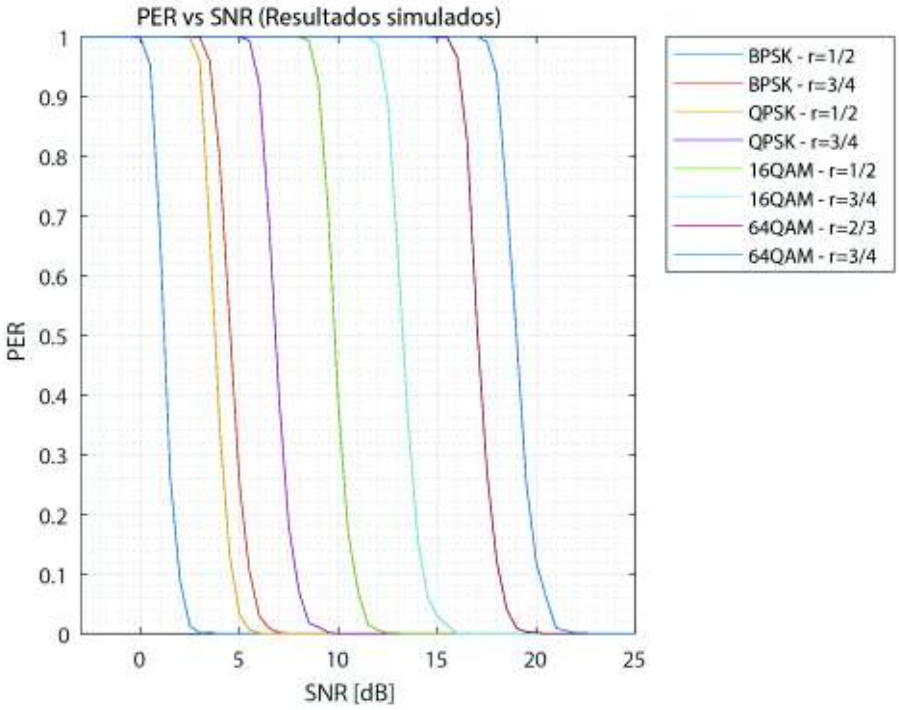


Figura 3.2: Resultados simulados de la PER para todas las configuraciones (semilogarítmico)

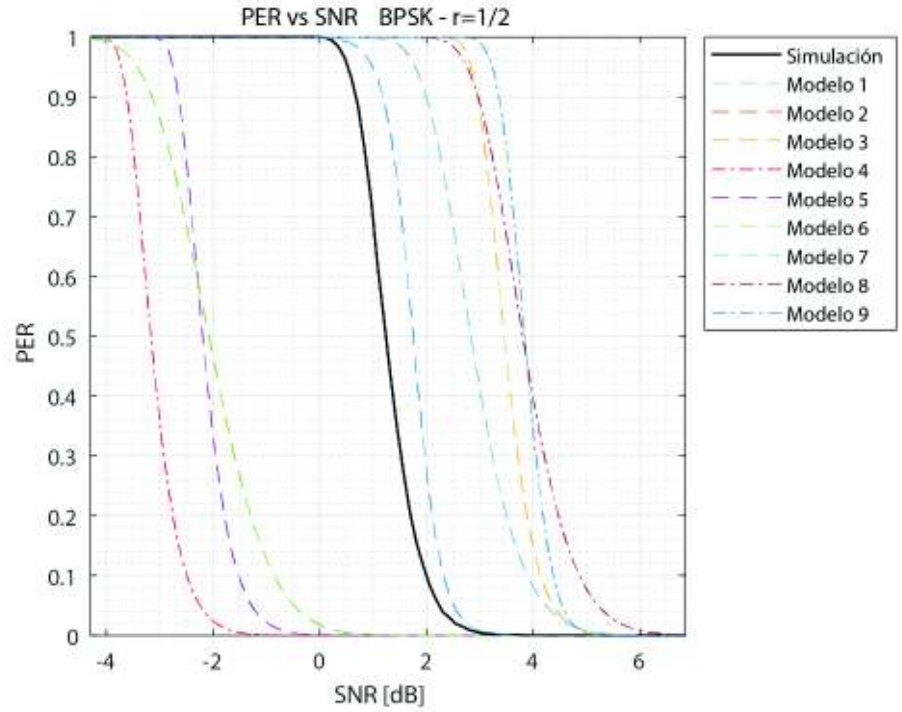
En estas figuras se observa que el comportamiento de la PER es el esperado, ya que para menor tasa de codificación y esquema de modulación la PER es mejor debido a que la comunicación es más robusta. La ubicación de la curva de QPSK $r = {}^{1}/_{2}$ a la izquierda de BPSK $r = {}^{3}/_{4}$ a diferencia de las otras curvas se explica con el hecho de que, para QPSK se modula un bit más por símbolo que para BPSK, mientras que de 16QAM a QPSK el incremento es de 2 bits así como de 64QAM a 16QAM, siendo la pérdida de robustez menor entre BPSK y QPSK, lo cual se observa como curvas más cercanas entre sí.

3.2. Comparación de modelos teóricos

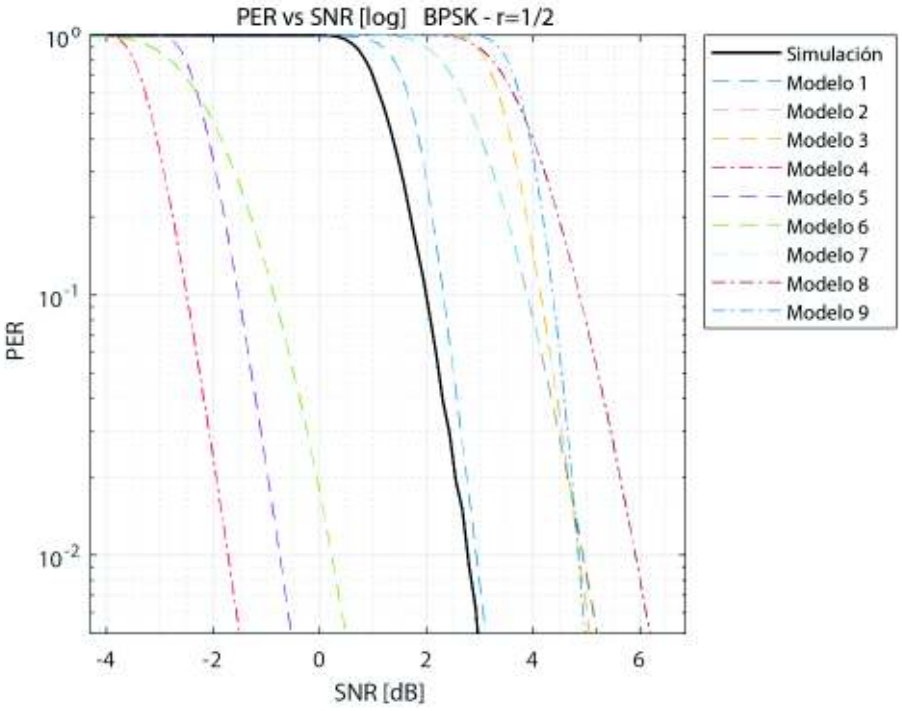
Las siguientes figuras presentan los resultados de la PER vs. SNR. Se presenta dos gráficas por cada configuración, la una en escala lineal y la otra en escala semilogarítmica (en el eje y). En cada gráfica se presentan las curvas de cada modelo y se comparan con los datos de la PER simulados correspondientes. Así, un resumen de las figuras se presenta en la Tabla 3.1.

Tabla 3.1: Figuras de la comparación de modelos teóricos de la PER

Modulación	Tasa de codificación (r)	Escala	Figura
BPSK	$1/2$	Lineal	3.3 **(a)**
BPSK	$1/2$	Semilogarítmica	3.3 **(b)**
BPSK	$3/4$	Lineal	3.4 **(a)**
BPSK	$3/4$	Semilogarítmica	3.4 **(b)**
QPSK	$1/2$	Lineal	3.5 **(a)**
QPSK	$1/2$	Semilogarítmica	3.5 **(b)**
QPSK	$3/4$	Lineal	3.6 **(a)**
QPSK	$3/4$	Semilogarítmica	3.6 **(b)**
16QAM	$1/2$	Lineal	3.7 **(a)**
16QAM	$1/2$	Semilogarítmica	3.7 **(b)**
16QAM	$3/4$	Lineal	3.8 **(a)**
16QAM	$3/4$	Semilogarítmica	3.8 **(b)**
64QAM	$2/3$	Lineal	3.9 **(a)**
64QAM	$2/3$	Semilogarítmica	3.9 **(b)**
64QAM	$3/4$	Lineal	3.10 **(a)**
64QAM	$3/4$	Semilogarítmica	3.10 **(b)**

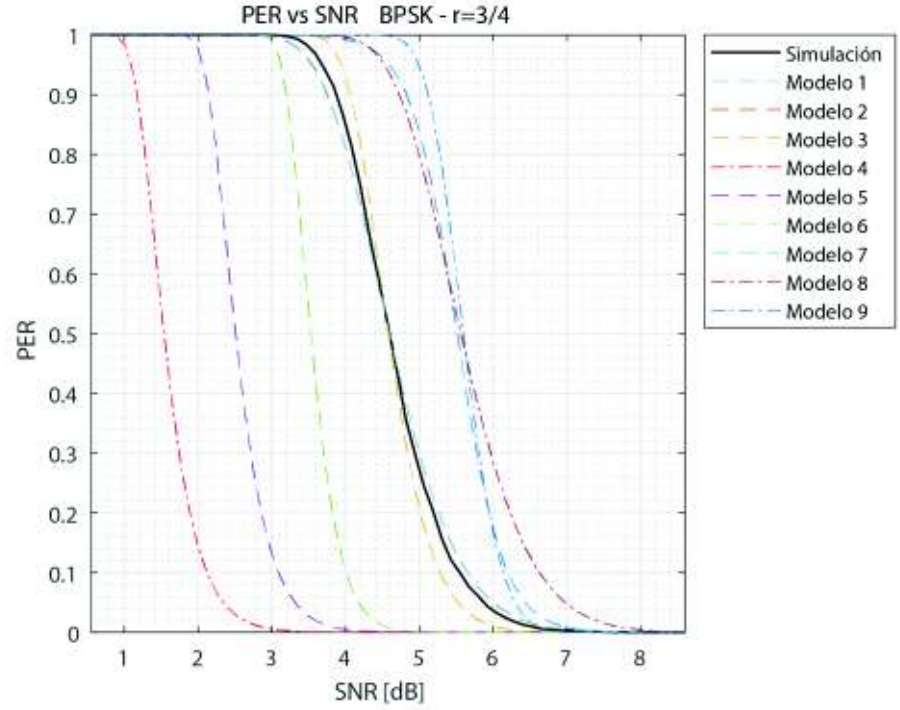


(a) Lineal

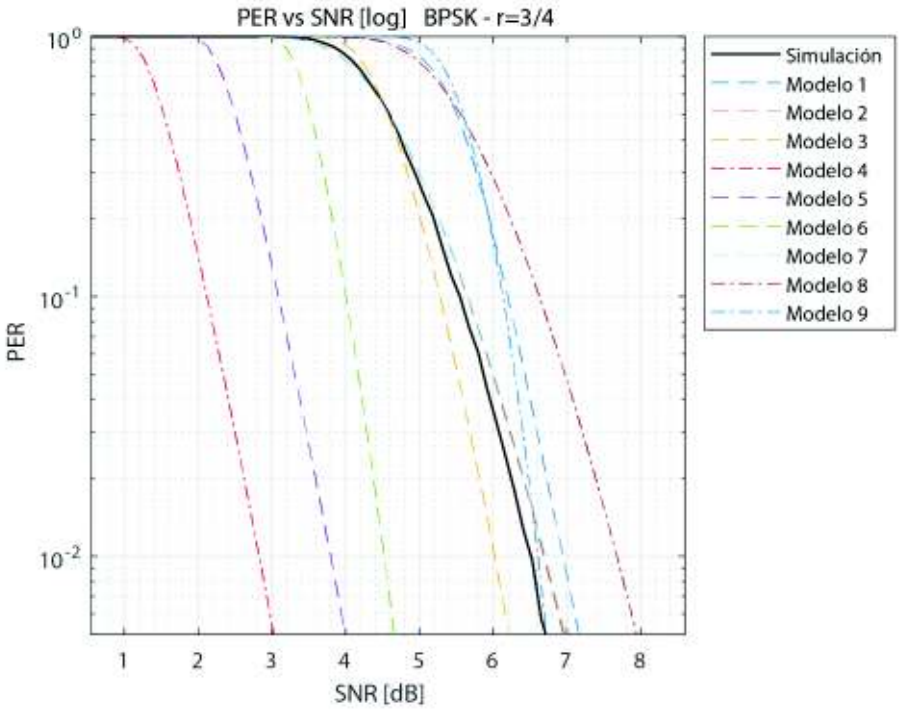


(b) Semilogarítmica

Figura 3.3: Comparación de modelos teóricos con resultados simulados para BPSK r=$^1/_2$ (**(a)** Escala lineal, **(b)** Escala semilogarítmica)

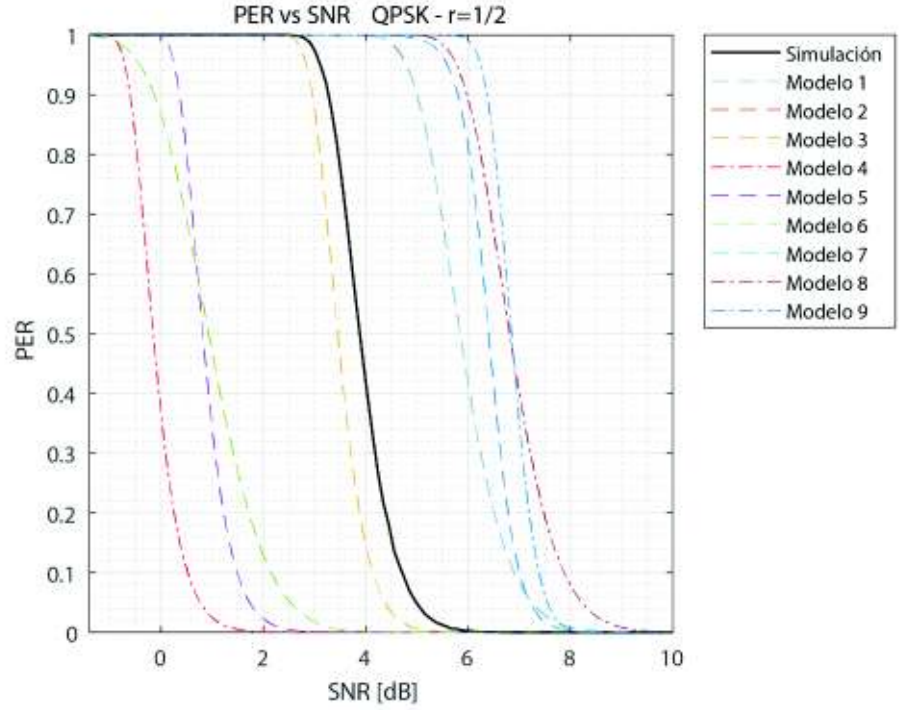


(a) Lineal

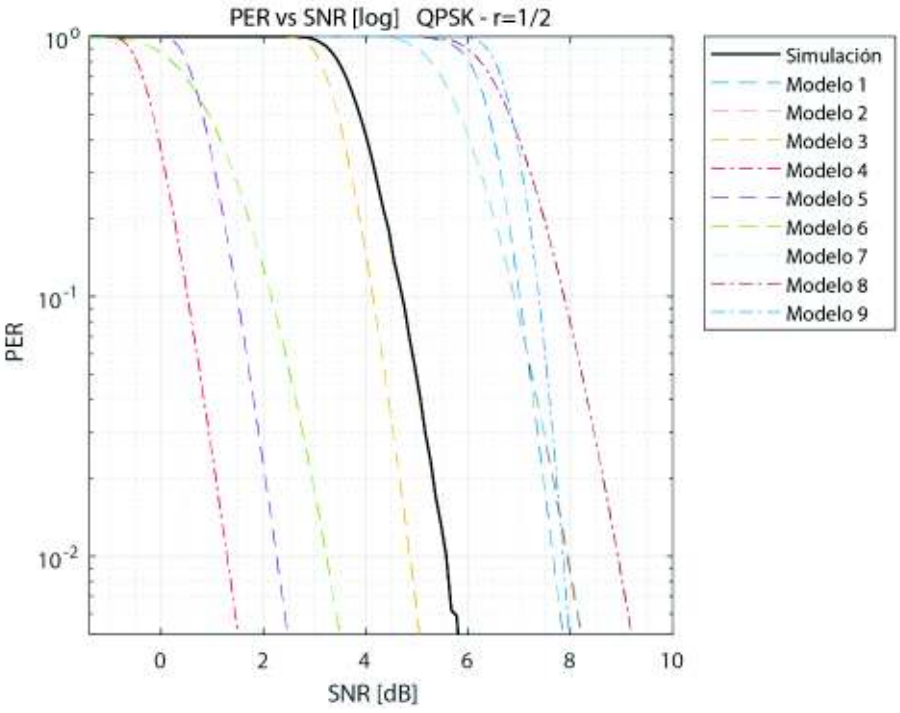


(b) Semilogarítmica

Figura 3.4: Comparación de modelos teóricos con resultados simulados para BPSK r=$^{3}/_{4}$ (**(a)** Escala lineal, **(b)** Escala semilogarítmica)

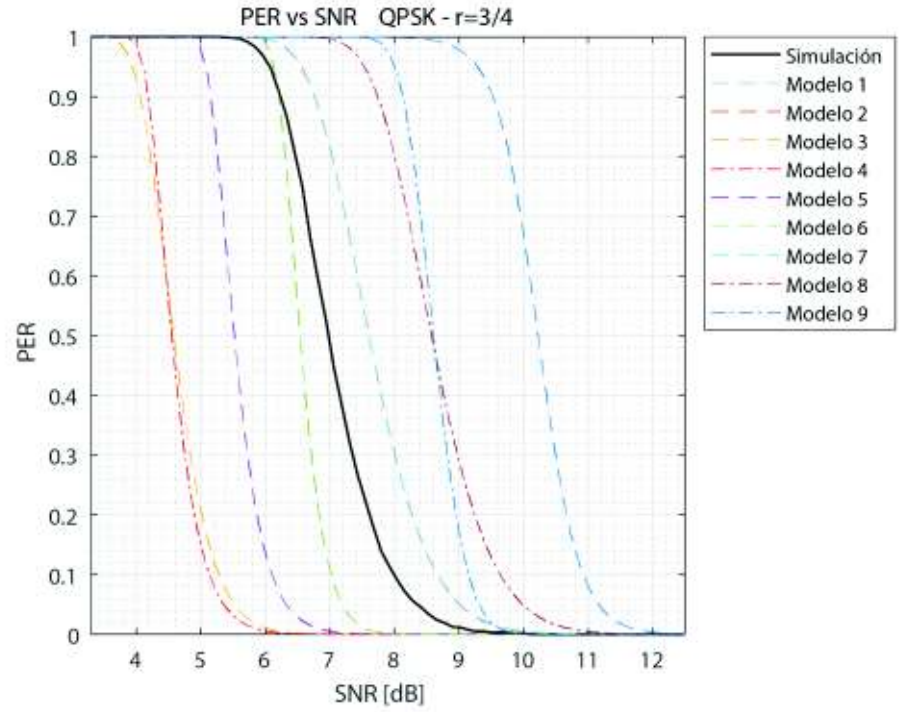


(a) Lineal

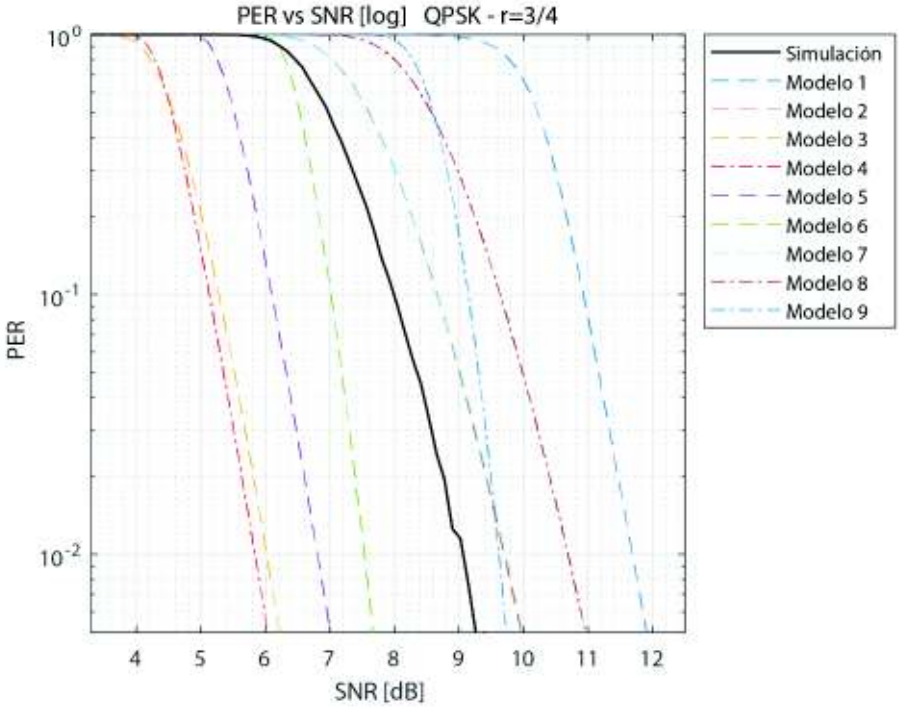


(b) Semilogarítmica

Figura 3.5: Comparación de modelos teóricos con resultados simulados para QPSK r=$^1/_2$ (**(a)** Escala lineal, **(b)** Escala semilogarítmica)

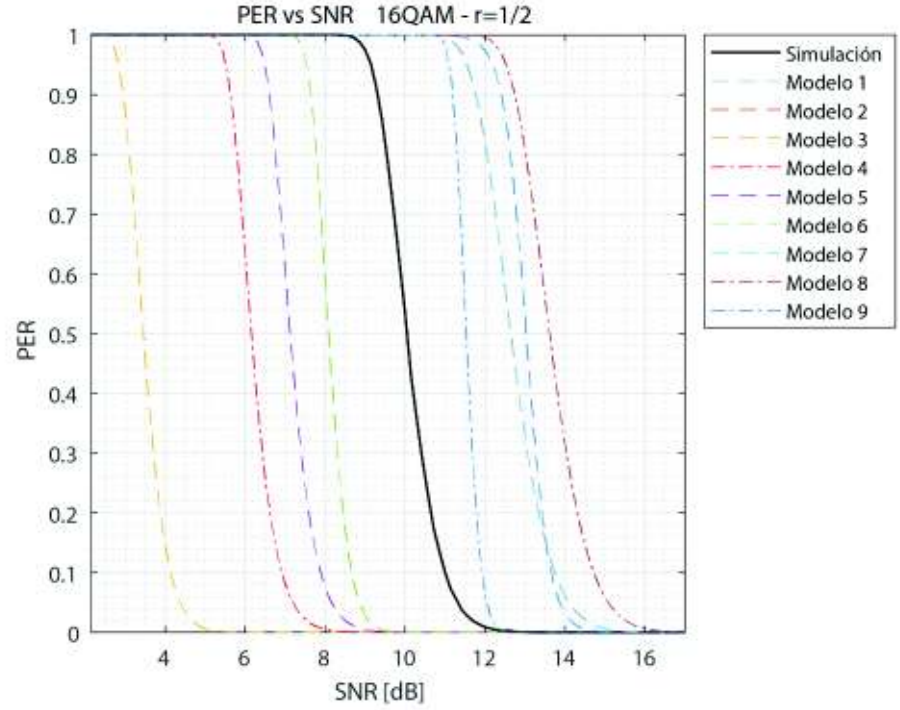


(a) Lineal

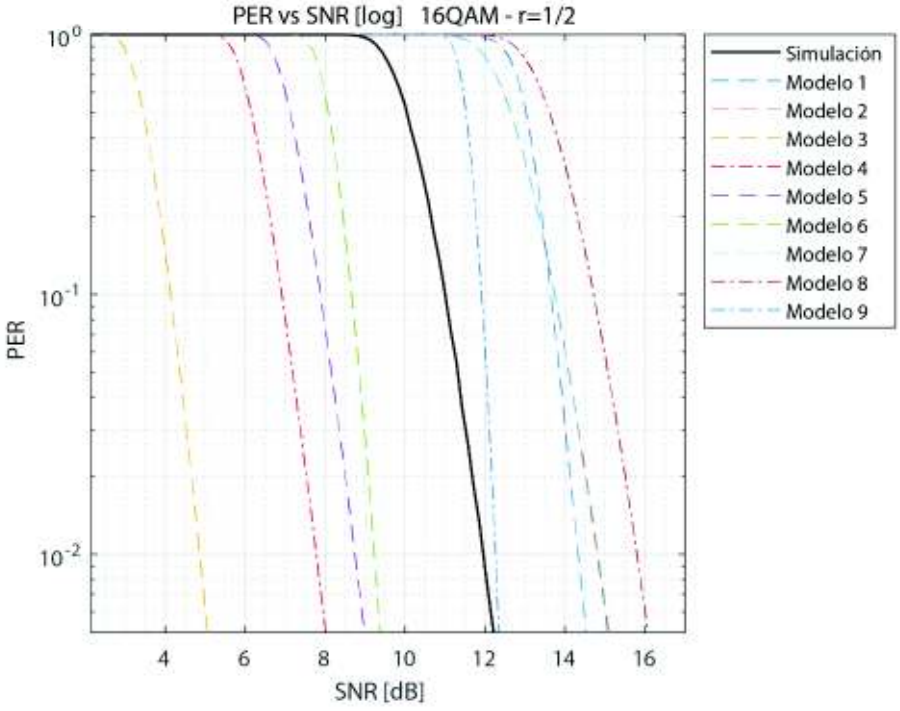


(b) Semilogarítmica

Figura 3.6: Comparación de modelos teóricos con resultados simulados para QPSK r=$^{3}/_{4}$ (**(a)** Escala lineal, **(b)** Escala semilogarítmica)

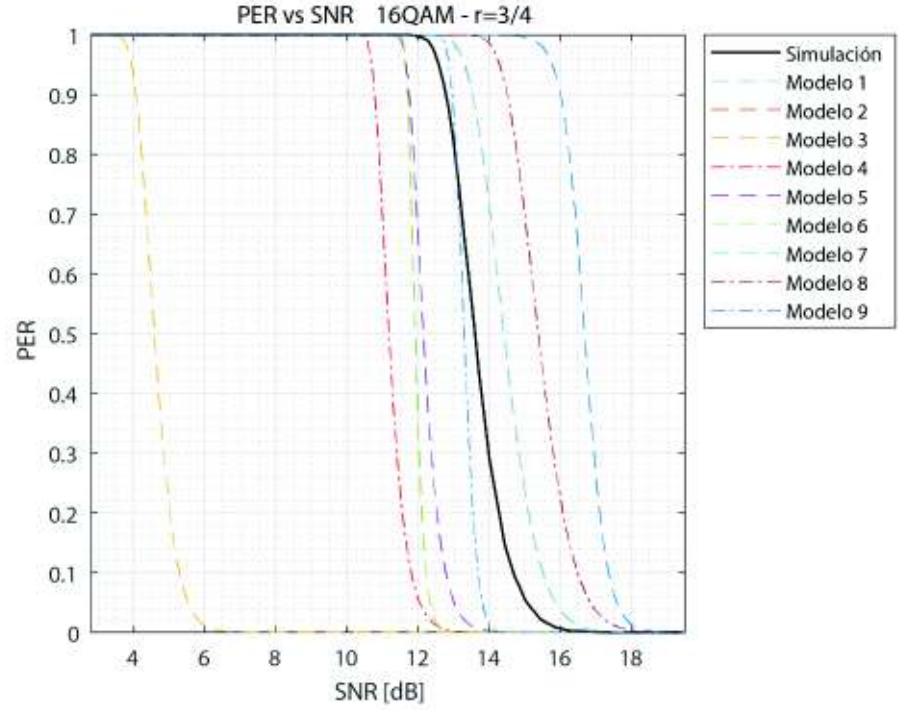


(a) Lineal

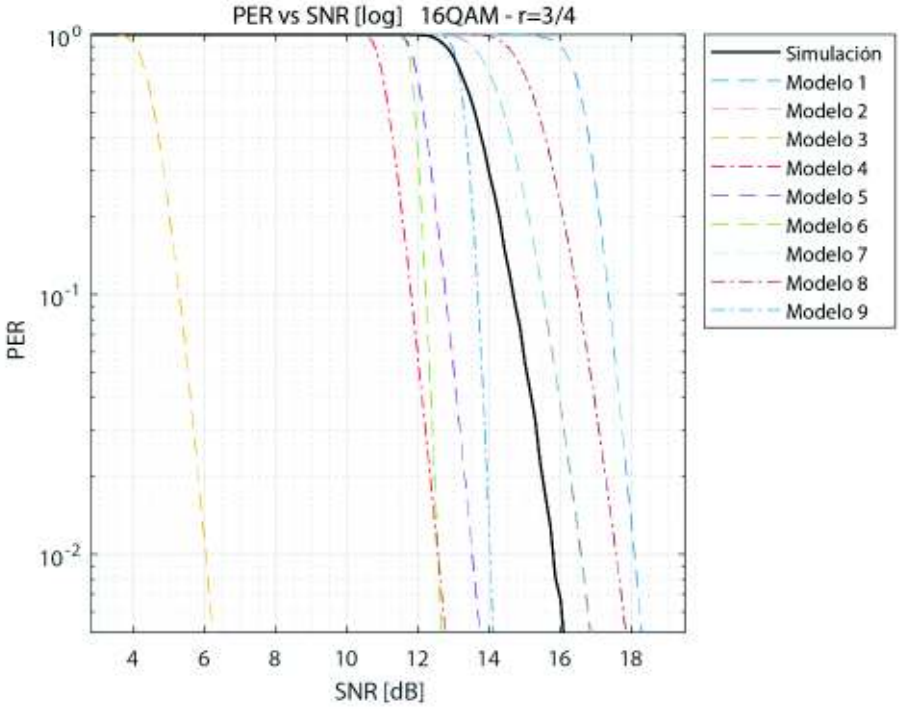


(b) Semilogarítmica

Figura 3.7: Comparación de modelos teóricos con resultados simulados para 16QAM r=$^1/_2$ (**(a)** Escala lineal, **(b)** Escala semilogarítmica)

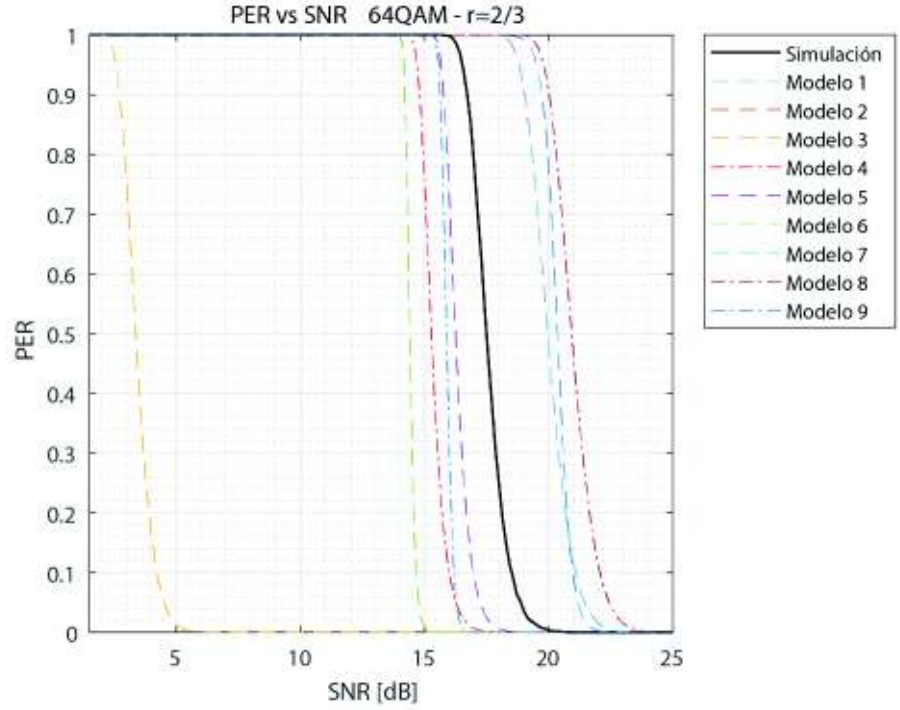


(a) Lineal

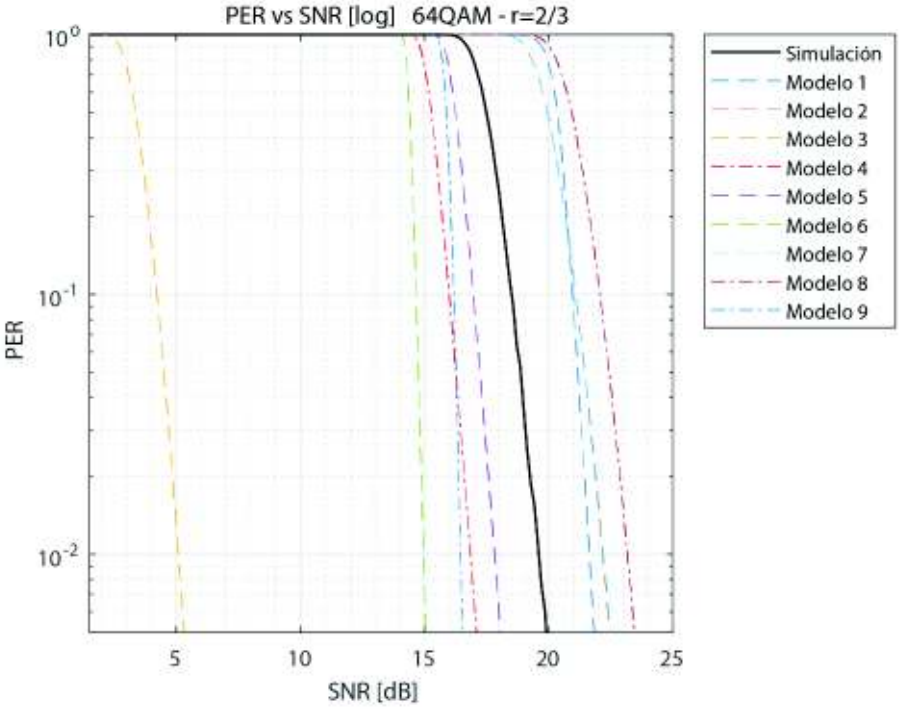


(b) Semilogarítmica

Figura 3.8: Comparación de modelos teóricos con resultados simulados para 16QAM r=$^{3}/_{4}$ (**(a)** Escala lineal, **(b)** Escala semilogarítmica)

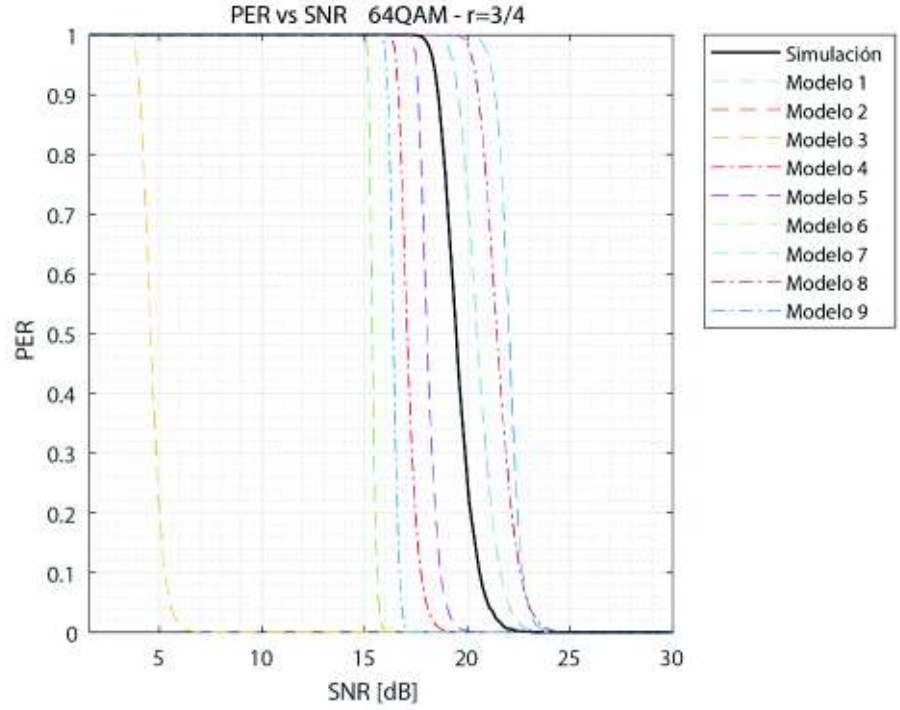


(a) Lineal

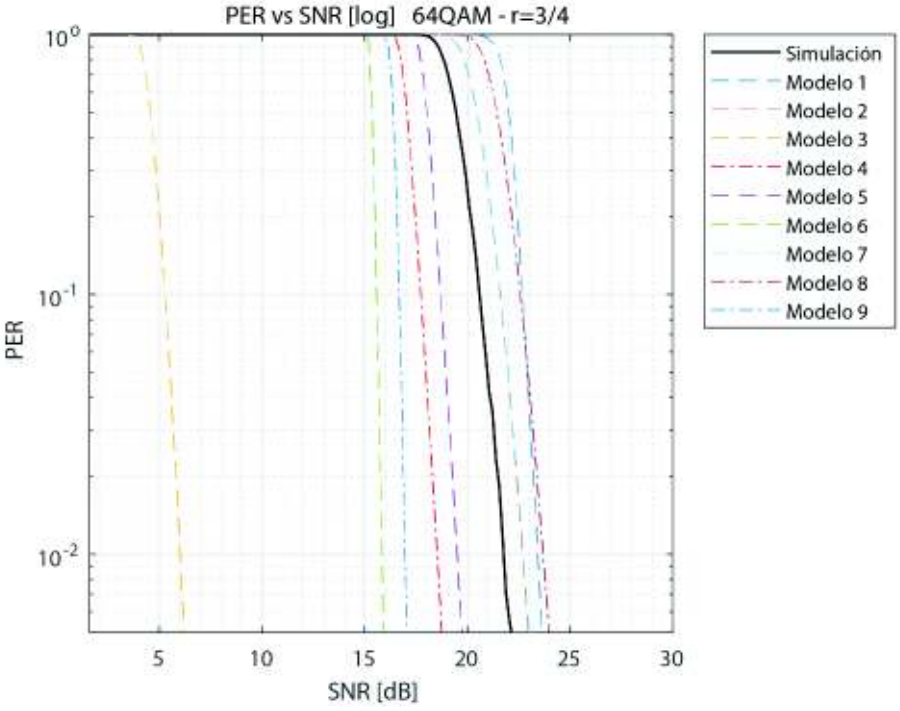


(b) Semilogarítmica

Figura 3.9: Comparación de modelos teóricos con resultados simulados para 64QAM $r=^{2}/_{3}$ (**(a)** Escala lineal, **(b)** Escala semilogarítmica)



(a) Lineal



(b) Semilogarítmica

Figura 3.10: Comparación de modelos teóricos con resultados simulados para 64QAM r=$^{3}/_{4}$ (**(a)** Escala lineal, **(b)** Escala semilogarítmica)

3.2.1. Análisis cualitativo

Se considerará cada una de las configuraciones y de qué manera se comporta cada uno de los modelos teóricos en tanto la PER con respecto a los resultados simulados.

