INGENIERÍA EN TELECOMUNICACIONES

IMPLEMENTACIÓN DE ESTACIONES RECEPTORAS PARA EL MONITOREO DEL MOVIMIENTO ANIMAL

Gabriel Iván Garcés
Para aspirar al grado de Ingeniero en Telecomunicaciones por la Universidad Nacional de Río Negro

Jorge Cogo
Director

Karina Laneri
Co-directora

6 de Noviembre de 2024

Lugar de trabajo - CITECCA - UNRN & FIESTIN - CAB (CNEA)

Universidad Nacional de Río Negro - Sede Andina
Argentina

Para mi familia, para mis amigos, gracias de corazón.

"Ce ne sont pas seulement les liens du sang qui forment la parenté, mais ceux du coeur et de l' intelligence."

Montesquieu

Índice de contenidos

Índice de contenidos **3**

1. Introducción **5**

2. Sistema ATLAS **7**

2.1. Componentes Principales 8

2.2. Contenido de paquetes de transmisores de ATLAS 9

2.3. Estado del Arte de los Métodos de Demodulación 10

3. Esquemas de Modulación y Demodulación **11**

3.1. Modulacion BFSK (*Binary Frequency-Shift Keying*) 11

3.1.1. Ancho de Banda FSK 12

3.2. Demodulación FSK 13

3.2.1. Probabilidad de Error de Bit BFSK 13

3.2.2. Demodulación mediante correladores 15

3.2.3. Demodulación mediante Discriminador de Frecuencia 18

3.2.4. Demodulación mediante un PLL 19

3.2.5. Análisis de la señal de entrada 20

4. Lazo de Seguimiento de Fase (*Phase-Locked Loop*) **25**

4.1. PLL de tiempo continuo 25

4.2. PLL de tiempo discreto 26

4.3. Diseño de un PLL 27

4.4. Implementación de PLL de tiempo discreto de señales complejas 36

4.4.1. Uso del PLL complejo como Demodulador PLL-FSK 39

4.4.2. Diseño Final del PLL de tiempo discreto de señales complejas . 40

5. Códigos Gold **45**

5.1. Ruido Pseudoaleatorio (*Pseudorandom noise*) 45

5.2. Códigos Gold 46

5.2.1. Secuencias de longitud máxima (secuencias-m) 46

5.2.2. Construcción de los códigos Gold 47

6. Resultados **51**
6.1. Generación de Códigos Gold y análisis de correlación 51
6.2. Análisis del desempeño de Tasa de Error de Bit (BER) 54
6.3. Análisis de desempeño de sincronismo 59

7. Conclusiones **71**

Bibliografía **75**

A. Código MATLAB **79**
A.1. Modulador BFSK 79
A.2. Demodulador PLL-FSK 80
A.3. Demodulador Mediante Discriminador de Frecuencias 83
A.4. Demodulación mediante Correladores 84
A.5. BER PLL-FSK 85
A.6. BER Discriminador 87
A.7. BER Correladores 88
A.8. Estimación de BER para los tres Demoduladores 90
A.9. Espectros de la primera zona de Nyquist para distintas f_s 93
A.10. Espectros Error de Fase para distintas f_s 96
A.11. Respuestas en Frecuencias para distintas Frecuencias de Muestreo . . . 99
A.12. Generador de Código Gold 101
A.13. Correlación Cruzada para Código Gold de 31 bits en un canal sin ruido 102
A.14. Correlación Cruzada para Código Gold de 4095 bits en un canal sin ruido 105
A.15. Correlación entre Error Filtrado y Código Gold de 31 bits en un canal ideal 108
A.16. Correlación entre Error Filtrado y Código Gold de 4095 bits en un canal ideal 110
A.17. Sincronización con Gold 31 Bits sin señal al principio y final 112
A.18. Sincronización con Gold 31 Bits con secuencia de unos y menos unos alternados al principio y final 116
A.19. Sincronización con Gold 4095 Bits sin señal al principio y final 119
A.20. Sincronización con Gold 4095 Bits con secuencia de unos y menos unos alternados al principio y al final 122
A.21. Obtención de los umbrales de decisión para el Demodulador BFSK mediante Discriminador de Frecuencias 125

Capítulo 1

Introducción

La fauna silvestre enfrenta desafíos crecientes debido a la expansión de las actividades humanas, la fragmentación de hábitats y el cambio climático. Estas presiones han generado una necesidad urgente de herramientas precisas y accesibles para el monitoreo de especies, particularmente en áreas remotas o de difícil acceso. En este contexto, los sistemas de rastreo se han convertido en una herramienta clave para estudiar el comportamiento, los desplazamientos y los patrones de migración de una amplia variedad de especies. Tradicionalmente, los biólogos y ecólogos han recurrido a sistemas comerciales basados en algún Sistema Global de Navegación por Satélite (GNSS, por sus siglas en inglés [Montenbruck and Teunissen, 2017]), como el GPS, para rastrear animales en tiempo real o de manera asincrónica. Sin embargo, estos sistemas presentan limitaciones importantes en ciertos entornos, particularmente en áreas boscosas o con densa vegetación, donde las señales satelitales son débiles o inexistentes. En regiones boscosas, como lo son los bosques patagónicos, los sistemas GNSS suelen fallar en proporcionar datos precisos debido a la falta de cobertura confiable. El sistema ATLAS ha surgido como una alternativa a explorar en estos entornos desafiantes. Este sistema de rastreo de vida silvestre utiliza tecnología de radiofrecuencia para obtener la localización de animales en tiempo real mediante el uso de transmisores ligeros y una red de estaciones receptoras. Una de las principales ventajas de ATLAS es su potencial capacidad para funcionar en áreas donde las señales GNSS no son detectables o donde éstas no tienen la precisión necesaria para estudios detallados de comportamiento animal.

La precisión de la estimación de ubicación de sistemas como ATLAS depende en gran medida de las técnicas de modulación y demodulación empleadas en la transmisión y recepción de señales. En este contexto, la sincronización precisa entre transmisores y receptores representa uno de los desafíos fundamentales para garantizar la calidad de la solución de la posición. El Lazo de Seguimiento de Fase emerge como una solución tecnológica que, desde su concepción inicial como un dispositivo analógico hasta sus implementaciones digitales contemporáneas, ha demostrado ser una herramienta versátil

en múltiples aplicaciones. Su capacidad para realizar el seguimiento preciso tanto de la fase como de la frecuencia de señales sinusoidales lo convierte en un componente esencial en sistemas de comunicación digital, donde se utiliza extensivamente para la demodulación de señales moduladas en frecuencia, la sincronización de portadora y la temporización de símbolos.

En el contexto específico de los sistemas de monitoreo de fauna silvestre existe la necesidad de identificar y seguir múltiples dispositivos simultáneamente. En este escenario, los códigos Gold emergen como una herramienta fundamental debido a sus excelentes propiedades de correlación cruzada, buenas propiedades de auto-correlación y su capacidad para generar grandes conjuntos de Códigos distintos entre sí pertenecientes a una misma familia.

El presente trabajo se enfoca en tres aspectos principales:

- El análisis y optimización de los métodos de demodulación empleados en el sistema ATLAS, abordando tres esquemas principales: técnicas basadas en correladores, discriminadores de frecuencia y Lazo de Seguimiento de Fase.
- Un análisis del Lazo de Seguimiento de Fase, desde sus fundamentos teóricos hasta su diseño como demodulador BFSK.
- La evaluación del rendimiento de estos demoduladores BFSK diferentes en la recepción de señales binarias aleatorias moduladas, analizando su desempeño en diferentes condiciones de ruido y, además, la capacidad de sincronización bajo diferentes condiciones de ruido y frecuencias de muestreo con códigos Gold.

Capítulo 2

Sistema ATLAS

El propósito de este capítulo es ofrecer una descripción detallada del funcionamiento del sistema ATLAS desde sus aplicaciones exitosas hasta sus componentes técnicos y potenciales mejoras en el futuro.

ATLAS ha sido implementado con éxito en varios países y ha proporcionado resultados altamente satisfactorios en el monitoreo de fauna voladora. Los estudios más destacados incluyen:

- Israel (Valle de Hula): Uno de los primeros sitios de implementación de ATLAS, donde se ha utilizado para rastrear aves. La precisión alcanzada en esta región ha sido notable, con un margen de error de 5 metros en condiciones ideales [1], [2].

- Países Bajos (Mar de Wadden): ATLAS ha sido utilizado para rastrear la migración de aves a través del Mar de Wadden. Este ecosistema de humedales ofrece un desafío único debido a la presencia de agua y las grandes distancias que cubren los animales. A pesar de estas condiciones, el sistema ha logrado proporcionar datos precisos sobre las rutas migratorias y el comportamiento de las especies [3].

- Reino Unido: Otros estudios recientes han demostrado la eficacia de ATLAS en el seguimiento de aves y murciélagos en estas regiones. El sistema ha sido fundamental para estudiar patrones de vuelo y zonas de alimentación de especies en peligro de extinción, contribuyendo a proyectos de conservación [4].

Sin embargo, la aplicación de ATLAS a especies terrestres representa un nuevo desafío y una posible innovación en el campo del rastreo de fauna, dado que por ejemplo, numerosas especies con interés en conservación en nuestro país requieren de un sistema de seguimiento por radiotelemetría preciso, portable, de bajo costo y adaptable a cada especie a monitorear.

2.1. Componentes Principales

El sistema ATLAS es un ejemplo de una tecnología denominada reverse-GPS o sistema de localización inversa. A diferencia de los sistemas GNSS tradicionales, donde los satélites transmiten señales y los receptores calculan su posición [Montenbruck and Teunissen, 2017], en ATLAS la relación se invierte: las etiquetas transmisoras en los animales envían señales de radio a un conjunto de estaciones receptoras fijas, que se encargan de calcular la posición del transmisor utilizando la diferencia en los tiempos de llegada (TDOA, por sus siglas en inglés) de las señales [5]. El sistema ATLAS está conformado por los siguientes componentes principales descriptos a continuación.

Dispositivo transmisor: El corazón del sistema ATLAS es un pequeño transmisor, denominado etiqueta (*tag*, en inglés) de radiofrecuencia que se coloca en los animales que se desean rastrear. Estos transmisores son de bajo costo, ligeros y consumen poca energía, lo que les permite funcionar durante largos periodos sin necesidad de mantenimiento frecuente. En la mayoría de los casos, los transmisores operan en la banda de 434 MHz utilizando modulación por desplazamiento de frecuencia binaria (BFSK) o modulación por desplazamiento de frecuencia gausiana (GFSK). Los modelos de segunda generación, como los basados en el chip TI CC1310 [6], permiten una transmisión eficiente, con velocidades de hasta 4 Mb/s y un rango de transmisión adecuado para áreas de gran extensión. En la Figura 2.1 se puede observar una fotografía de los transmisores del sistema ATLAS, cuyo tamaño es 19 mm por 11 mm [5].

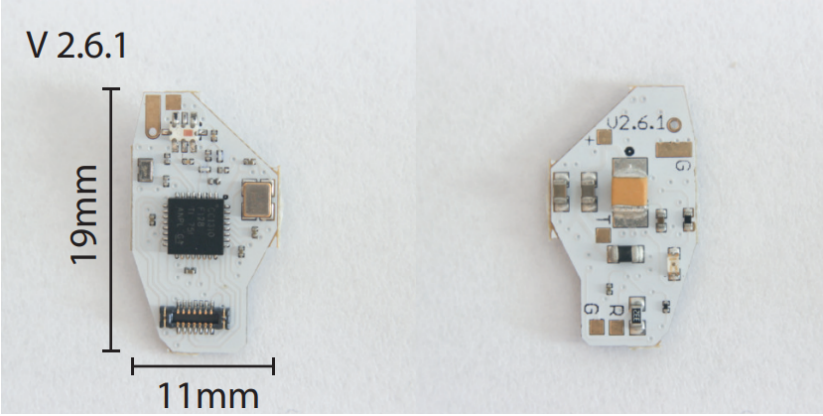

Figura 2.1: Fotografía de los transmisores del sistema ATLAS.

Estaciones receptoras: Las estaciones receptoras son dispositivos fijos equipados con antenas y hardware de procesamiento. Estas estaciones se sincronizan entre sí utilizando relojes internos de alta precisión, lo que les permite medir con exactitud la diferencia en los tiempos de llegada de las señales transmitidas por las etiquetas. Cada estación recibe la señal, la demodula y calcula el tiempo exacto de arribo. La clave de este sistema radica en la capacidad de las estaciones para comunicarse entre sí y comparar los tiempos de llegada, permitiendo así la localización precisa del transmisor

mediante técnicas de triangulación.

Algoritmo de triangulación: La técnica de triangulación utilizada en ATLAS es similar a la que se emplea en otros sistemas de localización, pero sin hacer uso de señales satelitales. El algoritmo utiliza las diferencias en los tiempos de llegada (TDOA) para calcular la posición del transmisor. Cada estación proporciona un dato de tiempo de arribo, y al comparar las diferencias de estos tiempos entre múltiples estaciones, es posible estimar la posición del transmisor en el espacio. Este proceso requiere una alta sincronización entre las estaciones receptoras para minimizar el error en la localización. En la Figura 2.2 se puede observar el caso particular en el que se tiene un transmisor con un *tag* particular y tres estaciones receptoras dispuestas para realizar el cálculo de su posición estimada.

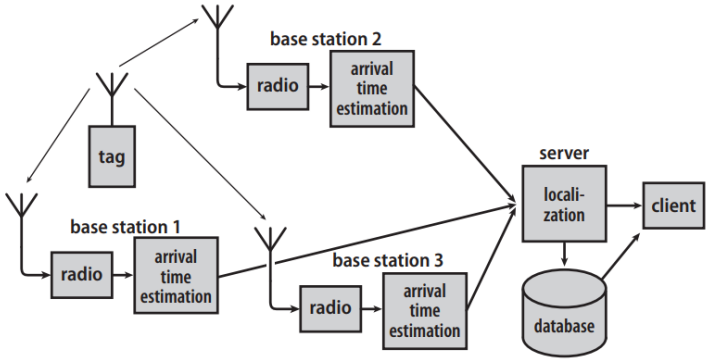

Figura 2.2: Componentes principales del Sistema ATLAS.

2.2. Contenido de paquetes de transmisores de ATLAS

Los paquetes transmitidos por los transmisores del sistema ATLAS están estructurados de manera que faciliten la identificación y seguimiento de los *tags*. A continuación, se detalla cómo están conformados los paquetes y las implicancias de esta particularidad en la detección y demodulación de las señales [7].

Los transmisores de ATLAS utilizan modulación FSK Binaria (BFSK) con una frecuencia central de 434 MHz, una desviación de frecuencia de 381 kHz, y una tasa de símbolos de 1 M bit/s. Cada paquete tiene una longitud fija de 8192 bits, lo que permite la transmisión de secuencias pseudoaleatorias, con cada *tag* emitiendo una secuencia conocida, única y que difiere de la de los demás *tags*.

Los paquetes carecen de una palabra de sincronización explícita, aunque se incluye un preámbulo 010101 que se repite al inicio de la transmisión. La falta de una palabra de sincronización en los paquetes de los transmisores ATLAS introduce varias implicancias para la recepción y sincronización de las señales. En sistemas que incluyen una palabra de sincronización, ésta sirve como un marcador claro para que el receptor identifique

el inicio de un paquete. En ausencia de este marcador, el receptor debe recurrir a otros métodos de detección, como la correlación con secuencias conocidas, metodología utilizada en este sistema para la identificación de los respectivos *tags* transmisores. Sin embargo, esto agrega una capa de complejidad, ya que el receptor debe estar atento a la aparición de la señal en cualquier momento.

2.3. Estado del Arte de los Métodos de Demodulación

Para demodular BFSK existen varios métodos, entre ellos:

- Demodulación mediante Lazo de Seguimiento de Fase (PLL, por sus siglas en inglés).
- Demodulación mediante correladores (también llamado Demodulador en Cuadratura).
- Demodulación mediante discriminador de frecuencias.

Estos métodos son en los que se ahondará en este trabajo final de grado. Por otra parte, además, los sistemas de rastreo basados en TDOA dependen en gran medida de la precisión con la que se puedan medir los tiempos de llegada de las señales de radio y, el cálculo del tiempo de llegada (TOA, por sus siglas en inglés) se realiza mediante correlación cruzada de códigos pseudoaleatorios que permiten comparar la señal recibida con la señal de la etiqueta esperada [7]. Esto permite identificar el punto en el tiempo donde las señales coinciden, lo que se considera como el momento exacto de llegada. Por ende en este trabajo también se abordará este punto.

El sistema ATLAS ofrece una solución innovadora y eficiente para el monitoreo de fauna en una variedad de entornos, particularmente en aquellos donde los sistemas GNSS no funcionan adecuadamente. Su capacidad para ofrecer una alta precisión de localización, junto con su bajo costo y peso ligero, lo convierten en una herramienta invaluable para estudios ecológicos y proyectos de conservación. A medida que la tecnología continúe avanzando y los algoritmos de demodulación y triangulación se perfeccionen, es probable que ATLAS se expanda a nuevas aplicaciones, incluyendo el seguimiento de especies terrestres en áreas de difícil acceso. Este sistema tiene el potencial de aplicabilidad para el monitoreo de fauna en regiones como la Patagonia, ofreciendo una solución eficaz a los desafíos de rastreo en entornos complejos.

Capítulo 3

Esquemas de Modulación y Demodulación

3.1. Modulacion BFSK (*Binary Frequency-Shift Keying*)

BFSK se refiere a la **modulación binaria por desplazamiento de frecuencia**. La FSK binaria es una forma de modulación de ángulo, de amplitud constante, parecida a la modulación convencional de frecuencia (FM), pero la señal moduladora es una señal binaria que varía entre dos valores discretos de tensión, y no es una forma de onda analógica que cambie continuamente [8]. Se trata de la forma más simple de modulación FSK donde los dos bits 0 y 1 corresponden a dos frecuencias distintas, la primer frecuencia F_1 que es la frecuencia inferior comunmente llamada frecuencia de espacio y, F_2 que es la frecuencia superior comunmente llamada frecuencia de marca. Los bits pueden ser traducidos a símbolos a través de las relaciones:

$$\begin{aligned} 0 &\rightarrow -1 \\ 1 &\rightarrow +1 \end{aligned} \tag{3.1}$$

Esto nos permite escribir las dos frecuencias F_i con $i \in \{1, 2\}$:

$$F_i = F_c + (-1)^i \cdot \Delta_F = F_c \pm \Delta_F \tag{3.2}$$

donde F_c es la frecuencia portadora nominal y Δ_F es la desviación de frecuencia pico con respecto a esta frecuencia portadora que se introducirá en la señal modulada en cada Tiempo de Bit. La forma de onda de la señal se puede escribir como:

$$s(t) = A\cos\left(2\pi F_i t + \phi\right) = A\cos\left[2\pi\big(F_c \pm \Delta_F\big)t + \phi\right] \tag{3.3}$$

Que es equivalente a:

$$s(t) = \Re\{Ae^{j(2\pi F_i t+\phi)}\} = \Re\left\{Ae^{\left[j\left(2\pi\left(F_c \pm \Delta_F\right)t+\phi\right)\right]}\right\} \tag{3.4}$$

para $k \cdot T_b \leq t \leq (k+1) \cdot T_b$, con T_b el tiempo de bit, siendo k un entero que indica el k-ésimo bit, y ϕ una fase arbitraria.

La Figura 3.1 muestra una forma de onda BFSK para un flujo aleatorio de datos a una velocidad de $R_b = 1/T_b$. Se debe tener en cuenta que no estamos distinguiendo entre un período de bit y un período de símbolo porque ambos son lo mismo para una técnica de modulación binaria.

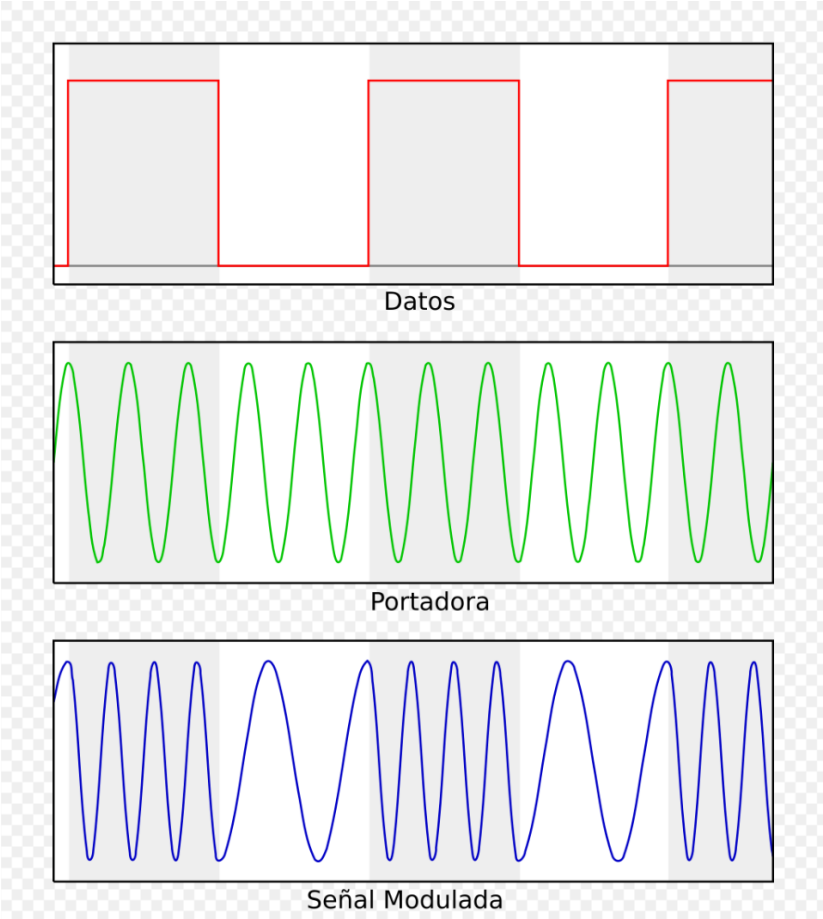

Figura 3.1: Modulación Binaria FSK.

En este trabajo utilizaremos la representación compleja (IQ) de las señales moduladas en BFSK por lo que la señal modulada utilizada viene dada por la ecuación:

$$s_{IQ}(t) = Ae^{\left[j\left(2\pi\left(F_c \pm \Delta_F\right)t+\phi\right)\right]} = A\cos\left(2\pi F_i t + \phi\right) + jA\,\mathrm{sen}\left(2\pi F_i t + \phi\right) \tag{3.5}$$

3.1.1. Ancho de Banda FSK

Si consideramos el ancho de banda del primer nulo a nulo para dos bits consecutivos modulados (bit cero y bit uno), entonces el ancho de banda resultante para la señal

BFSK está dado por:

$$B = 2\frac{1}{T_b} + 2\Delta_F \tag{3.6}$$

3.2. Demodulación FSK

Existen dos técnicas de demodulación de una señal FSK:

- Coherente
- No coherente

La demodulación coherente se basa en el conocimiento de la fase y la frecuencia de la señal portadora, mientras que la demodulación no coherente no requiere sincronización de fase. La demodulación FSK coherente es capaz de ofrecer un mejor desempeño en términos de probabilidad de error de bit, pero la FSK no coherente es más sencilla de implementar y se utiliza para la mayoría de las transmisiones FSK. Este último tipo de demodulación es el que se utilizará en este trabajo.

3.2.1. Probabilidad de Error de Bit BFSK

Probabilidad de Error de Bit BFSK mediante detección coherente:

La probabilidad teórica de error de bit (P_{eb}) de una señal FSK binaria detectada de forma coherente sobre ruido gaussiano blanco aditivo (AWGN, por sus siglas en inglés), donde los dos símbolos son ortogonales y equiprobables, viene dada por [9]:

$$P_{eb} = Q\left(\sqrt{\frac{E_b}{N_0}}\right) = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E_b}{2N_0}}\right) \tag{3.7}$$

donde $E_b = \frac{1}{2}A^2$ es la energía por bit de la señal FSK, donde A es la amplitud de la misma y, donde N_0 es la densidad espectral de potencia de ruido.

Probabilidad de Error de Bit BFSK mediante detección no coherente:

La probabilidad teórica de error de bit de una señal FSK binaria detectada de forma no coherente sobre ruido AWGN, donde los dos símbolos son ortogonales y equiprobables, viene dada por [9]:

$$P_{eb} = \frac{1}{2}e^{\frac{-E_b}{2N_0}} \tag{3.8}$$

La Probabilidad de Error de Bit teórica se calculó bajo las condiciones clave que el conjunto de señales a utilizar son ortogonales entre sí, siendo $\frac{1}{T_b}$ la separación entre

lóbulos del conjunto de señales a utilizar, en este caso $s_1(t)$ y $s_2(t)$. Además, el demodulador a utilizar es un Detector No Coherente como el que se puede observar en la Figura 3.2 que consta de dos canales con filtros pasa banda, cada uno con un ancho de banda $B = \frac{1}{T_b}$, y detectores de envolvente. Aquí, no se necesita conocer la fase de la señal transmitida pero si las frecuencias F_1 y F_2 transmitidas. En la Figura 3.3 se puede observar una comparación entre las probabilidad de error de bit coherente contra su par no coherente, donde podemos observar que en el caso de la demodulación BFSK no coherente se necesita aproximadamente 1 dB más de E_b/N_0 en comparación con la demodulación BFSK coherente para una misma probabilidad de error. A partir de esto verificamos el rendimiento de los métodos de demodulación midiendo su Tasa de Error de Bit (BER).