El análisis se enfoca en la distancia percibida entre una curva teórica y la simulada correspondiente, así como en la similitud entre sus formas. Se toma en cuenta la pendiente y la concavidad de las curvas teóricas con respecto a las simuladas.

a. BPSK $r = {}^{1}/_{2}$

- **Modelo 1:** es el modelo que se aproxima más a los resultados simulados, teniendo un *offset* hacia la derecha inferior en módulo al resto de modelos; presenta una forma cercana a la de los datos simulados con una pendiente similar, pero no tanto como los modelos 3, 4 y 5.
- **Modelo 2:** se aproxima a los datos simulados bastante bien para los valores más pequeños de SNR, pero para mayores valores diverge; presenta una forma cercana a la de los datos simulados con una pendiente similar, pero es más abierta y diverge de los datos rápidamente junto a los modelos 7 y 8.
- **Modelo 3:** se encuentra relativamente alejada de los datos simulados; presenta una forma muy cercana a la de los datos simulados con una pendiente muy similar, siendo la tercera más cercana detrás de los modelos 4 y 5.
- **Modelo 4:** se encuentra muy alejada de los datos simulados; presenta una forma muy cercana a la de los datos simulados con una pendiente muy similar, siendo la más cercana junto al modelo 5.
- **Modelo 5:** se encuentra muy alejada de los datos simulados; presenta una forma muy cercana a la de los datos simulados con una pendiente muy similar, siendo la más cercana junto al modelo 4.
- **Modelo 6:** se encuentra alejada de los datos simulados, acortando distancia a medida que aumenta la SNR; presenta una forma distinta a la de los datos simulados con una pendiente muy alejada, pero una tendencia similar.
- **Modelo 7:** se aproxima a los datos simulados bastante bien para los valores más pequeños de SNR, pero para mayores valores diverge; presenta una forma cercana a la de los datos simulados con una pendiente similar, pero es más abierta y diverge de los datos rápidamente junto a los modelos 2 y 8.
- **Modelo 8:** se encuentra muy alejada de los datos simulados; presenta una forma

cercana a la de los datos simulados con una pendiente similar, pero es más abierta y diverge de los datos rápidamente junto a los modelos 2 y 7.

- **Modelo 9:** se encuentra alejada de los datos simulados, aproximándose más a medida que aumenta la SNR; presenta una forma distinta a la de los datos simulados con una pendiente muy alejada, y una tendencia distinta.

El modelo que más se acerca es el modelo 1, sin embargo hay mejores candidatos en tanto la forma de la curva, puntualmente los modelos 4 y 5. Los modelos que pueden considerarse viables por su forma son los modelos 3, 4 y 5. El modelo 9 se abre mucho hacia abajo, mientras que los modelos 2, 6, 7 y 8 se abren mucho hacia arriba. Por otro lado, todos los modelos excepto el 1 y el 7 están demasiado alejados de los datos simulados.

b. BPSK $r = {}^{3}/_{4}$

- **Modelo 1:** relativamente alejado a valores simulados, pero acercándose a mayores valores de SNR; presenta una forma similar pero de pendiente notablemente más pronunciada.
- **Modelo 2:** se acerca mucho a los valores simulados, ligeramente menos que el modelo 7; presenta una forma muy similar de muy cercana pendiente, al igual que los modelos 7 y 8.
- **Modelo 3:** está relativamente cercano a los datos simulados, pero se aleja a mayor SNR; presenta una forma similar pero de pendiente notablemente más pronunciada.
- **Modelo 4:** se encuentra muy alejada de los datos simulados; presenta una forma similar pero de pendiente notablemente más pronunciada, muy cercana al modelo 5.
- **Modelo 5:** se encuentra muy alejada de los datos simulados; presenta una forma similar pero de pendiente notablemente más pronunciada, muy cercana al modelo 4.
- **Modelo 6:** se encuentra alejada de los datos simulados, aumentando la distancia a medida que aumenta la SNR; presenta una forma distinta, más cóncava.
- **Modelo 7:** se acerca mucho a los valores simulados; presenta una forma muy similar de muy cercana pendiente, al igual que los modelos 2 y 8.
- **Modelo 8:** está relativamente alejada de los valores simulados, alejándose más a mayor SNR; presenta una forma muy similar de muy cercana pendiente, al igual que los modelos 2 y 7.

- **Modelo 9:** se encuentra relativamente alejada de los datos simulados, aproximándose más a medida que aumenta la SNR, hasta cortar la curva de datos simulados; presenta una forma distinta, más cóncava.

A simple vista, hay tres modelos con una suficiente cercanía: los modelos 2, 3 y 7. Aunque, si se ve la gráfica semilogarítmica son los modelos 2 y 7 los que presentan una mayor y más consistente cercanía a los datos simulados. Los modelos 1 y 8 se encuentran relativamente cerca pero no lo suficiente, y se acercan de manera irregular con respecto a los otros cercanos. Por otro lado los modelos 2, 7 y 8 son los que más se acercan a la forma de los datos simulados, siguiéndoles los modelos 1, 3, 4 y 5 con mayores pendientes, y finalmente los modelos 6 y 9 que presentan pendientes muy grandes.

c. QPSK $r = {}^{1}/_{2}$

- **Modelo 1:** relativamente alejado a valores simulados, acercándose a mayores valores de SNR; presenta una forma similar pero de pendiente más pronunciada.
- **Modelo 2:** alejado a los valores simulados, similar al modelo 7; presenta forma similar de pendiente menos pronunciada, al igual que los modelos 7 y 8.
- **Modelo 3:** modelo más cercano a los datos simulados, pero se aleja a mayor SNR; presenta una forma similar de pendiente más pronunciada, pero más cercana que el modelo 1.
- **Modelo 4:** se encuentra muy alejado de los datos simulados; presenta una forma muy similar de pendiente muy similar, muy cercana al modelo 5.
- **Modelo 5:** se encuentra muy alejado de los datos simulados; presenta una forma muy similar de pendiente muy similar, muy cercana al modelo 4.
- **Modelo 6:** se encuentra alejado de los datos simulados, disminuyendo ligeramente la distancia a medida que aumenta la SNR; presenta una forma distinta, ligeramente más cóncava y de menor pendiente.
- **Modelo 7:** alejado a los valores simulados, similar al modelo 2; presenta forma similar de pendiente menos pronunciada, al igual que los modelos 2 y 8.
- **Modelo 8:** se encuentra muy alejado de los datos simulados; presenta forma similar de pendiente menos pronunciada, al igual que los modelos 2 y 7.
- **Modelo 9:** se encuentra alejado de los datos simulados, aproximándose más a medida que aumenta la SNR; presenta una forma muy distinta, considerablemente más cóncava.

El modelo que más se acerca a los datos simulados es el modelo 3. El resto se encuentran a una distancia considerable, siendo el 6 el menos lejano de estos. Ninguno de los modelos tiene una fuerte coincidencia en la forma de la curva simulada, pero se tiene como candidato a los modelos 1, 3, 4 y 5. Los modelos 2, 6, 7 y 8 presentan una menor pendiente, mientras que el modelo 9 presenta una mayor pendiente.

d. QPSK $r = {}^{3}/_{4}$

- **Modelo 1:** está muy alejado a valores simulados de manera uniforme; presenta una forma similar pero de pendiente más pronunciada.
- **Modelo 2:** está cercano a valores simulados de manera uniforme y ligeramente más que el modelo 7; presenta una forma muy similar de muy cercana pendiente, al igual que los modelos 7 y 8.
- **Modelo 3:** está considerablemente alejado de los datos simulados; presenta una forma similar pero de pendiente notablemente más pronunciada.
- **Modelo 4:** está considerablemente alejado de los datos simulados, ligeramente más que el modelo 3; presenta una forma similar pero de pendiente más pronunciada y más cóncavo, muy cercana al modelo 5.
- **Modelo 5:** se encuentra alejado de los datos simulados, alejándose más a mayores valores de SNR; presenta una forma similar pero de pendiente más pronunciada y más cóncavo, muy cercana al modelo 4.
- **Modelo 6:** se encuentra cercano a los datos simulados para bajos valores de SNR, pero rápidamente se aleja a mayores valores de SNR; presenta una forma distinta, más cóncava y de pendiente más pronunciada.
- **Modelo 7:** está cercano a valores simulados de manera uniforme; presenta una forma muy similar de muy cercana pendiente, al igual que los modelos 2 y 8.
- **Modelo 8:** está relativamente alejado de los valores simulados, alejándose más a mayor SNR; presenta una forma muy similar de muy cercana pendiente, al igual que los modelos 2 y 7.
- **Modelo 9:** se encuentra alejado de los datos simulados a bajo SNR, aproximándose más a medida que aumenta la SNR, hasta cortar la curva de datos simulados; presenta una forma distinta, más cóncava y de pendiente mucho más pronunciada.

Para valores bajos de SNR, el modelo 6 es el más próximo, pero a mayores valores, los modelos 2 y 7 son consistentemente más cercanos a los valores simulados. El modelo 9

se llega a acercar, pero de una manera brusca, y los modelos 1, 3, 4, 5 y 8 se encuentran alejados. Existen 3 modelos con una forma muy aproximada: los modelos 2, 7 y 8, siguiéndoles los modelos 1, 3, 4 y 5 con mayores pendientes, y finalmente los modelos 6 y 9 que presentan pendientes muy grandes y tienen una forma más cóncava.

e. 16QAM $r = {}^{1}/_{2}$

- **Modelo 1:** relativamente alejado a valores simulados, pero acercándose a mayores valores de SNR; presenta una forma similar pero de pendiente notablemente más pronunciada.
- **Modelo 2:** relativamente alejado a los valores simulados, acercándose ligeramente a mayores valores de SNR, ligeramente menos que el modelo 7; presenta una forma similar de cercana pendiente, al igual que los modelos 7 y 8.
- **Modelo 3:** está muy alejado de los datos simulados; presenta una forma similar pero de pendiente notablemente más pronunciada y ligeramente más cóncava.
- **Modelo 4:** se encuentra considerablemente alejado de los datos simulados; presenta una forma similar de considerablemente más cercana pendiente, similar al modelo 5.
- **Modelo 5:** se encuentra alejado de los datos simulados; presenta una forma similar de considerablemente más cercana pendiente, similar al modelo 4.
- **Modelo 6:** se encuentra alejado de los datos simulados, pero menos a menores valores de SNR; presenta una forma distinta, más cóncava.
- **Modelo 7:** relativamente alejado a los valores simulados, acercándose ligeramente a mayores valores de SNR; presenta una forma similar de cercana pendiente, al igual que los modelos 2 y 8.
- **Modelo 8:** está considerablemente alejado de los valores simulados; presenta una forma similar de cercana pendiente, al igual que los modelos 2 y 7.
- **Modelo 9:** se encuentra relativamente alejada de los datos simulados, aproximándose más a medida que aumenta la SNR, hasta cortar la curva de datos simulados; presenta una forma muy distinta, considerablemente más cóncava.

No hay modelos lo suficientemente cercanos a ser considerados. El modelo 9 se acerca, pero con una caída brusca por lo que no es un buen candidato. Los dos modelos más próximos son el 1 y el 6. En tanto la forma hay 2 candidatos buenos, pero que no coinciden demasiado bien: los modelos 4 y 5. Otros tres modelos que pueden ser tomados en cuenta son los modelos 2, 7 y 8, que se aproximan en forma menos que el 4 y el 5, pero no por

mucho. Los modelos restantes presentan una mayor pendiente, particularmente el modelo 9.

f. 16QAM $r = {}^{3}/_{4}$

- **Modelo 1:** relativamente alejado a valores simulados, pero acercándose a mayores valores de SNR; presenta una forma similar pero de pendiente notablemente más pronunciada.
- **Modelo 2:** se acerca considerablemente a los valores simulados, ligeramente menos que el modelo 7; presenta una forma muy similar de pendiente extremadamente cercana, al igual que los modelos 7 y 8.
- **Modelo 3:** está muy alejado a los datos simulados; presenta una forma similar pero de pendiente notablemente más pronunciada, ligeramente más cóncava.
- **Modelo 4:** se encuentra relativamente alejado de los datos simulados, alejándose ligeramente más a mayor SNR; presenta una forma similar pero de pendiente notablemente más pronunciada, muy cercana al modelo 5.
- **Modelo 5:** se encuentra relativamente cercano a los datos simulados para valores de SNR pequeños, pero se aleja a mayores valores de SNR; presenta una forma similar pero de pendiente notablemente más pronunciada, muy cercana al modelo 4.
- **Modelo 6:** se encuentra alejada de los datos simulados, aumentando la distancia a medida que aumenta la SNR; presenta una forma distinta, mucho más cóncava.
- **Modelo 7:** se acerca considerablemente a los valores simulados; presenta una forma muy similar de pendiente extremadamente cercana, al igual que los modelos 2 y 8.
- **Modelo 8:** está relativamente alejado de los valores simulados de manera uniforme; presenta una forma muy similar de pendiente extremadamente cercana, al igual que los modelos 2 y 7.
- **Modelo 9:** se encuentra cercano a los valores simulados para pequeños valores de SNR, pero se aleja rápidamente mientras aumentan; presenta una forma distinta, mucho más cóncava.

Aunque el modelo 9 se acerca mucho para valores bajos de SNR, no es un buen candidato por su brusca caída. Los modelos que más se acercan a los valores simulados son el 2 y el 7, siguiéndoles el 5 y el 8. En tanto la forma, hay 3 muy buenos candidatos: los modelos 2, 7 y 8. Con mayor pendiente les siguen los modelos 1, 3, 4 y 5, y finalmente los modelos 6

y 9 con pendientes muy grandes.

g. 64QAM $r = {}^{2}/_{3}$

- **Modelo 1:** está relativamente alejado a valores simulados, pero acercándose a mayores valores de SNR; presenta una forma ligeramente similar pero de pendiente notablemente más pronunciada.
- **Modelo 2:** está relativamente alejado a valores simulados uniformemente, ligeramente más que el modelo 7; presenta una forma muy similar de muy cercana pendiente, pero ligeramente más cóncava, al igual que los modelos 7 y 8.
- **Modelo 3:** está muy alejado a los datos simulados; presenta una forma similar pero de pendiente más pronunciada y más cóncava.
- **Modelo 4:** se encuentra alejada de los datos simulados, aumentando la distancia ligeramente a mayor SNR; presenta una forma similar pero de pendiente ligeramente más pronunciada, muy cercana al modelo 5.
- **Modelo 5:** se encuentra relativamente cercano a los datos simulados, aumentando la distancia a mayores valores de SNR; presenta una forma similar pero de pendiente ligeramente más pronunciada, muy cercana al modelo 4.
- **Modelo 6:** se encuentra muy alejado de los datos simulados, aumentando la distancia a medida que aumenta la SNR; presenta una forma distinta, mucho más cóncava.
- **Modelo 7:** está relativamente alejado a valores simulados uniformemente; presenta una forma muy similar de muy cercana pendiente, pero ligeramente más cóncava, al igual que los modelos 2 y 8.
- **Modelo 8:** está alejado a valores simulados uniformemente; presenta una forma muy similar de muy cercana pendiente, pero ligeramente más cóncava, al igual que los modelos 2 y 7.
- **Modelo 9:** se encuentra muy alejado de los datos simulados, aumentando la distancia a medida que aumenta la SNR; presenta una forma distinta, mucho más cóncava.

El mejor candidato por cercanía es el modelo 5, pero sí presenta una distancia considerable a los valores simulados. Le siguen los modelos 1 y 4. En tanto la forma hay 3 candidatos principales: los modelos 2, 7 y 8, teniendo a otro candidato menos prioritario, el modelo 3. El resto presenta pendientes muy grandes, especialmente los modelos 6 y 9.

h. 64QAM $r = {}^{3}/_{4}$

- **Modelo 1:** relativamente alejado a valores simulados, pero acercándose a mayores valores de SNR; presenta una forma similar pero de pendiente notablemente más pronunciada.
- **Modelo 2:** se acerca a los valores simulados, ligeramente más que el modelo 7; presenta una forma muy similar de pendiente extremadamente cercana, al igual que los modelos 7 y 8.
- **Modelo 3:** está muy alejado a los datos simulados; presenta una forma similar pero de pendiente notablemente más pronunciada y ligeramente más cóncavo.
- **Modelo 4:** se encuentra relativamente alejado de los datos simulados, alejándose ligeramente más a mayores valores de SNR; presenta una forma similar pero de pendiente notablemente más pronunciada, muy cercana al modelo 5.
- **Modelo 5:** se encuentra relativamente acercado a los datos simulados, alejándose más a mayores valores de SNR; presenta una forma similar pero de pendiente notablemente más pronunciada, muy cercana al modelo 4.
- **Modelo 6:** se encuentra considerablemente alejado de los datos simulados, aumentando la distancia a medida que aumenta la SNR; presenta una forma distinta, mucho más cóncava.
- **Modelo 7:** se acerca a los valores simulados; presenta una forma muy similar de pendiente extremadamente cercana, al igual que los modelos 2 y 8.
- **Modelo 8:** está relativamente acercado a los valores simulados uniformemente; presenta una forma muy similar de pendiente extremadamente cercana, al igual que los modelos 2 y 7.
- **Modelo 9:** se encuentra considerablemente alejado de los datos simulados, aumentando la distancia a medida que aumenta la SNR; presenta una forma distinta, mucho más cóncava.

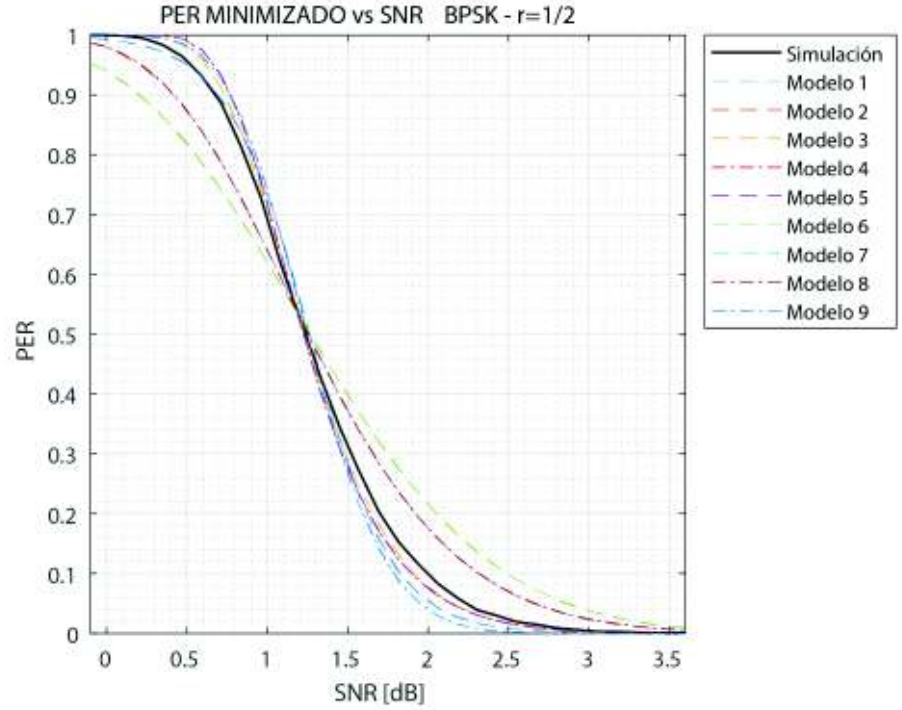
Existen dos modelos lo suficientemente cercanos: el modelo 2 y el 7. Después de ellos les sigue el modelo 8 y el 5 en ese orden. El modelo 9 presenta una caída muy brusca, así que no se lo toma en cuenta. En tanto la forma hay 3 muy buenos candidatos: los modelos 2, 7 y 8. El resto de los modelos presentan pendientes demasiado grandes, especialmente los modelos 6 y 9.

3.3. Comparación de modelos teóricos minimizados

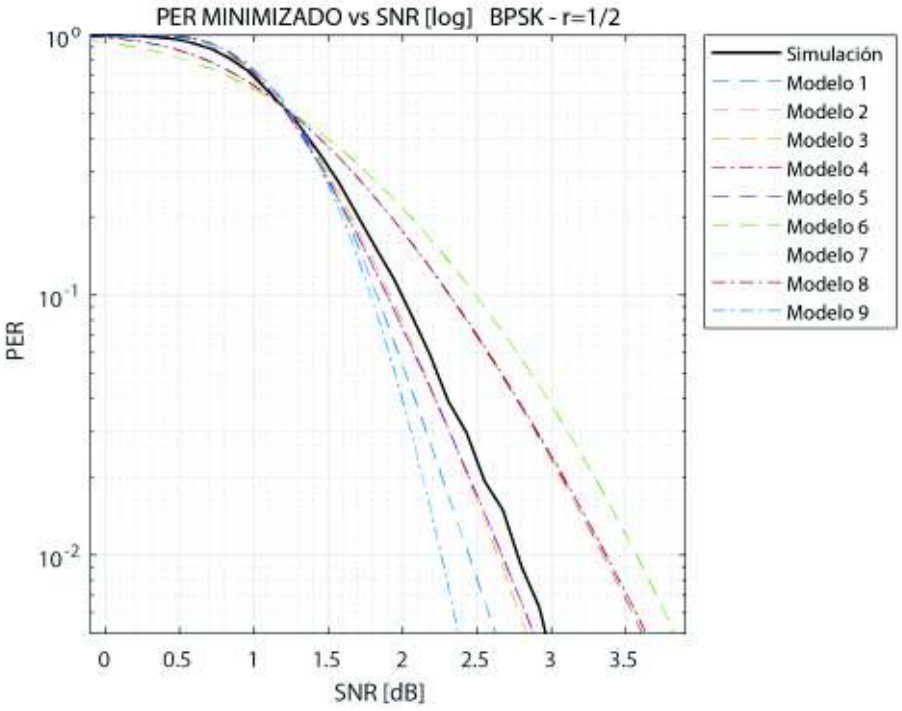
Las siguientes figuras presentan los resultados de la PER vs. SNR, considerando curvas teóricas minimizadas con su *offset* de SNR respectivo. Se presenta dos gráficas por cada configuración, la una en escala lineal y la otra en escala semilogarítmica (en el eje y). En cada gráfica se presentan las curvas de cada modelo minimizado y se comparan con los datos de la PER simulados correspondientes. Así, un resumen de las figuras se presenta en la Tabla 3.2.

Tabla 3.2: Figuras de la comparación de modelos teóricos minimizados de la PER

Modulación	**Tasa de codificación (r)**	**Escala**	**Figura**
BPSK	$1/2$	Lineal	3.11 **(a)**
BPSK	$1/2$	Semilogarítmica	3.11 **(b)**
BPSK	$3/4$	Lineal	3.12 **(a)**
BPSK	$3/4$	Semilogarítmica	3.12 **(b)**
QPSK	$1/2$	Lineal	3.13 **(a)**
QPSK	$1/2$	Semilogarítmica	3.13 **(b)**
QPSK	$3/4$	Lineal	3.14 **(a)**
QPSK	$3/4$	Semilogarítmica	3.14 **(b)**
16QAM	$1/2$	Lineal	3.15 **(a)**
16QAM	$1/2$	Semilogarítmica	3.15 **(b)**
16QAM	$3/4$	Lineal	3.16 **(a)**
16QAM	$3/4$	Semilogarítmica	3.16 **(b)**
64QAM	$2/3$	Lineal	3.17 **(a)**
64QAM	$2/3$	Semilogarítmica	3.17 **(b)**
64QAM	$3/4$	Lineal	3.18 **(a)**
64QAM	$3/4$	Semilogarítmica	3.18 **(b)**

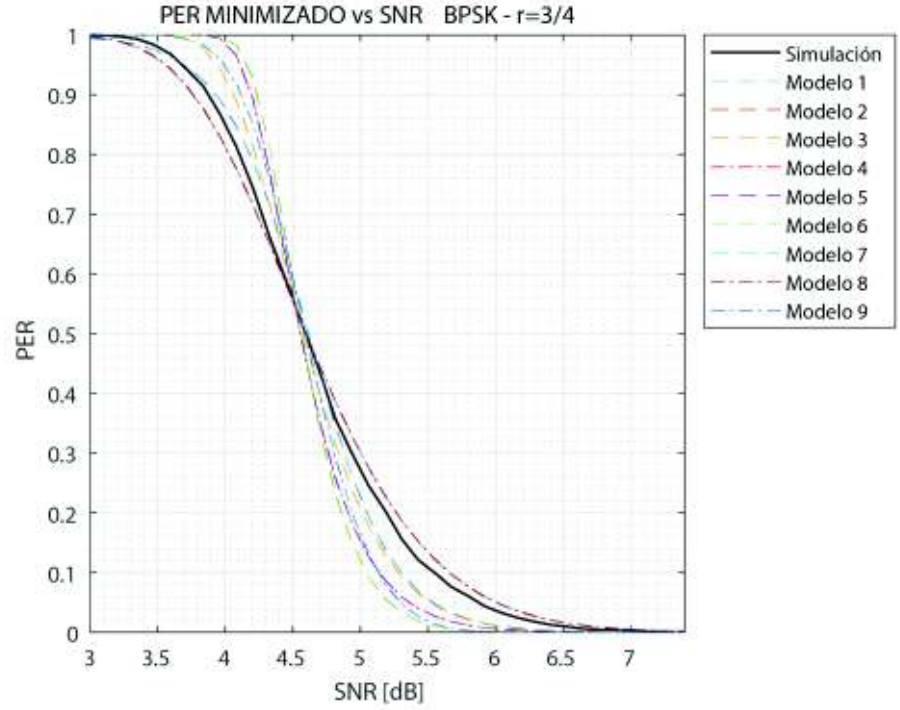


(a) Lineal

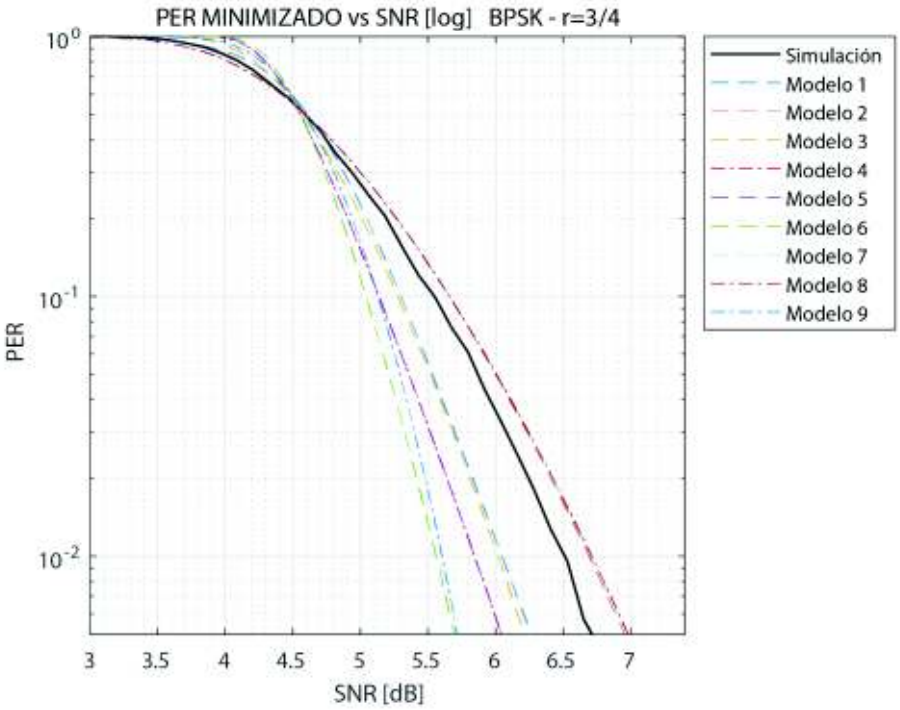


(b) Semilogarítmica

Figura 3.11: Comparación de modelos teóricos minimizados con resultados simulados para BPSK r=$1/2$ (**(a)** Escala lineal, **(b)** Escala semilogarítmica)

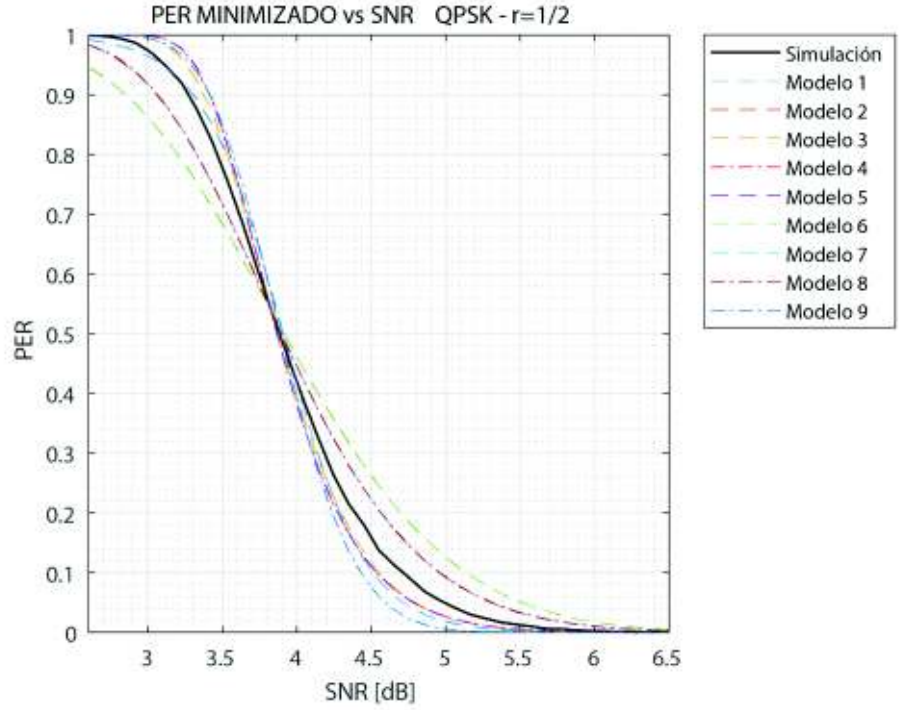


(a) Lineal

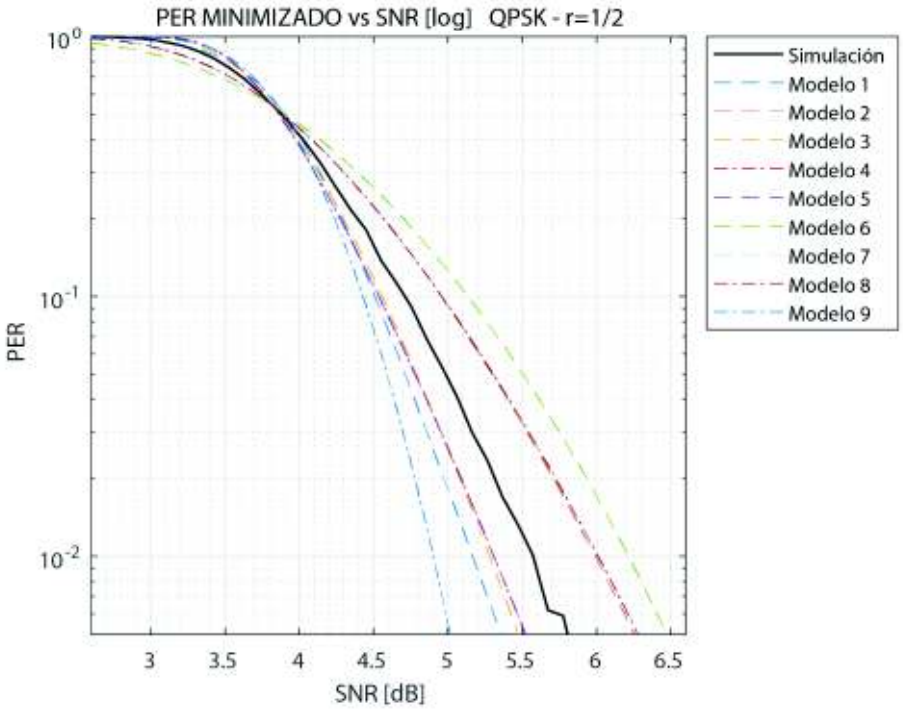


(b) Semilogarítmica

Figura 3.12: Comparación de modelos teóricos minimizados con resultados simulados para BPSK r=$^3/_4$ (**(a)** Escala lineal, **(b)** Escala semilogarítmica)

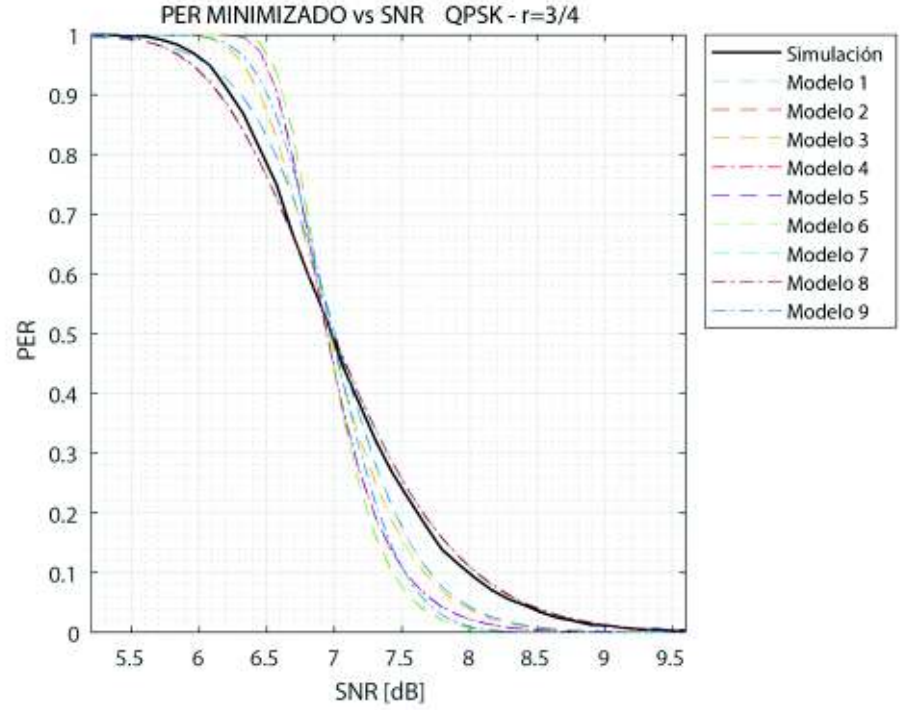


(a) Lineal

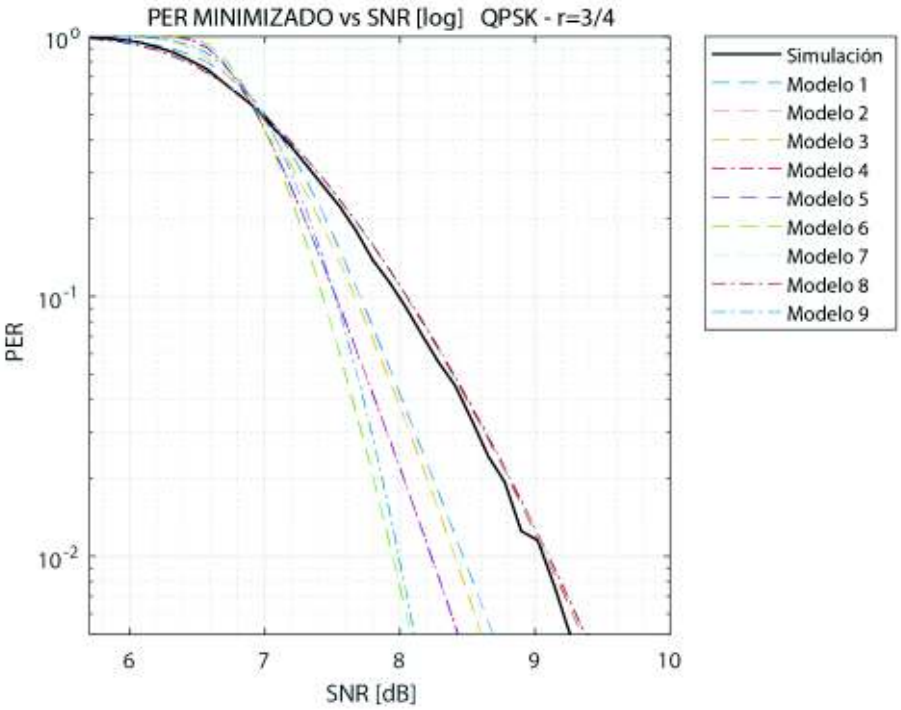


(b) Semilogarítmica

Figura 3.13: Comparación de modelos teóricos minimizados con resultados simulados para QPSK r=$1/2$ (**(a)** Escala lineal, **(b)** Escala semilogarítmica)