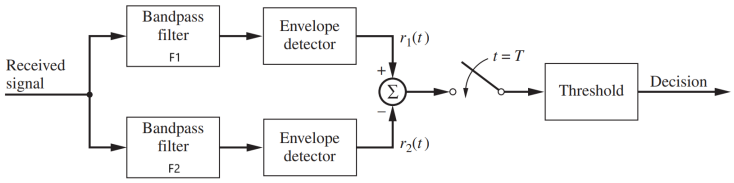

Figura 3.2: Detección no coherente de FSK mediante detectores de envolvente.

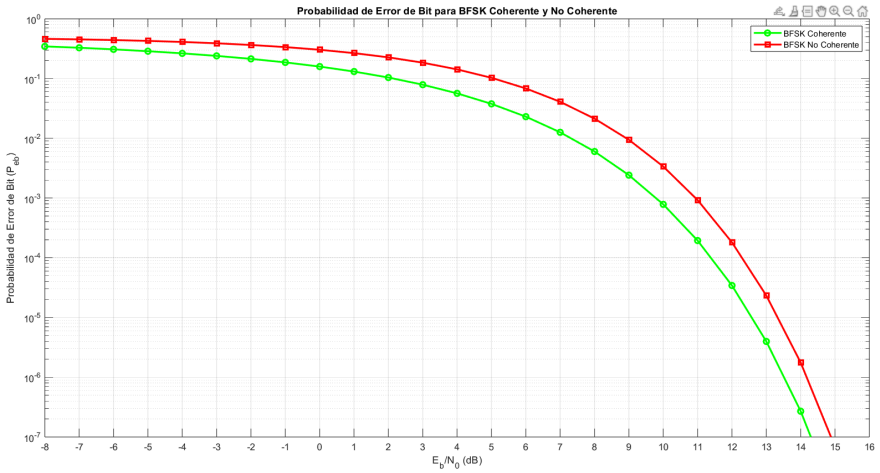

Figura 3.3: Probabilidades de Error de Bit BFSK mediante detección Coherente (curva verde) y mediante detección No Coherente (curva roja).

Ahora bien, también existen distintas formas de demodulación no coherente. Las técnicas simuladas en esta tesis fueron la demodulación mediante correladores, mediante Discriminador de Frecuencias y, mediante un Lazo de Seguimiento de Fase (PLL), que se explican a continuación.

3.2.2. Demodulación mediante correladores

Se puede implementar un demodulador para la detección no coherente mediante correladores o también denominado demodulador en cuadratura [9]. En la Figura 3.4 se ilustran los canales en fase (I) y en cuadratura (Q) utilizados para detectar una señal binaria FSK (BFSK) de manera no coherente.

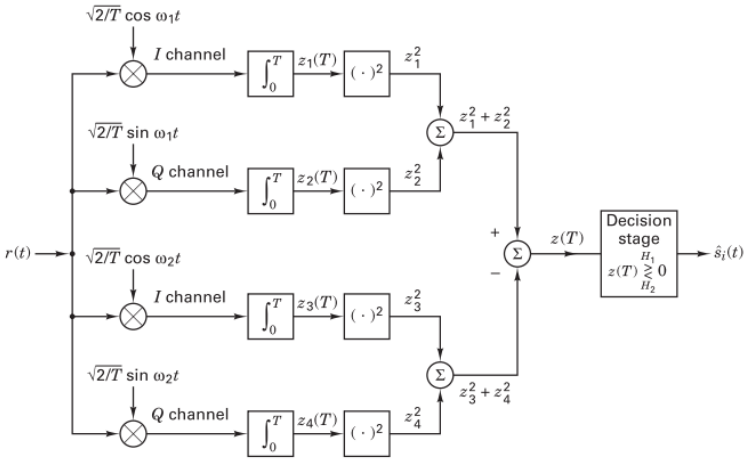

Figura 3.4: Demodulador en cuadratura.

Las dos ramas superiores que conforman el primer brazo están configuradas para detectar la señal con frecuencia F_1 y las dos ramas inferiores que conforman el segundo brazo están configuradas para detectar la señal con frecuencia F_2.

Imaginemos que la señal de entrada es $r(t) = A\cos(\omega_1 t) + n(t)$, siendo $\omega_1 = 2\pi F_1$. Se puede observar que, con fase igual cero, la señal recibida coincide exactamente en frecuencia y fase con la señal de referencia de la rama superior del primer brazo. En ese caso, el integrador de productos de la rama superior debería producir el máximo valor de salida. La segunda rama, debería producir una salida cercana a cero (dependiendo del nivel de ruido aditivo) ya que su señal de referencia $\text{sen}(\omega_1 t)$ es ortogonal a la señal $r(t)$. Por su parte, las ramas tercera y cuarta también deberían producir salidas cercanas a cero, ya que sus señales de referencia también son ortogonales a la señal $r(t)$. Esta señal $r(t)$ descripta es solo una componente del total de lo que implicaría la transmisión de una señal BFSK. Si se diera el caso en el que la señal $r(t)$ tuviese un desfase de 90° respecto del caso anterior, con lo cual la señal de entrada sería $r(t) = A\,\text{sen}(\omega_1 t) + n(t)$, la segunda rama del primer brazo produciría la salida máxima y el resto de las ramas tanto del primer brazo como del segundo brazo producirían salidas iguales a cero. En un escenario mixto entre el primer y segundo caso, es decir que la señal recibida posea un desfase arbitrario, el primer brazo será el que produzca la máxima salida respecto del segundo brazo. De manera equivalente esto se puede extrapolar para señales con frecuencia F_2 para el segundo brazo.

Los bloques que siguen a los integradores realizan una elevación al cuadrado para

evitar la aparición de valores negativos. Por lo tanto, en el primer brazo se suman z_1^2 del canal I y z_2^2 del canal Q y, en el segundo brazo se suman z_3^2 del canal I y z_4^2 del canal Q. Por último, en la etapa final a partir de un valor umbral especifico $z(T)$ se elige la señal con frecuencia F_1 o la señal con frecuencia F_2, dependiendo que par de detectores de energía produce la salida máxima.

Similitud entre implementaciones de demodulación BFSK

La demodulación BFSK no coherente puede implementarse mediante filtros pasa banda como se detalló en la sección 3.2.1, como también mediante correladores como se detalló en la sección anterior. Ahora bien, el correlador implementa una versión de filtro pasabanda, por lo que, estas dos implementaciones resultan ser matemáticamente similares.

En primer lugar, ambas implementaciones comparten una estructura fundamental similar: utilizan dos canales o "brazos" paralelos, cada uno dedicado a detectar la presencia de una de las frecuencias características de la señal BFSK (F_1 o F_2). En el caso de los filtros, esto se logra mediante filtros pasa banda sintonizados a estas frecuencias específicas, mientras que en el caso de los correladores, se utiliza la multiplicación de la señal por referencias sinusoidales a estas mismas frecuencias.

En segundo lugar, la similitud matemática entre ambas implementaciones viene dada por la relación siguiente, cuando en la implementación con correladores multiplicamos la señal de entrada $r(t)$ por las referencias $\cos(\omega_1 t)$ y $\text{sen}(\omega_1 t)$ y luego integramos el resultado, estamos efectivamente realizando una operación equivalente a un filtrado pasa banda centrado en la frecuencia F_1. Lo que se extiende al segundo brazo.

En términos prácticos, ambas implementaciones buscan detectar la energía presente en cada una de las frecuencias características. En la implementación con filtros, esto se logra mediante el filtrado pasa banda seguido de detectores de envolvente. Por otro lado, en la implementación con correladores, se obtiene mediante la multiplicación por referencias en fase y cuadratura, integración y posterior detección de energía a través de la suma de los cuadrados de las componentes en fase y cuadratura ($I^2 + Q^2$).

Ortogonalidad entre tonos

Ahora bien ¿Cómo se puede determinar que las señales con frecuencias F_1 y F_2 son ortogonales entre sí, tal que en el demodulador en cuadratura, una señal de una frecuencia dada produzca una salida cercana a cero en el brazo cuya señal de referencia no posee esa frecuencia determinada? Los tonos F_1 y F_2 manifiestan ortogonalidad si, para un tono transmitido en F_1, la envolvente muestreada del filtro de salida del receptor sintonizado en F_2 es cero. Se asegura dicha ortogonalidad entre tonos si cualquier par de tonos en el conjunto tiene una separación de frecuencia que sea múltiplo de $1/T_b$.

Para que un tono detectado de manera no coherente manifieste una señal de salida máxima en su filtro receptor asociado y una señal de salida cero en cualquier filtro vecino, el pico del espectro del tono 1 debe coincidir con uno de los cruces por cero del espectro del tono 2. La diferencia de frecuencia entre el centro del lóbulo principal espectral y el primer cruce por cero representa el espacio mínimo requerido. En la Figura 3.5 se logra observar este hecho.

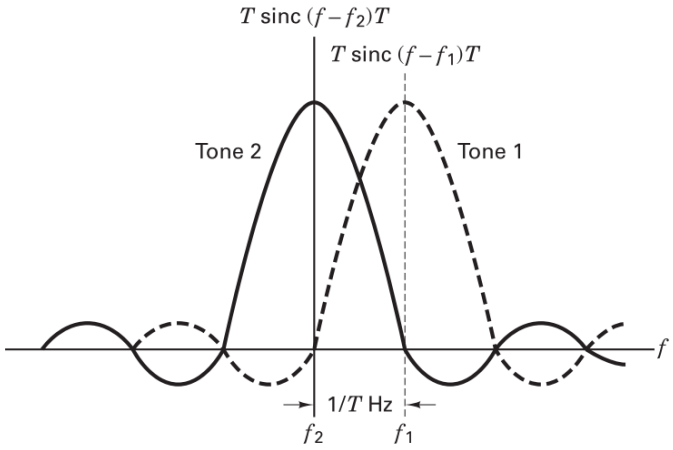

Figura 3.5: Espaciado mínimo de tonos para señalización FSK ortogonal.

Una característica clave del método de demodulación mediante correladores es que requiere un conocimiento preciso de las frecuencias F_1 y F_2. En este esquema, las señales de referencia del demodulador deben coincidir exactamente con las frecuencias de la señal modulada que se desea detectar. Si las frecuencias portadoras no son conocidas de antemano, entonces sería necesario implementar un mayor número de correladores, cada uno sintonizado a diferentes frecuencias posibles dentro del espectro de la señal, y luego seleccionar la salida correspondiente al correlador que produzca el valor máximo. Este proceso incrementa significativamente la complejidad del sistema, ya que implicaría diseñar una serie de correladores paralelos o un sistema de barrido de frecuencias que evalúe las distintas posibilidades. Con lo cual se puede decir que este tipo de demodulador mediante correladores es un demodulador "Semi-Coherente" dado que es necesario conocer previamente las dos frecuencias F_1 y F_2, mientras que la fase de las mismas no es necesario conocerlas.

En contraste, los demoduladores basados en discriminadores de frecuencia, como los sistemas que emplean un derivador más un detector de envolvente, ó aquellos que utilizan un PLL, no requieren un conocimiento previo de las frecuencias exactas de las señales moduladas. En estos métodos, la detección de las señales se realiza a través del seguimiento de los cambios en la frecuencia de la señal recibida, lo que los hace inherentemente más flexibles en escenarios donde las frecuencias portadoras pueden variar o no ser conocidas con precisión. Esto es especialmente útil en sistemas don-

de las frecuencias de transmisión pueden variar ligeramente debido a efectos como el desplazamiento Doppler o la falta de precisión en los osciladores del transmisor.

3.2.3. Demodulación mediante Discriminador de Frecuencia

Este tipo de receptor toma prestados conceptos de la FM analógica. La FM se puede demodular convirtiendo los cambios de frecuencia en cambios de amplitud. A continuación, se pueden aplicar las técnicas utilizadas para demodular la modulación de amplitud (AM). El receptor utilizará un Discriminador de Frecuencias que consiste en un diferenciador para convertir las frecuencias en amplitudes y, a continuación, un detector de envolvente más un filtro pasa-bajos para eliminar la frecuencia portadora y dejar los datos. En la Figura 3.6 se puede observar un esquema de este tipo de demodulador BFSK.

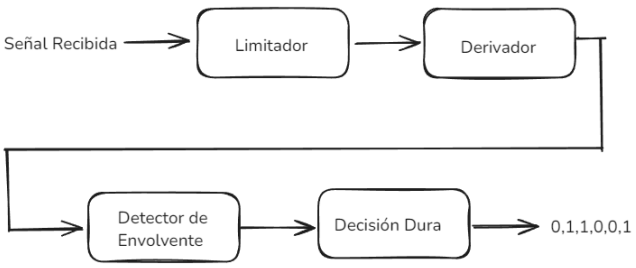

Figura 3.6: Diagrama de Bloques de Demodulador mediante Discriminador de Frecuencias.

Ahora bien, antes del derivador agregamos un limitador. Un limitador es un componente que restringe el rango de valores de amplitud de una señal. En comunicaciones y procesamiento de señales, un limitador se utiliza para evitar que la señal exceda ciertos niveles, lo cual es útil para prevenir distorsiones y mantener la señal dentro de un rango operativo deseado. En este caso particular será útil dado que las señales que demodularemos pasan a través de un canal con ruido gaussiano aditivo.

¿Por qué debería ir antes del derivador? El derivador toma la tasa de cambio de la señal, conviertiendo variaciones de frecuencia en variaciones de amplitud. Para que esto funcione bien la señal no debería tener variaciones de amplitud, como puede suceder debido al ruido aditivo, en donde la señal puede tener picos o valores extremos. El limitador corrige dichas variaciones de amplitud.

Esto en términos matemáticos se lo puede definir de la siguiente manera. Sea la señal recibida:

$$s(t) = Ae^{\left[j\left(2\pi\left(F_c \pm \Delta_F\right)t+\phi\right)\right]} + n_c(t) \tag{3.9}$$

Donde $n_c(t)$ es el ruido complejo gaussiano sumado en el canal de transmisión. Con lo cual, esta señal también puede ser vista como:

$$s(t) = A_n e^{\left[j\left(2\pi\left(F_c \pm \Delta_F\right)t + \phi + \phi_c(t)\right)\right]} \tag{3.10}$$

En donde se denota el efecto del ruido complejo sobre la amplitud y la fase de la señal. Ahora bien, dado el efecto en la amplitud, limitaremos la amplitud de la señal a 1, calculando el ángulo de la señal compleja e insertandoló en una exponencial compleja de amplitud A = 1 queda:

$$s_{limitada}(t) = e^{\left[j\left(2\pi\left(F_c \pm \Delta_F\right)t + \phi + \phi_c(t)\right)\right]} \tag{3.11}$$

En la misma se puede ver que el efecto del ruido aditivo afecta a la fase de la señal. Posteriormente, la señal pasa a través de un diferenciador para convertir variaciones de frecuencia en variaciones de amplitud, con lo cual, su derivada con respecto al tiempo t es:

$$\frac{d}{dt}s(t) = j\left[2\pi\left(F_c \pm \Delta_F\right) + \frac{d}{dt}\phi_c(t)\right]e^{\left[j\left(2\pi\left(F_c \pm \Delta_F\right)t + \phi + \phi_c(t)\right)\right]} \tag{3.12}$$

Por su parte, el detector de envolvente es necesario para separar la portadora de alta frecuencia de los datos digitales de baja frecuencia modulados en amplitud, con lo cual la señal finalmente tendrá la forma de:

$$s_{envolvente}(t) = \left|\frac{d}{dt}s(t)\right| = \left|2\pi\left(F_c \pm \Delta_F\right) + \frac{d}{dt}\phi_c(t)\right| \tag{3.13}$$

Por lo que a partir de aquí tomamos la decisión dura de si se trata de un bit igual a uno por encima de un umbral óptimo o igual a cero por debajo. En nuestro caso utilizamos el promedio de la señal comparado con el promedio de la señal en el tiempo de bit de interés para decidir. Por lo que si la envolvente es mayor que el umbral óptimo se decide por un bit igual a **uno** y, en caso contrario, se decide por un bit igual a **cero**.

Un análisis que subyace a partir de lo explicado en esta sección es que el Discriminador de Frecuencias es más robusto frente a errores en las frecuencia nominales F_1 y F_2 en la señal recibida, en comparación al demodulador mediante correladores, dado que el discriminador se basa en detectar variaciones instantáneas de la frecuencia de la señal.

3.2.4. Demodulación mediante un PLL

Uno de los circuitos más comunes que se usa para demodular señales FSK binarias es el lazo de seguimiento de fase (PLL). Un demodulador PLL-FSK funciona en forma

parecida a un demodulador PLL-FM. Cuando la entrada al PLL se desplaza entre las frecuencias de marca y de espacio, la tensión de error en la salida del comparador de fases sigue el corrimiento de frecuencias. Entonces, en un caso ideal sin ruido, como sólo hay dos frecuencias de entrada (F_1 y F_2) también sólo hay dos tensiones de error de salida. Uno representa un 1 lógico, y el otro a un 0 lógico, así, la salida es una representación en dos niveles (binaria) de la entrada FSK. En general, la frecuencia natural del Oscilador Controlado Por Tensión (VCO, por sus siglas en inglés) del PLL se iguala a la frecuencia central del modulador FSK. En consecuencia, los cambios en la tensión de error siguen a los cambios de la frecuencia analógica de entrada, y son simétricos respecto a 0 V, en el caso de un PLL analógico. En la Figura 3.7 se muestra un demodulador PLL-FSK básico al que le ingresa una señal modulada en FSK. Posteriormente un comparador de fases (también denominado detector de fases) mide la diferencia de fases entre esta señal y la señal que se encuentra a la salida del VCO. Esta diferencia de fase es filtrada por el denominado Filtro de Lazo y a la salida del mismo se obtiene una señal cuya envolvente sirve para, mediante un Filtro Adaptado, tomar la decisión dura sobre cuales son los bits demodulados [8].

En las próximas secciones se detallará sobre el proceso de diseño de un Lazo de Seguimiento de Fase para utilizarlo como demodulador PLL-FSK.

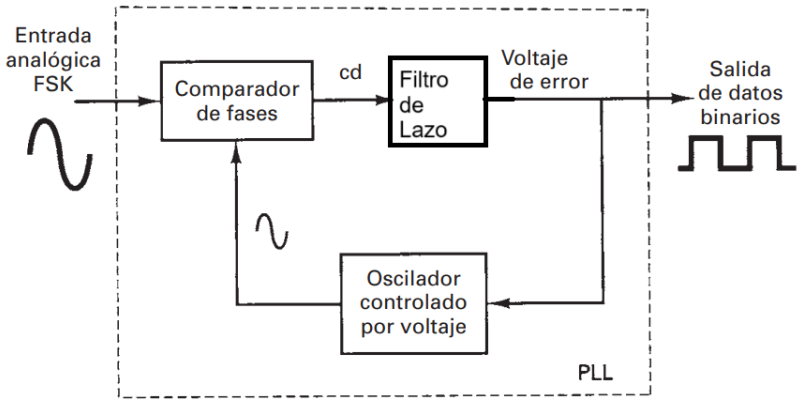

Figura 3.7: Demodulador PLL-FSK.

3.2.5. Análisis de la señal de entrada

Como se mencionó en la sección 2.2, las señales BFSK transmitidas de ATLAS poseen la frecuencia central de portadora en 434 MHz, ahora bien, para nuestras simulaciones aproximaremos la desviación de símbolo en 500 kHz para aprovechar las propiedades de ortogonalidad descriptas en la sección 3.2.2. Por su parte, la tasa de símbolos, que en este caso es igual a la tasa de bits dado que por el tipo de modulación utilizada un símbolo representa un bit, es igual a 1 M bit/s. Por lo tanto, la duración de bit es de $T_b = 10^{-6}$ s. Luego, el Ancho de Banda de la señal transmitida se puede

obtener a partir de la ecuación 3.6:

$$B = 2 \cdot 1\text{MHz} + 2 \cdot 500\text{kHz} = 3\text{MHz}$$

El contenido de los paquetes enviados está conformado por un código pseudoaleatorio (*PN code*) y cada etiqueta transmite una secuencia conocida diferente, sin palabra de sincronización, para poder detectarla e identificarla.

A esta señal modulada, le realizamos un muestreo pasabanda [10]. Por lo que ya en un primer momento podemos determinar que existirá una conversión descendente (*down-conversion*) de la señal pasabanda original desde la zona de Nyquist que contiene la señal modulada BFSK hacia la primera de zona de Nyquist como se muestra en la Figura 3.8. En este caso particular, el espectro de la señal de interés se encuentra en la sexta zona de Nyquist y, debido al muestreo pasabanda de la misma, la señal se replica en la primera zona de Nyquist. Con lo cual, el espectro de la señal luego de realizar el muestreo pasabanda se encontrará centrada en una frecuencia intermedia (FI) a partir de la cual podemos demodular.

Sin embargo, debemos tener cuidado cuando la zona de Nyquist en la cual se encuentra el espectro de nuestra señal de interés es una zona de Nyquist de índice par, ya que, el espectro replicado en la primera zona de Nyquist será una versión invertida respecto del espectro original [11], como se puede observar en la Figura 3.9.

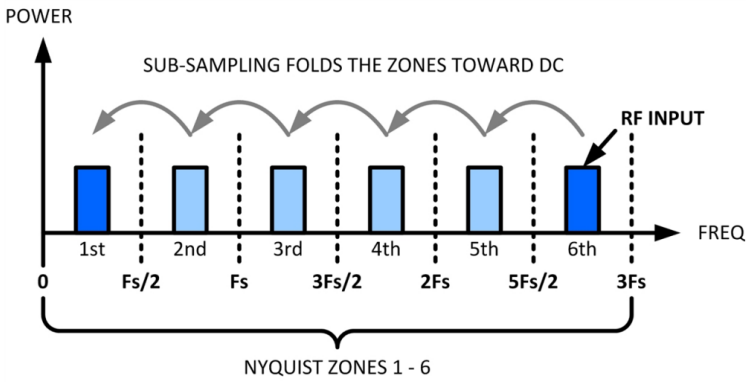

Figura 3.8: Los componentes de frecuencia de orden superior se pliegan en las zonas de Nyquist inferiores al realizar el submuestreo.

A modo de ejemplo, si establecemos que la frecuencia de muestreo de la señal es $f_s = 8$ MHz, se genera una conversión descendente del espectro situado entre los 432 MHz y los 436 MHz hacia la primera zona de Nyquist que es la banda situada entre banda base y 4 MHz, dada la conversión descendente del ancho de banda de la señal original comprendida desde los 432.5 MHz y los 435.5 MHz. El cálculo se realizó de la siguiente manera:

$$\frac{fc}{B} = \frac{fc}{\frac{fs}{2}} = \frac{434 \cdot 10^6}{\frac{8 \cdot 10^6}{2}} = 108{,}5$$

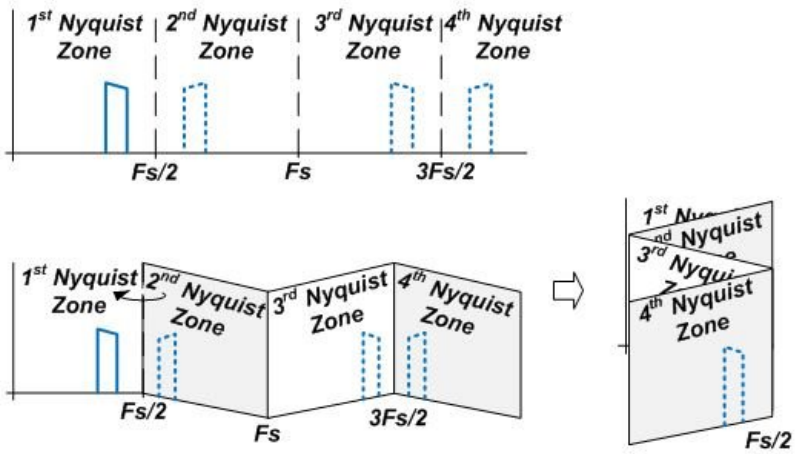

Figura 3.9: Zonas de Nyquist plegables hacia atrás.