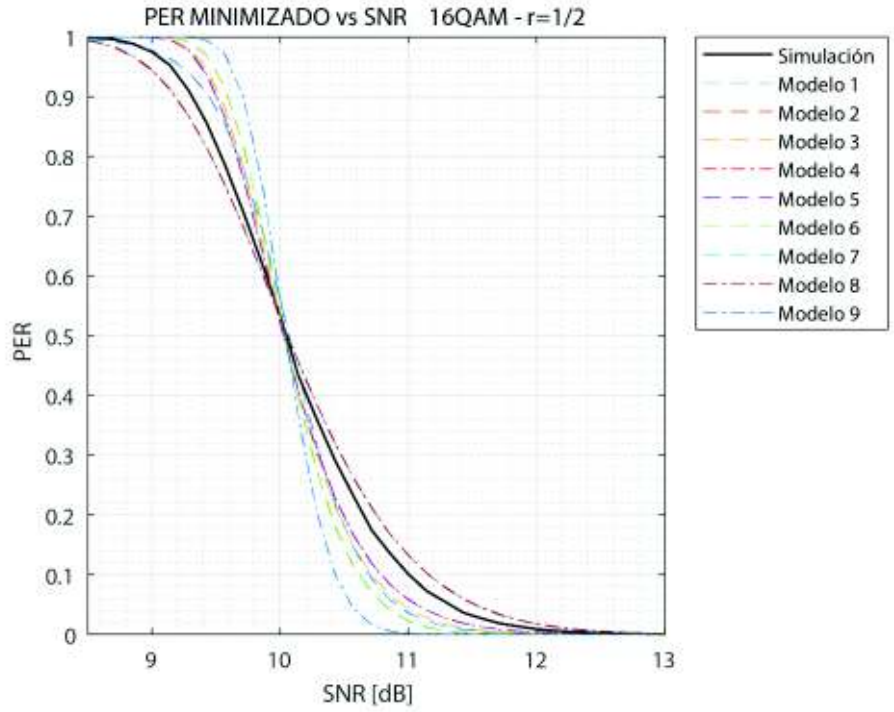


(a) Lineal

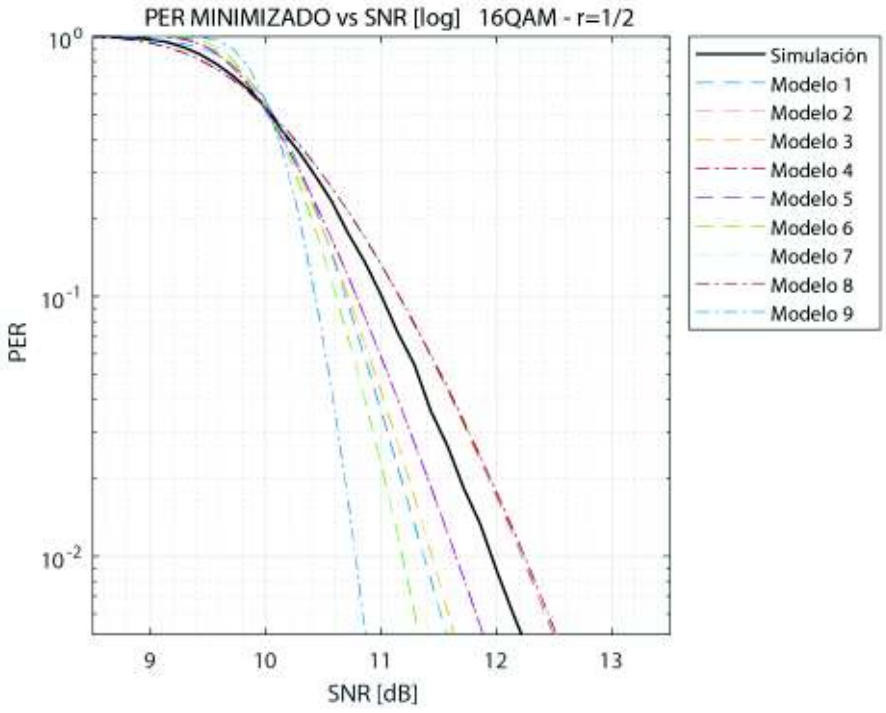


(b) Semilogarítmica

Figura 3.14: Comparación de modelos teóricos minimizados con resultados simulados para QPSK r=$^3/_4$ (**(a)** Escala lineal, **(b)** Escala semilogarítmica)

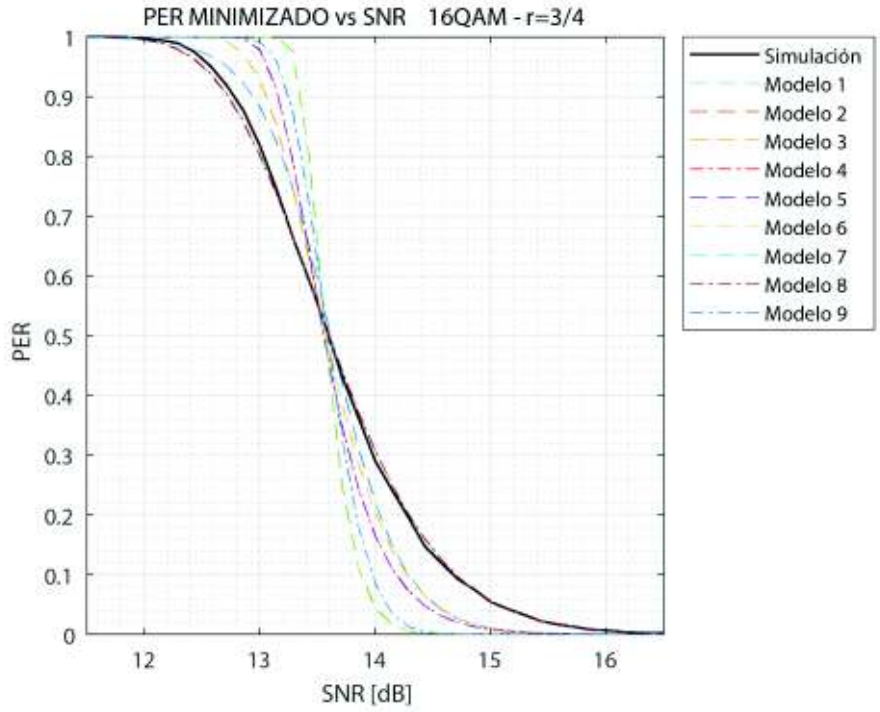


(a) Lineal

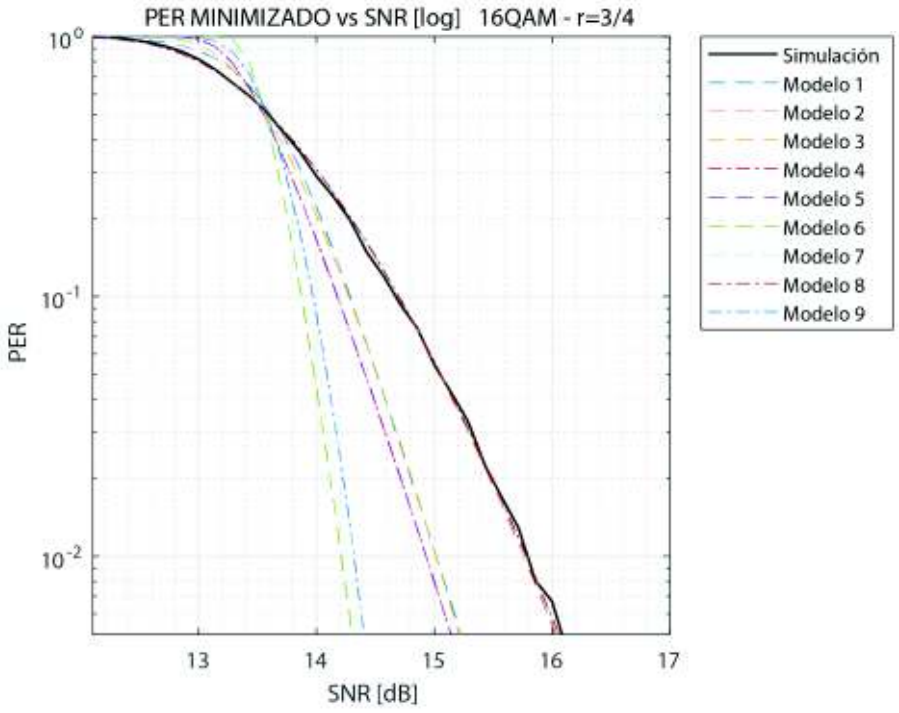


(b) Semilogarítmica

Figura 3.15: Comparación de modelos teóricos minimizados con resultados simulados para 16QAM r=$1/2$ (**(a)** Escala lineal, **(b)** Escala semilogarítmica)

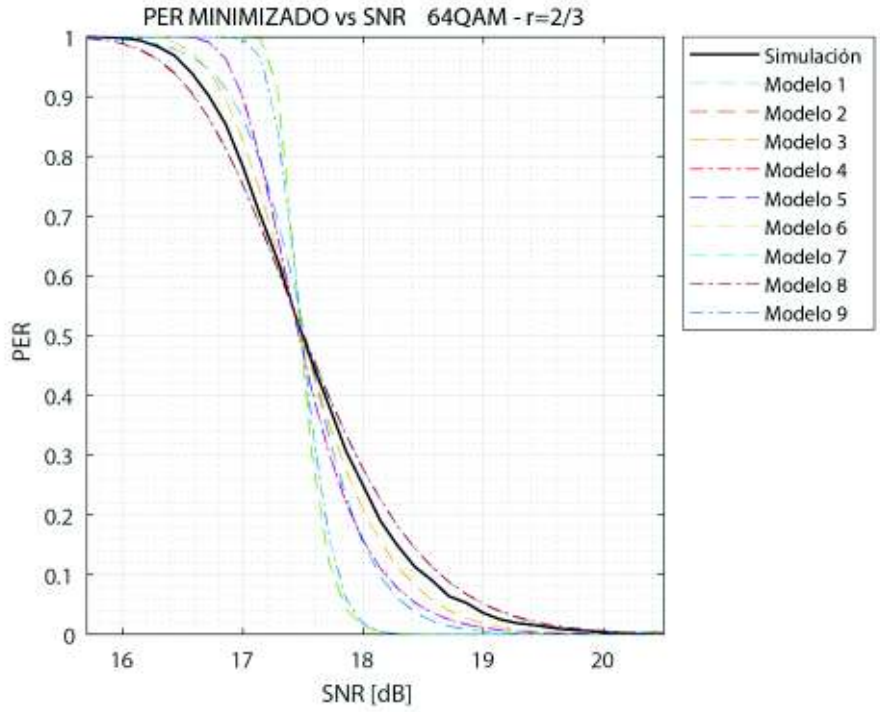


(a) Lineal

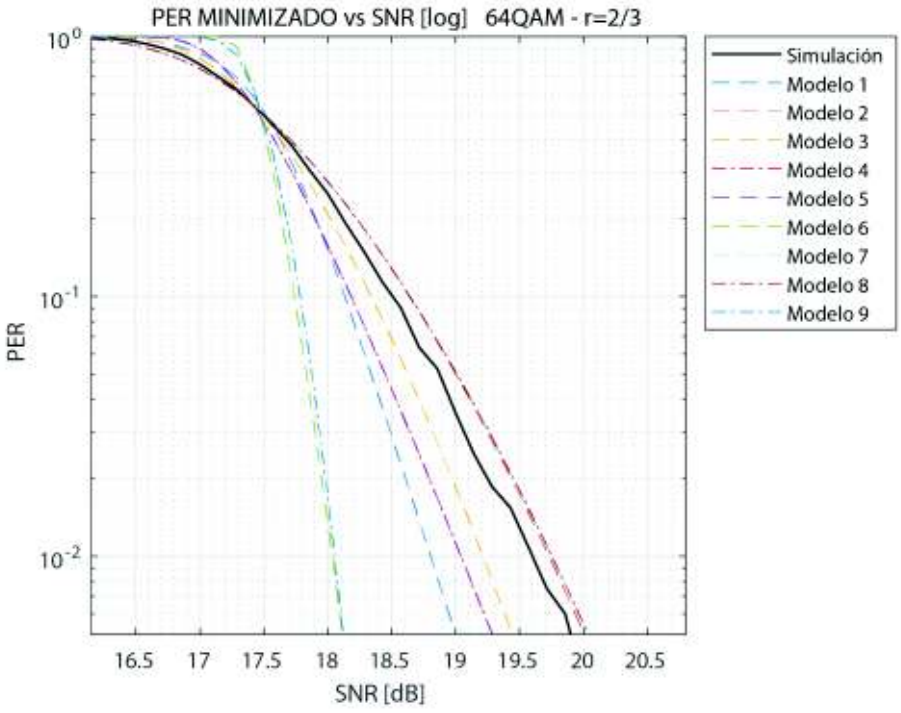


(b) Semilogarítmica

Figura 3.16: Comparación de modelos teóricos minimizados con resultados simulados para 16QAM r=$^3/_4$ (**(a)** Escala lineal, **(b)** Escala semilogarítmica)

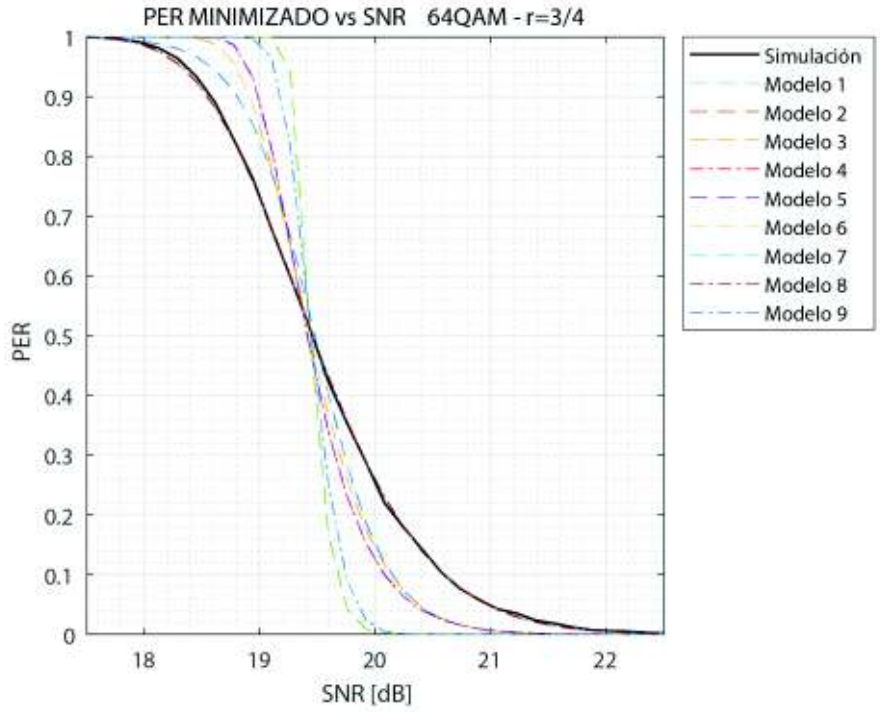


(a) Lineal

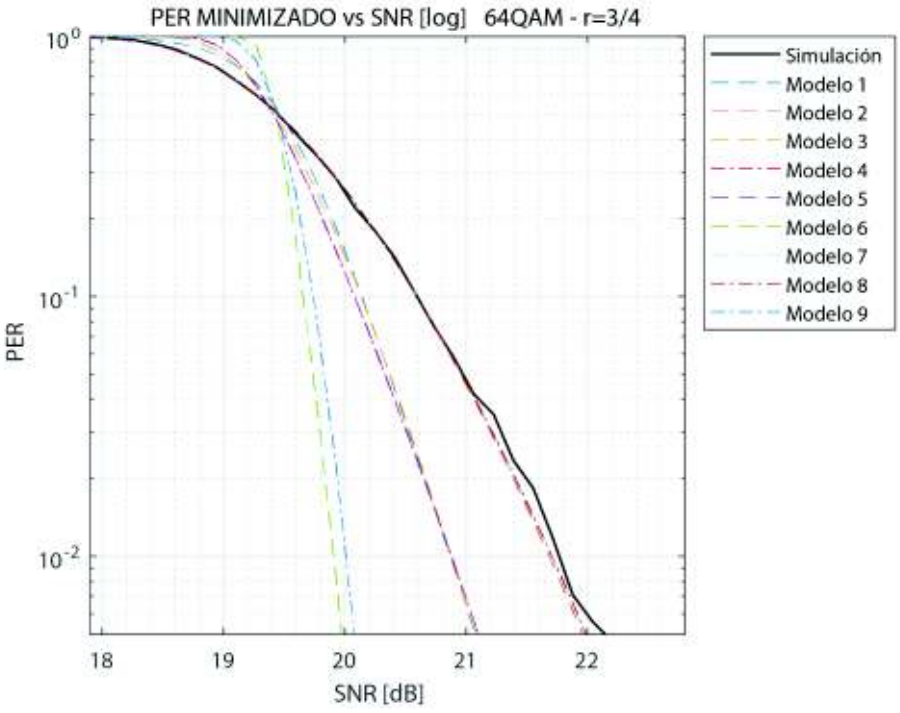


(b) Semilogarítmica

Figura 3.17: Comparación de modelos teóricos minimizados con resultados simulados para 64QAM r=$^2/_3$ (**(a)** Escala lineal, **(b)** Escala semilogarítmica)



(a) Lineal



(b) Semilogarítmica

Figura 3.18: Comparación de modelos teóricos minimizados con resultados simulados para 64QAM r=$^3/_4$ (**(a)** Escala lineal, **(b)** Escala semilogarítmica)

3.3.1. Análisis cuantitativo

Con los resultados de las figuras, el *script* ANALISIS_RESULTADOS y los contenidos de las variables C_MIN y E_MIN, se obtiene la Tabla 3.3 que indica los *offsets* de SNR minimizados por cada configuración y cada modelo, y la Tabla 3.4 que indica los errores mínimos por modelo para cada configuración.

Tabla 3.3: *Offsets* minimizados de modelos teóricos

M	r	Mod. 1	Mod. 2	Mod. 3	Mod. 4	Mod. 5	Mod. 6	Mod. 7	Mod. 8	Mod. 9
BPSK	$1/2$	-0,4884	-1,5787	-2,2227	4,3877	3,4186	3,3486	-1,5793	-2,5484	-2,5876
BPSK	$3/4$	-0,8972	0,0046	-0,0021	3,0175	2,0484	1,0378	0,0036	-0,9655	-1,0030
QPSK	$1/2$	-2,4963	-1,9460	0,4189	4,0186	3,0495	2,9830	-1,9467	-2,9158	-2,9551
QPSK	$3/4$	-3,2194	-0,6151	2,3857	2,3937	1,4246	0,4139	-0,6162	-1,5853	-1,6251
16QAM	$1/2$	-2,9602	-2,5891	6,5763	3,8678	2,8987	1,9476	-2,5902	-3,5593	-1,4868
16QAM	$3/4$	-3,0627	-0,8079	8,9948	2,3781	1,4090	1,6595	-0,8093	-1,7784	0,2938
64QAM	$2/3$	-2,8356	-2,4532	14,1234	2,1874	1,2183	3,0776	-2,4545	-3,4236	1,6047
64QAM	$3/4$	-2,5541	-1,0126	14,8523	2,3456	1,3765	4,0659	-1,0141	-1,9832	3,0470

Tabla 3.4: Errores minimizados de modelos teóricos

M	r	Mod. 1	Mod. 2	Mod. 3	Mod. 4	Mod. 5	Mod. 6	Mod. 7	Mod. 8	Mod. 9
BPSK	$1/2$	0,01837	0,09507	0,00649	0,01525	0,01525	0,23597	0,09456	0,09456	0,03471
BPSK	$3/4$	0,03187	0,01474	0,06123	0,17622	0,17622	0,25712	0,01444	0,01444	0,13654
QPSK	$1/2$	0,02823	0,07164	0,02524	0,04182	0,04182	0,21587	0,07107	0,07107	0,07459
QPSK	$3/4$	0,04473	0,00667	0,08563	0,21462	0,21462	0,30356	0,00631	0,00631	0,17270
16QAM	$1/2$	0,05490	0,01820	0,06831	0,04564	0,04564	0,12500	0,01789	0,01789	0,28452
16QAM	$3/4$	0,07538	0,00239	0,11051	0,19987	0,19987	0,56377	0,00212	0,00212	0,38039
64QAM	$2/3$	0,07913	0,01461	0,01941	0,09558	0,09558	0,56550	0,01425	0,01425	0,46845
64QAM	$3/4$	0,09555	0,00071	0,13283	0,18759	0,18759	0,68848	0,00053	0,00053	0,52311

Los valores óptimos por configuración se encuentran coloreados. Se considera que el *offset* es un indicador de **exactitud** y el error mínimo un indicador de **precisión**.

De esta manera, se observa que ningún modelo es particularmente exacto, ya que en cada configuración el más exacto es distinto a las otras, a excepción del modelo 9 que es el más exacto para 16QAM $r = 1/2$ y 16QAM $r = 3/4$ y el modelo 3 que abarca las configuraciones BPSK $r = 3/4$ y QPSK $r = 1/2$, mientras que los modelos 8 y 4 que no son los más exactos en ninguna configuración.

En tanto precisión, sin embargo, se observa que existen 3 modelos que abarcan todos los casos. El modelo 3 abarca 2 configuraciones: BPSK $r = {}^{1}/_{2}$ y QPSK $r = {}^{1}/_{2}$. El modelo 7 abarca 3 configuraciones: BPSK $r = {}^{3}/_{4}$, 16QAM $r = {}^{1}/_{2}$ y 16QAM $r = {}^{3}/_{4}$; mientras que el modelo 8 abarca también 3 configuraciones: QPSK $r = {}^{3}/_{4}$, 64QAM $r = {}^{2}/_{3}$ y 64QAM $r = {}^{3}/_{4}$. Entonces se puede ver que existe un modelo candidato por precisión en el modelo 8 o en el 7.

En tanto sobreposición, solamente en la configuración $QPSK\ r = {}^{1}/_{2}$ el modelo más preciso coincide con el más exacto, siendo el 3.

Un resumen de esta información se puede apreciar en la Tabla 3.5.

Tabla 3.5: Resumen de resultados cuantitativos de los modelos de la PER

Modulación	**Tasa de codificación (r)**	**Modelo más exacto**	***Offset* mínimo**	**Modelo más preciso**	**Error mínimo**	***Offset* de error mínimo**
BPSK	$^{1}/_{2}$	1	-0,488367	3	0,006489	-2.222734
BPSK	$^{3}/_{4}$	3	-0.002113	7	0,014435	0.003591
QPSK	$^{1}/_{2}$	3	0,418851	3	0,025243	0.418851
QPSK	$^{3}/_{4}$	6	0,413932	8	0,006313	-1.585335
16QAM	$^{1}/_{2}$	9	-1,486833	7	0,017889	-2.590182
16QAM	$^{3}/_{4}$	9	0,293790	7	0,002121	-0.809250
64QAM	$^{2}/_{3}$	5	1,218265	8	0,014254	-3.423560
64QAM	$^{3}/_{4}$	2	-1,012557	8	0,000529	-1.983217

3.4. Análisis de resultados

Tomando en cuenta el análisis cualitativo y el cuantitativo se puede aseverar lo siguiente:

- Los modelos que más se asemejan a la forma de los datos simulados son el 3, el 7 y el 8.
- El modelo 9 es el más exacto en 2 configuraciones: 16QAM $r = {}^{1}/_{2}$ y 16QAM $r = {}^{3}/_{4}$, pero no se lo puede considerar por la forma drásticamente distinta a la curva simulada de estos casos.

- El modelo 9 puede descartarse del análisis, ya que no es el más preciso en ninguna configuración y su valía como el más exacto en 2 queda descartada.
- El modelo 4 puede descartarse del análisis, al no ser el más exacto ni el más preciso en ninguna configuración.
- En la configuración 64QAM $r = {}^{3}/_{4}$, el *offset* del modelo 7 es muy próximo al *offset* mínimo del modelo 2, por lo que se puede tomar al modelo 7 como el *offset* óptimo en esta configuración sin perder mucha exactitud.
- El modelo 2 puede descartarse del análisis, ya que no es el más preciso en ninguna configuración y su valía como el más exacto en un modelo puede pasar al modelo 7 sin perder mucha exactitud.
- En tanto precisión, los modelos 7 y 8 son intercambiables por sus valores de error casi iguales.
- En la configuración QPSK $r = {}^{3}/_{4}$, se puede considerar el modelo 7 como candidato de exactitud debido a que es el segundo más cercano después del modelo 6 para poder descartar al modelo 6 del análisis.

Con este análisis se tiene las siguientes preferencias por configuración:

- BPSK $r = {}^{1}/_{2}$: **Exactitud:** Modelo 1 **Precisión:** Modelo 3
- BPSK $r = {}^{3}/_{4}$: **Exactitud:** Modelo 7 **Precisión:** Modelo 7
- QPSK $r = {}^{1}/_{2}$: **Exactitud:** Modelo 3 **Precisión:** Modelo 3
- QPSK $r = {}^{3}/_{4}$: **Exactitud:** Modelo 6 o 7 **Precisión:** Modelo 8 o 7
- 16QAM $r = {}^{1}/_{2}$: **Exactitud:** Modelo 6 **Precisión:** Modelo 7 u 8
- 16QAM $r = {}^{3}/_{4}$: **Exactitud:** Modelo 7 **Precisión:** Modelo 7 u 8
- 64QAM $r = {}^{2}/_{3}$: **Exactitud:** Modelo 5 **Precisión:** Modelo 8 o 7
- 64QAM $r = {}^{3}/_{4}$: **Exactitud:** Modelo 7 **Precisión:** Modelo 8 o 7

3.5. Producto demostrable

Se considera como producto demostrable el conjunto de *scripts* y funciones que permiten:

- La obtención de datos de PER simulados de un sistema de comunicación IEEE

802.11p para todas las configuraciones permitidas.

- La obtención de datos de PER teóricos dados por los modelos listados en la sección 1.3.7 para todas las configuraciones permitidas de acuerdo al estándar IEEE 802.11p.
- La optimización del error de cada curva de PER teórica con respecto a la correspondiente simulada mediante *offsets* en SNR.
- La comparación gráfica y matemática (basada en errores y *offsets*) entre los datos de PER teóricos con respecto a los respectivos simulados para todas las configuraciones permitidas por el estándar IEEE 802.11p.

Debido a que el presente trabajo es enfocado hacia la investigación, los usuarios de estos *scripts* son personas que desean hacer uso de la capa física IEEE 802.11p para futuros estudios. El perfil incluye conocimientos de MATLAB.

Una guía de utilización de estos *scripts* es presentada en la sección 2.5. Los parámetros que los usuarios podrían cambiar de acuerdo a sus necesidades se presentan en los scripts SIMULACION y SIMULACION_ANGOSTO y son:

- Longitud de los paquetes.
- Número de paquetes a enviar.
- Rangos de SNR.

3.5.1. Funcionalidad

El producto tiene una funcionalidad puntual en el ámbito investigativo, la cual consiste en la obtención y comparación de datos de PER simulados con teóricos obtenidos de modelos específicos (Sección 1.3.7) para un sistema de comunicación IEEE 802.11p.

Como se puede evidenciar en este capítulo, se han presentado los resultados de esta comparación de manera gráfica y matemática, incluyendo un proceso de minimización de error. Además, los *scripts* y funciones fueron diseñados a partir de, y cumplen con los estándares IEEE 802.11 [8] e IEEE 802.11p [30] de manera íntegra.

3.5.2. Desempeño

Por la naturaleza de la simulación, la obtención inicial de los datos toma una cantidad considerable de tiempo, siendo en promedio 40 horas. Sin embargo, esto es esperado, y los datos

son guardados continuamente (por cada configuración) lo cual permite una ejecución por partes, y provee protección ante errores imprevistos en medio de la ejecución. Esta fase se evalúa de acuerdo a los resultados obtenidos

Luego de la obtención de datos iniciales (SIMULACION y SIMULACION_ANGOSTO), la ejecución del resto de *scripts* del plan toma pocos minutos. Esta parte se evalúa en tiempo real.

De manera general los *scripts* y funciones son implementados de manera eficiente y eficaz.

Las gráficas obtenidas mediante el *script* GRAFICACION_RESULTADOS contienen toda la información necesaria para su interpretación y son presentadas en dos tipos: datos de PER adimensionales y datos de PER en dB. Esto permite una mejor apreciación de los datos gráficos.

El *script* ANALISIS_RESULTADOS nos da un resumen textual de los resultados. Si se desea datos más detallados, se inspecciona las variables C_MIN y E_MIN.

De esta manera se obtienen los resultados esperados en tiempos apropiados y con un apropiado nivel de detalle.

3.5.3. Diseño robusto e implementación

El producto cumple por completo con lo esperado. Esto se evidencia en la evaluación de tiempo real y en los resultados expuestos en este capítulo.

El producto se encuentra parametrizado con respecto a:

- Longitud de los paquetes.
- Número de paquetes a enviar.
- Rangos de SNR.

Estos parámetros tienen un valor por defecto, con los cuales se han presentado los resultados expuestos, pero pueden ser cambiados en los *scripts* SIMULACION y SIMULACION_ANGOSTO si se desea. Esto es posible gracias a la flexibilidad de trabajar con *scripts* y funciones.

Más allá de esto, durante la evaluación no es necesario realizar cambio, ajuste o reparación alguna. Sin embargo, debido a que este producto es presentado en *scripts* es posible la

inspección del código para cualquier adecuación que sea necesaria, o para tomar parte del mismo y aislarlo. Esto es posible gracias a la presentación en código del producto.

3.5.4. Facilidad de uso

Los usuarios esperados del producto son investigadores. Los usuarios podrían emplear el producto fácilmente ayudados por el plan de ejecución de la sección 2.5. Además ubicar y comprender rápidamente cómo cambiar los parámetros de la simulación de manera simple, debido a la facilidad de uso y de modificación del producto.

El plan de ejecución permite la utilización del producto de una manera eficiente y efectiva, ya que se encuentra automatizado. Dado que los resultados obtenidos cumplen con lo especificado al inicio de esta sección, se esperan buenos niveles de satisfacción.

4. CONCLUSIONES Y RECOMENDACIONES

Con los resultados obtenidos, se establecen los modelos a utilizar en cada configuración con el *offset* respectivo.

4.1. Conclusiones

En este trabajo de titulación se implementó en MATLAB la capa física del estándar IEEE 802.11p y se obtuvieron resultados de PER para las distintas configuraciones permitidas por el mismo, sobre rangos de SNR adecuados. Luego se implementaron los modelos de la PER establecidos en el primer capítulo y se compararon los resultados de los mismos con los correspondientes de la simulación, con lo cual se tiene las siguientes conclusiones:

- Los modelos que deben utilizarse por cada configuración (velocidad de transmisión) considerando lo precisos que son con respecto a los datos simulados, son los expuestos en la Tabla 4.1 con sus *offsets* de SNR respectivos.

Tabla 4.1: Modelos escogidos por configuración con su offset respectivo

M	r	Modelo	*Offset* SNR	Diferencia con *offset* mínimo
BPSK	$1/2$	3	-2,222734	1,734367
BPSK	$3/4$	7	0,003591	0,001478
QPSK	$1/2$	3	0,418851	0,000000
QPSK	$3/4$	7	-0,616235	0,202303
16QAM	$1/2$	7	-2,590182	1,103348
16QAM	$3/4$	7	-0,809250	0,515460
64QAM	$2/3$	7	-2,454460	1,236195
64QAM	$3/4$	7	-1,014116	0,001559

- Los modelos más cercanos a los datos simulados sin aplicar *offsets* por cada configuración escogidos están expuestos en la Tabla 4.2.
- El modelo 7 es el que tiene una forma más cercana a los valores simulados de manera general.

Tabla 4.2: Modelos más cercanos por configuración

M	r	Modelo
BPSK	$1/2$	1
BPSK	$3/4$	3
QPSK	$1/2$	3
QPSK	$3/4$	6
16QAM	$1/2$	6
16QAM	$3/4$	7
64QAM	$2/3$	5
64QAM	$3/4$	7

- En tanto tasas de codificación considerando *offset*, para $r = 2/3$ y $r = 3/4$ se utiliza el modelo 7, mientras que para $r = 1/2$ se utiliza el modelo 3, a menos que se trate de modulación 16QAM, en cuyo caso se utiliza también el modelo 7 de acuerdo a la Tabla 4.1.
- Si no se considera *offset*, en tanto tasas de codificación, se puede usar el modelo 7 para $r = 3/4$ y el modelo 5 para $r = 2/3$. Para $r = 1/2$ no hay un modelo que se ajuste a todos los resultados, y se deberá tomar por casos (esquema de modulación) de acuerdo a los resultados expuestos en la sección 3.4.
- Si se toma en cuenta solo el esquema de modulación tomando en cuenta *offset*, se puede tomar el modelo 7 para los esquemas de modulación por cuadratura (16QAM y 64QAM), mientras que para los esquemas por modulación digital por frecuencia (BPSK y QPSK) no hay modelo que se ajuste a todos los resultados. Se consideran los *offsets* de acuerdo a la Tabla 4.1.
- Si no se toma en cuenta el *offset* y se toma en cuenta la modulación, no existe un único modelo más útil por esquema de modulación.
- De entre los modelos teóricos estudiados, ninguno de ellos se adapta de manera precisa y exacta a todas las configuraciones de esquema de modulación y tasa de codificación convolucional. Sin embargo, se pueden utilizar distintos modelos por casos, con *offsets* en SNR para obtener los resultados esperados.
- De manera alternativa, el modelo 1 es particularmente de interés, ya que su parametrización en tanto constantes fijas por configuración permite, en teoría,

realizar un ajuste de la curva con respecto a la simulada (o a cualquier otra teórica) mediante *software* o métodos matemáticos.

4.2. Recomendaciones

Para futuros trabajos se recomienda:

- Considerar realizar la misma comparación teórica, simulando un canal más sofisticado, por ejemplo con *fading*, para la obtención de datos simulados.
- Considerar la validación de los resultados obtenidos mediante experimentos controlados.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] F. Abrate, A. Vesco, and R. Scopigno. An Analytical Packet Error Rate Model for WAVE Receivers. *2011 IEEE Vehicular Technology Conference (VTC Fall), San Francisco, CA*, pages 1–5, 2011.

[2] A. Mahmood and R. Jäntti. Packet Error Rate Analysis of Uncoded Schemes in Block-Fading Channels Using Extreme Value Theory. *IEEE Communications Letters*, 21:208–211, 2017.

[3] Marvin Simon and Mohamed-Slim Alouini. *Digital Communication over Fading Channels: A Unified Approach to Performance Analysis*. Wiley-Interscience, Estados Unidos, 2000.

[4] Yoo-Seung Song and Hyun-Kyun Choi. Analysis of V2V Broadcast Performance Limit for WAVE Communication Systems Using TwoPath Loss Model. *ETRI Journal*, 39:213–221, 2017.

[5] R. Khalili and K. Salamatian. A new analytic approach to evaluation of packet error rate in wireless networks. *3rd Annual Communication Networks and Services Research Conference (CNSR'05)*, 3:333–338, 2005.

[6] Y. Xi, A. Burr, J. Wei, and D. Grace. A General Upper Bound to Evaluate Packet Error Rate over Quasi-Static Fading Channels. *IEEE Transactions on Wireless Communications*, 10(5):1373–1377, 2011.

[7] IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments. *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009)*, pages 1–51, 2010.

[8] IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY)

Specifications. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pages 1–2793, 2012.

[9] L. Qi. Research on intelligent transportation system technologies and applications. *2008 Workshop on Power Electronics and Intelligent Transportation System*, pages 529–531, Aug 2008.

[10] E. C. Eze, S. Zhang, and E. Liu. Vehicular ad hoc networks (VANETs): Current state, challenges, potentials and way forward. *2014 20th International Conference on Automation and Computing, Cranfield*, pages 176–181, 2014.

[11] J. Sigüenza. Evaluación de la capa física del estándar IEEE 802.11p para redes vehiculares en un canal AWGN. *EPN*, 2017.

[12] Yasser Toor, Paul Muhlethaler, Anis Laouiti, and Arnaud de La Fortelle. Vehicle ad hoc networks: Applications and related technical issues. IEEE Communications Surveys & Tutorials. *Communications Surveys & Tutorials, IEEE.*, 10:74–88, 2008.

[13] Muhammad Akbar, Chaudhary Muhammad Asim Rasheed, and Amir. Qayyum. VANET Architectures and Protocol Stacks: A Survey. 1970.

[14] Olivia Brickley, Martin Koubek, Susan Rea, and Dirk Pesch. A network centric simulation environment for CALM-based cooperative vehicular systems. 2010.

[15] A. M. Abdelgader and W. Lenan. The physical layer of the IEEE 802.11 p WAVE communication standard: the specifications and challenges. *Proceedings of the World Congress on Engineering and Computer Science*, 2:22–24, 2014.

[16] Shereen Hamato, Sharifah Hafizah Syed Ariffin, and Norsheila Fisal. Overview of Wireless Access in Vehicular Environment (WAVE) Protocols and Standards. *Indian Journal of Science and Technology*, 7, 2013.

[17] Elyes ben Hamida, Hassan Noura, and Wassim Znaidi. Security of cooperative intelligent transport systems: Standards, threats analysis and cryptographic countermeasures. 4, 07 2015.

[18] Devin Akin and Jim Geier. *Certified Wireless Analysis Professional™ Official Study Guide*. McGraw-Hill/Osborne, Estados Unidos, 2004.

[19] Shu Lin and Daniel J Costello. *Error Control Coding*. 01 2004.

[20] Y. Q. Shi, Xi Min Zhang, Zhi-Cheng Ni, and N. Ansari. Interleaving for combating bursts of errors. *IEEE Circuits and Systems Magazine*, 4(1):29–42, 2004.

[21] Mario Pineda-Ruelas. *Enteros, aritmética modular y grupos finitos*. 03 2014.

[22] John Proakis and Masoud Salehi. *Digital Communications*. McGraw-Hill, Estados Unidos, 2008.

[23] Vijay Garg. *Wireless Communications and Networking*. Morgan Kaufmann, Estados Unidos, 2007.

[24] D C. Shah, Baban Rindhe, and Santosh Narayankhedkar. Effects of cyclic prefix on ofdm system. pages 420–424, 01 2010.

[25] David Tse and Pramod Viswanath. Fundamentals of wireless communication / d. tse, p. viswanath. 11 2018.

[26] G Breed. Bit Error Rates: Fundamental Concepts and Measurement Issues. *High Freq. Elec.*, 1:46–48, 2003.

[27] Bernard Sklar. *Digital communications: fundamentals and applications*. Prentice Hall Communications Engineering and Emerging Technologies Series. Prentice-Hall PTR, 2001.