Por lo que:

$$4 \text{ MHz} \cdot 108 = 432 \text{ MHz}$$

$$4 \text{ MHz} \cdot 109 = 436 \text{ MHz}$$

Con lo cual, la zona de Nyquist será la zona número 109 que es una zona impar, por lo que no se invierte el espectro de la misma en su replicado en la primera zona de Nyquist, situándolo hacia una frecuencia intermedia (FI) igual a 2 MHz . Esto se puede corroborar con las simulaciones que se observan en la Figura 3.10 y en la Figura 3.11 que representan los espectros de entrada para 10.000 realizaciones por cada frecuencia de muestreo, a un E_b/N_0 constante igual a 10 dB, de 1.000 bits aleatorios modulados por cada realización y, ruido de fase aleatorio adicional en el rango comprendido entre 0 y 2π. Siendo las frecuencias de muestreo $f_s = 4$ MHz, 5 MHz, 6 MHz, 7 MHz, 8 MHz, 21 MHz, 30 MHz y, 60 MHz.

En consecuencia, se pasa desde el espectro de la señal BFSK modulada cuya transmisión en el canal AWGN se encuentra centrada en la frecuencia de portadora $f_c = 434$ MHz como se puede observar en la Figura 3.12, el cual está conformado por 10.000 realizaciones de 1.000 bits aleatorios por cada realización y representado mediante un muestreo pasabajo, a los espectros con las frecuencias intermedias FI observados en las Figuras 3.10 y 3.11. La frecuencia de muestreo utilizada en cada caso determina el rango de visualización del espectro en el gráfico, abarcando desde $-f_s/2$ hasta $f_s/2$, es decir, la primera zona de Nyquist.

En el presente análisis, se muestran los espectros de señales FSK moduladas y el ruido correspondiente para distintas frecuencias de muestreo, utilizando simulaciones promedio sobre varias realizaciones. Cada figura contiene dos curvas: una que representa el espectro de la señal FSK modulada y otra que corresponde al espectro del ruido, en color azul y naranja, respectivamente. El espectro del ruido se observa plano en todas

las figuras, lo cual es característico del ruido blanco aditivo gaussiano, ya que, este tipo de ruido tiene una distribución uniforme de energía a lo largo de todas las frecuencias, lo que se refleja en su espectro constante en todo el rango mostrado.

Por último, se puede observar que para el caso en el que la frecuencia de muestreo es igual a 7 MHz, se produce una conversión descendente a 0 Hz (*Zero-IF*).

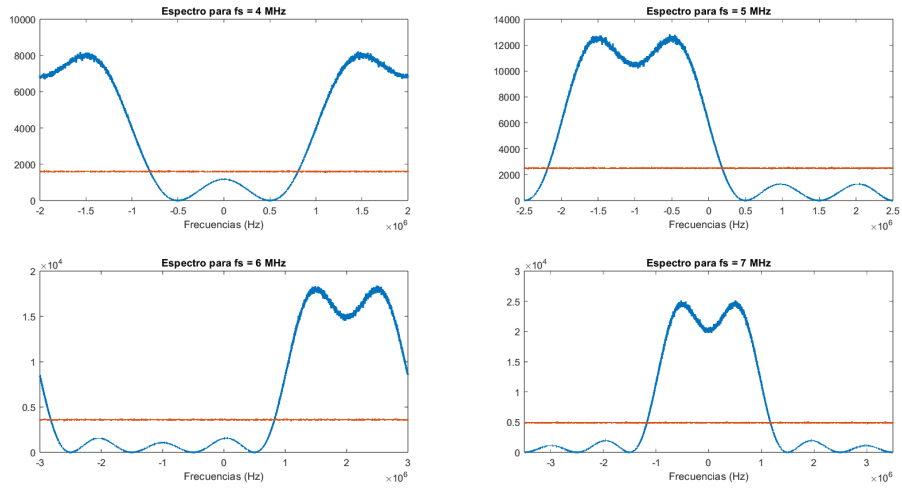

Figura 3.10: Réplicas del espectro de la señal en la primera zona de Nyquist después del proceso de muestreo pasabanda para las frecuencias de muestreo $f_s = 4$ MHz, 5 MHz, 6 MHz y, 7 MHz.

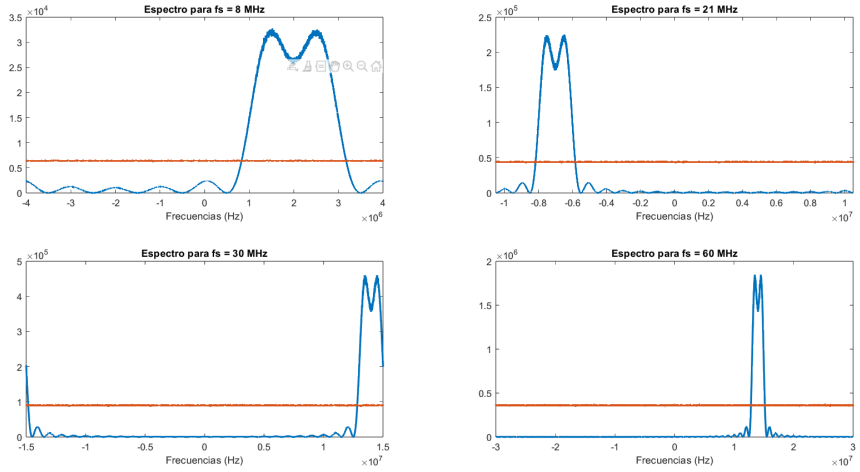

Figura 3.11: Réplicas del espectro de la señal en la primera zona de Nyquist después del proceso de muestreo pasabanda para las frecuencias de muestreo $f_s = 8$ MHz, 21 MHz, 30 MHz y, 60 MHz.

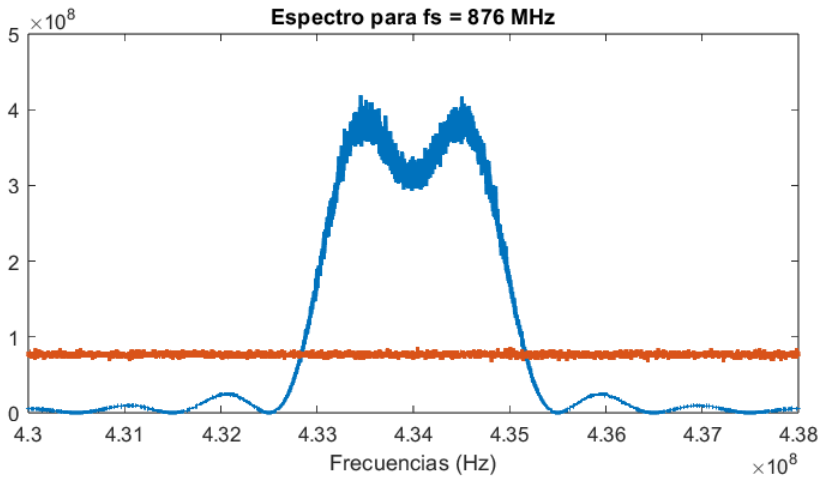

Figura 3.12: Espectro de la señal transmitida muestreada con $f_s = 876$ MHz.

Capítulo 4

Lazo de Seguimiento de Fase (*Phase-Locked Loop*)

4.1. PLL de tiempo continuo

Se puede considerar al Lazo de Seguimiento de Fase como un dispositivo que rastrea la fase y la frecuencia de una sinusoide. Los PLL se utilizan en sistemas de comunicación digital para sincronizar los osciladores locales en el receptor con los osciladores utilizados por el transmisor (sincronización de fase de portadora) o para sincronizar el reloj de datos en el receptor con el reloj de datos utilizado en la fuente de datos (sincronización de temporización de símbolos). En la Figura 4.1 se muestra el esquema básico de un PLL.

El mismo posee tres componentes básicos:

- Detector de fase.
- Filtro de lazo.
- Oscilador controlado por tensión (VCO).

El detector de fase es un dispositivo cuya salida es una función $g(\cdot)$ de la diferencia de fase entre las dos entradas. Debido a que la entrada del lazo y la salida del VCO forman las entradas al detector de fase, la salida del detector de fase es $g(\theta(t) - \hat{\theta}(t))$. La diferencia $\theta(t) - \hat{\theta}(t)$ se llama error de fase y se denota $\theta_e(t)$. El error de fase es filtrado por el filtro de lazo para producir una tensión de control $v(t)$ que se utiliza para establecer la fase del VCO. La salida del VCO $y(t) = \cos(\omega_0 t + \hat{\theta}(t))$ está relacionada con la entrada $v(t)$ a través de la relación de fase:

$$\hat{\theta}(t) = k_0 \int_{-\infty}^{t} v(x)dx \tag{4.1}$$

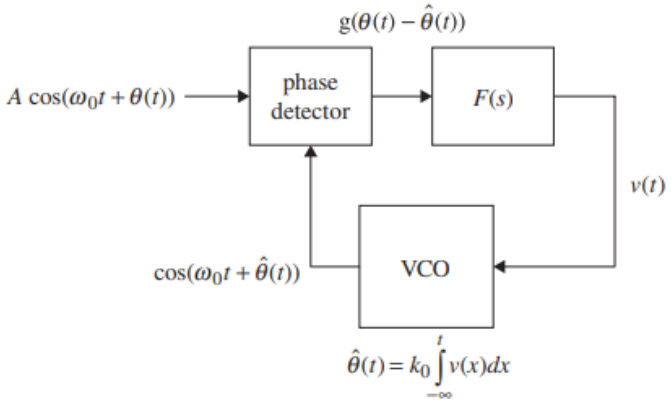

Figura 4.1: Estructura básica de un PLL.

donde k_0 es una constante de proporcionalidad, llamada ganancia del VCO, que tiene unidades de radianes/voltio. El lazo ajusta la tensión de control $v(t)$ para producir una estimación de fase $\hat{\theta}(t)$ que lleva el error de fase a cero.

Ahora bien, para lograr un error de fase cero en estado estacionario para un desplazamiento de fase constante, el filtro de lazo debe tener una ganancia proporcional distinta de cero. Así mismo, para lograr un error de fase cero en estado estacionario para un desplazamiento de frecuencia constante, el filtro de lazo debe tener una ganancia proporcional infinita.

Un filtro de lazo que satisface estas condiciones es el *Filtro de Lazo Proporcional más Integrador*, cuya abreviatura es "Filtro PI" [12]. Este tipo de filtro es el que se utilizará en este trabajo. En la Figura 4.2 se ilustra un PLL de tiempo continuo de segundo orden con un Filtro PI.

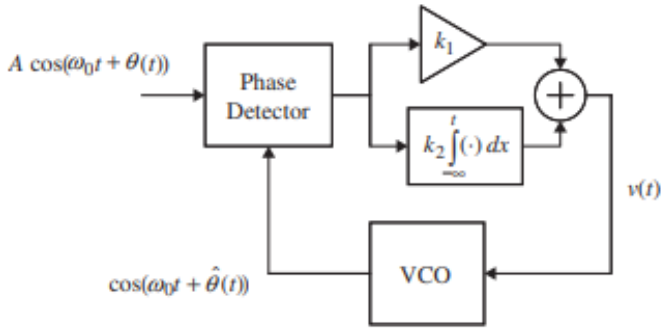

Figura 4.2: PLL de tiempo continuo de segundo orden con filtro PI.

4.2. PLL de tiempo discreto

Los PLL de tiempo discreto se utilizan en sistemas de datos muestreados. Su estructura consiste en un detector de fase de tiempo discreto, un filtro de lazo de tiempo

discreto y un sintetizador digital directo (DDS) que están dispuestos en un lazo de retroalimentación de la misma manera que lo estaban el detector de fase, el filtro de lazo y el VCO para el PLL de tiempo continuo. Las muestras de una sinusoide con frecuencia Ω_0 rad/muestra y de fase $\theta(nT)$ forman la entrada al PLL. El detector de fase de tiempo discreto calcula una función de la diferencia de fase entre la entrada y la salida del DDS. Esta diferencia de fase es el error de fase de tiempo discreto. El error de fase se filtra mediante el filtro de lazo y se ingresa al DDS. El DDS es una versión de tiempo discreto del VCO. En la Figura 4.3 se muestra un PLL básico de tiempo discreto.

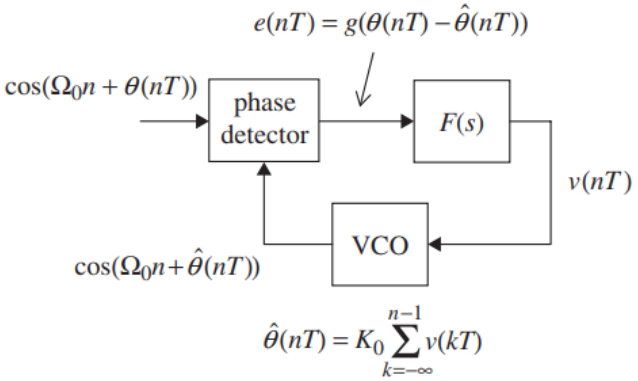

Figura 4.3: Estructura básica de un PLL de tiempo discreto.

En las siguientes secciones se realizará la implementación de un PLL de tiempo discreto con señales de entrada y salida complejas, dado que es el que se utilizará como demodulador PLL-FSK en este trabajo.

4.3. Diseño de un PLL

El comportamiento de un PLL está determinado por dos parámetros:

- Factor de Amortiguamiento (*Damping Factor*).
- Ancho de Banda de Ruido de Lazo (*Loop Noise Bandwidth*).

Ancho de Banda del PLL

Otra forma de caracterizar el PLL es calcular el ancho de banda de la respuesta en frecuencia del PLL. El ancho de banda del lazo es función tanto del Factor de Amortiguamiento como de la frecuencia natural (ω_n) del PLL. Mas adelante se discutirá que el PLL actúa como un filtro pasa-bajos al momento de *trackear* una señal y, que la frecuencia natural del PLL puede considerarse una medida aproximada del ancho de banda del Lazo, con lo cual, el ancho de banda de 3 dB denotado como ω_{3dB} se

obtiene estableciendo $|H(j\omega)|^2 = 1/2$ y resolviendo para ω. No obstante, si bien el ancho de banda de 3 dB es un concepto familiar, no es una medida muy útil de ancho de banda para un PLL. En consecuencia, se usa una medida más útil denominada Ancho de Banda de Ruido Equivalente (B_n). El ancho de banda de ruido equivalente de un sistema lineal con función de transferencia $H(j\omega)$ es el ancho de banda (medido en Hertz) de un Filtro Pasa-Bajos rectangular ficticio con la misma área que $|H(j\omega)|^2$. Esto se lo puede observar en la Figura 4.4 en la cual el mencionado filtro pasa-bajos ideal se representa con línea punteada. Cuando se diseña un PLL que posee un Filtro

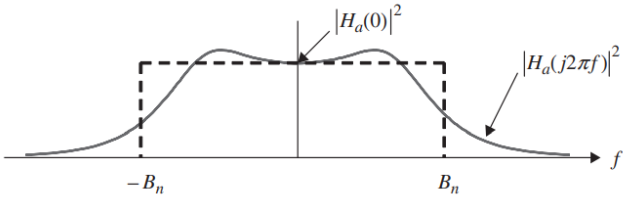

Figura 4.4: Diagrama conceptual para el cálculo del ancho de banda del ruido.

de Lazo del tipo "Proporcional más Integrador" (PI), el ancho de banda de ruido de lazo o ancho de banda de ruido equivalente B_n se define como [12]:

$$B_n = \frac{\omega_n}{2}\left(\zeta + \frac{1}{4\zeta}\right) \tag{4.2}$$

Cuyo valor aumenta a medida que aumenta el Factor de Amortiguamiento utilizado en el diseño.

Tiempo de adquisición del PLL

Cualquier PLL requiere de un período de tiempo distinto de cero para reducir un error de fase a cero, es decir para que la señal de salida del PLL logre *trackear* a la señal de entrada. Durante las etapas iniciales de adquisición, la tensión de entrada al VCO en el caso analógico o al DDS para el caso digital se ajusta para producir una salida cuya frecuencia coincida con la frecuencia de la señal de entrada. Esta fase inicial del proceso de adquisición se denomina enganche de frecuencia. Una vez que se logra el enganche de frecuencia, se requiere un período de tiempo adicional para reducir el error de fase del lazo a un nivel bajo. Este lapso del proceso de adquisición se denomina enganche de fase. En consecuencia, el tiempo de adquisición denotado como T_{LOCK} se puede aproximar mediante la suma del tiempo para lograr el enganche en frecuencia T_{FL} y el tiempo para lograr el enganche de fase T_{PL}. Con lo cual:

$$T_{LOCK} \approx T_{FL} + T_{PL} \tag{4.3}$$

En donde a T_{FL} y T_{PL} se los puede aproximar como:

$$T_{FL} \approx \frac{4 \cdot (\Delta_F)^2}{B_n^3} \tag{4.4}$$

$$T_{PL} \approx \frac{1{,}3}{B_n} \tag{4.5}$$

Por lo que, como se puede observar en las ecuaciones, el tiempo de enganche será mayor cuanto mayor sea la desviación en frecuencia de la señal de entrada respecto de la frecuencia de la señal del VCO o el DDS (equivalente en nuestro caso a un Oscilador Controlado Numéricamente: NCO, por sus siglas en inglés) y, además será mayor cuanto menor sea el ancho de banda de ruido equivalente.

Relación de compromiso en la elección del ancho de banda

Existe una relación de compromiso (*trade-off*) en la correcta elección del Ancho de Banda del PLL. Sabemos que elegir un pequeño ancho de banda de ruido filtrará la mayor parte del ruido de la señal de interés y por extensión las frecuencias que caen en la banda de supresión. Por otro lado, elegir un gran ancho de banda de ruido puede *trackear* variaciones de fase rápidas, es decir, frecuencias más altas, pero a su vez por extensión permitirá que entre más ruido a través del lazo. Sin embargo, cuanto mayor sea el ancho de banda, más rápida será la adquisición y, a su vez, cuanto menor sea el ancho de banda menor será el error de seguimiento. Por lo tanto, la adquisición rápida y el buen seguimiento plantean demandas contrapuestas en el diseño de PLL. El tiempo de adquisición se puede reducir a expensas de un mayor error de seguimiento. El error de seguimiento se puede reducir a expensas de un mayor tiempo de adquisición. Un buen diseño equilibra los dos criterios de rendimiento, dependiendo de la aplicación. En nuestro caso particular en donde estamos implementando un demodulador PLL-FSK, nos interesa tener un buen nivel de relación señal a ruido (SNR, por sus siglas en inglés) por lo que debemos ceder en el tiempo de enganche del PLL. En este trabajo por cuestiones de diseño predefinimos el ancho de banda de ruido equivalente del PLL y luego, a partir de la ecuación 4.2, obtenemos la frecuencia natural del PLL que es la que nos determina hasta que frecuencia se puede considerar plana la respuesta en frecuencia (*flat top*). Despejando de la ecuación del Ancho de Banda del PLL, tenemos que:

$$\omega_n = \frac{2B_n}{\zeta + \frac{1}{4\zeta}} \tag{4.6}$$

Por lo que la frecuencia natural del PLL será:

$$f_n = \frac{\omega_n}{2\pi} \tag{4.7}$$

Factor de Amortiguamiento (*Damping Factor*)

El proceso de adquisición de fase en un PLL exhibe un comportamiento oscilatorio al principio que puede ser controlado por el denominado Factor de Amortiguamiento. Para una señal de entrada dada, un PLL se comporta de manera diferente para diferentes valores de Factor de Amortiguamiento ζ.

En primer lugar, cuando el Factor de Amortiguamiento $\zeta < 1$, los polos de este lazo de segundo orden son pares conjugados complejos y la respuesta del bucle presenta oscilaciones amortiguadas, la respuesta del lazo exhibe oscilaciones amortiguadas en forma de sobreimpulsos y subimpulsos. Un sistema de segundo orden con transitorios sinusoidales amortiguados se denomina subamortiguado. En segundo lugar, cuando $\zeta > 1$, los polos son reales y distintos y la respuesta del lazo es la suma de exponenciales decrecientes con lo que el comportamiento oscilatorio desaparece con grandes ζ. Un sistema de este tipo se denomina sobreamortiguado. Por último, cuando $\zeta = 1$, los polos son reales y repetidos y, la respuesta está en algún lugar entre oscilaciones amortiguadas y exponenciales decrecientes y el PLL se denomina críticamente amortiguado. En la Figura 4.5 se puede observar este comportamiento en la respuesta al escalón en el tiempo para un PLL con un filtro PI.

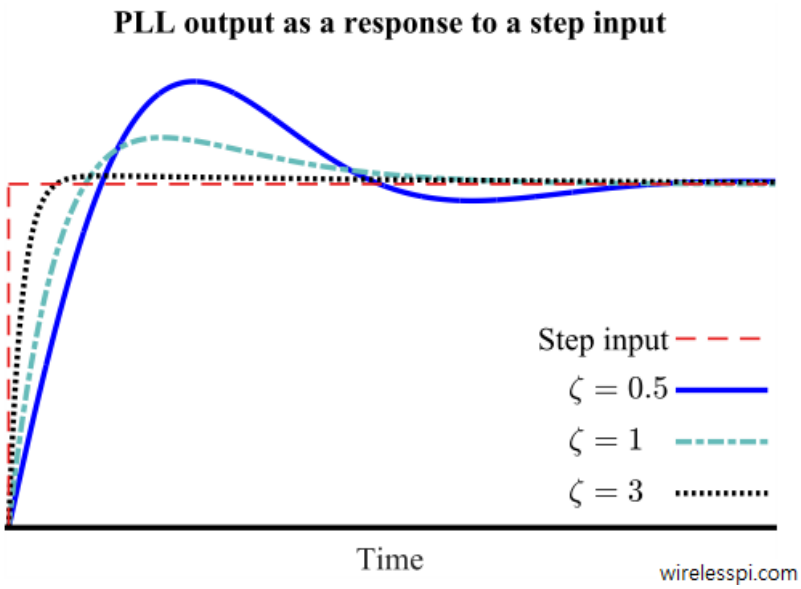

Figura 4.5: Respuesta Temporal al Escalón para un PLL con filtro PI.

En esta misma figura se ve reflejado que un gran Factor de Amortiguamiento no produce sobreimpulsos y genera un tiempo de convergencia corto desde el estado transitorio hacia el estado estacionario, como sucede en el caso del $\zeta = 3$, mientras que uno pequeño exhibe una convergencia relativamente larga y con oscilaciones amortiguadas. Valores típicos de diseño se encuentran en el intervalo de 0,5 a 2 en aplicaciones prácticas. Por los motivos mencionados en nuestro demodulador PLL-FSK utilizamos un Factor de Amortiguamiento igual a 2 (dos). Por último, cabe destacar que se logra un

buen equilibrio entre los dos casos con un Factor de Amortiguamiento de $\frac{1}{\sqrt{2}} = 0,707$ y, en general, es un valor de uso frecuente.

Filtro de Lazo: Proporcional + Integrador

Como sugiere el nombre, un filtro PI tiene un componente proporcional y un componente integrador. El término proporcional es una simple ganancia de K_1 a la salida del filtro ya que aporta una señal que es proporcional a la entrada del filtro que, en nuestro caso, es el error de fase. Por lo que la componente proporcional del error de fase filtrado será:

$$e_1[n] = K_1\theta_e[n] \tag{4.8}$$

El término integrador es un integrador ideal con una ganancia de K_2 a la salida del filtro ya que aporta una señal que es proporcional a la integral de la señal de entrada. Con lo cual la componente integradora del error de fase filtrado será:

$$e_2[n] = e_2[n-1] + K_2\theta_e[n] \tag{4.9}$$

Como se mencionó anteriormente, esta componente es necesaria para llevar el error de estado estacionario en la salida del PLL a cero, en presencia de un error de frecuencia. Posteriomente, se produce la combinación de las componentes proporcional e integradora, lo que conduce a la salida del filtro de lazo:

$$v[n] = e_1[n] + e_2[n] \tag{4.10}$$

En la Figura 4.6 se observa como se incorpora un filtro PI en el modelo PLL lineal, resultando el diagrama de bloques de tiempo discreto expuesto allí, en donde cabe aclarar que la notación $D1$ representa un retraso de un tiempo de muestra (equivalente a Z^{-1}).

Sintetizador digital directo (DDS)

Un DDS genera una señal sinusoidal de tiempo discreto y de valores discretos con una fase lo más cercana posible a la fase de la señal de referencia. Imaginemos dos escenarios para nuestra comprensión, supongamos que $\theta[n] = \theta(nT)$ es cero y, por lo tanto, la frecuencia instantánea de la señal de entrada es $2\pi\frac{fc}{fs}$, siendo la frecuencia natural del DDS igual a la frecuencia de la señal de entrada. Como consecuencia, el DDS también opera a la misma frecuencia y el error de fase $\theta_e[n]$ es cero, entonces la salida del detector de error de fase idealmente debe ser cero, lo que a su vez, conduce a una salida de filtro de lazo de cero.