[28] J M. Blair, C A. Edwards, and J H. Johnson. Rational chebyshev approximations for the inverse of the error function. *Mathematics of Computation - Math. Comput.*, 30, 10 1976.

[29] Jamal Salah. A note on gamma function. *International Journal of Modern Sciences and Engineering Technology (IJMSET)*, 2:58–64, 2015.

[30] Task Group p. IEEE P802.11p: Wireless Access in Vehicular Environments (WAVE), draft standard ed. *IEEE Computer Society*, 2006.

6. ANEXOS

ANEXO I: Diagramas de Flujo

ANEXO II: Algoritmos

ANEXO III: Código

ANEXO I

En este anexo se adjuntan los diagramas de flujo para cada *script* y función definidos en el Capítulo 2.

Nomenclatura para diagramas de flujo

Todos los diagramas de flujo seguirán la nomenclatura presentada en la Figura I.1.

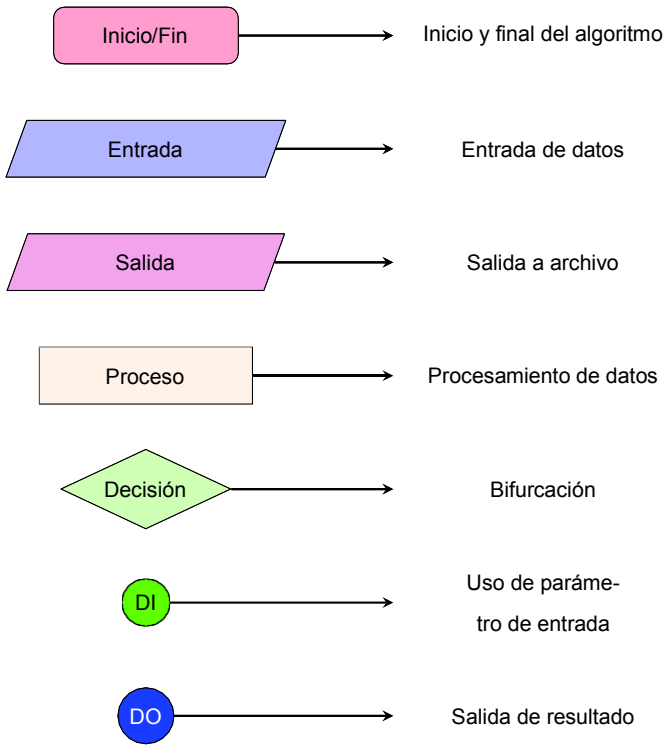


Figura I.1: Nomenclatura para los diagramas de flujo

Diagramas de flujo

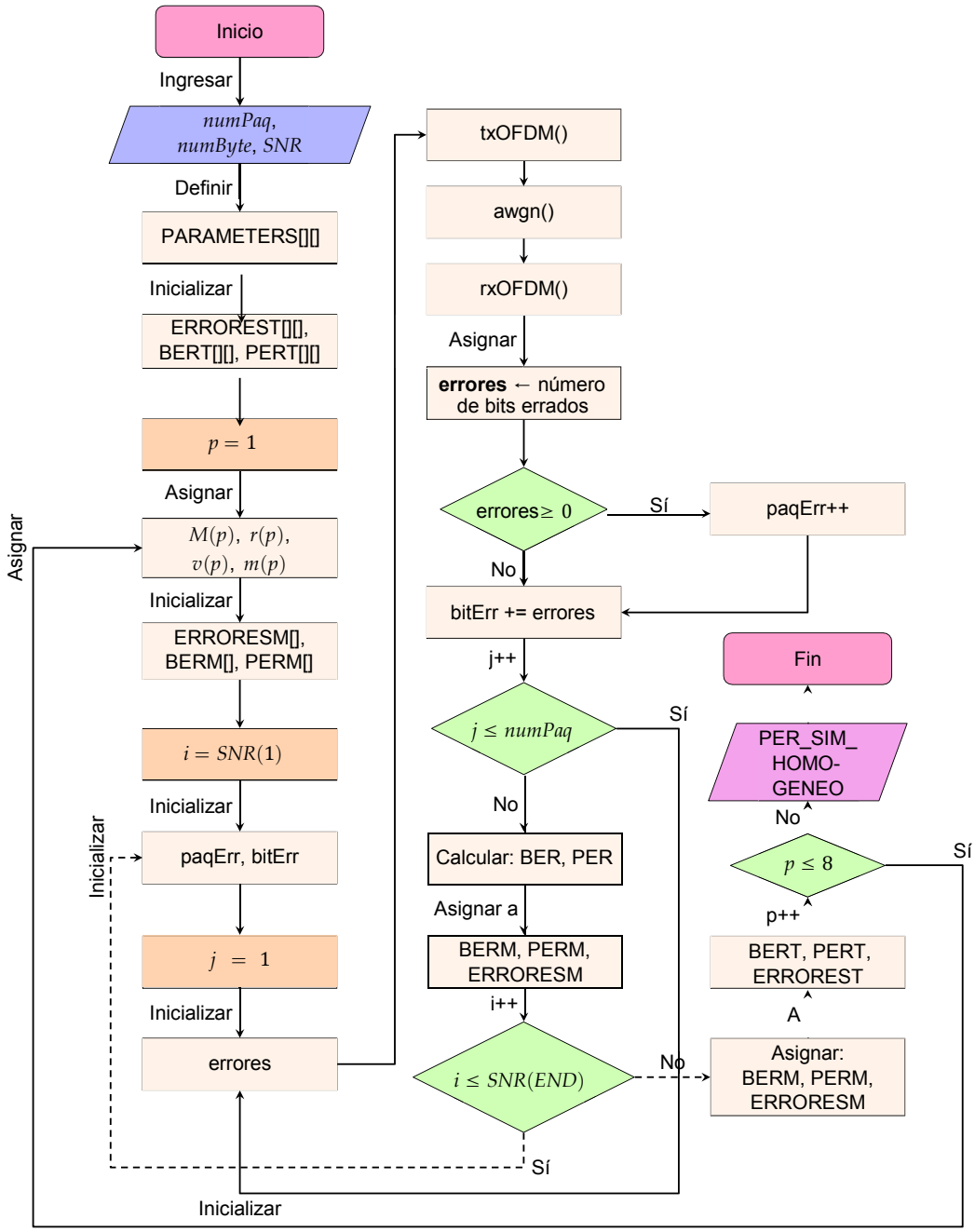


Figura I.2: Diagrama de flujo de la simulación principal (SNR homogéneo)

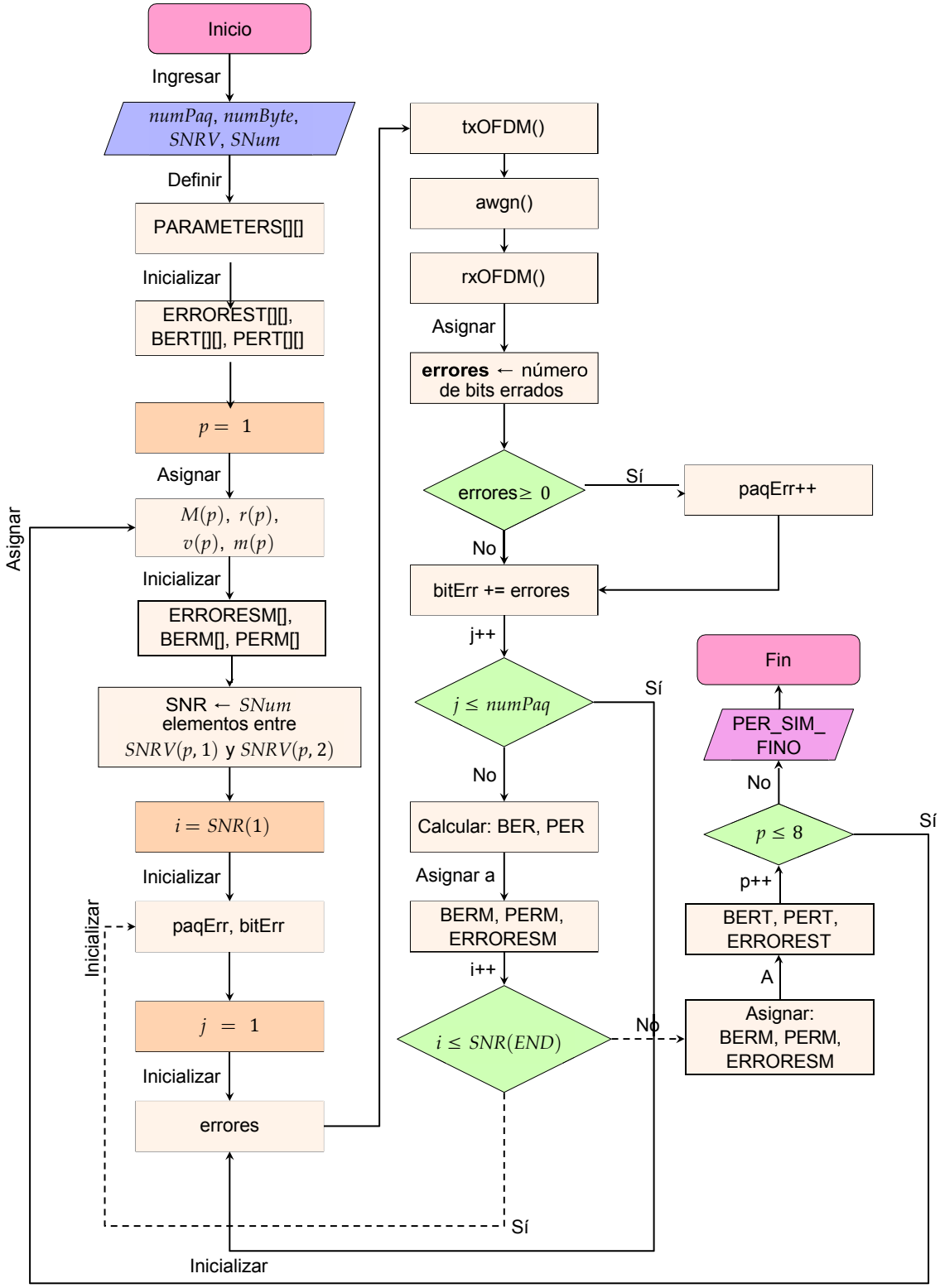


Figura I.3: Diagrama de flujo de la simulación principal (SNR angosto)