Sin embargo, si $\theta_e[n]$ no fuese cero al principio, el detector de error de fase desa-

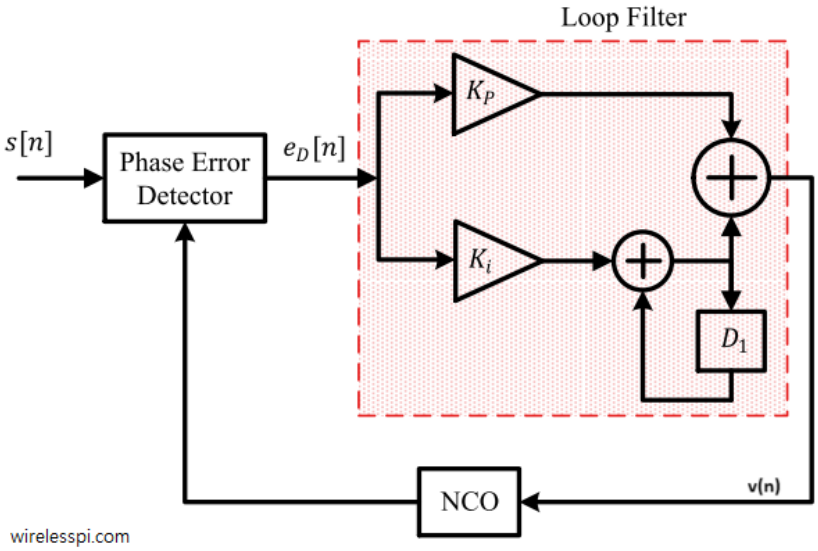

Figura 4.6: PLL digital con filtro PI.

rrollaría una señal de salida distinta de cero, la cual tendría una pendiente positiva o negativa (dependiendo de $\theta_e[n]$), el filtro de lazo posteriormente generaría una señal finita y, esto haría que el DDS cambiara su fase de tal manera que haría que $\theta_e[n]$ sea cero de nuevo. El DDS ajusta su fase de salida basado en su señal de entrada como:

$$\hat{\theta}(nT) = K_0 \sum_{k=-\infty}^{n-1} v(kT) \tag{4.11}$$

donde K_0 es una constante de proporcionalidad conocida como ganancia del oscilador. A partir de esta expresión, podemos ver que un DDS actúa como un acumulador de fase ya que ejecuta una integración diferencial hacia atrás para acumular su entrada. Y como se mencionó anteriormente, a diferencia del VCO, la ganancia K_0 del acumulador de fase se puede configurar fácilmente a un valor fijo, por ejemplo 1.

Por último, podemos reescribir la salida del DDS como:

$$\hat{\theta}[n] = K_0 \sum_{i=-\infty}^{n-1} v[i] = K_0 \sum_{i=-\infty}^{n-2} v[i] + K_0 \cdot v[n-1] = \hat{\theta}[n-1] + K_0 \cdot v[n-1]$$

Selección de Constantes de Lazo:

Una vez determinado el Ancho de Banda y el Factor de Amortiguamiento del filtro de lazo PI, hay cuatro constantes que se deben determinar: K_0 , K_D , K_1 y K_2. En un sistema de tiempo discreto, la ganancia del sintetizador digital directo (DDS) K_0 se puede fijar fácilmente a un valor determinado, por lo que en este caso lo fijamos en 1. De igual manera, la ganancia del detector de error de fase K_D en analógico se calcula de acuerdo con la estructura y la expresión resultante del detector de error de fase.

Pero en tiempo discreto puede tratarse como un parámetro fijo alrededor del cual se diseña el resto del lazo, en este caso también configuramos K_D como igual a 1.

Con K_0 y K_D establecidos, los coeficientes del filtro PI: ganancia proporcional K_1 y ganancia integradora K_2, son las dos incógnitas que necesitamos para diseñar nuestro demodulador PLL-FSK. Suponiendo que el ancho de banda del ruido de lazo es pequeño en comparación con la frecuencia de muestreo, la teoría de sistemas de control establece las siguientes relaciones entre las constantes de lazo y los parámetros de lazo [12]:

$$K_1 = \frac{4\zeta\theta_n}{(K_0K_D)\cdot(1+2\zeta\theta_n+\theta_n^2)} \tag{4.12}$$

$$K_2 = \frac{4\theta_n^2}{(K_0K_D)\cdot(1+2\zeta\theta_n+\theta_n^2)} \tag{4.13}$$

Por lo que en nuestro caso:

$$K_1 = \frac{4\zeta\theta_n}{1+2\zeta\theta_n+\theta_n^2} \tag{4.14}$$

$$K_2 = \frac{4\theta_n^2}{1+2\zeta\theta_n+\theta_n^2} \tag{4.15}$$

Donde θ_n se refiere al Ancho de Banda Normalizado del PLL y, se lo define como:

$$\theta_n = \frac{B_nT_s}{\zeta+\frac{1}{4\zeta}} \tag{4.16}$$

Por último, la función de transferencia de un PLL discreto con Filtro de Lazo Proporcional más Integrador viene dada por la ecuación [12]:

$$H(z) = \frac{K_DK_0(K_1+K_2)z^{-1} - K_DK_0K_1z^{-2}}{1-2\left(1-\frac{1}{2}K_DK_0(K_1+K_2)\right)z^{-1}+(1-K_DK_0K_1)z^{-2}} \tag{4.17}$$

PLL como Filtro Pasabajos

El propósito de emplear un PLL en un receptor de comunicaciones es rastrear una forma de onda entrante en fase y frecuencia. Esta señal de entrada está corrupta por el ruido gaussiano aditivo. En una configuración de este tipo, un receptor enganchado con la fase de la señal recibida de entrada debe en la salida, aproximar esta señal original adecuadamente mientras elimina la mayor cantidad de ruido posible.

El Receptor utiliza un VCO en el caso analógico ó un DDS en el caso digital con una frecuencia cercana a la frecuencia de portadora esperada en la señal recibida. A través del filtro de lazo, el PLL promedia la salida del detector de error de fase durante un período de tiempo y sigue sintonizando su oscilador en función de este promedio. Si la señal de entrada tiene una frecuencia estable, este promedio a largo plazo produce

un seguimiento de fase muy preciso, eliminando así una cantidad significativa de ruido. En un escenario de este tipo, la entrada al PLL es una señal ruidosa mientras que la salida es una versión limpia de la entrada. Por lo tanto, podemos decir que cuando funciona como un sistema de seguimiento lineal, un Lazo de Seguimiento de Fase es un filtro que pasa la señal y rechaza el ruido.

En la Figura 4.7 se muestra la respuesta en frecuencia de un PLL con filtro de lazo PI para distintos Factores de Amortiguamiento y se demuestra que en efecto es un filtro pasa-bajos. La figura también revela que el espectro de este PLL como filtro pasa-bajos es aproximadamente plano entre cero y la frecuencia natural del PLL. Esto implica que el PLL debería poder rastrear variaciones de fase y frecuencia en la señal de referencia siempre que estas variaciones permanezcan la zona plana del filtro.

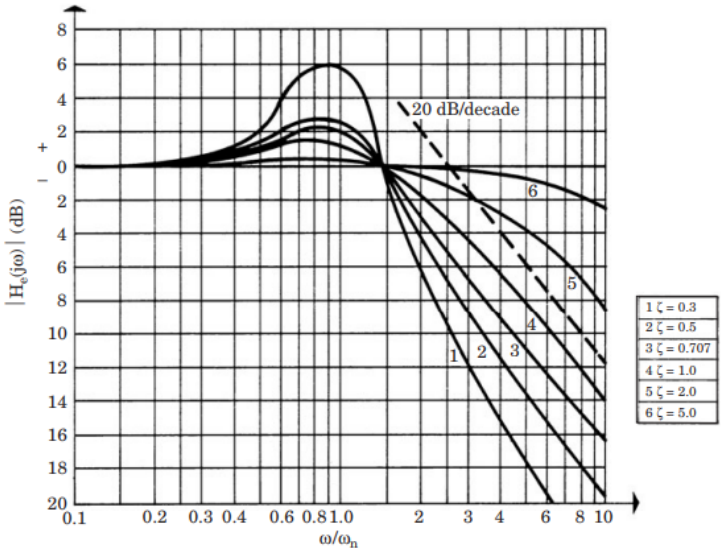

Figura 4.7: Respuesta de frecuencia del PLL de segundo orden con Factores de Amortiguamiento 0.3, 0.5, 0.707, 1.0, 2.0 y 5.0.

Parámetros Clave de un PLL

A continuación comentamos algunos parámetros clave que rigen su rendimiento. Comenzamos con los parámetros que especifican el rango de frecuencia dentro del cual se puede operar un PLL [13].

El **Rango Retención** (Hold Range) es el rango dentro del cual un PLL siempre quedará enganchado. Para el PLL de segundo orden este valor es teóricamente infinito.

El **Rango de Atracción** (Pull-in Range) se refiere a cuando el error de frecuencia de la señal de referencia en un estado desenganchado se reduce por debajo de un valor crítico, entonces, la acumulación del error de fase comienza a desacelerarse, lo que conduce a un enganche eventual del PLL. Para un PLL de segundo orden, esto es infinito.

El **Rango de Separación** (Pull-out Range) se define como el escalón de frecuencia (*frequency step*) máximo que se puede aplicar a un PLL enganchado en fase sin que pierda el enganche. Un bucle de segundo orden teóricamente siempre puede recuperar el enganche, pero si el escalón de frecuencia de la señal de entrada es mayor que el Pull-out Range puede haber varios saltos de ciclo antes de que el PLL recupere el enganche. Es decir el PLL pierde temporalmente la sincronización durante varios ciclos de señal, con lo cual se busca que la desviación de frecuencia sea menor a este valor. Este fenómeno es común cuando el ruido o las perturbaciones en la señal son demasiado fuertes o cuando el sistema PLL está operando al límite de su capacidad de seguimiento. Una aproximación del mismo viene dado por la ecuación:

$$\Delta\omega_{PO} = 1{,}8\omega_n(\zeta + 1) \tag{4.18}$$

Ahora bien, si el desfase de frecuencia se reduce por debajo de otro valor, denominado **Rango de Enganche** (Lock Range) el PLL se engancha dentro de una única nota de batido entre las frecuencias de referencia y de salida. Esto ocurre cuando la diferencia de frecuencia entre la señal de referencia y la señal de salida del PLL es lo suficientemente pequeña tal que se genera un batido, que es la diferencia entre estas dos frecuencias. Si la diferencia es suficientemente pequeña, el PLL es capaz de engancharse o sincronizarse con la señal de referencia. Normalmente, el rango de frecuencia de operación (*operating range*) de un PLL está restringido a este valor, que para un PLL de segundo orden viene dado por:

$$\Delta\omega_L = 2\zeta\omega_n \tag{4.19}$$

En la Figura 4.8 se puede observar como están limitados estos parámetros, siendo que $\Delta\omega_H > \Delta\omega_P > \Delta\omega_{PO} > \Delta\omega_L$

Por último, partir de estas frecuencias angulares, podemos obtener las frecuencias lineales en Hz como:

$$f_n = \frac{\omega_n}{2\pi} \tag{4.20}$$

$$\Delta f_{PO} = \frac{\Delta\omega_{PO}}{2\pi} \tag{4.21}$$

$$\Delta f_L = \frac{\Delta\omega_L}{2\pi} \tag{4.22}$$

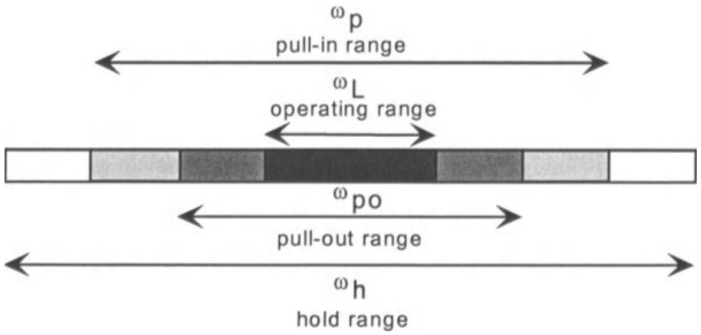

Figura 4.8: Parámetros Clave de un PLL.

4.4. Implementación de PLL de tiempo discreto de señales complejas

El siguiente Lazo de Seguimiento de Fase que se describirá a continuación está diseñado para rastrear la fase de una señal exponencial compleja. Fue elegido dado que proporciona un seguimiento preciso en sistemas FSK en IQ, con una respuesta robusta y controlada frente a variaciones de fase y frecuencia, que mejora la precisión y estabilidad en la recuperación de datos.