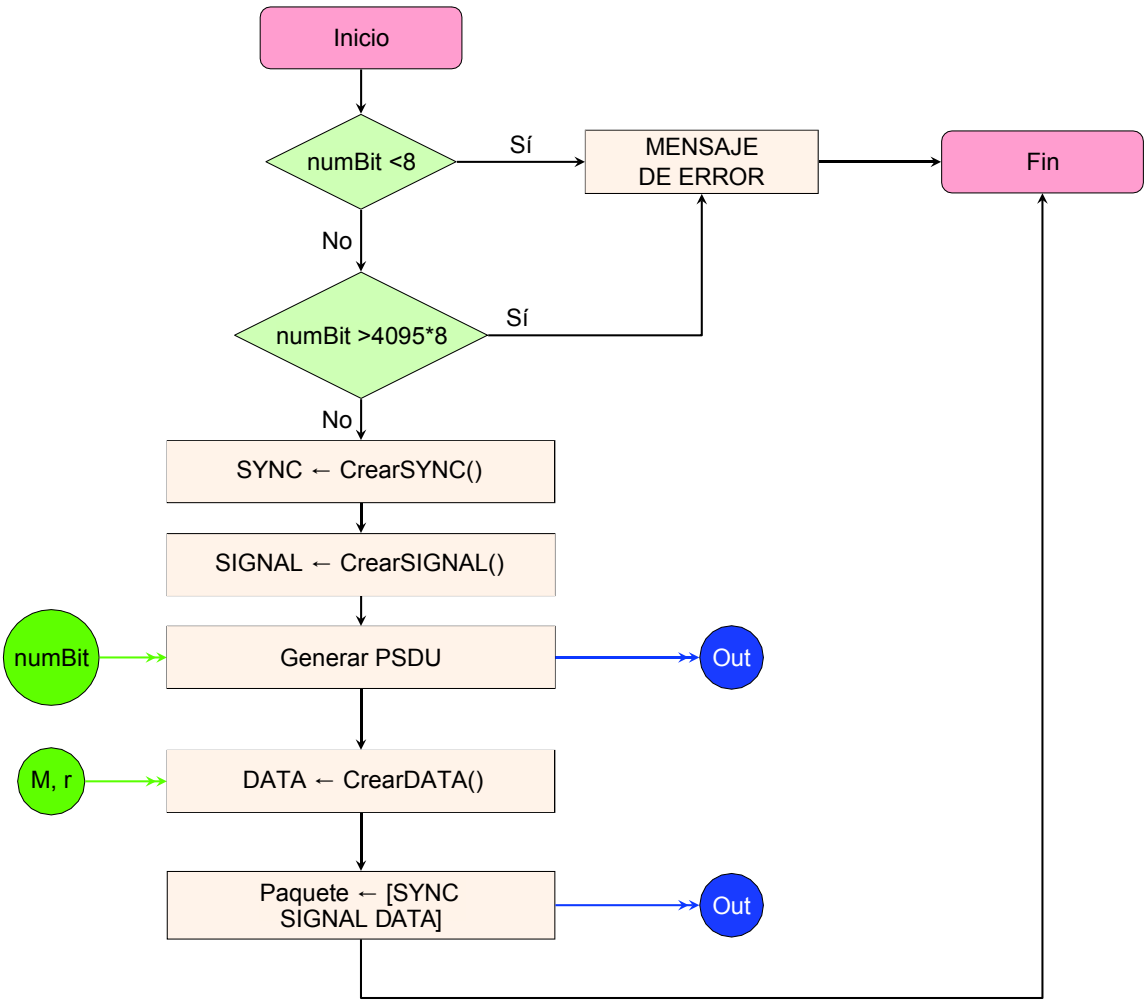


Figura I.4: Diagrama de flujo del módulo de Tx

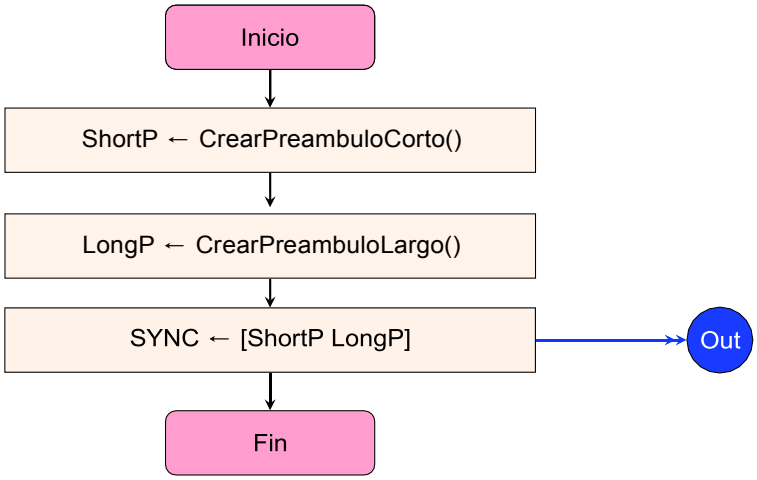


Figura I.5: Diagrama de flujo de la generación del SYNC

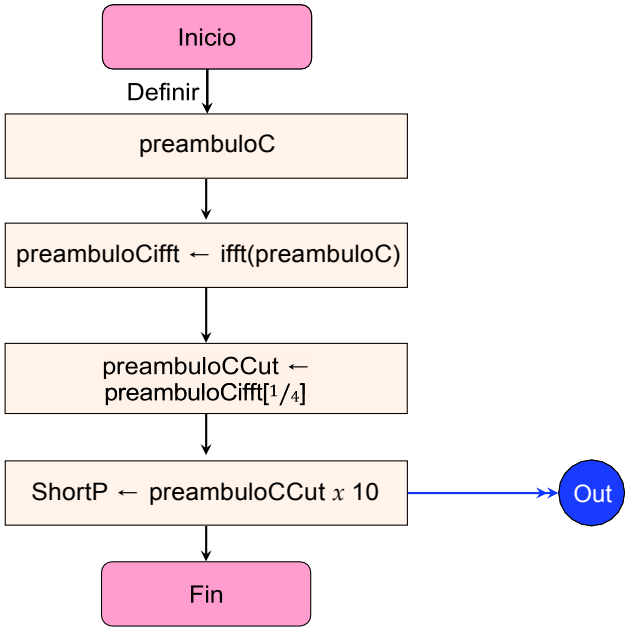


Figura I.6: Diagrama de flujo de la generación del preámbulo corto

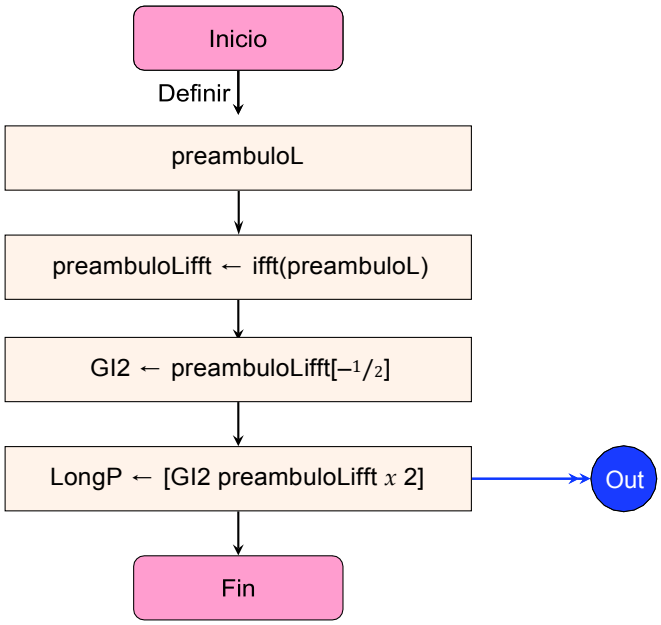


Figura I.7: Diagrama de flujo de la generación del preámbulo corto

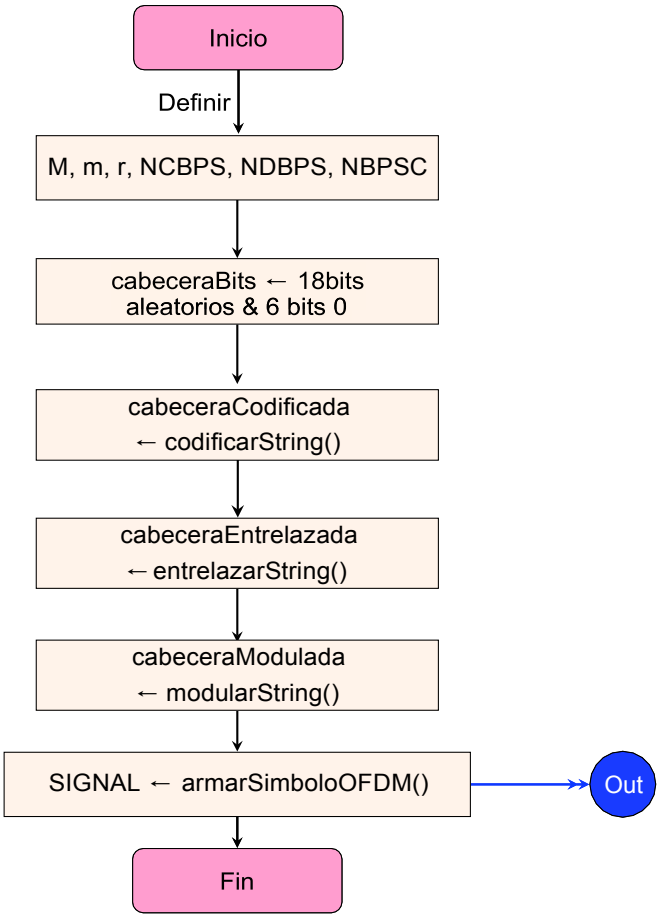


Figura I.8: Diagrama de flujo de la generación de SIGNAL

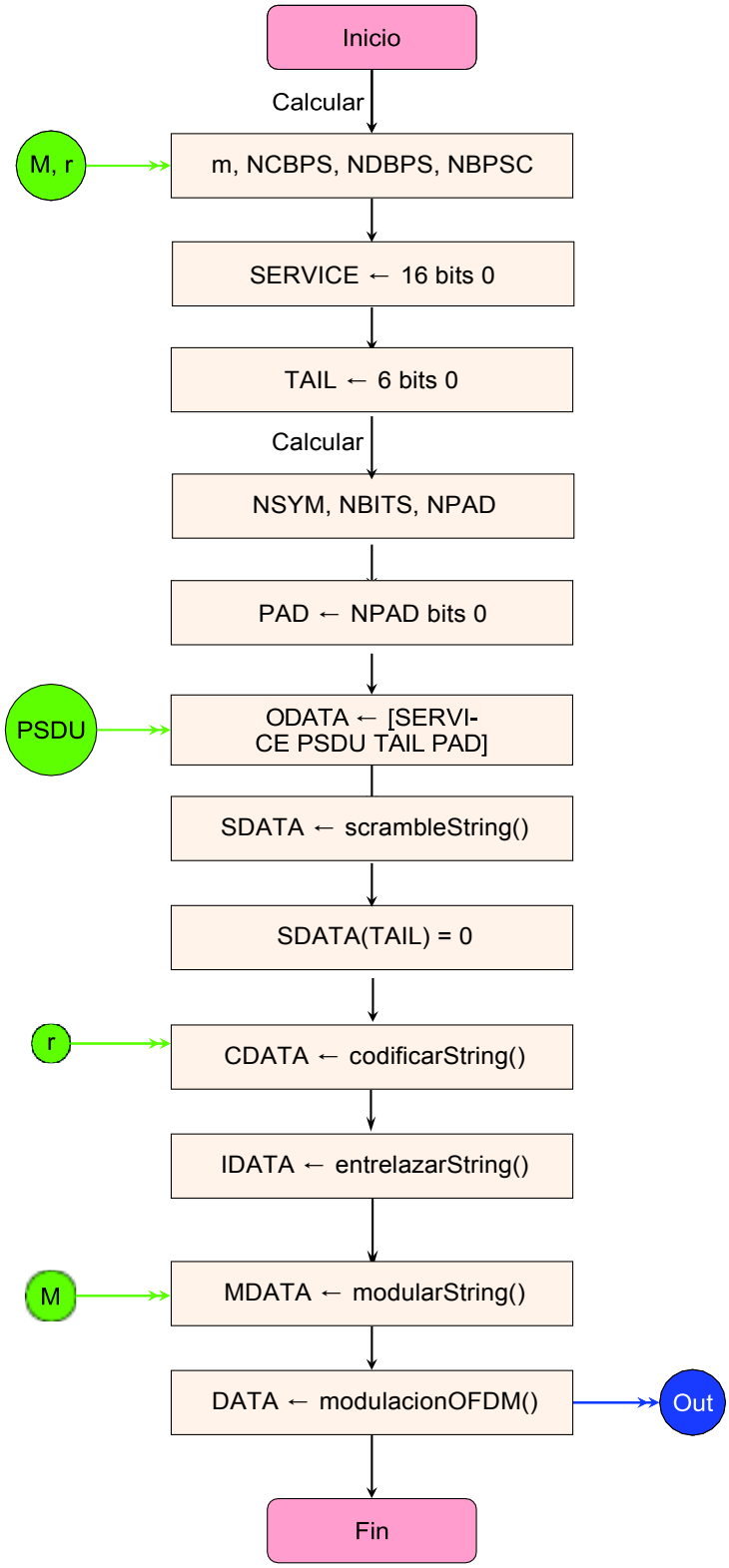


Figura I.9: Diagrama de flujo de la generación de DATA

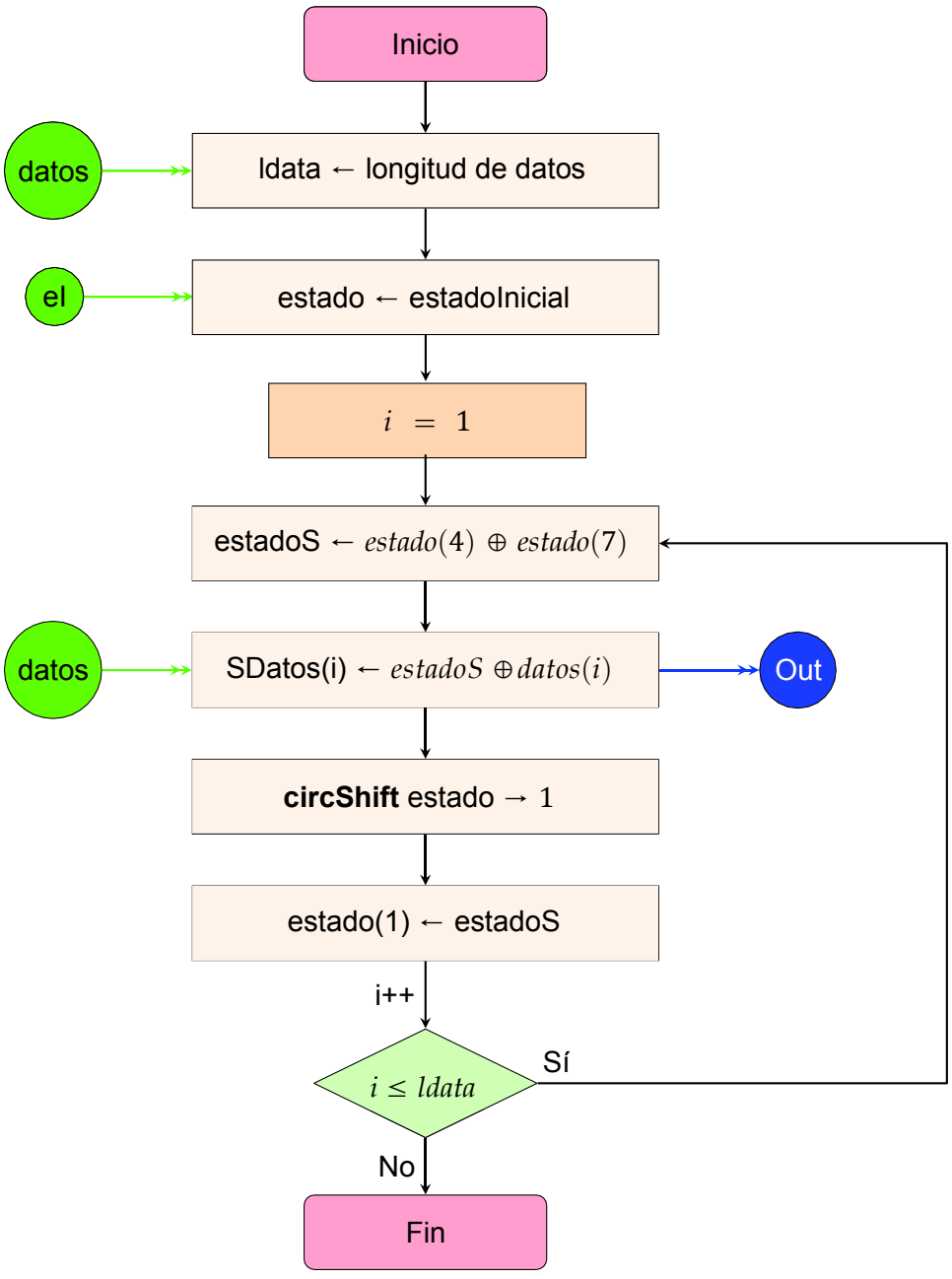


Figura I.10: Diagrama de flujo de la aleatorización

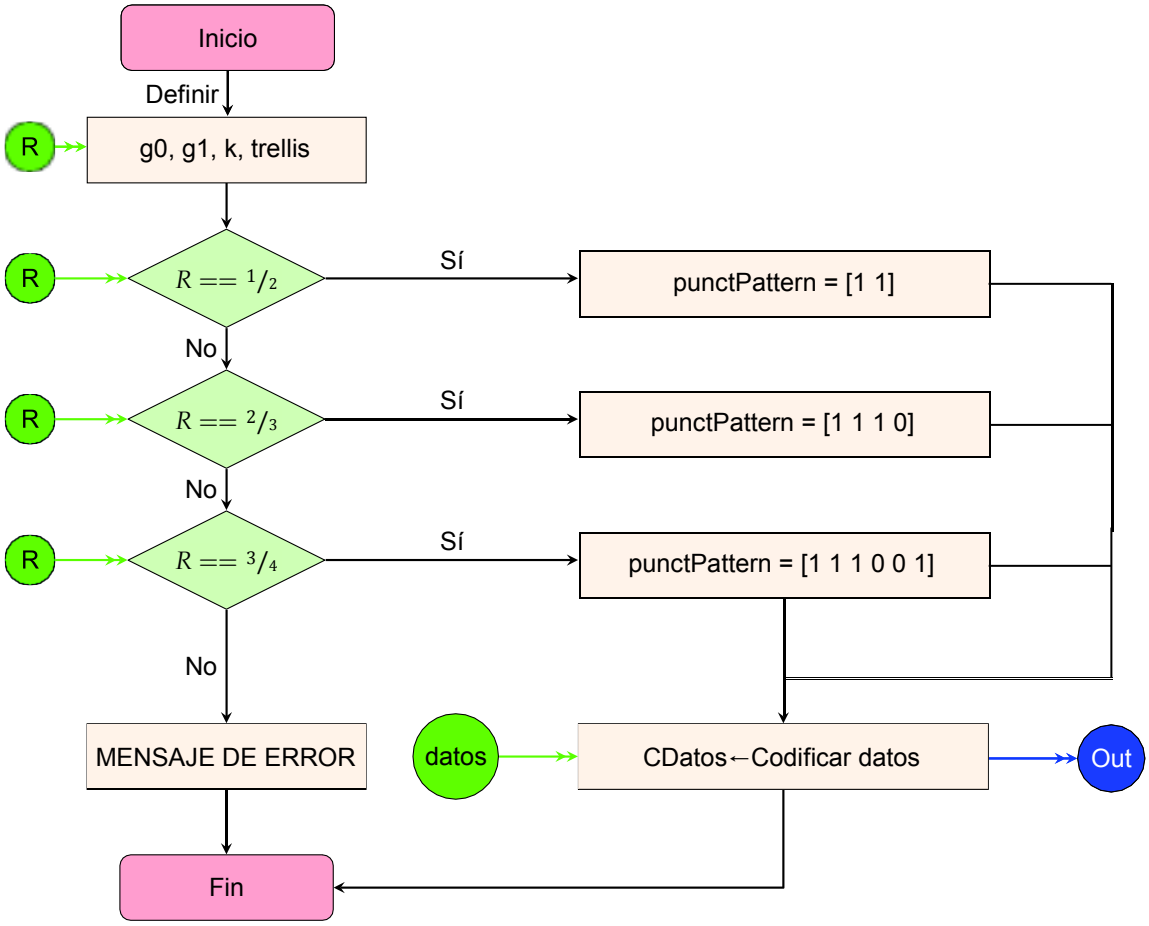


Figura I.11: Diagrama de flujo de la codificación

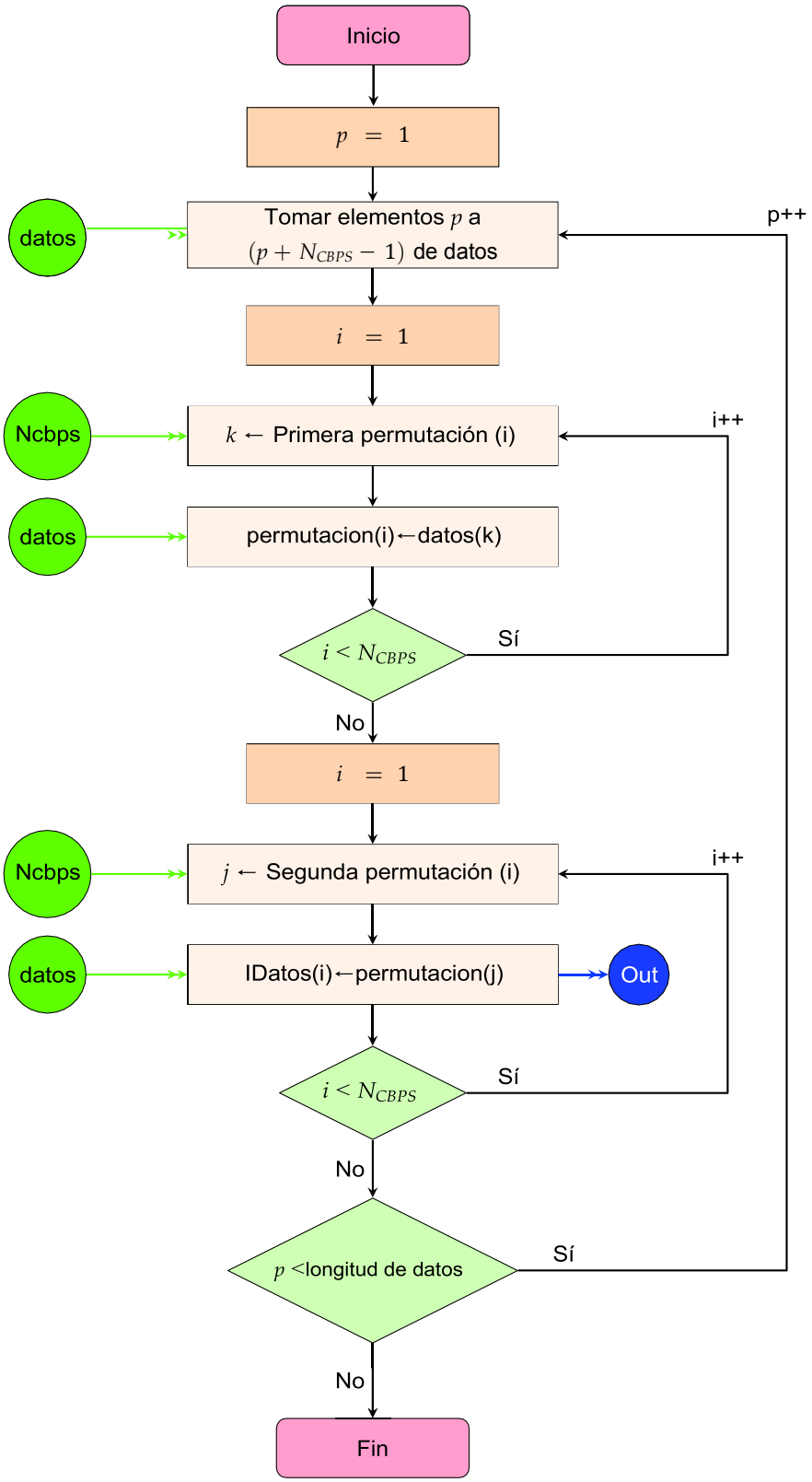


Figura I.12: Diagrama de flujo del entrelazado

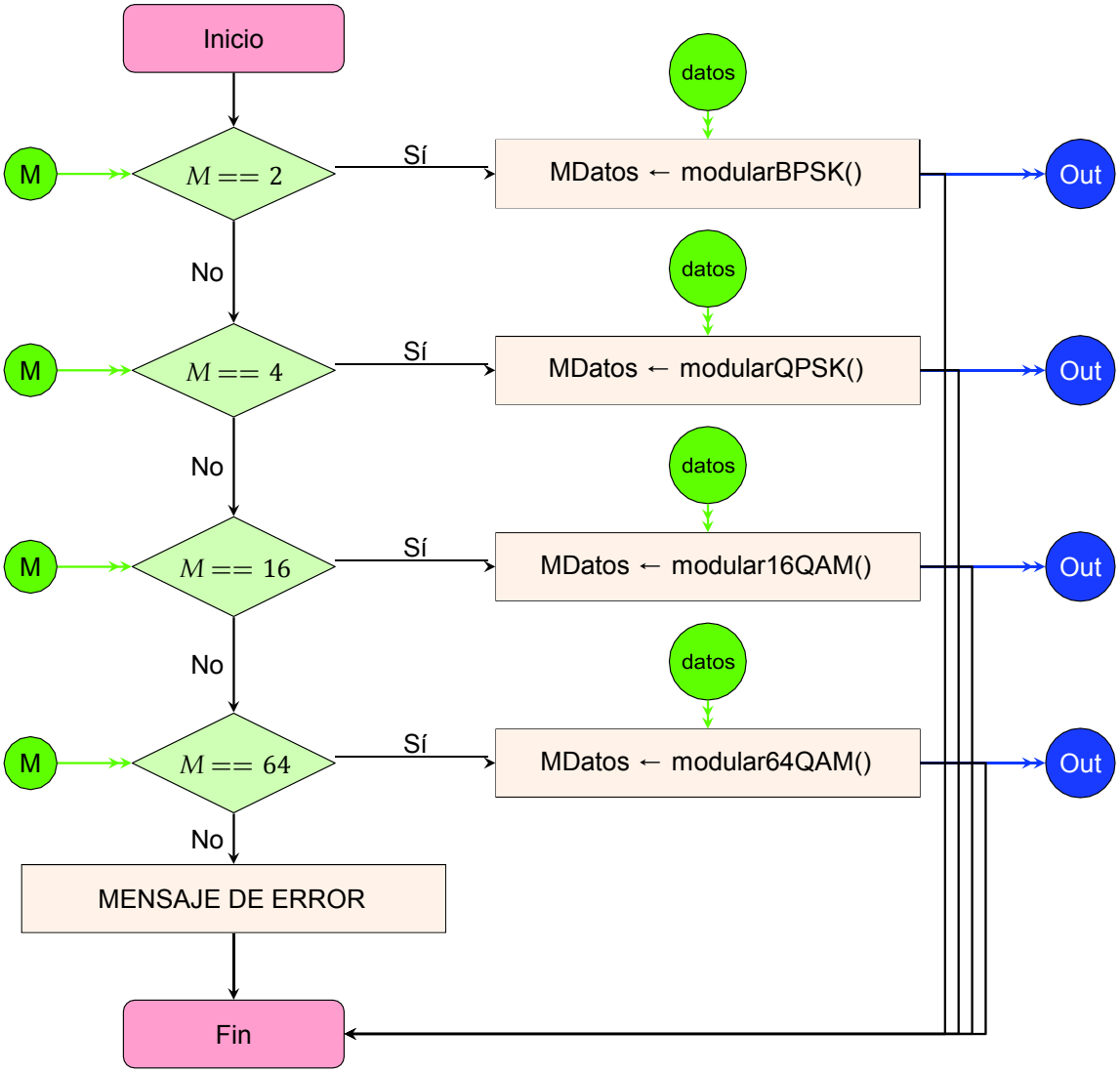


Figura I.13: Diagrama de flujo del modulado

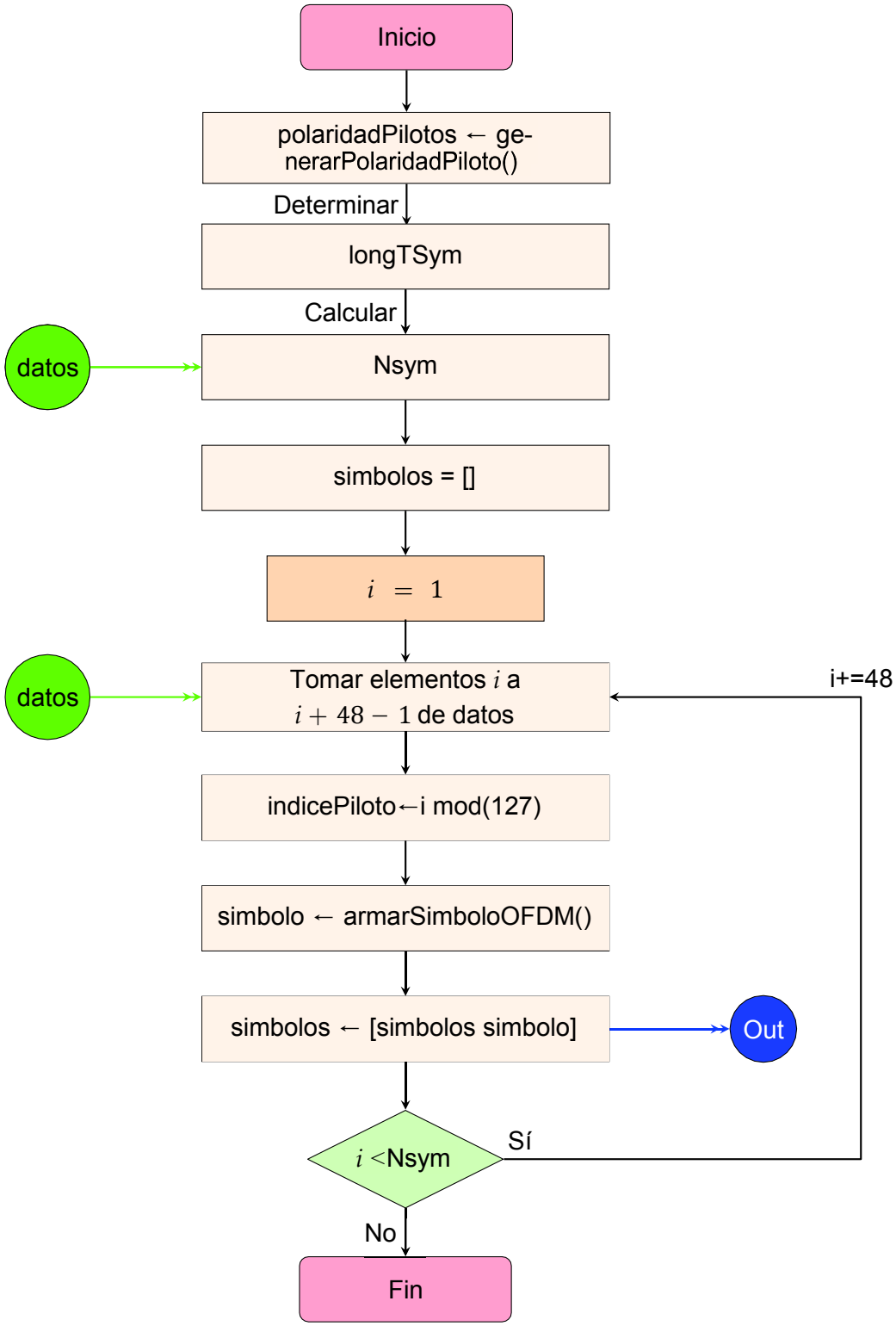


Figura I.14: Diagrama de flujo de la modulación OFDM

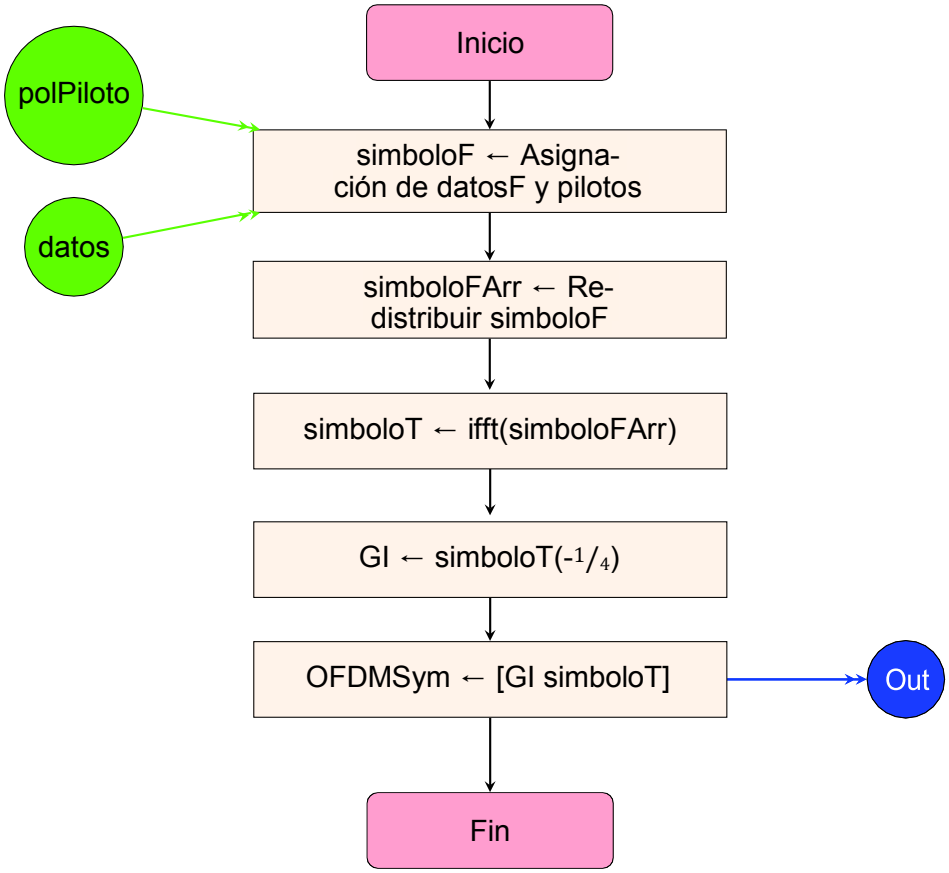


Figura I.15: Diagrama de flujo de la formación de símbolo OFDM

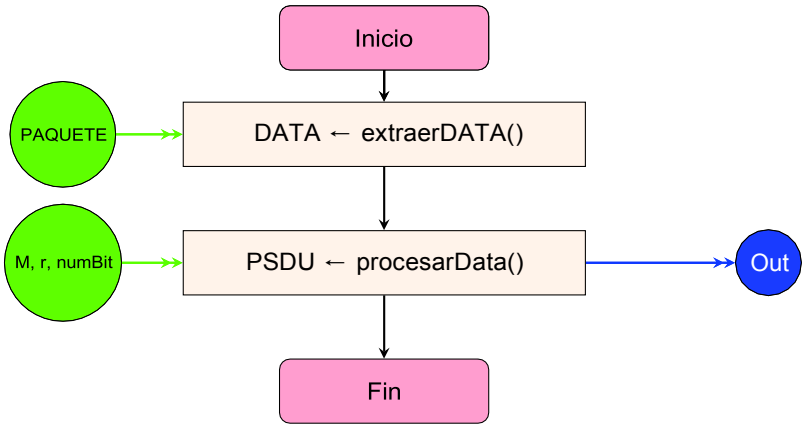


Figura I.16: Diagrama de flujo del módulo de Rx

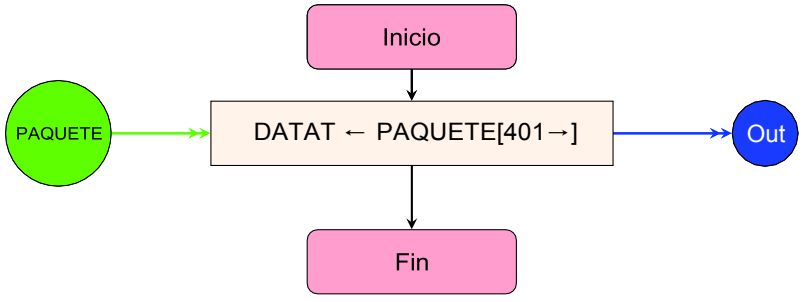


Figura I.17: Diagrama de flujo de la extracción del DATA

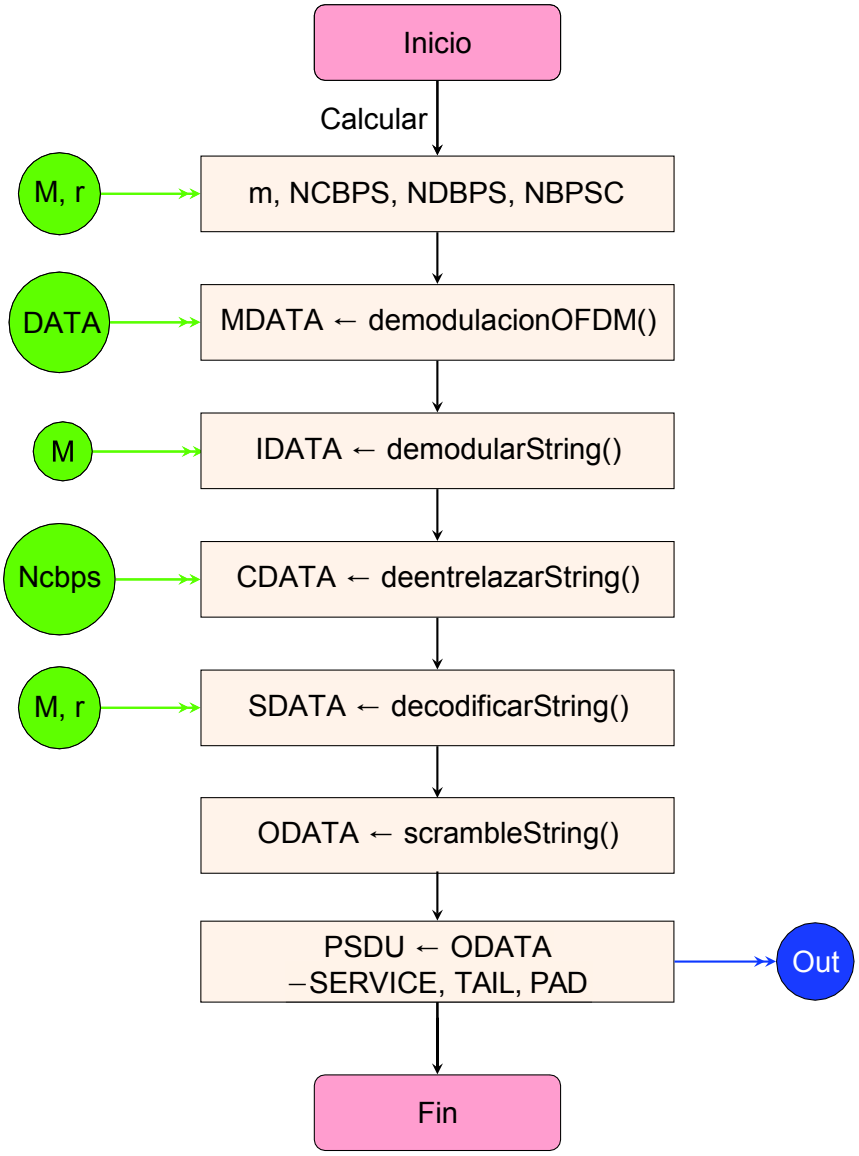


Figura I.18: Diagrama de flujo del procesamiento de DATA

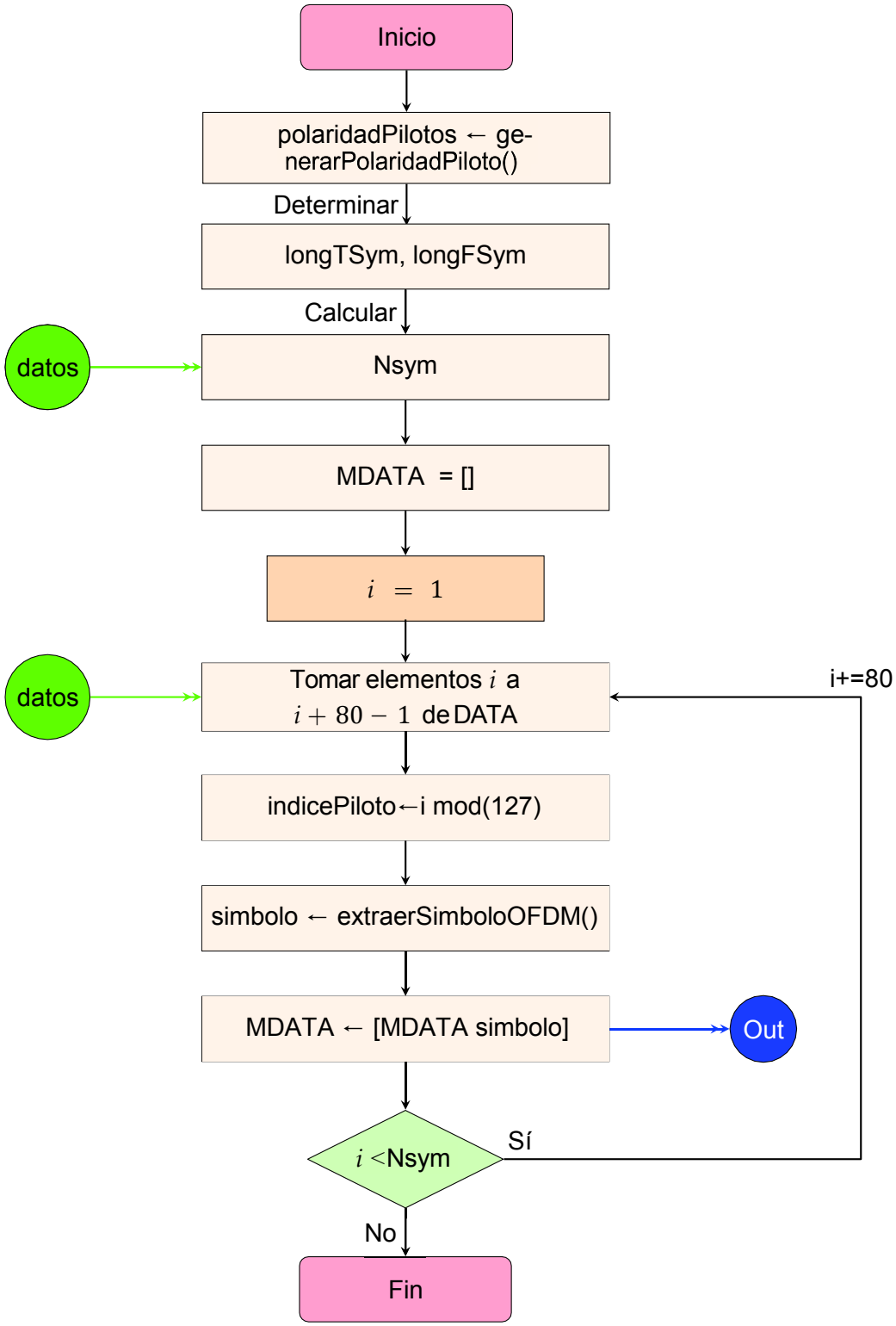


Figura I.19: Diagrama de flujo de la demodulación OFDM

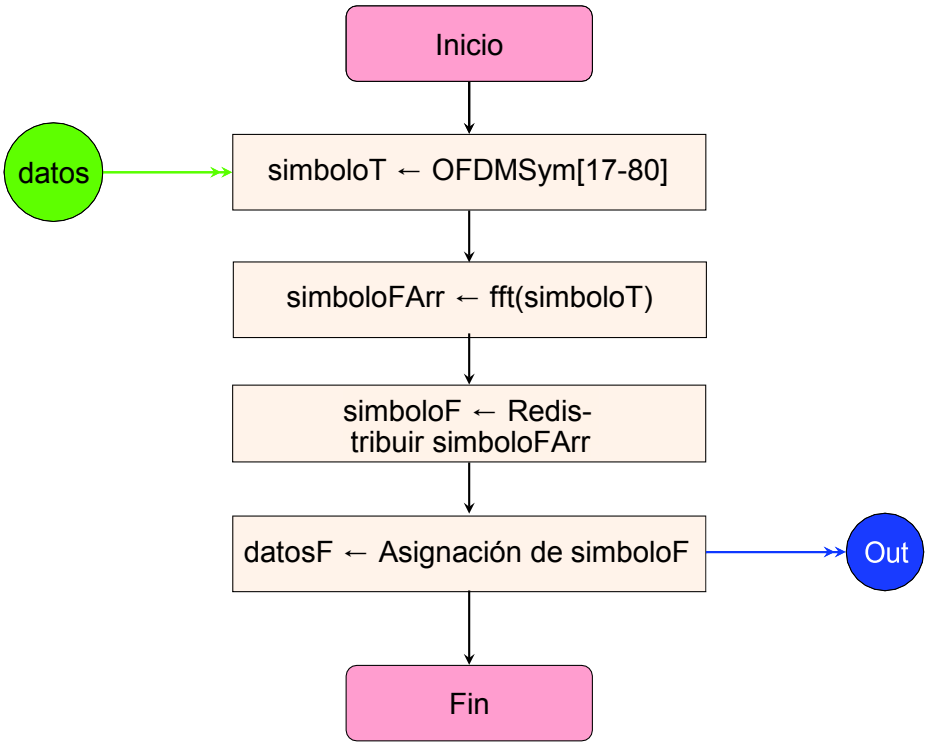


Figura I.20: Diagrama de flujo del mapeo de símbolo OFDM

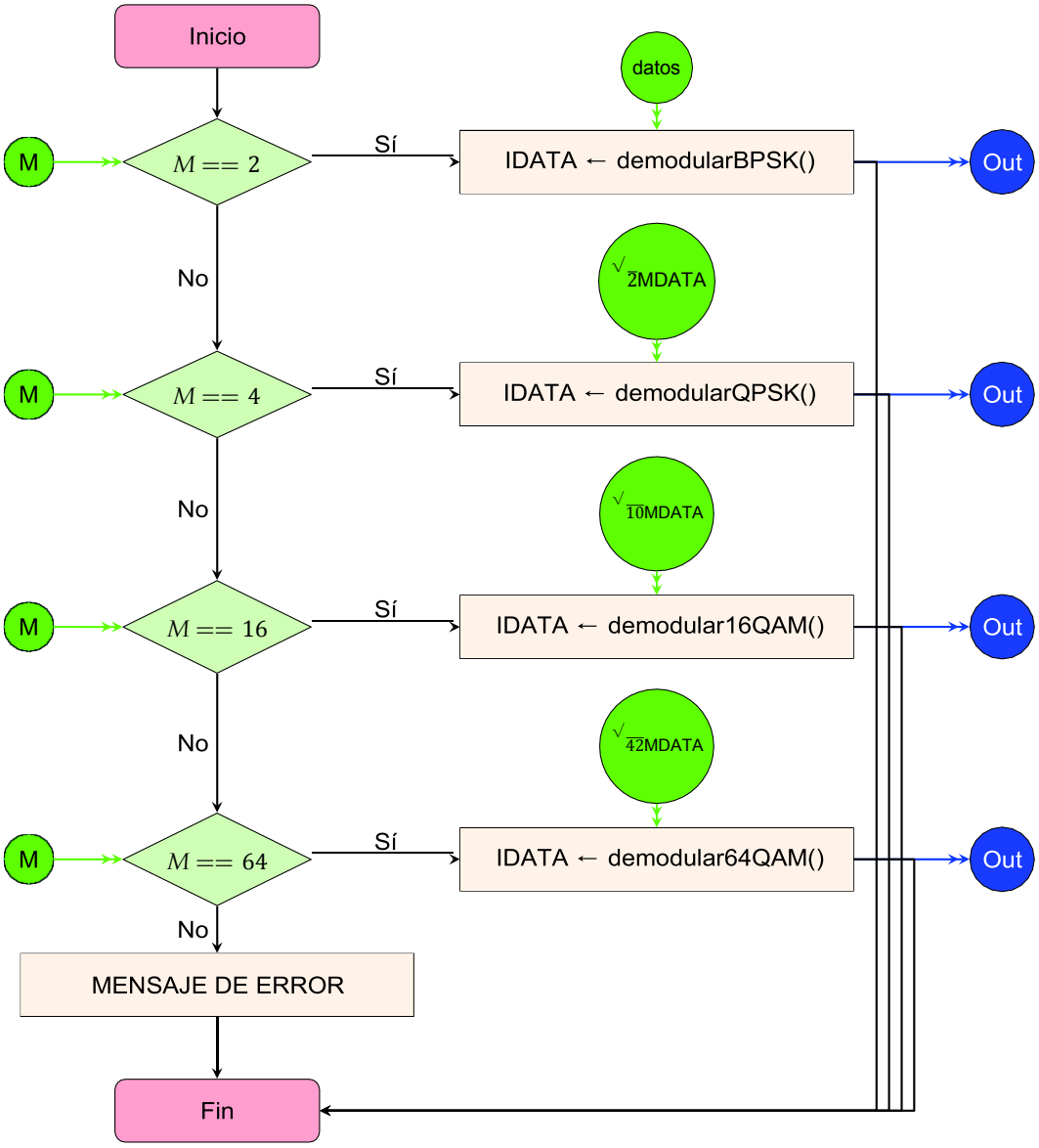


Figura I.21: Diagrama de flujo de la demodulación

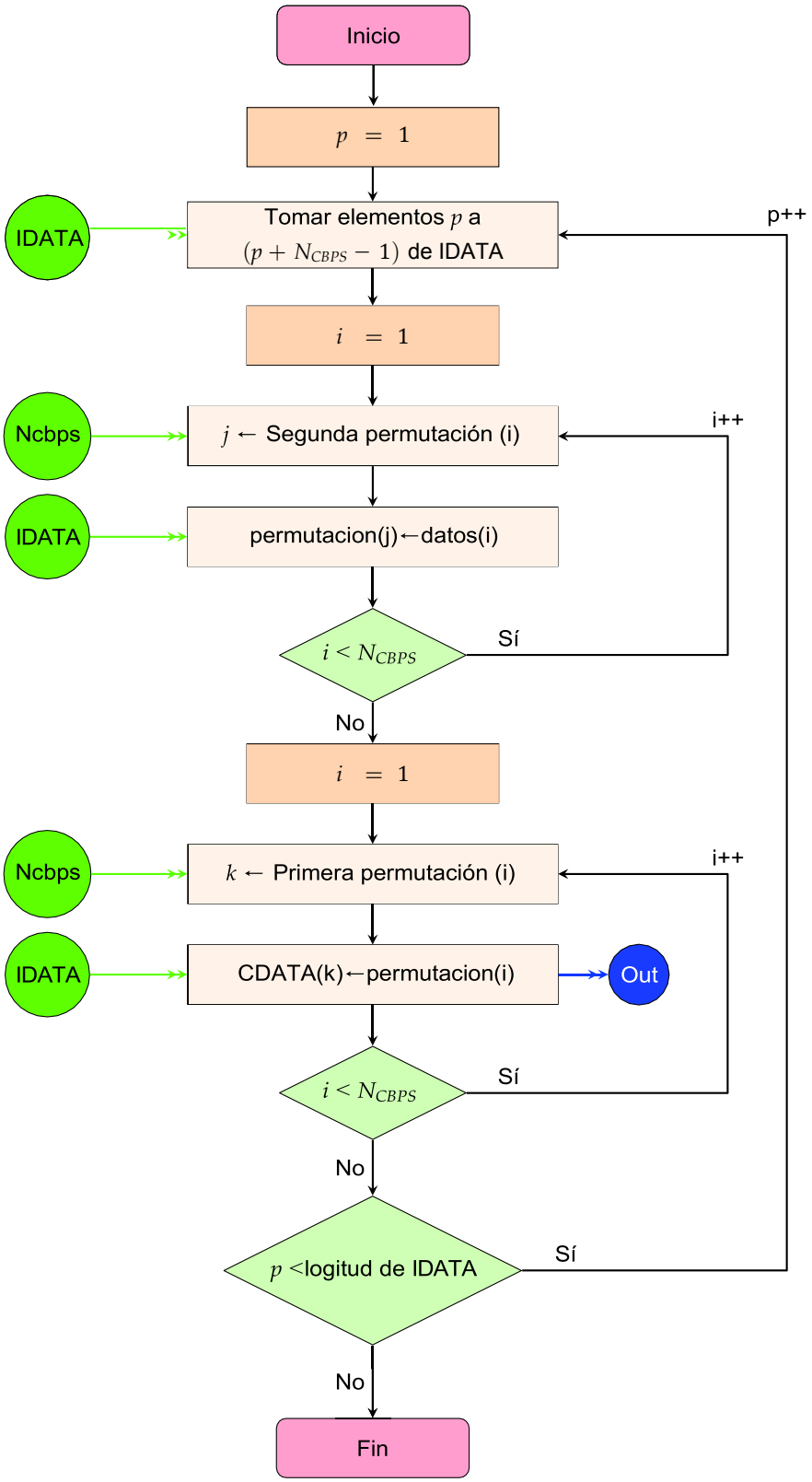


Figura I.22: Diagrama de flujo del deentrelazado

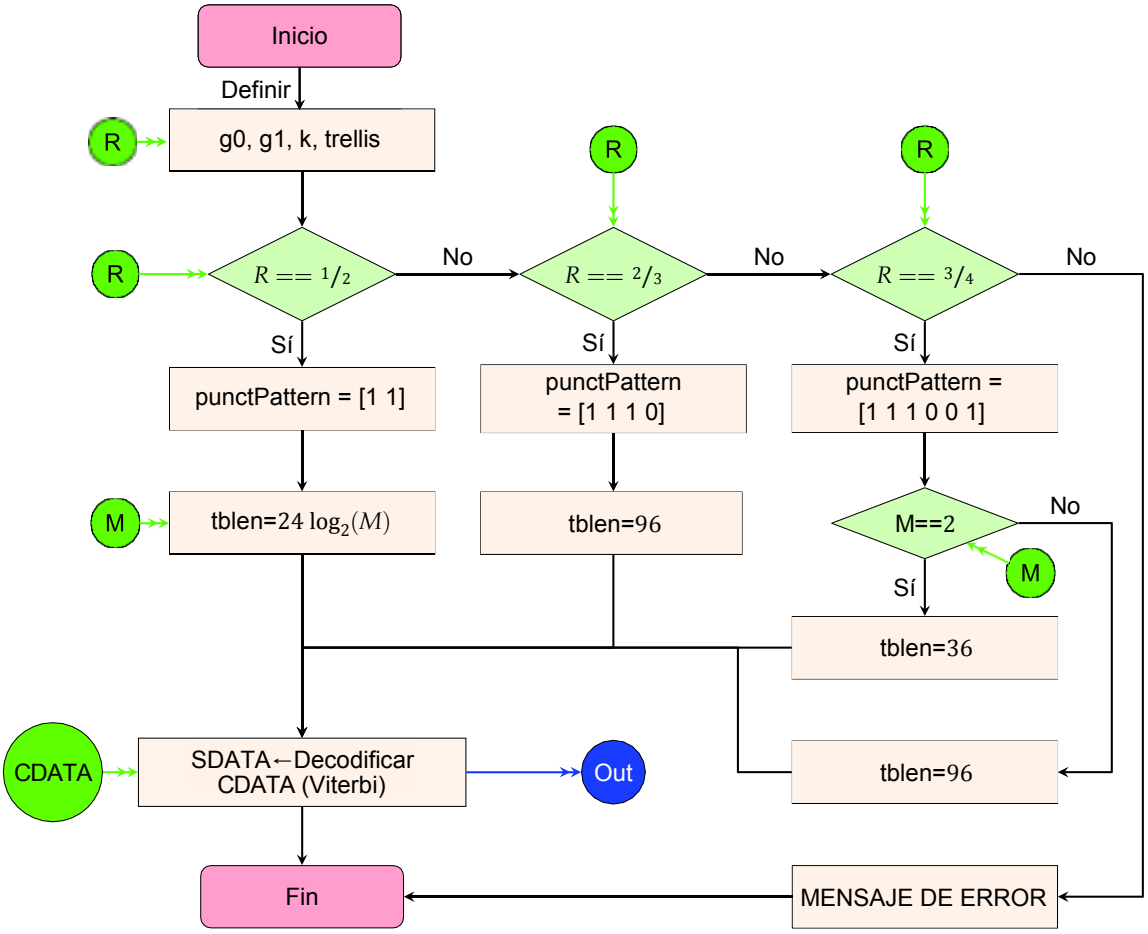


Figura I.23: Diagrama de flujo de la decodificación

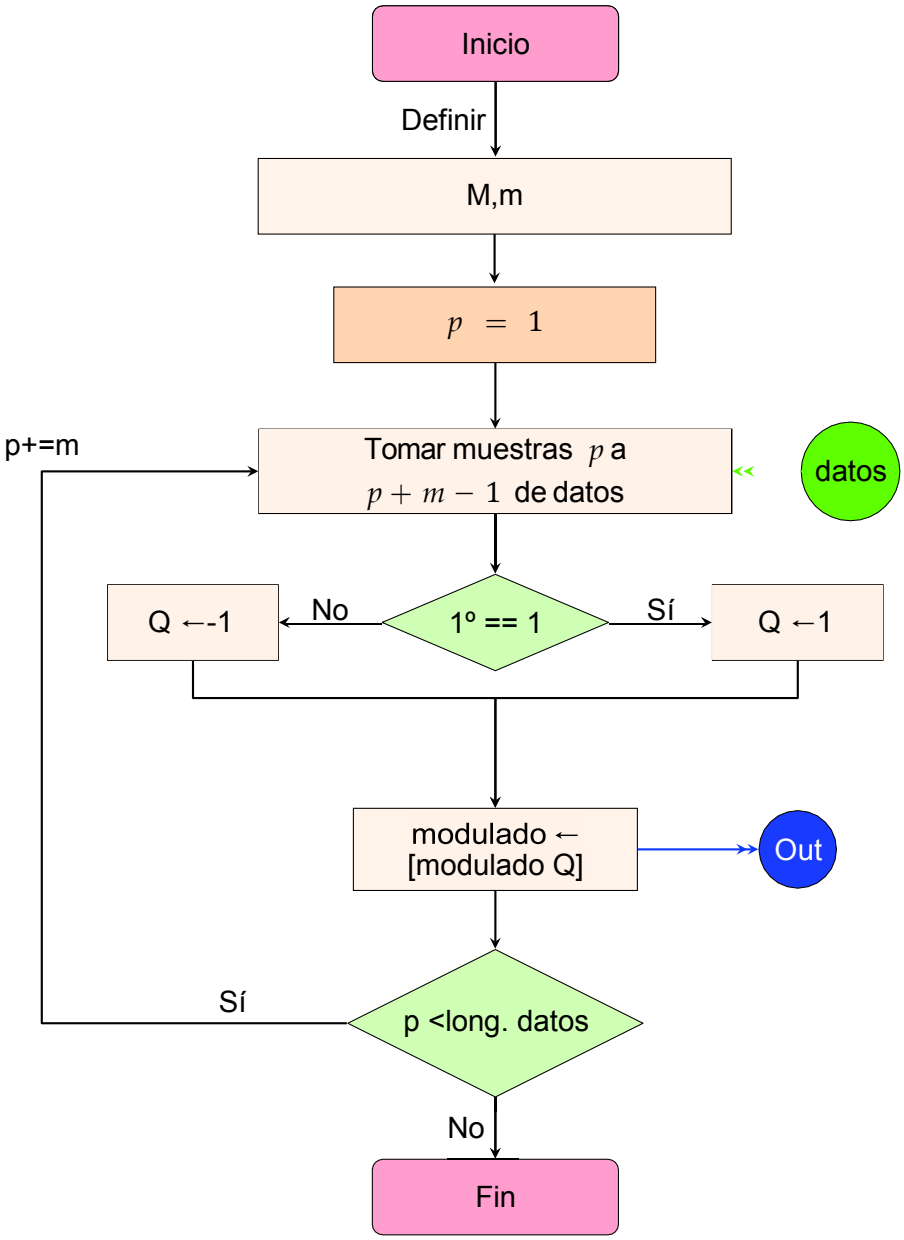


Figura I.24: Diagrama de flujo de modulación BPSK

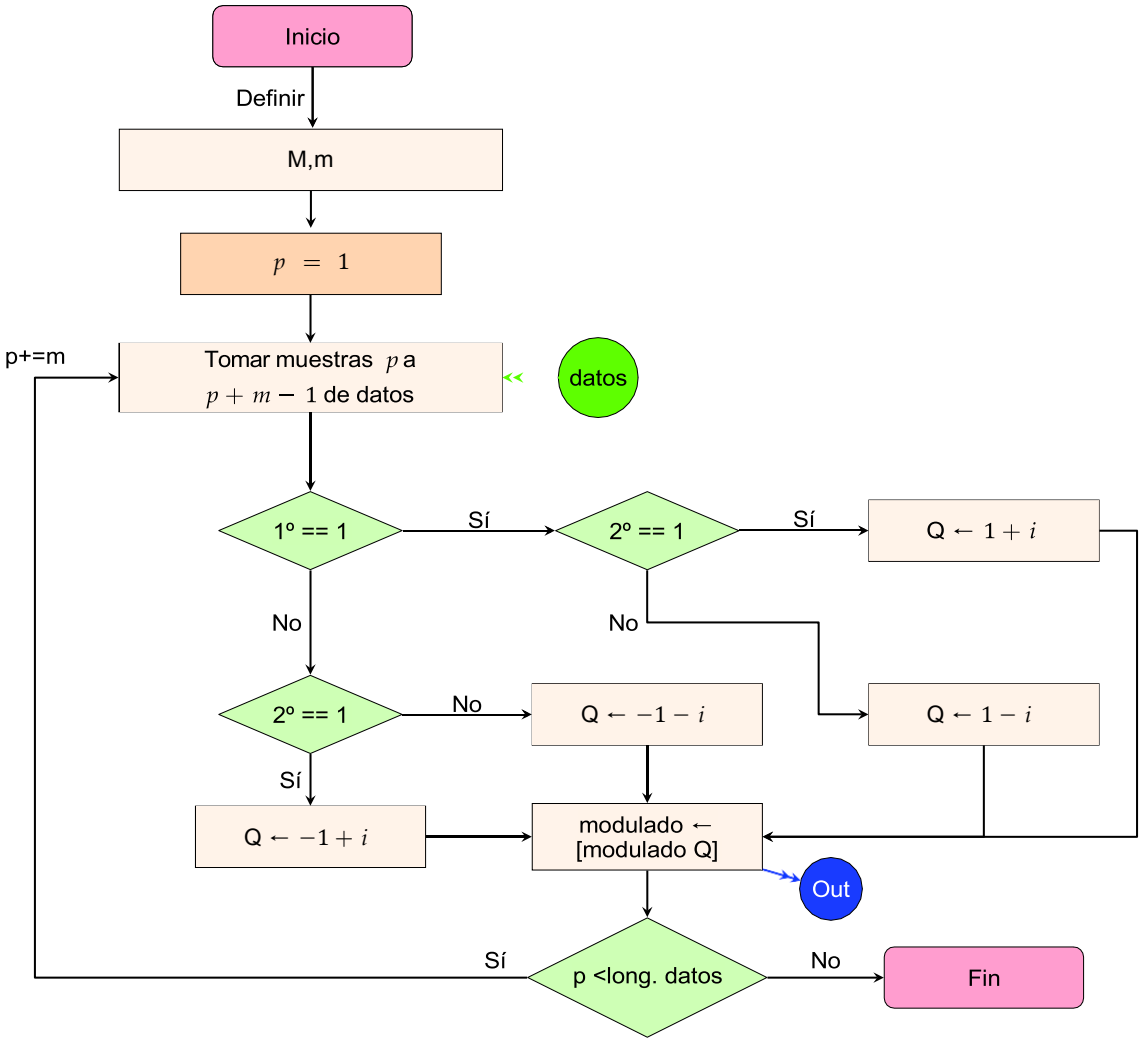


Figura I.25: Diagrama de flujo de modulación QPSK

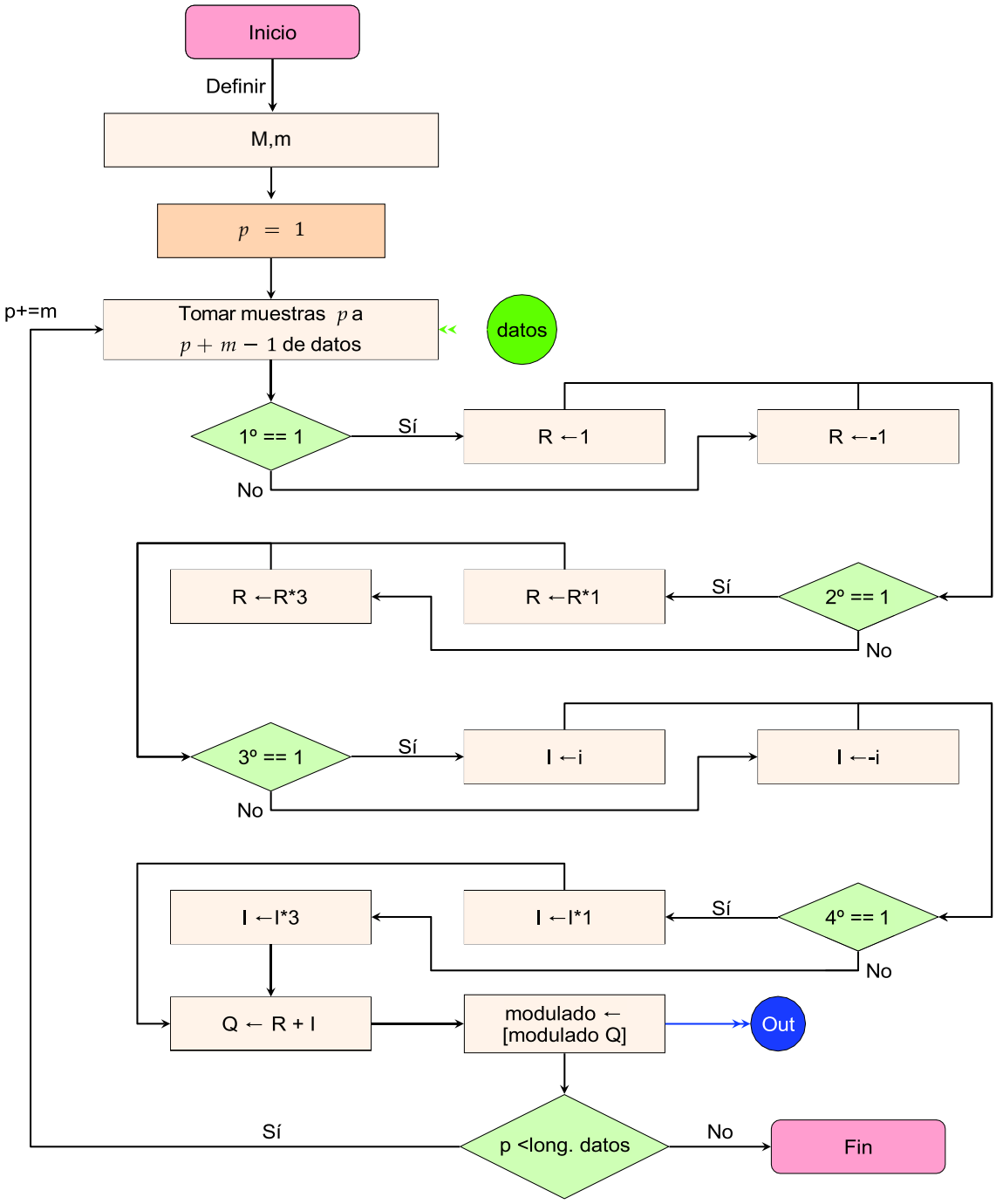


Figura I.26: Diagrama de flujo de modulación 16QAM

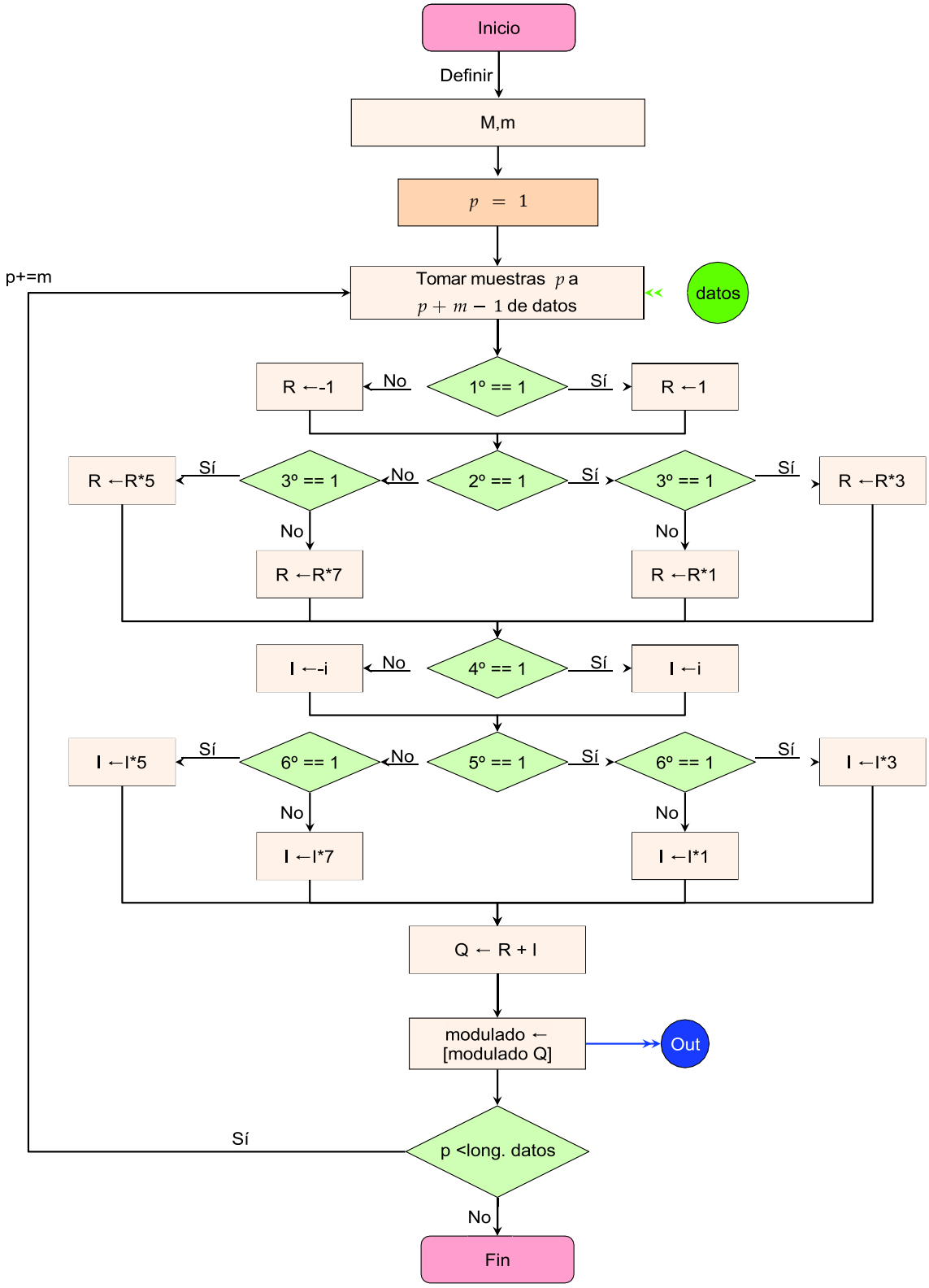


Figura I.27: Diagrama de flujo de modulación 64QAM

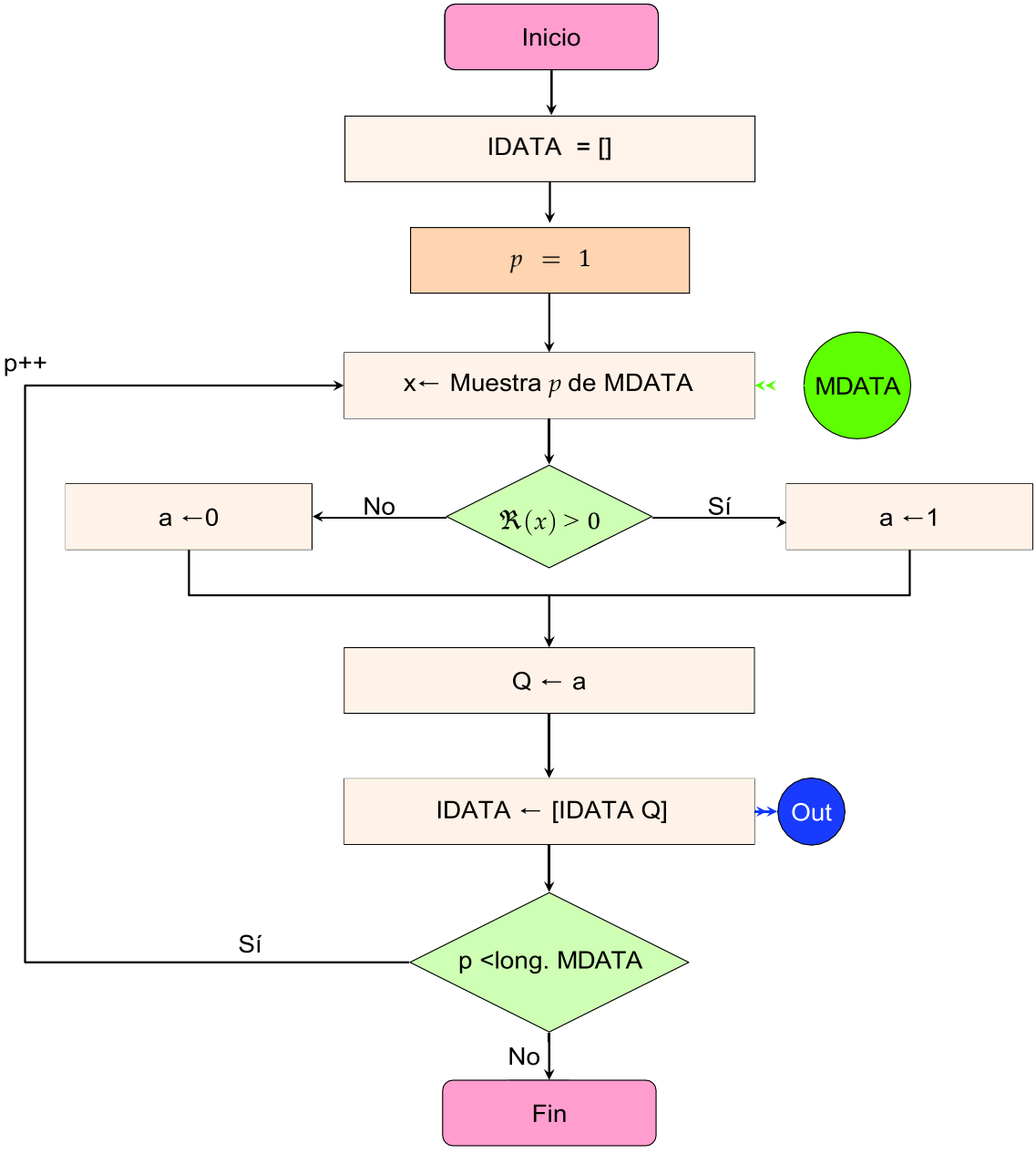


Figura I.28: Diagrama de flujo de demodulación BPSK

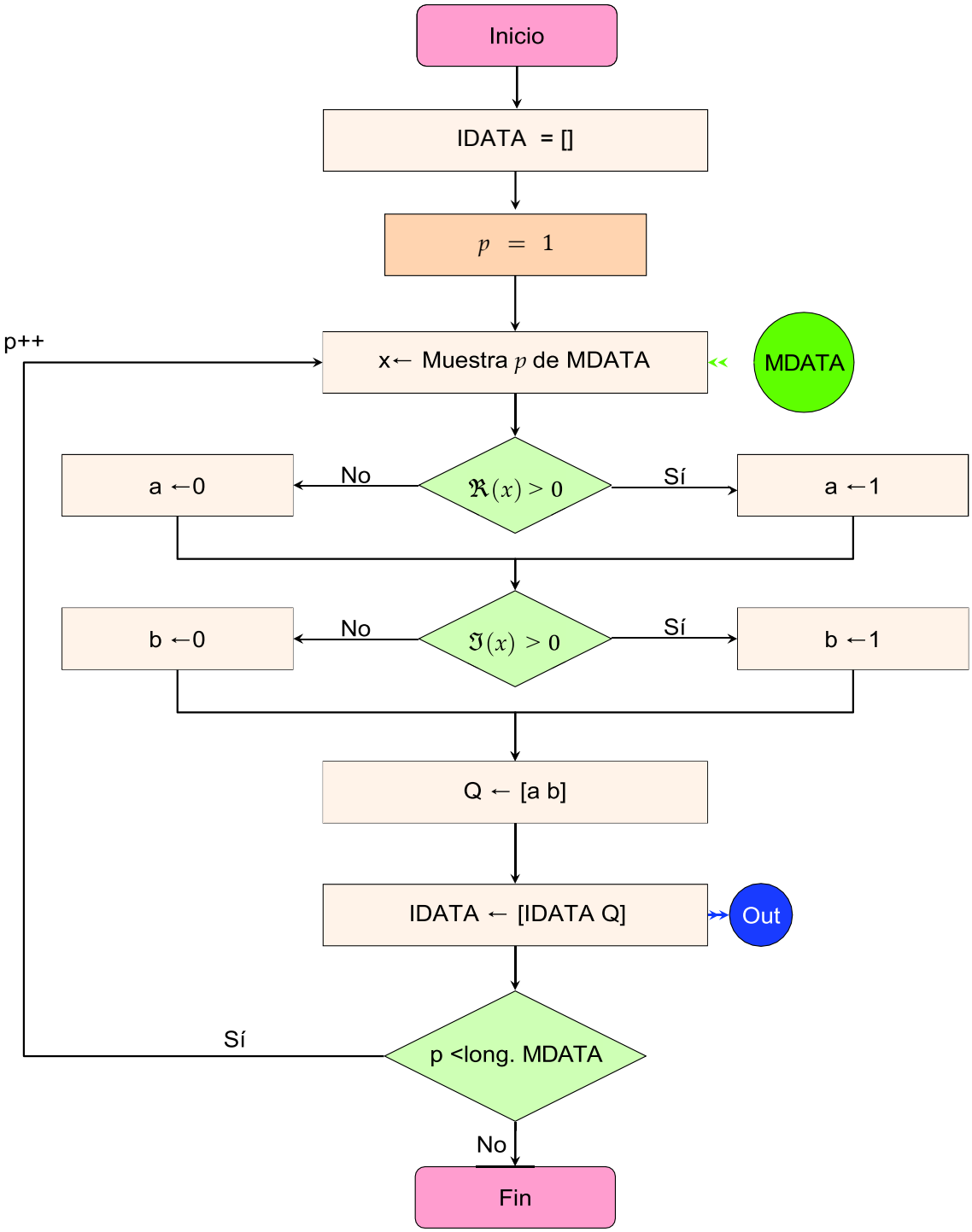


Figura I.29: Diagrama de flujo de demodulación QPSK

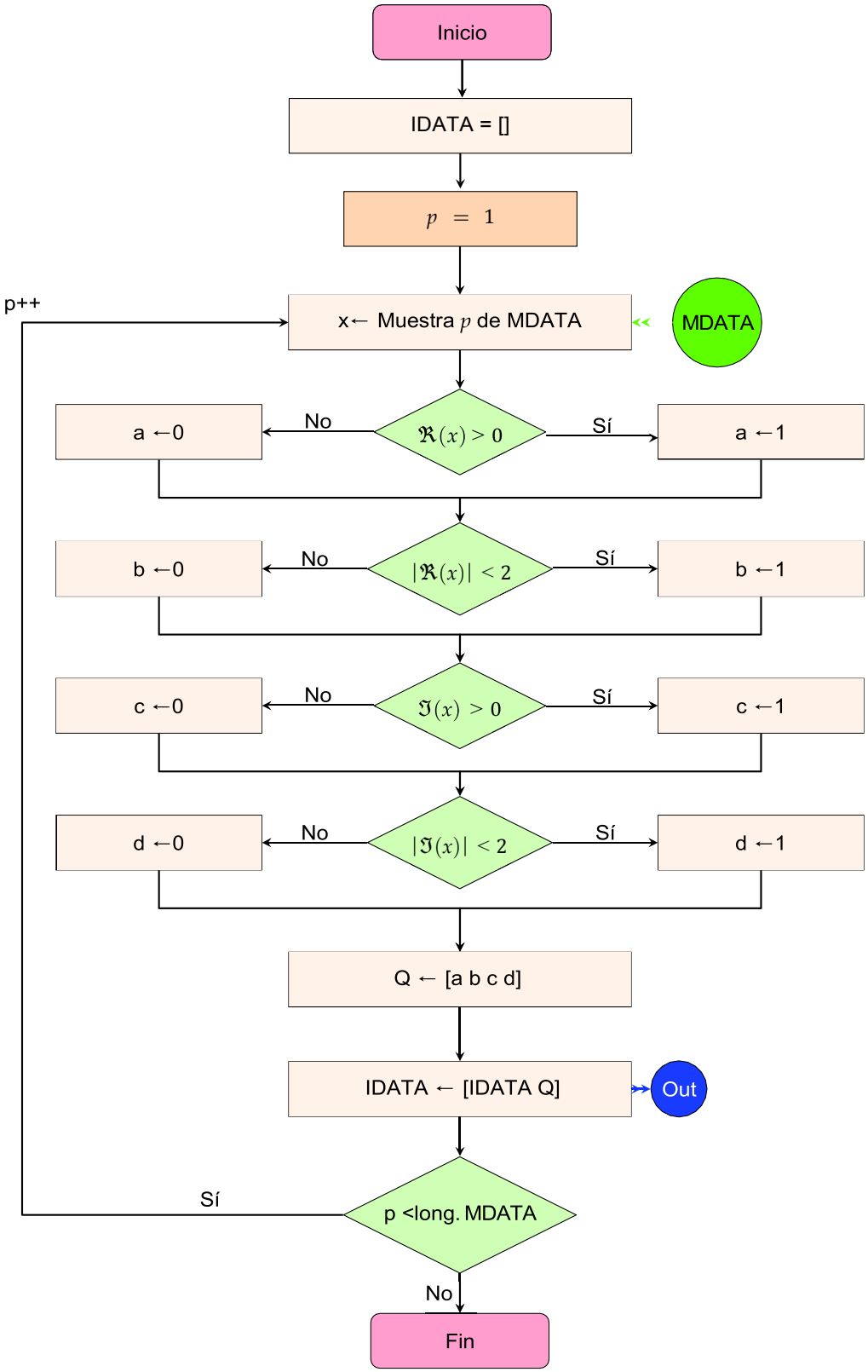


Figura I.30: Diagrama de flujo de demodulación 16QAM

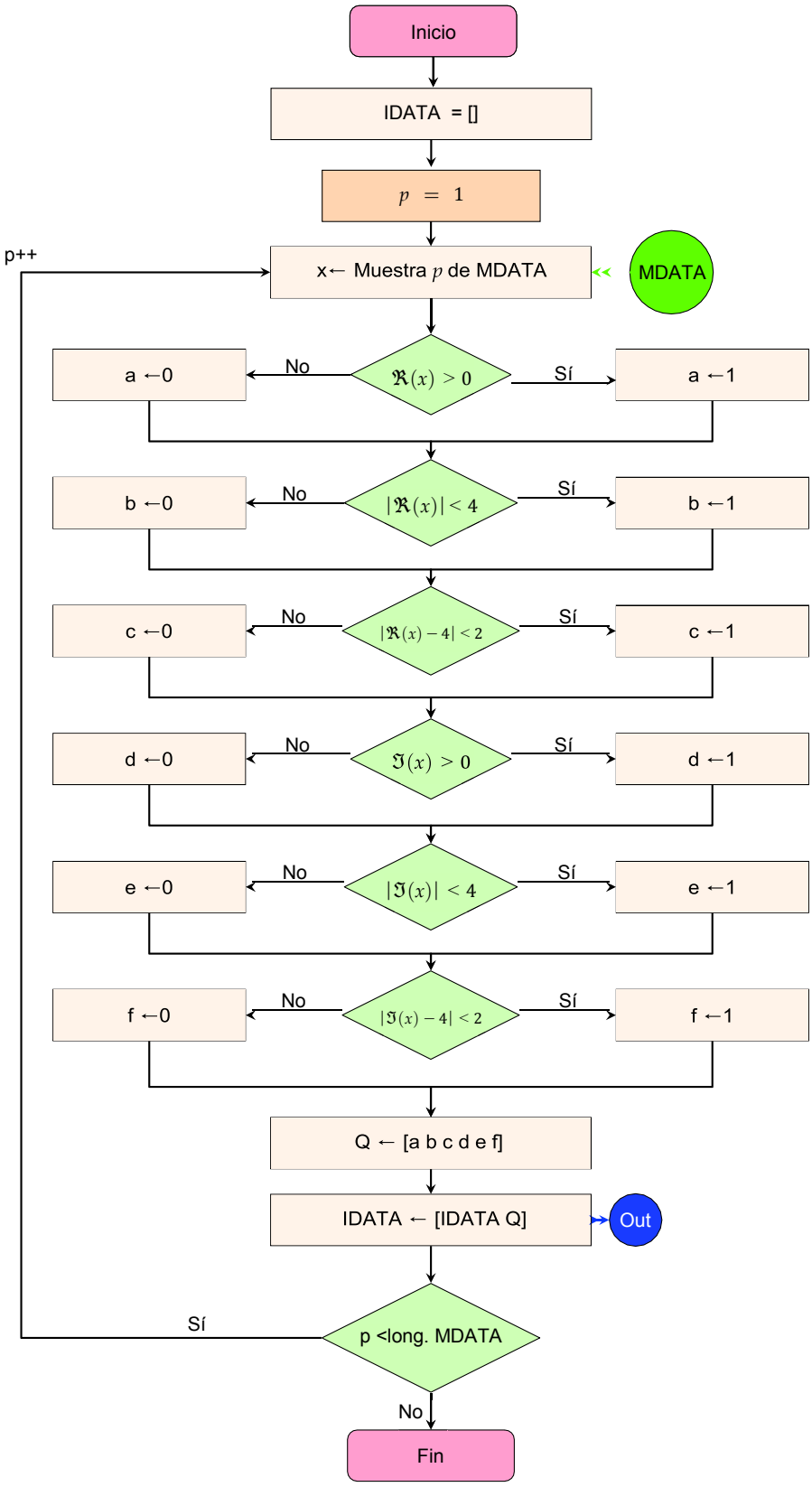


Figura I.31: Diagrama de flujo de demodulación 64QAM

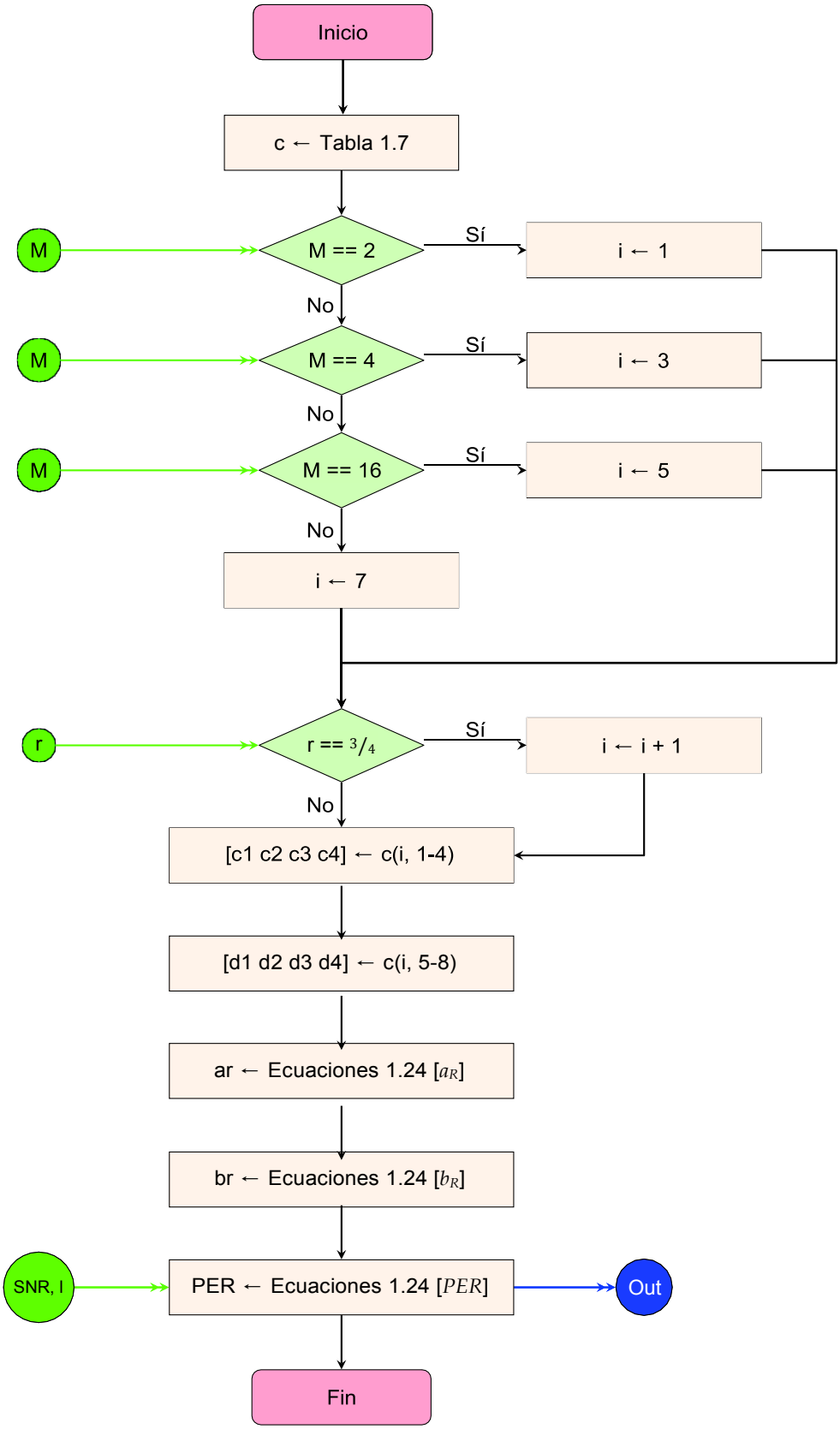


Figura I.32: Diagrama de flujo del modelo teórico 1

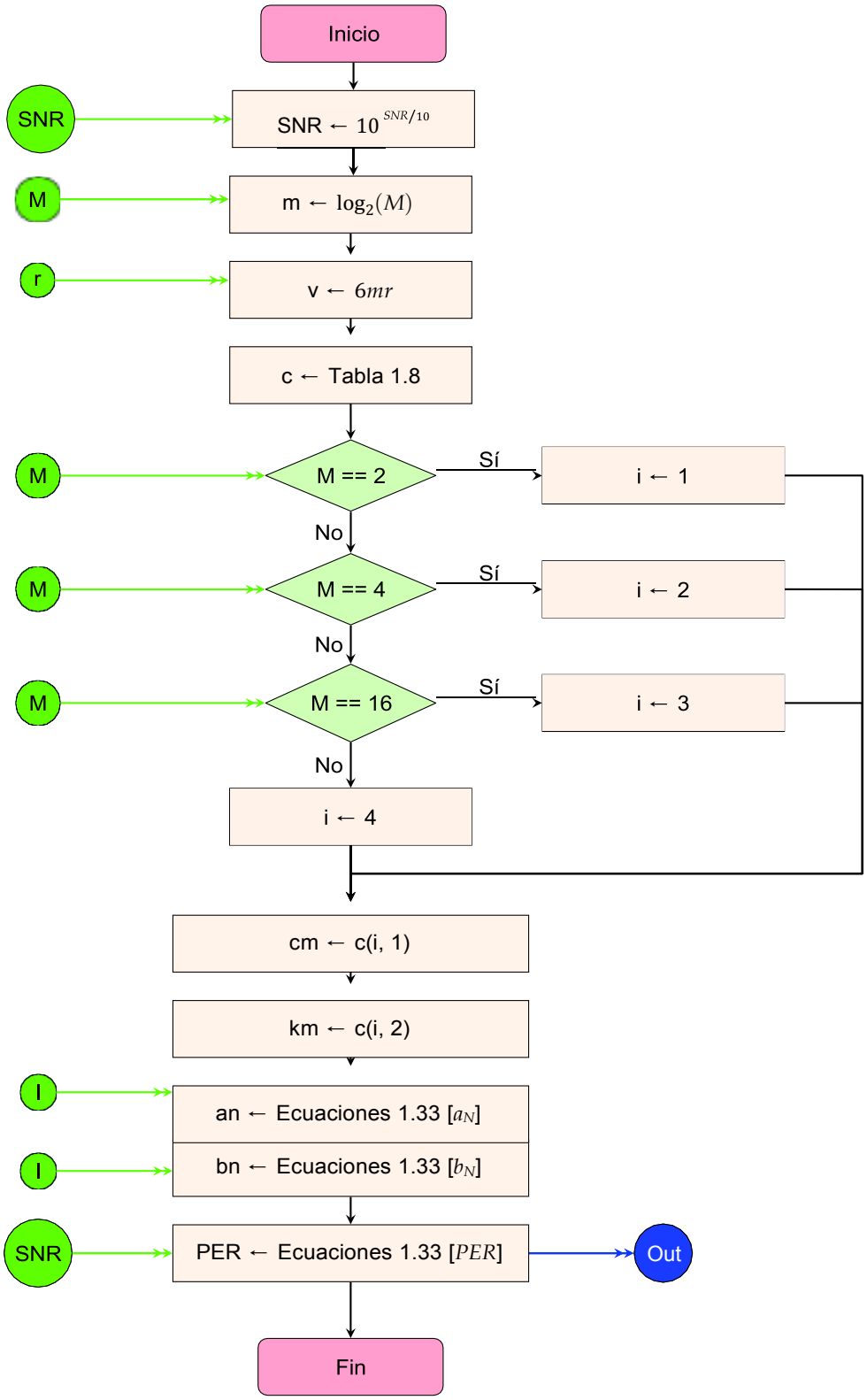


Figura I.33: Diagrama de flujo del modelo teórico 2

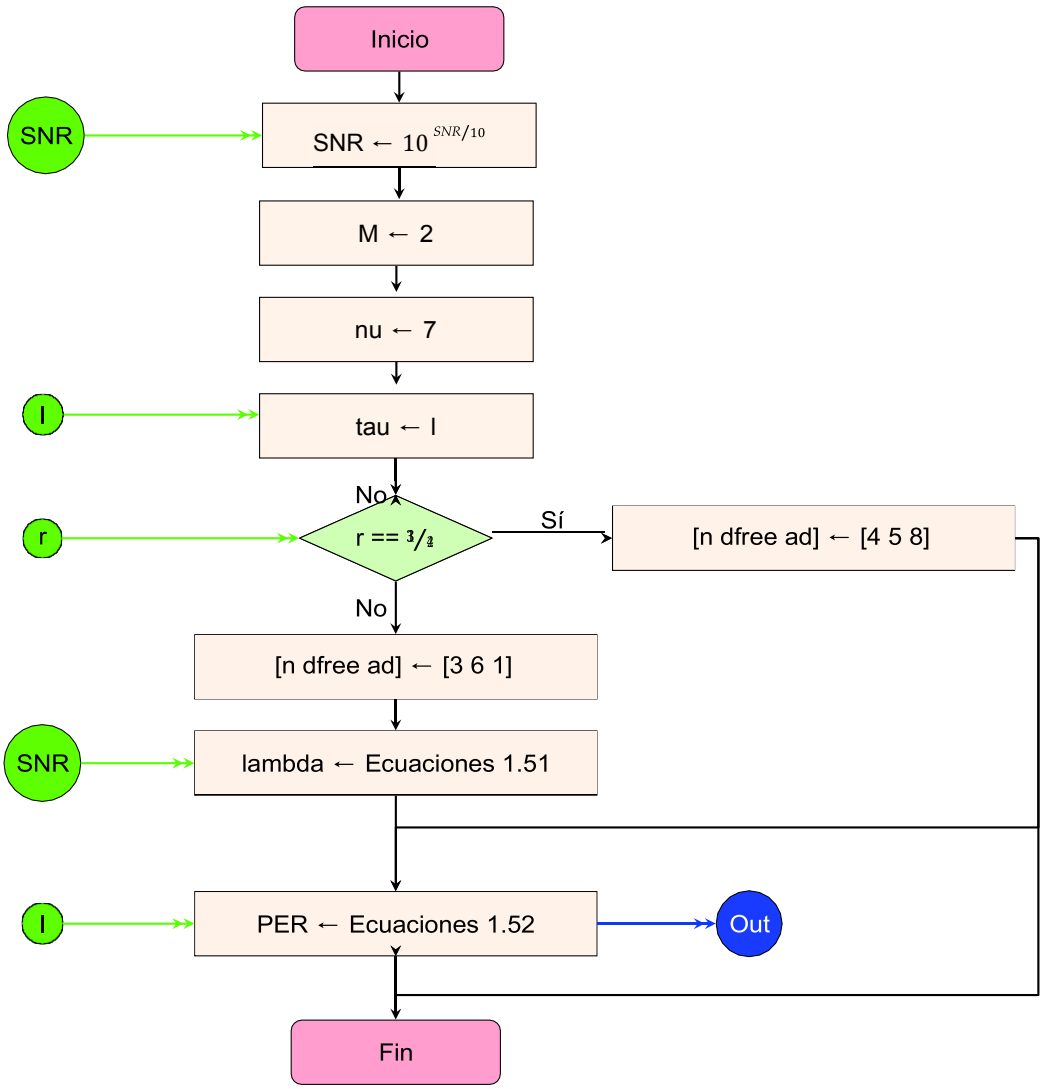


Figura I.34: Diagrama de flujo del modelo teórico 3

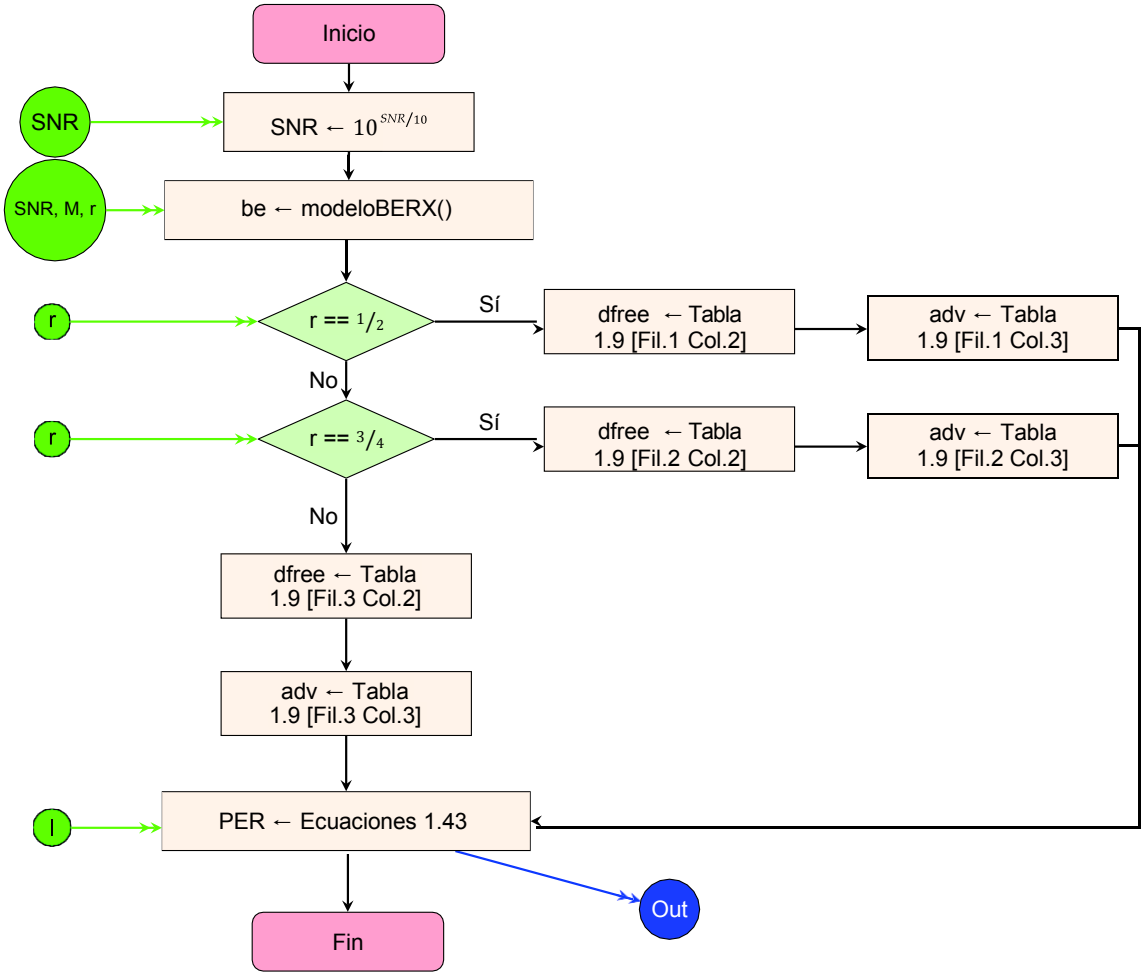


Figura I.35: Diagrama de flujo del modelo teórico i, con i = 4 (modeloBERX() = modeloBER1()), i = 5 (modeloBERX() = modeloBER2()) o i = 6 (modeloBERX() = modeloBER3())

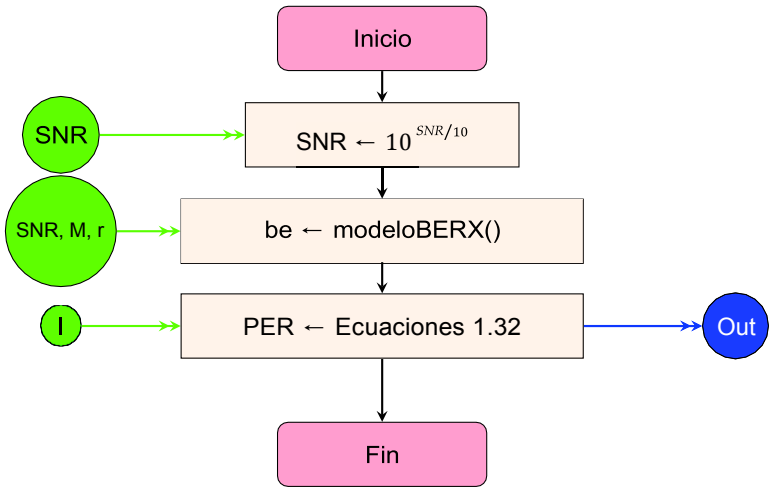


Figura I.36: Diagrama de flujo del modelo teórico i, con i = 7 (modeloBERX() = modeloBER1()), i = 8 (modeloBERX() = modeloBER2()) o i = 9 (modeloBERX() = modeloBER3())

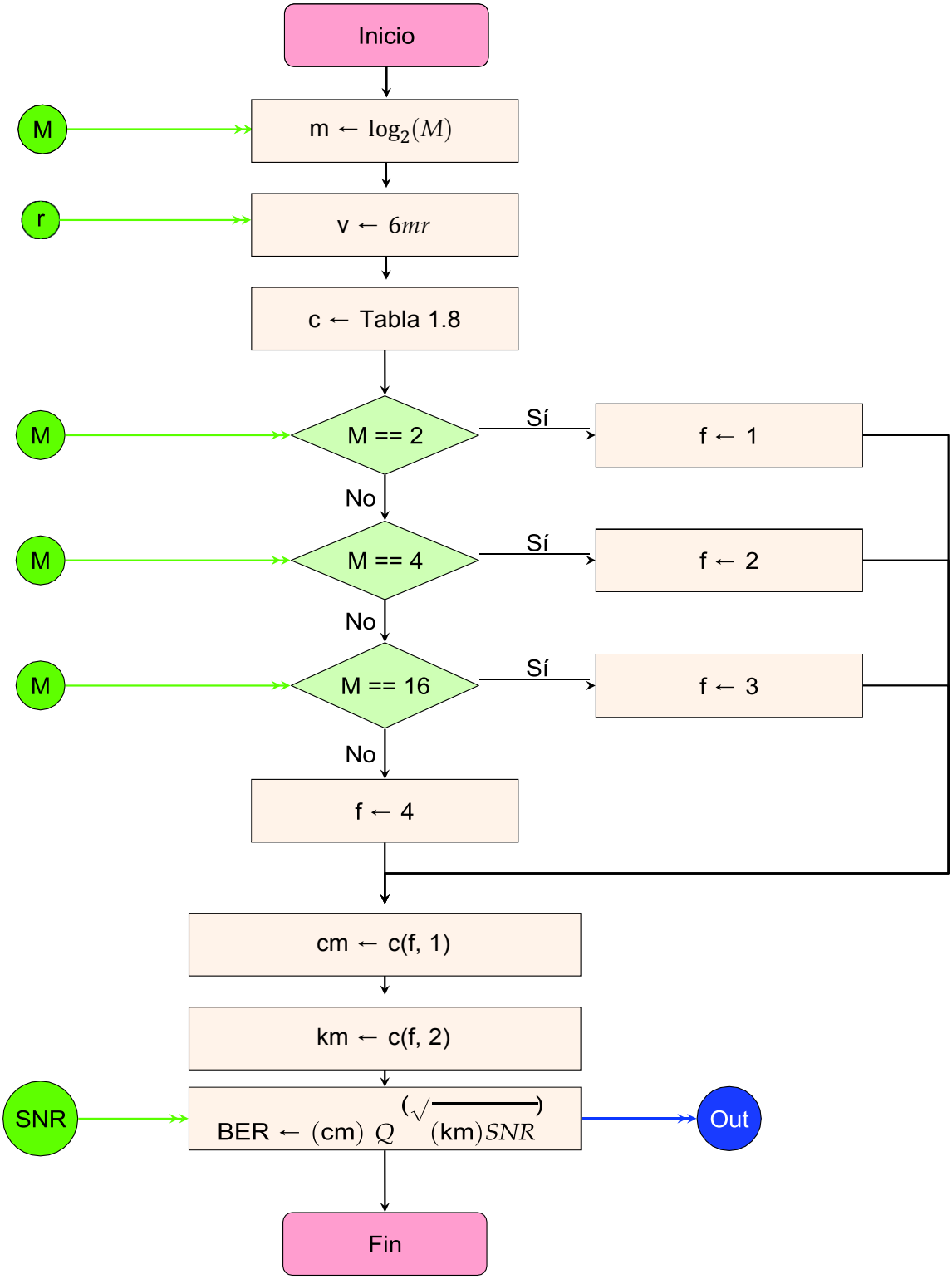


Figura I.37: Diagrama de flujo del modelo teórico del BER 1

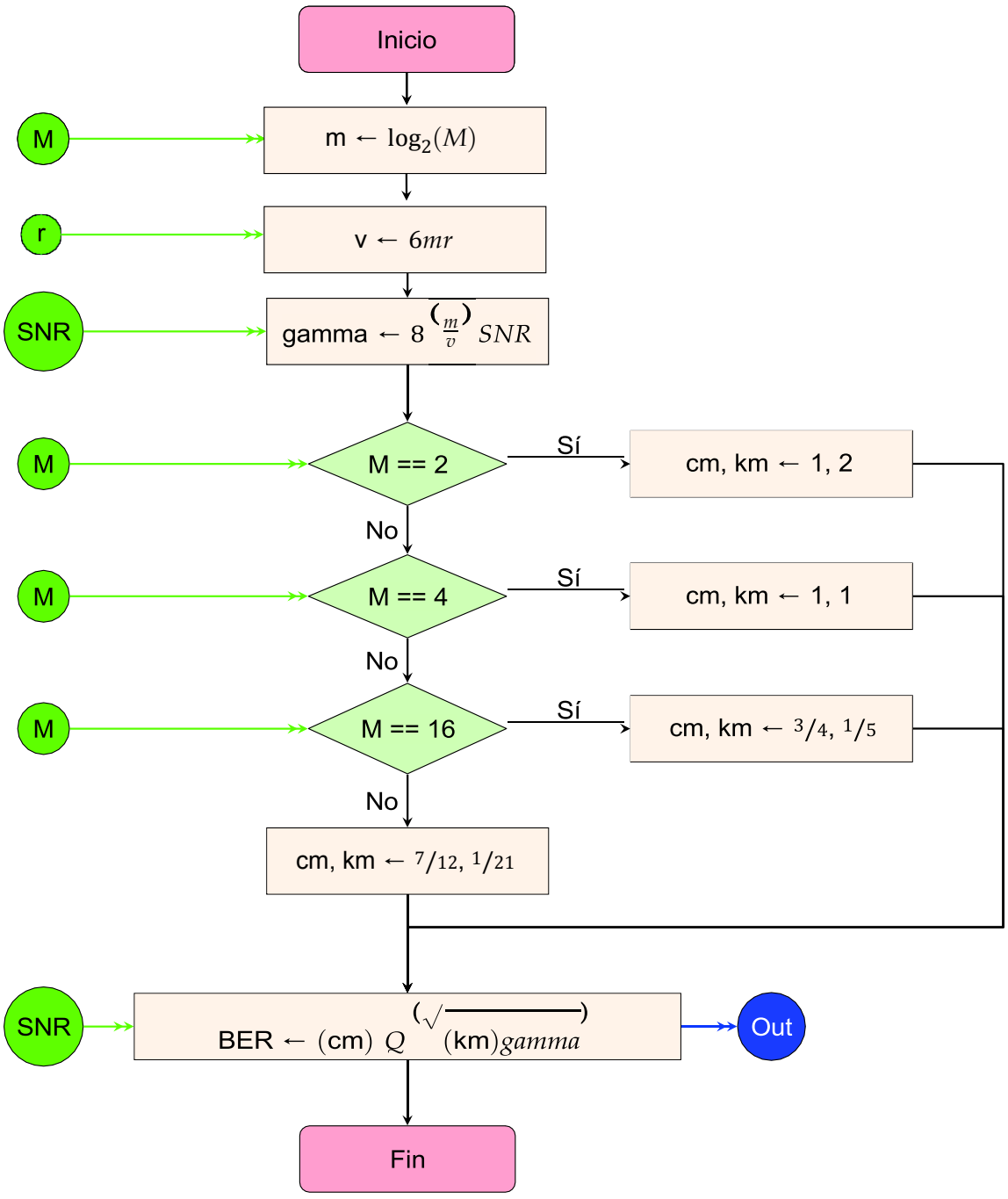


Figura I.38: Diagrama de flujo del modelo teórico del BER 2

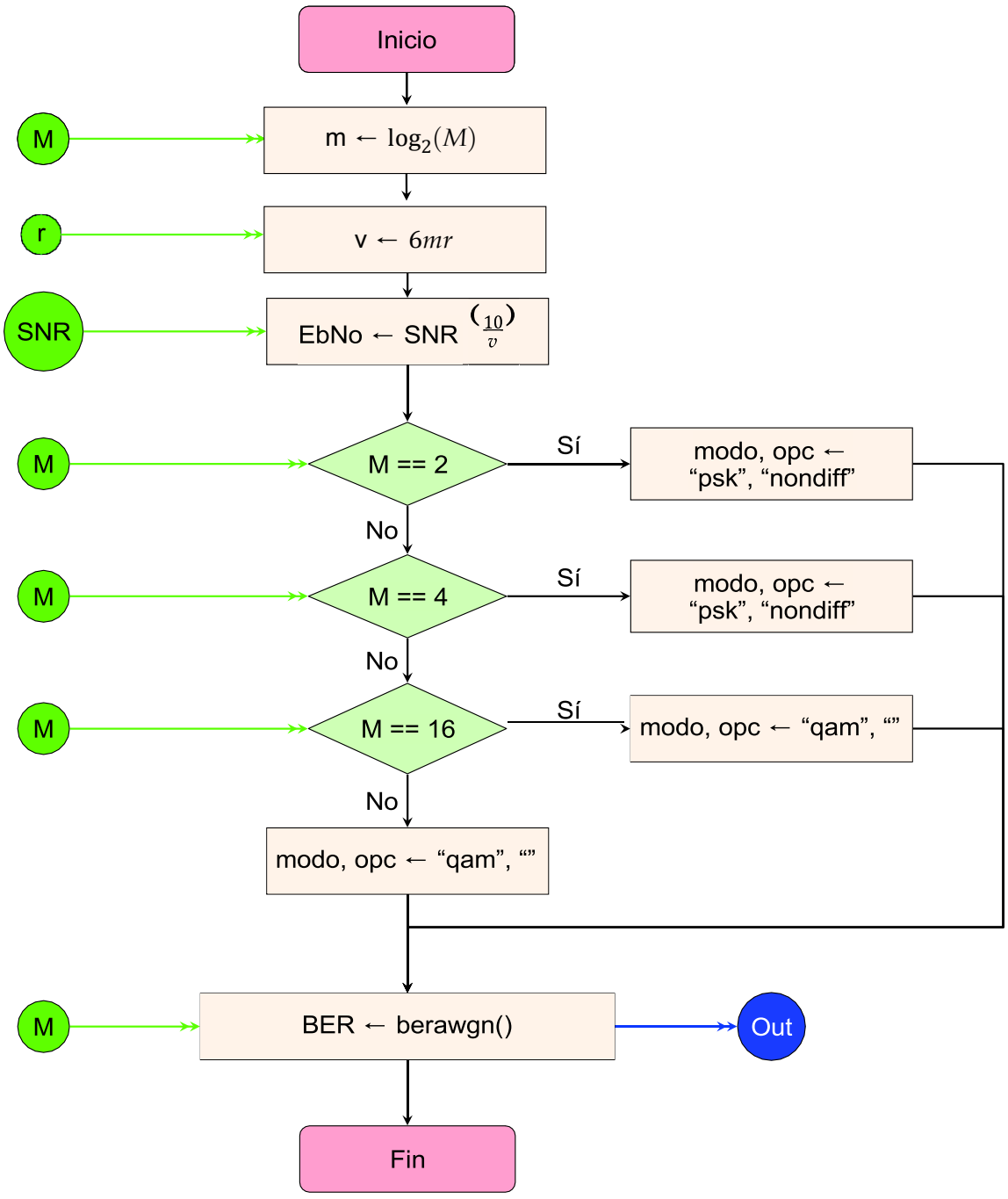


Figura I.39: Diagrama de flujo del modelo teórico del BER 3

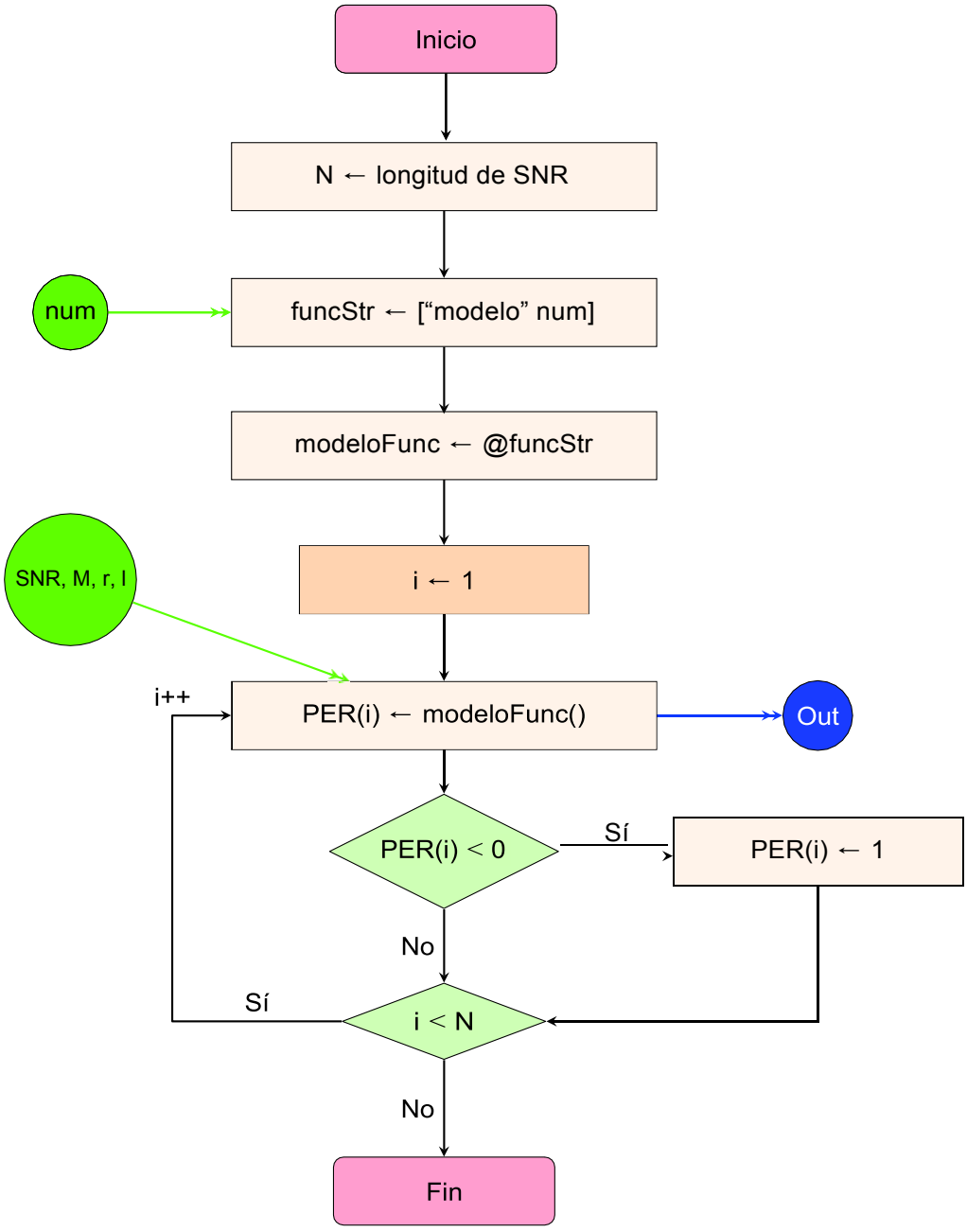


Figura I.40: Diagrama de flujo del script de evaluación de modelos

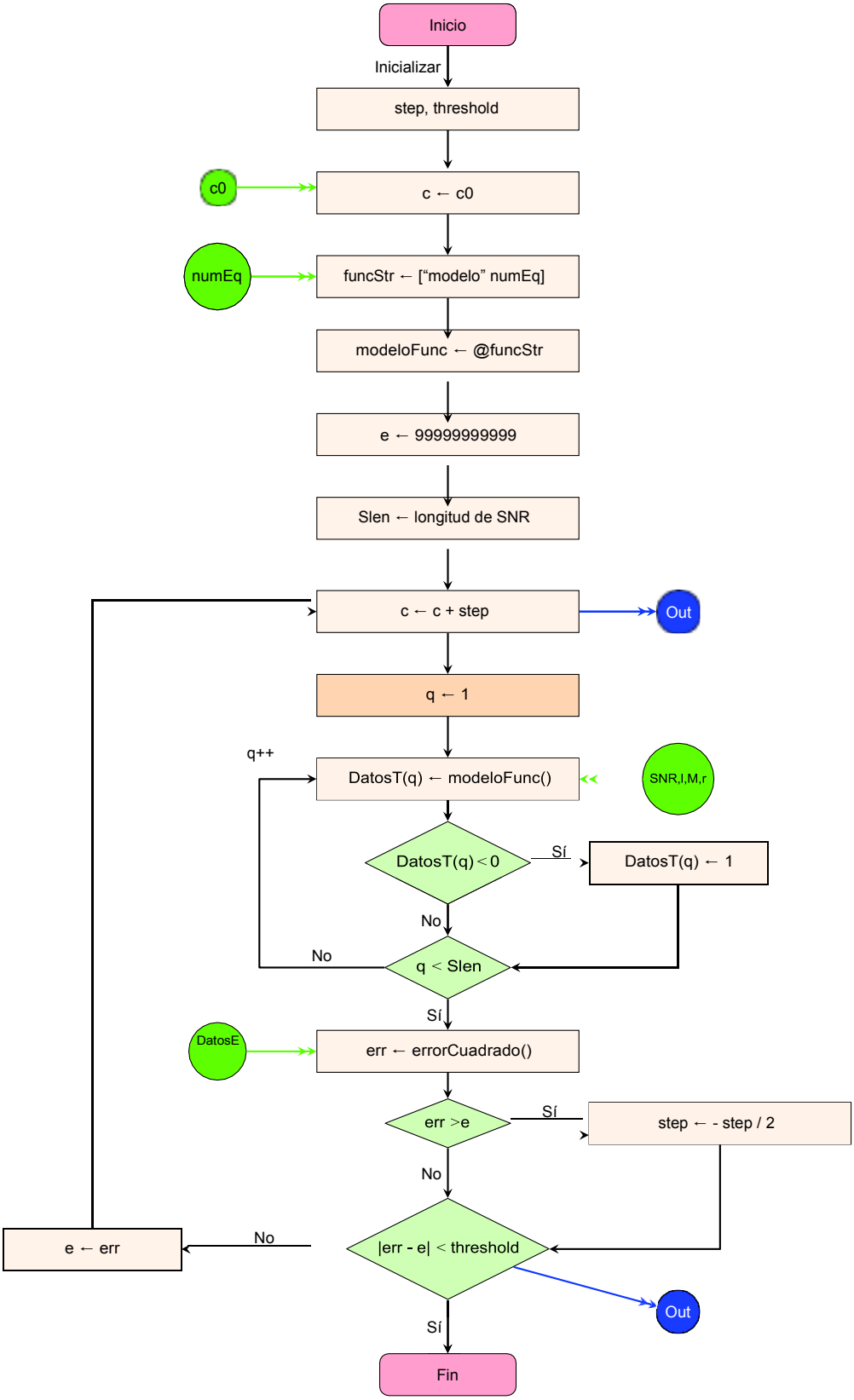


Figura I.41: Diagrama de flujo de la minimización del error

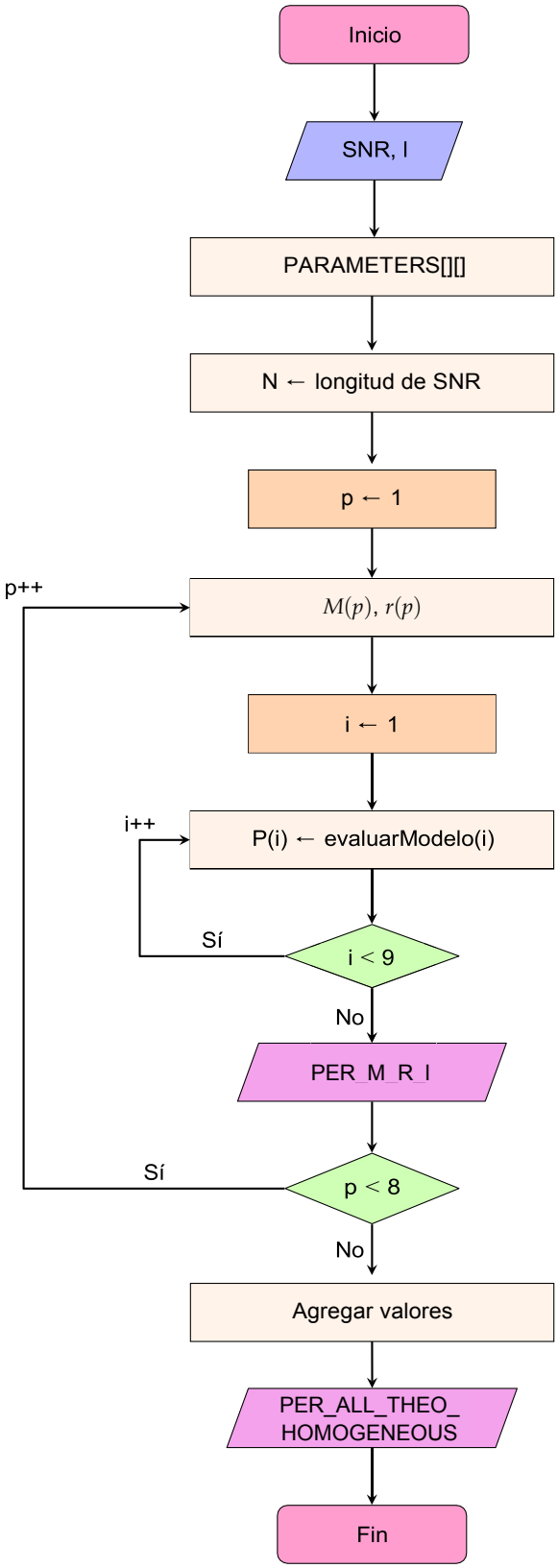


Figura I.42: Diagrama de flujo del análisis del PER teórico

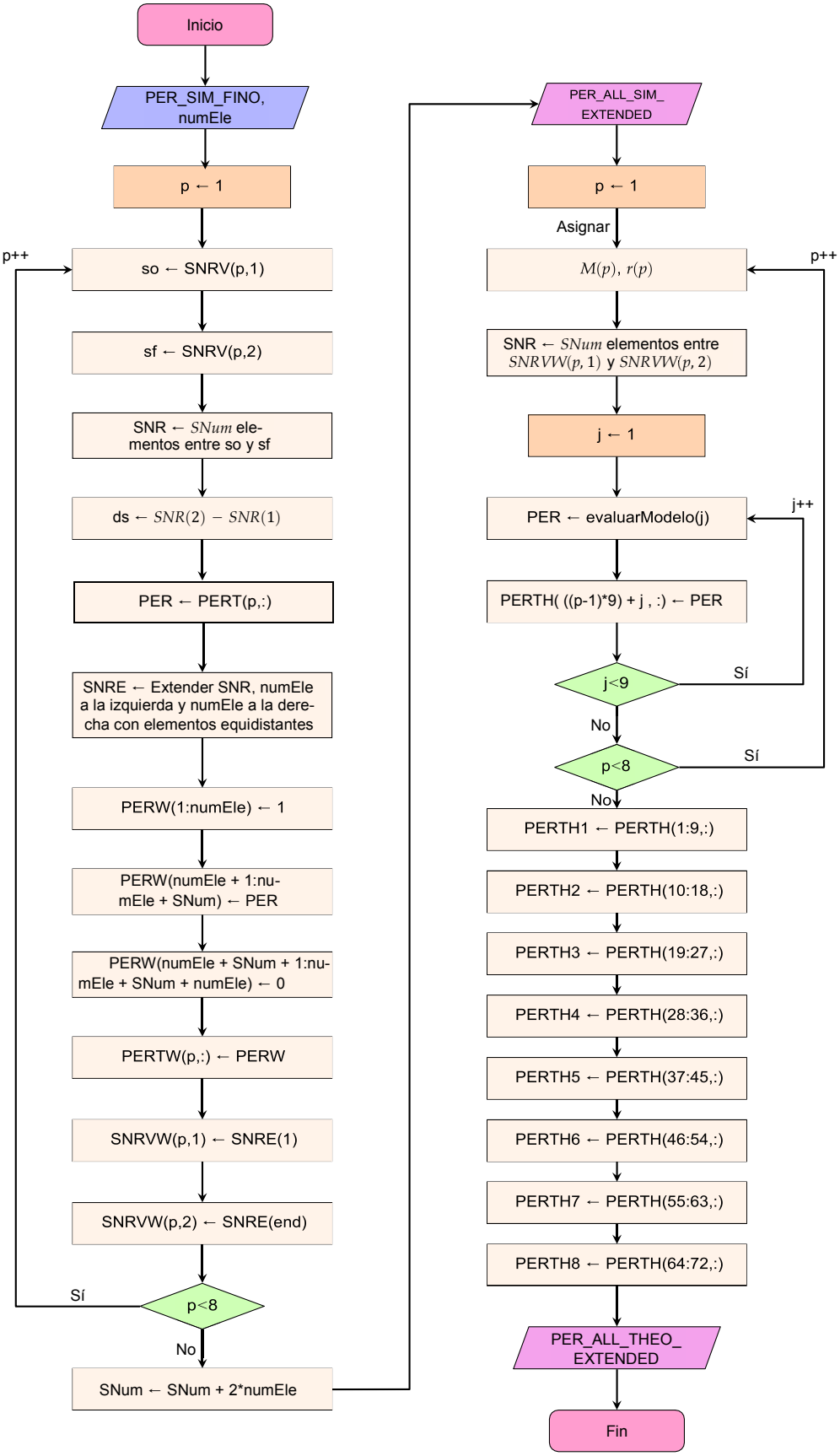


Figura I.43: Diagrama de flujo de extensión de PER

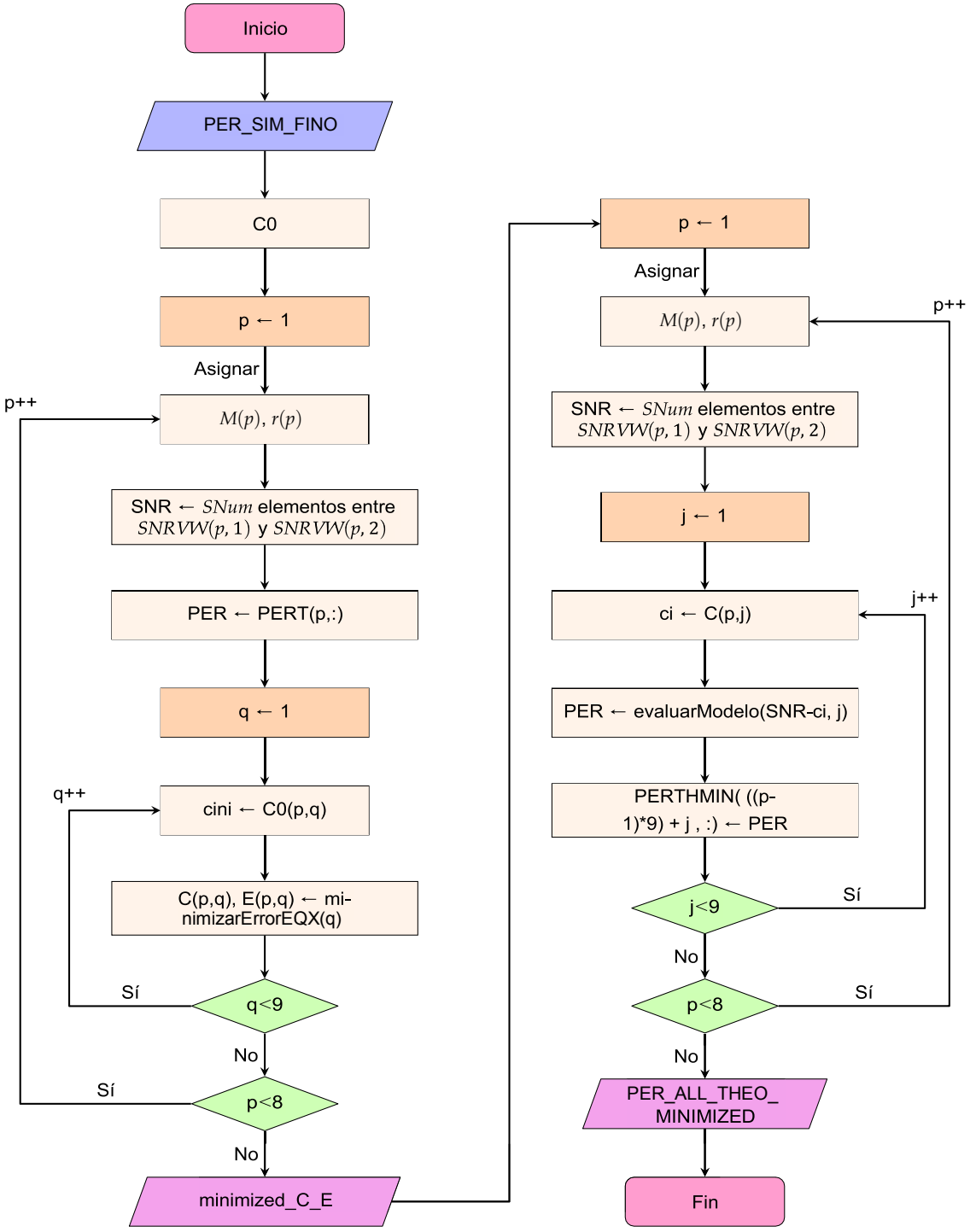


Figura I.44: Diagrama de flujo de comparación de valores de PER

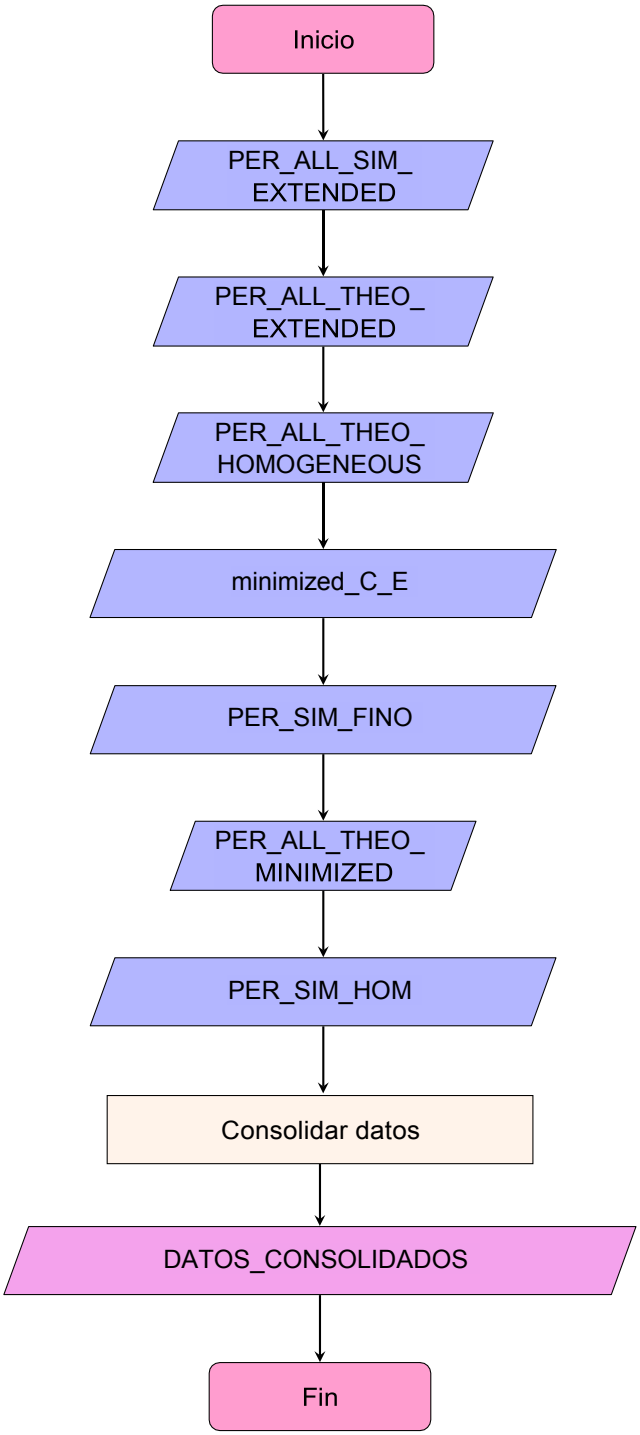


Figura I.45: Diagrama de flujo de consolidación de datos

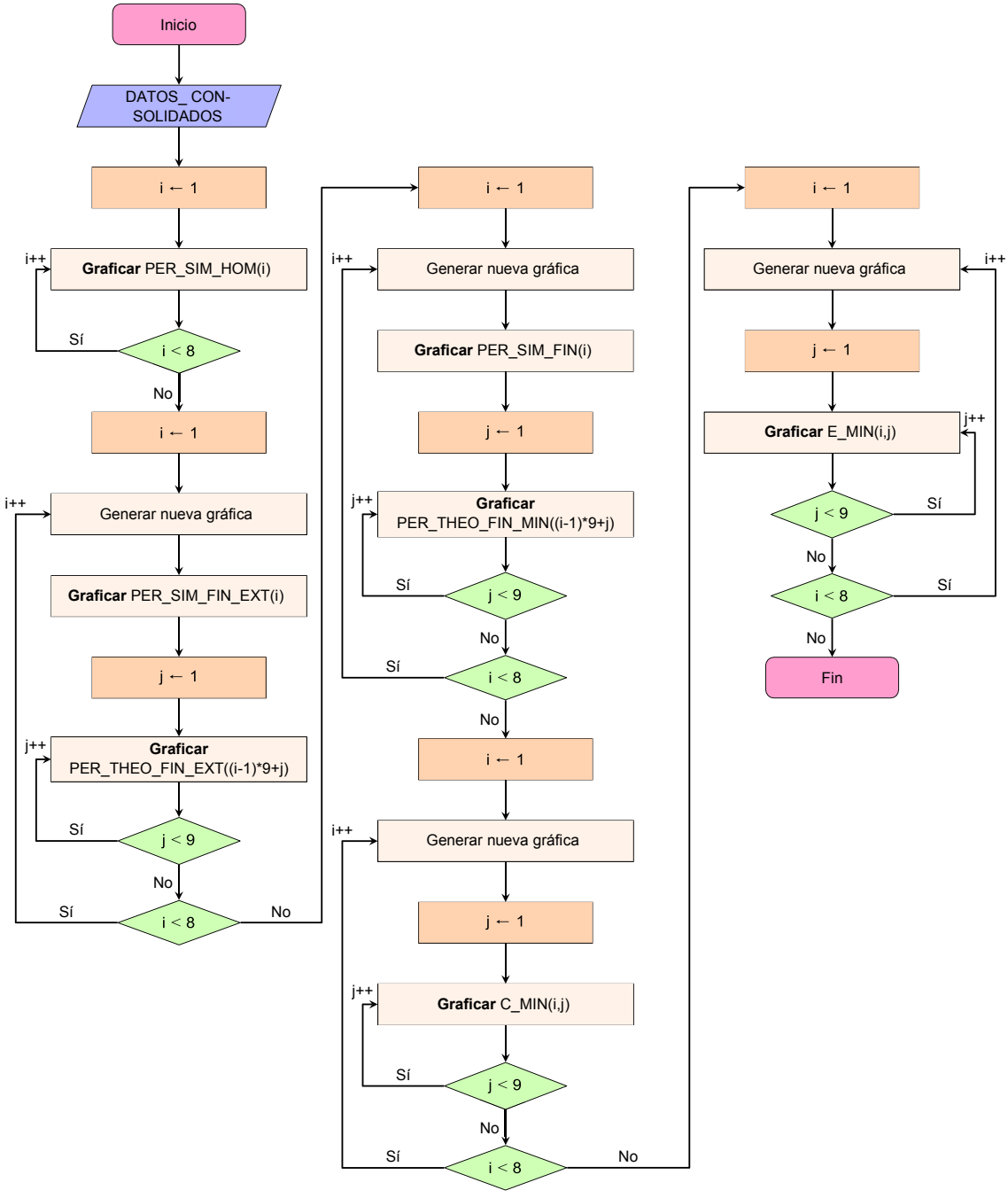


Figura I.46: Diagrama de flujo de la graficación de resultados

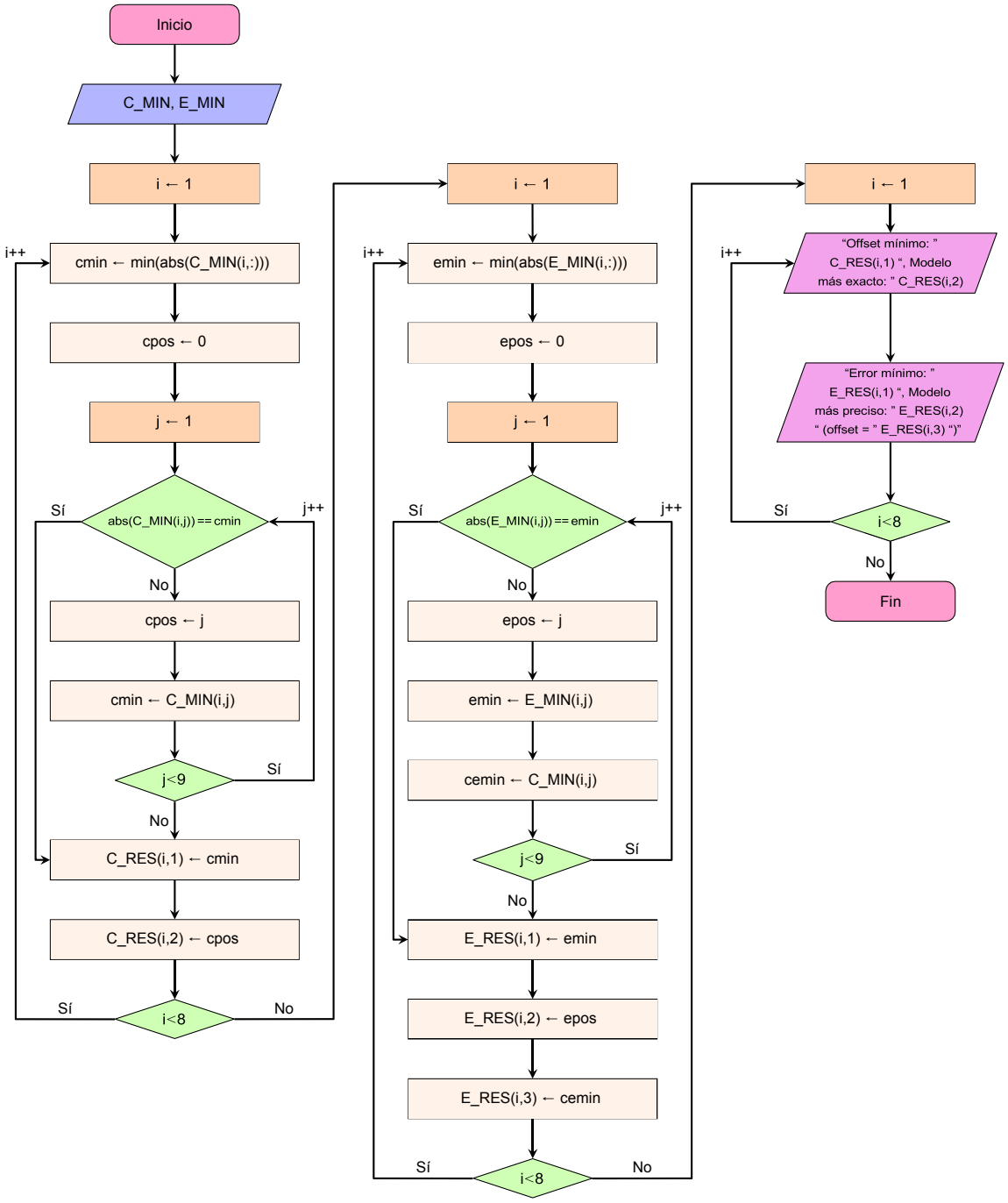


Figura I.47: Diagrama de flujo del análisis de resultados

ANEXO II

En este anexo se adjuntan los algoritmos para cada *script* y función definidos en el Capítulo 2.

Algoritmo 1: Simulacion (SNR homogéneo)

Salida: BERT, PERT, ERROREST
Leer: numPaq, numByte, SNR;
PARAMETERS[][] ← Asignar valores (De acuerdo a [8] y [30]);
ERROREST, BERT, PERT ← Matrices vacías de $8\text{x}105$;
p ← 1;
numConf ← 8;
mientras *p* $\leq$ *numConf* **hacer**
    M ← PARAMETERS(p)(1);
    r ← PARAMETERS(p)(2);
    v ← PARAMETERS(p)(3);
    ERRORESM, BERM, PERM ← Arreglos vacíos;
    i ← 1;
    mientras *i* $\leq longitud(SNR)$ **hacer**
        paqErr, bitErr ← 0 ;
        j ← 1;
        mientras *j* $\leq$ *numPaq* **hacer**
            errores ← 0;
            DatosTX, stringBitsTx ← txOFDM(numBit, M, r);
            DatosRX ← awgn(DatosTX, SNR(i));
            stringBitsRx ← rxOFDM(DatosRX, numBit, M, r);
            errores ← $\sum |stringBitsRx - stringBitsTx|$;
            si *errores* $>$ *0* **entonces**
                paqErr ← paqErr + 1 ;
            fin
            bitErr ← bitErr + errores ;
            j ← j + 1;
        fin
        BER ← bitErr / (numPaq * numBit);
        PER ← paqErr / numPaq;
        ERRORESM ← **Concatenar** ERRORESM bitErr;
        BERM ← **Concatenar** BERM BER;
        PERM ← **Concatenar** PERM PER;
        i ← i + 1;
    fin
    ERROREST[p] ← ERRORESM;
    BERT[p] ← BERM;
    PERT[p] ← PERM;
    p ← p + 1;
fin
devolver *BERT, PERT, ERROREST*

Algoritmo 2: Simulacion (SNR angosto)

Salida: BERT, PERT, ERROREST

Leer: numPaq, numByte, SNRV, SNum;
PARAMETERS[][] ← Asignar valores (De acuerdo a [8] y [30]);
ERROREST, BERT, PERT ← Matrices vacías de 8x105;
p ← 1;
numConf ← 8;
mientras *p* $\leq$ *numConf* **hacer**
    M ← PARAMETERS(p)(1);
    r ← PARAMETERS(p)(2);
    v ← PARAMETERS(p)(3);
    ERRORESM, BERM, PERM ← Arreglos vacíos;
    SNR ← SNum elementos equidistantes entre SNRV(p,1) y SNRV(p,2);
    i ← 1;
    mientras *i* $\leq longitud(SNR)$ **hacer**
        paqErr, bitErr ← 0 ;
        j ← 1;
        mientras *j* $\leq$ *numPaq* **hacer**
            errores ← 0;
            DatosTX, stringBitsTx ← txOFDM(numBit, M, r);
            DatosRX ← awgn(DatosTX, SNR(i));
            stringBitsRx ← rxOFDM(DatosRX, numBit, M, r);
            errores ← $\sum |stringBitsRx - stringBitsTx|$;
            si *errores* $>$ *0* **entonces**
                paqErr ← paqErr + 1 ;
            fin
            bitErr ← bitErr + errores ;
            j ← j + 1;
        fin
        BER ← bitErr / (numPaq * numBit);
        PER ← paqErr / numPaq;
        ERRORESM ← **Concatenar** ERRORESM bitErr;
        BERM ← **Concatenar** BERM BER;
        PERM ← **Concatenar** PERM PER;
        i ← i + 1;
    fin
    ERROREST[p] ← ERRORESM;
    BERT[p] ← BERM;
    PERT[p] ← PERM;
    p ← p + 1;
fin
devolver *BERT, PERT, ERROREST*

Algoritmo 3: txOFDM()

Entrada: numBit, M, r
Salida: PAQUETE, PSDU

si *numBit* < *8* || *numBit* > *4095*8* **entonces**
    Mensaje de error;
    devolver *0*
en otro caso
    SYNC ← CrearSYNC();
    SIGNAL ← CrearSIGNAL();
    PSDU ← numBit bits aleatorios;
    DATA ← CrearDATA(PSDU, M, r);
    PAQUETE ← **Concatenar** SYNC SIGNAL DATA;
fin
devolver *PSDU, PAQUETE*

Algoritmo 4: CrearSYNC()

Salida: SYNC

ShortP ← CrearPreambuloCorto();
LongP ← CrearPreambuloLargo();
SYNC ← **Concatenar** ShortP LongP;
devolver *SYNC*

Algoritmo 5: CrearPreambuloCorto()

Salida: ShortP

preambuloC ← Asignar valores (De acuerdo a [8]);
preambuloC ← $\sqrt{13/6}$ * preambuloC;
preambuloCArr ← Redistribuir preambuloC (De acuerdo a [8]);
preambuloCifft ← IFFT(preambuloCArr);
preambuloCCut ← preambuloCifft(1 – 16);
ShortP ← **Concatenar** preambuloCCut **x 10**;
devolver *ShortP*

Algoritmo 6: CrearPreambuloLargo()

Salida: LongP

preambuloL ← Asignar valores (De acuerdo a [8]);
preambuloLArr ← Redistribuir preambuloL (De acuerdo a [8]);
preambuloLifft ← IFFT(preambuloLArr);
GI2 ← preambuloLifft(33 – 64);
LongP ← **Concatenar** GI2 preambuloLifft preambuloLifft;
devolver *LongP*

Algoritmo 7: CrearSIGNAL()

Salida: SIGNAL

M ← 2;
r ← $1/2$;
m ← $\log_2(M)$;
NBPSC ← m;
NCBPS ← NBPSC * 48;
NDBPS ← NCBPS * r;
cabeceraBits ← 18 bits aleatorios;
tail ← 6 bits 0;
cabeceraBits ← **Concatenar** cabeceraBits tail;
cabeceraCodificada ← codificarString(cabeceraBits, r);
cabeceraEntrelazada ← entrelazarString(cabeceraCodificada, NCBPS);
cabeceraModulada ← modularString(cabeceraEntrelazada, M);
SIGNAL ← armarSimboloOFDM(cabeceraModulada, 1);
devolver *SIGNAL*

Algoritmo 8: CrearDATA()

Entrada: PSDU, M, r
Salida: DATA

m ← $\log_2(M)$;
NBPSC ← m;
NCBPS ← NBPSC * 48;
NDBPS ← NCBPS * r;
SERVICE ← 16 bits 0;
TAIL ← 6 bits 0;
NSYM ← $\lceil longitud(PSDU) + 22 \rceil$ / NDBPS;
NBITS ← NSYM * NDBPS;
NPAD ← NBITS - $(longitud(PSDU) + 22)$;
PAD ← NPAD bits 0;
ODATA ← **Concatenar** SERVICE PSDU TAIL PAD ;
SDATA ← scrambleString(ODATA, [1 0 1 1 1 0 1]);
SDATA($17 + longitud(PSDU)$ – $17 + longitud(PSDU) + 6$) ← 6 bits 0;
CDATA ← codificarString(SDATA, r);
IDATA ← entrelazarString(CDATA, NCBPS);
MDATA ← modularString(IDATA, M);
DATA ← modulacionOFDM(MDATA);
devolver *DATA*;

Algoritmo 9: scrambleString()

Entrada: datos, estadoInicial
Salida: SDatos

```
1  SDatos ← longitud(datos) bits 0;
2  estado ← estadoInicial;
3  i ← 1 ;
4  mientras i ≤ longitud(datos) hacer
5      estadoS ← estado(4) ⊕ estado(7);
6      SDatos(i) ← estadoS ⊕ datos(i);
7      estado ← Circshift(estado → 1);
8      estado(1) ← estadoS;
9      i ← i + 1;
10 fin
11 devolver SDatos
```

Algoritmo 10: codificarString()

Entrada: datos, R
Salida: CDatos

```
1  g0 ← 133;
2  g1 ← 171;
3  k ← 7;
4  trellis ← Generar diagrama de Trellis (k, g0, g1);
5  inicio
6      seleccionar R hacer
7          caso 1/2 hacer
8              punctPattern ← [1 1];
9          fin
10         caso 1/2 hacer
11             punctPattern ← [1 1 1 0];
12         fin
13         caso 1/2 hacer
14             punctPattern ← [1 1 1 0 0 1];
15         fin
16         si no hacer
17             Mensaje de error;
18             devolver 0;
19         fin
20     fin
21 fin
22 punctPattern ← Invertir(punctPattern);
23 CDatos ← Codificación convolucional de datos (trellis, punctPattern);
24 devolver CDatos;
```

Algoritmo 11: entrelazarString()

Entrada: datos, Ncbps
Salida: IDatos

```
1  permutacion ← longitud(datos) bits 0;
2  IDatos ← longitud(datos) bits 0;
3  Nbpsc ← Ncbps / 48;
4  s ← max(Nbpsc/2, 1);
5  a ← 1;
6  mientras a ≤ longitud(datos)/Ncbps hacer
7      b ← 1;
8      mientras b ≤ Ncbps hacer
9          c ← Ncbps/16 *(b − 1)  mód (16) + ⌊(b 1)/16⌋ ;
10         k ← (a − 1)Ncbps + b;
11         i ← (a − 1)Ncbps + (c + 1);
12         permutacion(k) ← datos(i);
13         b ← b + 1;
14     fin
15     b ← 1;
16     mientras b ≤ Ncbps hacer
17         c ← s ⌊(b−1)/s⌋ + (b − 1) + Ncbps − ⌊16(b 1)/Ncbps⌋ mód (s) ;
18         i ← (a − 1)Ncbps + b;
19         j ← (a − 1)Ncbps + (c + 1);
20         IDatos(i) ← permutacion(j);
21         b ← b + 1;
22     fin
23     a ← a + 1;
24 fin
25 devolver IDatos
```

Algoritmo 12: modularString()

Entrada: datos, M
Salida: MDatos

```
seleccionar M hacer
    caso 2 hacer
        MDatos ← modularBPSK(MDATA);
    fin
    caso 4 hacer
        MDatos ← 1/√2 modularQPSK(MDATA);
    fin
    caso 16 hacer
        MDatos ← 1/√10 modular16QAM(MDATA);
    fin
    caso 64 hacer
        MDatos ← 1/√42 modular64QAM(MDATA);
    fin
    si no hacer
        Mensaje de error;
        devolver 0;
    fin
fin
devolver MDatos
```

Algoritmo 13: modulacionOFDM()

Entrada: datos
Salida: simbolos

```
polaridadPilotos ← generarPolaridadPiloto();
Nsym ← longitud(datos)/48;
simbolos←Arreglo vacío;
i←1;
mientras i≤Nsym hacer
    indicePiloto ← i mód 127 + 1;
    simbolo ← armarSimboloOFDM(datos( (i-1)*48 + 1 : i*48 ),
      polaridadPilotos(indicePiloto) );
    simbolos ← Concatenar simbolos simbolo;
fin
devolver simbolos
```

Algoritmo 14: armarSimboloOFDM()

Entrada: datosF, polPiloto
Salida: OFDMSym

```
simboloF ← datosF (De acuerdo a [8]);
simboloFArr ← Redistribuir simboloF (De acuerdo a [8]);
simboloT ← IFFT(simboloFArr);
GI ← simboloT(17 - 64);
OFDMSym ← Concatenar GI simboloT;
devolver OFDMSym;
```

Algoritmo 15: rxOFDM()

Entrada: PAQUETE, numBit, M, r
Salida: PSDU

```
1 DATA ← extraerDATA(PAQUETE);
2 PSDU ← procesarDATA(DATA, M, r, numBit);
3 devolver PSDU
```

Algoritmo 16: extraerData()

Entrada: PAQUETE
Salida: DATAT

```
1 DATAT ← PAQUETE(241→);
2 devolver DATAT
```

Algoritmo 17: procesarDATA()

Entrada: DATA, M, r, numBit
Salida: PSDU

```
1 m ← log2(M);
2 NBPSC ← m;
3 NCBPS ← NBPSC * 48;
4 NDBPS ← NCBPS * r;
5 MDATA ← demodulacionOFDM(DATA);
6 IDATA ← demodularString(MDATA, M);
7 CDATA ← deentrelazarString(IDATA, NCBPS);
8 SDATA ← decodificarString(CDATA, r, M);
9 ODATA ← scrambleString(SDATA, [1 0 1 1 1 0 1]);
10 PSDU ← ODATA(16 + 1 – 16 + numBit);
11 devolver PSDU;
```

Algoritmo 18: demodulacionOFDM()

Entrada: DATA
Salida: MDATA

```
1 polaridadPilotos ← generarPolaridadPiloto();
2 Nsym ← longitud(DATA)/80;
3 i←1;
4 MDATA←Arreglo vacío;
5 mientras i ≤ Nsym hacer
6     indicePiloto ← i  mód 127 + 1;
7     simbolo ← extraerSimboloOFDM(DATA( (i-1)*80 + 1 : i*80 ),
        polaridadPilotos(indicePiloto) );
8     MDATA ← Concatenar MDATA simbolo;
9 fin
10 devolver MDATA
```

Algoritmo 19: extraerSimboloOFDM()

Entrada: OFDMSym, polPiloto
Salida: datosF

```
1  simboloT ← OFDMSym(17–80);
2  simboloFArr ← IFFT(simboloT);
3  simboloF ← Redistribuir simboloFArr (De acuerdo a [8]);
4  datosF ← simboloF (valores correspondientes de acuerdo a [8]);
5  devolver datosF
```

Algoritmo 20: demodularString()

Entrada: MDATA, M
Salida: IDATA

```
1  seleccionar M hacer
2      caso 2 hacer
3          IDATA ← demodularBPSK(MDATA);
4      fin
5      caso 4 hacer
6          IDATA ← demodularQPSK(√2MDATA);
7      fin
8      caso 16 hacer
9          IDATA ← demodular16QAM(√10MDATA);
10     fin
11     caso 64 hacer
12         IDATA ← demodular64QAM(√42MDATA);
13     fin
14     si no hacer
15         Mensaje de error;
16         devolver 0;
17     fin
18 fin
19 devolver IDATA
```

Algoritmo 21: deentrelazarString()

Entrada: IDATA, Ncbps
Salida: CDATA

1 permutacion ← $longitud(IDATA)$ bits 0;
2 CDATA ← $longitud(IDATA)$ bits 0;
3 Nbpsc ← Ncbps / 48;
4 s ← **max**($Nbpsc/2$, 1);
5 a ← 1;
6 **mientras** $a \leq longitud(IDATA)/Ncbps$ **hacer**
7 b ← 1;
8 **mientras** $b \leq Ncbps$ **hacer**
9 c ← $s \lfloor (b-1)/s \rfloor + (b-1) + Ncbps - \left\lfloor \frac{16(b-1)}{Ncbps} \right\rfloor) \bmod (s)$;
10 i ← $(a-1)Ncbps + b$;
11 j ← $(a-1)Ncbps + (c+1)$;
12 permutacion(j) ← IDATA(i);
13 b ← b + 1;
14 **fin**
15 b ← 1;
16 **mientras** $b \leq Ncbps$ **hacer**
17 c ← $Ncbps/16$ *$(b-1) \bmod (16) + \left\lfloor \frac{(b-1)}{16} \right\rfloor$;
18 i ← $(a-1)Ncbps + b$;
19 j ← $(a-1)Ncbps + (c+1)$;
20 CDATA(i) ← permutacion(k);
21 b ← b + 1;
22 **fin**
23 a ← a + 1;
24 **fin**
25 **devolver** *CDATA*

Algoritmo 22: decodificarString()

Entrada: CDATA, R, M
Salida: SDATA

```
1  g0 ← 133;
2  g1 ← 171;
3  k ← 7;
4  trellis ← Generar diagrama de Trellis (k, g0, g1);
5  seleccionar R hacer
6      caso 1/2 hacer
7          punctPattern ← [1 1];
8          tblen ← 24*log2(M);
9      fin
10     caso 1/2 hacer
11         punctPattern ← [1 1 1 0];
12         tblen ← 96;
13     fin
14     caso 1/2 hacer
15         punctPattern ← [1 1 1 0 0 1];
16         si M == 2 entonces
17             tblen ← 36;
18         en otro caso
19             tblen ← 96;
20         fin
21     fin
22     si no hacer
23         Mensaje de error;
24         devolver 0;
25     fin
26 fin
27 punctPattern ← Invertir(punctPattern);
28 SDATA ← Decodificación por algoritmo de Viterbi de datos (trellis, punctPattern,
     tblen);
29 devolver SDATA;
```

Algoritmo 23: modularBPSK()

```
   Entrada: datos
   Salida: modulado
1  modulado ← Arreglo vacío ;
2  j ← 1 ;
3  mientras j ≤ longitud(datos) hacer
4      re← 0;
5      si datos(j) == 1 entonces
6          re ← 1;
7      en otro caso
8          re← - 1;
9      fin
10     modulado ← Concatenar modulado re ;
11     j ← j + 1;
12 fin
13 devolver modulado
```

Algoritmo 24: modularQPSK()

```
   Entrada: datos
   Salida: modulado
1  modulado ← Arreglo vacío ;
2  j ← 1 ;
3  mientras j ≤ longitud(datos) hacer
4      re← 0;
5      im← 0i;
6      si datos(j) == 1 entonces
7          re ← 1;
8      en otro caso
9          re← - 1;
10     fin
11     si datos(j+1) == 1 entonces
12         im ← 1i;
13     en otro caso
14         im ← −1i;
15     fin
16     modulado ← Concatenar modulado (re + im) ;
17     j ← j + 2;
18 fin
19 devolver modulado
```

Algoritmo 25: modular16QAM()

Entrada: datos
Salida: modulado

```
1  modulado ← Arreglo vacío ;
2  j ← 1 ;
3  mientras j ≤ longitud(datos) hacer
4      re← 0;
5      im← 0i;
6      si datos(j) == 1 entonces
7          re ← 1;
8      en otro caso
9          re← - 1;
10     fin
11     si datos(j+1) == 1 entonces
12         re ← re * 1;
13     en otro caso
14         re ← re * 3;
15     fin
16     si datos(j+2) == 1 entonces
17         im ← 1i;
18     en otro caso
19         im ← −1i;
20     fin
21     si datos(j+3) == 1 entonces
22         im ← im * 1;
23     en otro caso
24         im ← im * 3;
25     fin
26     modulado ← Concatenar modulado (re + im) ;
27     j ← j + 4;
28 fin
29 devolver modulado
```

Algoritmo 26: modular64QAM()

Entrada: datos
Salida: modulado

```
1  modulado ← Arreglo vacío ;
2  j ← 1 ;
3  mientras j ≤ longitud(datos) hacer
4      si datos(j) == 1 entonces
5          re ← 1;
6      en otro caso
7          re← - 1;
8      fin
9      si datos(j+1) == 1 entonces
10         si datos(j+2) == 1 entonces
11             re ← re * 3;
12         en otro caso
13             re ← re * 1;
14         fin
15     en otro caso
16         si datos(j+2) == 1 entonces
17             re ← re * 5;
18         en otro caso
19             re ← re * 7;
20         fin
21     fin
22     si datos(j+3) == 1 entonces
23         im ← 1i;
24     en otro caso
25         im ← −1i;
26     fin
27     si datos(j+4) == 1 entonces
28         si datos(j+5) == 1 entonces
29             im ← im * 3;
30         en otro caso
31             im ← im * 1;
32         fin
33     en otro caso
34         si datos(j+5) == 1 entonces
35             im ← im * 5;
36         en otro caso
37             im ← im * 7;
38         fin
39     fin
40     modulado ← Concatenar modulado (re + im) ;
41     j ← j + 6;
42 fin
43 devolver modulado
```

Algoritmo 27: demodularBPSK()

Entrada: MDATA
Salida: IDATA

```
1  IDATA ← Arreglo vacío ;
2  i ← 1 ;
3  mientras i ≤ longitud(MDATA) hacer
4      si ℜ(MDATA(i)) > 0 entonces
5          IDATA(i)← 1;
6      en otro caso
7          IDATA(i)← 0;
8      fin
9      i ← i + 1;
10 fin
11 devolver IDATA
```

Algoritmo 28: demodularQPSK()

Entrada: MDATA
Salida: IDATA

```
1  IDATA ← Arreglo vacío ;
2  i ← 1 ;
3  mientras i ≤ longitud(MDATA) hacer
4      j← (i-1)*2 + 1;
5      si ℜ(MDATA(i)) > 0 entonces
6          IDATA(j)← 1;
7      en otro caso
8          IDATA(j)← 0;
9      fin
10     si ℑ(MDATA(i)) > 0 entonces
11         IDATA(j+1)← 1;
12     en otro caso
13         IDATA(j+1)← 0;
14     fin
15     i ← i + 1;
16 fin
17 devolver IDATA
```

Algoritmo 29: demodular16QAM()

Entrada: MDATA
Salida: IDATA

```
1  IDATA ← Arreglo vacío ;
2  i ← 1 ;
3  mientras i ≤ longitud(MDATA) hacer
4      j← (i-1)*4 + 1;
5      si ℜ(MDATA(i)) > 0 entonces
6          IDATA(j)← 1;
7      en otro caso
8          IDATA(j)← 0;
9      fin
10     si |ℜ(MDATA(i))|< 2 entonces
11         IDATA(j+1)← 1;
12     en otro caso
13         IDATA(j+1)← 0;
14     fin
15     si ℑ(MDATA(i)) > 0 entonces
16         IDATA(j+2)← 1;
17     en otro caso
18         IDATA(j+2)← 0;
19     fin
20     si |ℑ(MDATA(i))|< 2 entonces
21         IDATA(j+3)← 1;
22     en otro caso
23         IDATA(j+3)← 0;
24     fin
25     i ← i + 1;
26 fin
27 devolver IDATA
```

Algoritmo 30: demodular64QAM()

Entrada: MDATA
Salida: IDATA

```
1  IDATA ← Arreglo vacío ;
2  i ← 1 ;
3  mientras i ≤ longitud(MDATA) hacer
4      j← (i-1)*6 + 1;
5      si ℜ(MDATA(i)) > 0 entonces
6          IDATA(j)← 1;
7      en otro caso
8          IDATA(j)← 0;
9      fin
10     si |ℜ(MDATA(i))| < 4 entonces
11         IDATA(j+1)← 1;
12     en otro caso
13         IDATA(j+1)← 0;
14     fin
15     si |ℜ(MDATA(i)) − 4| < 2 entonces
16         IDATA(j+2)← 1;
17     en otro caso
18         IDATA(j+2)← 0;
19     fin
20     si ℑ(MDATA(i)) > 0 entonces
21         IDATA(j+3)← 1;
22     en otro caso
23         IDATA(j+3)← 0;
24     fin
25     si |ℑ(MDATA(i))| < 4 entonces
26         IDATA(j+4)← 1;
27     en otro caso
28         IDATA(j+4)← 0;
29     fin
30     si |ℑ(MDATA(i)) − 4| < 2 entonces
31         IDATA(j+5)← 1;
32     en otro caso
33         IDATA(j+5)← 0;
34     fin
35     i ← i + 1;
36 fin
37 devolver IDATA
```

Algoritmo 31: modelo1()

Entrada: SNR, M, r, l
Salida: PER

```
1  c ← Tabla 1.7 (constantes c_i y d_i);
2  si M == 2 entonces
3      i← 1;
4  en otro caso
5      si M == 4 entonces
6          i← 3;
7      en otro caso
8          si M == 16 entonces
9              i← 5;
10         en otro caso
11             i← 7;
12         fin
13     fin
14 fin
15 si r == 3/4 entonces
16     i← i + 1;
17 fin
18 c1, c2, c3, c4 ← c(i,1), c(i,2), c(i,3), c(i,4);
19 d1, d2, d3, d4 ← c(i,5), c(i,6), c(i,7), c(i,8);
20 ar ← Ecuación 1.25 [a_R];
21 br ← Ecuación 1.26 [b_R];
22 PER ← Ecuación 1.24 [PER];
23 devolver PER
```

Algoritmo 32: modelo2()

Entrada: SNR, M, r, l
Salida: PER

```
SNR ← 10^(SNR/10);
m ← log2(M);
v ← 6mr;
c ← Tabla 1.8 (constantes cm y km);
si M == 2 entonces
    i ← 1;
en otro caso
    si M == 4 entonces
        i ← 2;
    en otro caso
        si M == 16 entonces
            i ← 3;
        en otro caso
            i ← 4;
        fin
    fin
fin
cm ← c(i,1);
km ← c(i,2);
an ← Ecuación 1.34 [aN];
bn ← Ecuación 1.35 [bN];
PER ← Ecuación 1.33 [PER];
devolver DATAT
```

Algoritmo 33: modelo3()

Entrada: SNR, M, r, l
Salida: PER

```
SNR ← 10^(SNR/10);
M ← 2;
nu ← 7;
tau ← l;
si r == 1/2 entonces
    n, dfree, ad ← 2, 10, 11 ;
en otro caso
    si r == 3/4 entonces
        n, dfree, ad ← 4, 5, 8 ;
    en otro caso
        n, dfree, ad ← 3, 6, 1 ;
    fin
fin
lambda ← Ecuaciones 1.51;
PER ← Ecuaciones 1.52;
devolver PER
```

Algoritmo 34: modeloX() (X=4,5,6)

Entrada: SNR, M, r, l
Salida: PER

```
1  SNR ← 10^(SNR/10);
2  be ← modeloBERX(SNR, M, r);
3  si r == 1/2 entonces
4      dfree ← Tabla 1.9 [Fila 1 Columna 2];
5      adv ← Tabla 1.9 [Fila 1 Columna 3];
6  en otro caso
7      si r == 3/4 entonces
8          dfree ← Tabla 1.9 [Fila 2 Columna 2];
9          adv ← Tabla 1.9 [Fila 2 Columna 3];
10     en otro caso
11         dfree ← Tabla 1.9 [Fila 3 Columna 2];
12         adv ← Tabla 1.9 [Fila 3 Columna 3];
13     fin
14 fin
15 PER ← Ecuaciones 1.43;
16 devolver PER
```

Algoritmo 35: modeloX() (X=7,8,9)

Entrada: SNR, M, r, l
Salida: PER

```
1  SNR ← 10^(SNR/10);
2  be ← modeloBERX(SNR, M, r);
3  PER ← Ecuaciones 1.32;
4  devolver PER
```

Algoritmo 36: modeloBER1()

Entrada: SNR, M, r
Salida: BER

```
m ← log2(M);
v ← 6mr;
c ← Tabla 1.8;
si M == 2 entonces
    f← 1;
en otro caso
    si M == 4 entonces
        f← 2;
    en otro caso
        si M == 16 entonces
            f← 3;
        en otro caso
            f← 4;
        fin
    fin
fin
cm ← c(f,1);
km ← c(f,2);
BER ← (cm) Q( sqrt( (km)SNR ) );
devolver BER
```

Algoritmo 37: modeloBER2()

Entrada: SNR, M, r
Salida: BER

```
m ← log2(M);
v ← 6mr;
c ← 8 (m/4) SNR;
si M == 2 entonces
    cm, km ← 1, 2;
en otro caso
    si M == 4 entonces
        cm, km ← 1, 1;
    en otro caso
        si M == 16 entonces
            cm, km ← 3/4, 1/5;
        en otro caso
            cm, km ← 7/12, 1/21;
        fin
    fin
fin
BER ← (cm) Q( sqrt( (km)SNR ) );
devolver BER
```

Algoritmo 38: modeloBER3()

Entrada: SNR, M, r
Salida: BER

```
m ← log2(M);
v ← 6mr;
EbNo ← SNR (10/v);
si M == 2 entonces
    modo, opc ← "psk", "nondiff" ;
en otro caso
    si M == 4 entonces
        modo, opc ← "psk", "nondiff";
    en otro caso
        si M == 16 entonces
            modo, opc ← "qam", "";
        en otro caso
            modo, opc ← "qam", "";
        fin
    fin
fin
BER ← berawgn(EbNo,modo,M,opc);
devolver BER
```

Algoritmo 39: evaluarModelo()

Entrada: SNR, M, r, l, num
Salida: c, e

```
N ← longitud de SNR;
funcStr ← ["modelo" num];
modeloFunc ← @funcStr;
i ← 1 ;
mientras i ≤ N hacer
    PER(i) ← modeloFunc(SNR(i), M, r, l) ;
    si PER(i) < 0 entonces
        PER(i) ← 1;
    fin
    i ← i + 1 ;
fin
devolver PER
```

Algoritmo 40: minimizarErrorEQX()

Entrada: SNR, DatosE, l, M, r, numEq, c0
Salida: c, e

```
step ← -1 ;
threshold ← 1x10^-12 ;
c ← c0 ;
funcStr ← ["modelo" numEq];
modeloFunc ← @funcStr;
e ← 99999999999;
Slen ← longitud(SNR);
mientras 1 == 1 hacer
    c ← c + step ;
    q← 1 ;
    mientras q≤Slen hacer
        DatosT(q) ← modeloX(SNR - c, l, M, r) ;
        si DatosT(q)<0 entonces
            DatosT(q) ← 1;
        fin
    fin
    err ← errorCuadrado(DatosE, DatosT);
    si err > e entonces
        step ← -step/2 ;
    fin
    si |err-e| < threshold entonces
        devolver c, e ;
    fin
    e ← err ;
fin
```

Algoritmo 41: ANALISIS_PER

```
Leer: SNR, l;
PARAMETERS[][] ← Asignar valores (De acuerdo a [8] y [30]);
N ← longitud(SNR) ;
mientras p ≤ 8 hacer
    M ← PARAMETERS(p)(1);
    r ← PARAMETERS(p)(2);
    mientras i ≤ 9 hacer
        P(i) ← evaluarModelo(SNR, M, r, l, num);
        i ← i + 1 ;
    fin
    Guardar: —- en PER_M_R_l;
    p ← p + 1 ;
fin
Agregar valores ;
Exportar: Resultados en PER_ALL_THEO_HOMOGENEOUS;
```

Algoritmo 42: EXTENSION_PER

Leer: PER_SIM_FINO, numEle;
mientras $p \leq 8$ **hacer**
    so $\leftarrow$ SNRV(p,1);
    sf $\leftarrow$ SNRV(p,2);
    SNR $\leftarrow$ $SNum$ elementos entre so y sf;
    ds $\leftarrow$ $SNR(2) - SNR(1)$;
    PER $\leftarrow$ PERT(p,:);
    SNRE $\leftarrow$ Agregar $numEle$ elementos equidistantes a la izquierda y derecha de SNR;
    PERW(1:numEle) $\leftarrow$ 1;
    PERW(numEle + 1:numEle + SNum) $\leftarrow$ PER;
    PERW((numEle + SNum + 1:numEle + SNum + numEle) $\leftarrow$ 0;
    PERTW(p,:) $\leftarrow$ PERW;
    SNRVW(p,1) $\leftarrow$ SNRE(1);
    SNRVW(p,2) $\leftarrow$ SNRE(end);
    p $\leftarrow$ p + 1 ;
fin
SNum $\leftarrow$ SNum + 2*numEle;
Exportar: —— **en** PER_ALL_SIM_EXTENDED;
mientras $p \leq 8$ **hacer**
    M $\leftarrow$ PARAMETERS(p)(1);
    r $\leftarrow$ PARAMETERS(p)(2);
    SNR $\leftarrow$ $SNum$ elementos equidistantes entre $SNRVW(p,1)$ y $SNRVW(p,2)$;
    mientras $j \leq 9$ **hacer**
        PER $\leftarrow$ evaluarModelo(SNR, M, r, l, num) ;
        PERTH(((p-1)*9) + j , :) $\leftarrow$ PER;
        j $\leftarrow$ j + 1;
    fin
    p $\leftarrow$ p + 1;
fin
Agregar valores ;
Exportar: Resultados **en** PER_ALL_THEO_EXTENDED;

Algoritmo 43: COMPARACION_PER

Leer: PER_SIM_FINO;
C0 ← Matriz con offsets iniciales por caso por configuración ;
mientras $p \leq 8$ **hacer**
    M ← PARAMETERS(p)(1);
    r ← PARAMETERS(p)(2);
    SNR ← $SNum$ elementos equidistantes entre $SNRVW(p,1)$ y $SNRVW(p,2)$;
    PER ← PERT(p,:);
    mientras $q \leq 9$ **hacer**
        cini ← C0(p,q);
        C(p,q), E(p,q) ← minimizarErrorEQX(SNR, DatosE, l, M, r, numEq, c0);
        q ← j + 1;
    fin
    p ← p + 1;
fin
Exportar: C, E **en** minimized_C_E;
mientras $p \leq 8$ **hacer**
    M ← PARAMETERS(p)(1);
    r ← PARAMETERS(p)(2);
    SNR ← $SNum$ elementos equidistantes entre $SNRVW(p,1)$ y $SNRVW(p,2)$;
    mientras $j \leq 9$ **hacer**
        ci ← C(p,j);
        PER ← evaluarModelo(SNR, M, r, l, num);
        PERTHMIN(((p1)*9) + j , :) ← PER;
        j ← j + 1;
    fin
    p ← p + 1;
fin
Exportar: Resultados **en** PER_ALL_THEO_MINIMIZED;

Algoritmo 44: CONSOLIDACION_DATOS

Leer: PER_ALL_SIM_EXTENDED;
Leer: PER_ALL_THEO_EXTENDED;
Leer: PER_ALL_THEO_HOMOGENEOUS;
Leer: minimized_C_E;
Leer: PER_SIM_FINO;
Leer: PER_ALL_THEO_MINIMIZED;
Leer: PER_SIM_HOM;
Consolidar datos ;
Exportar: Datos consolidados **en** DATOS_CONSOLIDADOS;

Algoritmo 45: GRAFICACION_RESULTADOS

```
1  Leer: DATOS_CONSOLIDADOS;
2  mientras i ≤ 8 hacer
3      Graficar: PER_SIM_HOM(i);
4      i ← i + 1 ;
5  fin
6  mientras i ≤ 8 hacer
7      Generar nueva figura ;
8      Graficar: PER_SIM_FIN_EXT(i);
9      mientras j ≤ 9 hacer
10         Graficar: PER_THEO_FIN_EXT((i-1)*9+j);
11         j ← j + 1;
12     fin
13     i ← i + 1;
14 fin
15 mientras i ≤ 8 hacer
16     Generar nueva figura ;
17     Graficar: PER_SIM_FIN(i);
18     mientras j ≤ 9 hacer
19         Graficar: PER_THEO_FIN_MIN((i-1)*9+j);
20         j ← j + 1;
21     fin
22     i ← i + 1;
23 fin
24 mientras i ≤ 8 hacer
25     Generar nueva figura ;
26     mientras j ≤ 9 hacer
27         Graficar: C_MIN((i-1)*9+j);
28         j ← j + 1;
29     fin
30     i ← i + 1;
31 fin
32 mientras i ≤ 8 hacer
33     Generar nueva figura ;
34     mientras j ≤ 9 hacer
35         Graficar: E_MIN((i-1)*9+j);
36         j ← j + 1;
37     fin
38     i ← i + 1;
39 fin
```

Algoritmo 46: ANALISIS_RESULTADOS

```
1  Leer: C_MIN, E_MIN de DATOS_CONSOLIDADOS;
2  mientras i ≤ 8 hacer
3      cmin ← min(abs(C_MIN(i,:)));
4      cpos ← 0 ;
5      mientras j ≤ 9 hacer
6          si abs(C_MIN(i,j)) == cmin entonces
7              j ← 9;
8          en otro caso
9              cpos ← j;
10             cmin ← C_MIN(i,j);
11         fin
12         j ← j + 1;
13     fin
14     C_RES(i,1) ← cmin;
15     C_RES(i,2) ← cpos;
16     i ← i + 1 ;
17 fin
18 mientras i ≤ 8 hacer
19     emin ← min(abs(E_MIN(i,:)));
20     epos ← 0 ;
21     mientras j ≤ 9 hacer
22         si abs(E_MIN(i,j)) == emin entonces
23             j ← 9;
24         en otro caso
25             epos ← j;
26             emin ← E_MIN(i,j);
27             cemin ← C_MIN(i,j);
28         fin
29         j ← j + 1;
30     fin
31     E_RES(i,1) ← emin;
32     E_RES(i,2) ← epos;
33     E_RES(i,3) ← cemin;
34     i ← i + 1 ;
35 fin
36 mientras i ≤ 8 hacer
37     Exportar: "Offset mínimo: " C_RES(i,1) ", Modelo más exacto: " C_RES(i,2) en
        PANTALLA;
38     Exportar: "Error mínimo:" E_RES(i,1) ", Modelo más preciso: " E_RES(i,2) "
        (offset = " E_RES(i,3) ")" en PANTALLA;
39     i ← i + 1 ;
40 fin
```

ANEXO III

En este anexo se adjunta un CD que contiene los diferentes *scripts* y funciones implementados, así como los archivos .mat que contienen los resultados generados y los archivos .eps que contienen las gráficas obtenidas.

ESCUELA POLITÉCNICA NACIONAL
"CAMPUS POLITÉCNICO JOSÉ RUBÉN ORELLANA RICAURTE"

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ORDEN DE EMPASTADO

De acuerdo con lo estipulado en el Art. 27 del Instructivo para la Implementación de la Unidad de Titulación en las Carreras y Programas Vigentes de la Escuela Politécnica Nacional, aprobado por Consejo Politécnico en sesión extraordinaria del 29 de abril de 2015 y por delegación del Decano, una vez verificado el cumplimiento de formato de presentación establecido, se autoriza la impresión y encuadernación final del Trabajo de Titulación presentado por:

XAVIER ALEJANDRO FLORES CABEZAS

Fecha de autorización: ***07 de enero de 2019***

M.Sc. YADIRA BRAVO
Subdecana

Alejandra P.