La señal de salida del DDS también es una exponencial compleja cuya fase es la estimación de la fase de entrada (asumiendo que las frecuencias discretas Ω_0 de ambas señales son idénticas). Por lo que, siendo la señal de entrada:

$$x(n) = Ae^{j(\Omega_0 n + \theta(n))} \tag{4.23}$$

Y la señal de salida del DDS:

$$y(n) = Ae^{j(\Omega_0 n + \hat{\theta}(n))} \tag{4.24}$$

El detector de fase calcula el error de fase calculando primero el producto de la entrada y el conjugado complejo de la salida DDS. Este producto viene dado por:

$$z(n) = x(n)y^*(n) = A^2 e^{j(\theta(n) - \hat{\theta}(n))} \tag{4.25}$$

La función $arg\cdot$ calcula la fase de la exponencial compleja usando un arcotangente de cuatro cuadrantes. Por lo que la salida del detector de fase será:

$$g(\theta_e) = \theta_e(n) = \theta(n) - \hat{\theta}(n) \tag{4.26}$$

Este detector de fase es lineal en el intervalo $-\pi \leq \theta_e \leq \pi$ y tiene una pendiente de uno. Por este motivo, $K_0 = 1$.

Por su parte, el DDS ajustará la fase de la señal de salida basado en la señal de Error Filtrado:

$$\hat{\theta}(nT) = K_0 \sum_{k=-\infty}^{n-1} v(kT) \tag{4.27}$$

que es el valor que ingresará al DDS. En la Figura 4.9 se muestra la implementación del PLL de señales complejas. Además, como se observa en la Figura 4.10 la curva-S del detector de fase de este PLL es lineal.

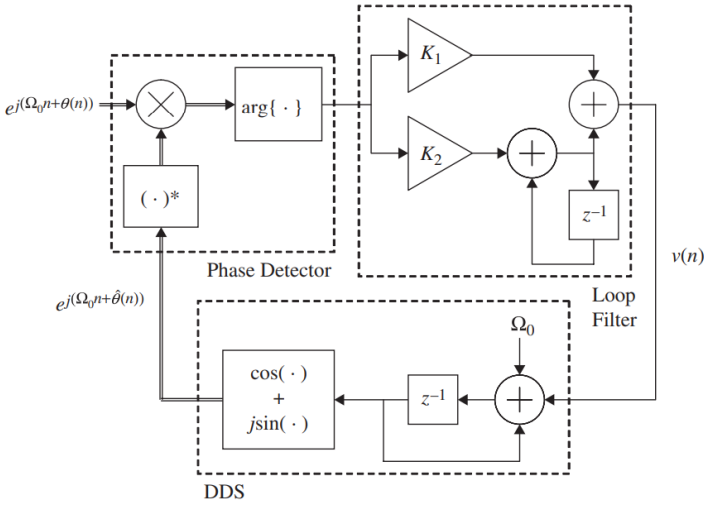

Figura 4.9: PLL de tiempo discreto de segundo orden con entradas y salidas complejas.

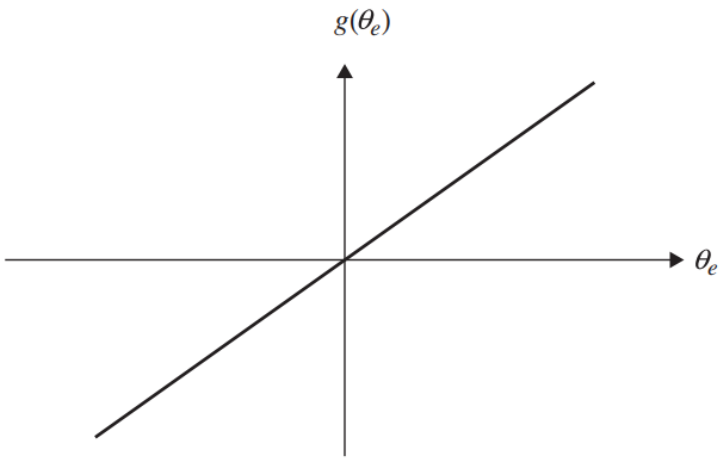

Figura 4.10: Curva-S del detector de fase utilizado.

Con lo cual, con un análisis de seguimiento de este PLL complejo si le ingresamos las sinusoides presentadas en esta sección, con un desfase relativo de π radianes, observaríamos resultados como en la Figura 4.11. Allí se puede observar como el Lazo de Seguimiento de Fase se engancha en fase y frecuencia a la señal de entrada en sus

partes reales, y como se alinea a ésta a medida que transcurre el tiempo, siendo la señal punteada, la señal de entrada y la solida la señal de salida del DDS. Además, en la Figura 4.12 se puede observar como el Error de Fase entre ambas señales comienza en π radianes, disminuye a $-0{,}5$ radianes y luego, a medida que transcurre el tiempo de muestra, se reduce a cero. En la Figura 4.13 podemos corroborar este hecho, dado que se logra observar como el argumento de la señal de salida del DDS (curva sólida) tiende al argumento lineal ($\theta_n + \pi u(n)$) de la señal de entrada (curva punteada), dado que, comienza en cero y finalmente se alinea con la entrada PLL (la diferencia entre estas dos curvas es el error de fase). Se puede observar que la línea sólida está debajo de la línea discontinua para las muestras 0 a 12, esto significa que el error de fase es positivo, como se ilustra en el gráfico de error de fase (Figura 4.12). Para las muestras 13 a 75, la línea sólida está por encima de la línea discontinua, lo que significa que el error de fase es negativo. Esto también lo confirma el gráfico de error de fase. El hecho de que la salida del DDS coincida con la entrada del PLL se confirma en los últimos tres gráficos mencionados (Figura 4.11, Figura 4.12, Figura 4.13).

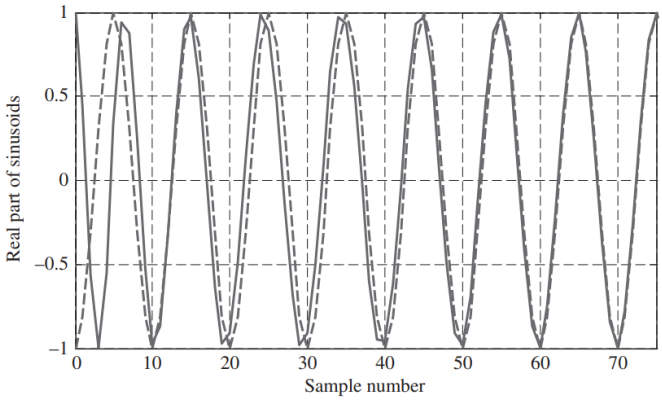

Figura 4.11: Parte real de la entrada exponencial al PLL (línea punteada) y la parte real de la salida DDS (línea sólida) en cada numero de muestra.

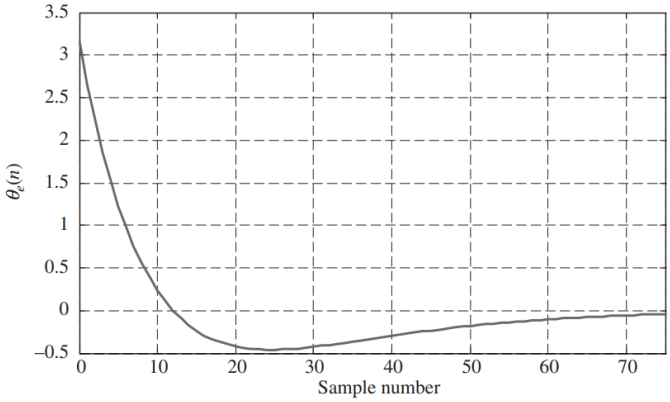

Figura 4.12: Error de fase θ_e en cada número de muestra.

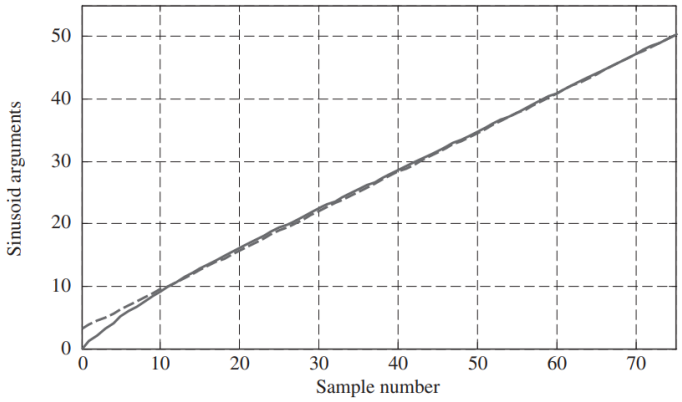

Figura 4.13: Argumento lineal de la exponencial compleja de entrada (línea punteada) y el argumento de la exponencial compleja de salida DDS (línea sólida) en cada numero de muestra.

4.4.1. Uso del PLL complejo como Demodulador PLL-FSK

Si analizamos a este PLL como Demodulador PLL-FSK, como ya sabemos la señal BFSK modulada y transmitida por el canal AWGN será de la forma:

$$x(n) = A_n e^{\left[j\left(\left(\frac{2\pi(F_c \pm \Delta_F)}{f_s}\right)n + \phi_1 + \phi_c(n)\right)\right]} \tag{4.28}$$

En donde la desviación máxima, Δ_F, en nuestro caso será de 500 kHz. Con lo cual la señal de salida del DDS, que *trackea* a esta señal de entrada será igual a:

$$y(n) = e^{\left[j\left(\left(\frac{2\pi(F_{dds})}{f_s}\right)n + \phi_2\right)\right]} \tag{4.29}$$

Aquí se puede notar que una de las ventajas que posee utilizar este tipo de demodulador, dado que la señal recibida contiene ruido aditivo tanto en amplitud como en fase, y este debe lidiar únicamente con el ruido introducido en la fase de la señal modulada y no con el ruido introducido en la amplitud de la misma, dado que solo trabaja con la fase (argumento) de la misma. Esto representa una ventaja frente al caso, por ejemplo, del Discriminador de Frecuencia que debe utilizar un limitador para lidiar con el ruido en amplitud de la señal recibida.

Posteriormente, esta señal discreta, que representa el error de fase en cada tiempo de muestra, se filtra en el filtro PI como se detalla en la 4.3, y luego el VCO ajusta su fase para seguir la frecuencia y la fase instantánea de la señal recibida. Ahora bien, como se mencionó anteriormente, es a partir del Error de Fase Filtrado $v(n)$ que se logra demodular las señales BFSK. En la Figura 4.14 se puede observar la representación de las doscientas muestras iniciales del Error de Fase y del Error de Fase Filtrado simulado en MATLAB, para un PLL complejo usado como demodulador PLL-FSK, utilizando una tasa de muestreo de 6 MHz. Y es que es a partir de esta señal de Error

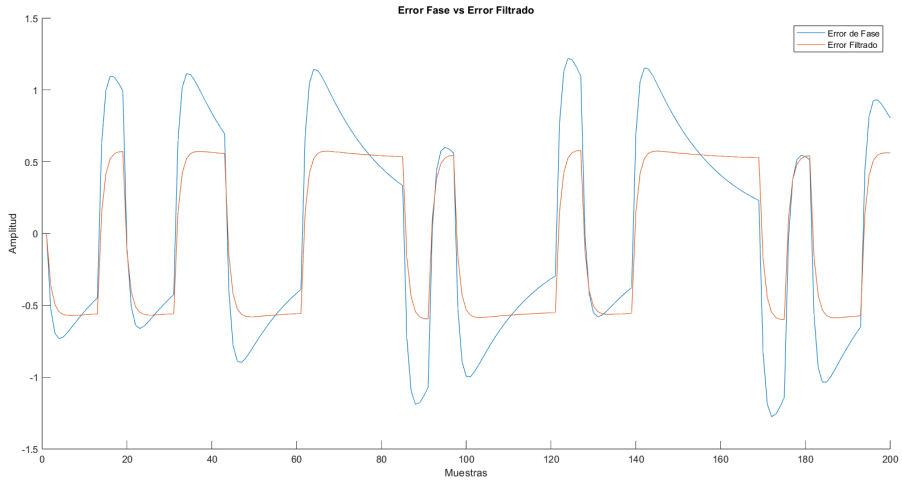

Figura 4.14: Error de Fase y Error de Fase Filtrado de un PLL de tiempo discreto de señales complejas que muestrea una señal BFSK a una tasa de 6 MHz, con un Ancho de Banda de Ruido Equivalente igual a 1.5 MHz y Factor de Amortiguamiento igual a 2.

de Fase Filtrado que se toma la decisión dura de cuales son los bits que contiene la señal recibida. Ésto se realiza basado en un umbral óptimo del filtro adaptado, que en el diseño del receptor realizado en este trabajo, será el promedio de la señal Error Filtrado sobre el total de la señal. En un receptor equivale a tomar el promedio de esta señal en una ventana de tiempo específica (diez mil bits por ejemplo, que es la cantidad de bits que se simularon en las imágenes mostradas en esta sección). Posteriormente, se calcula el promedio del error filtrado en cada tiempo de bit, se lo compara con este umbral óptimo y se determina si es un cero o un uno, respectivamente.

Adicionalmente, en la Figura 4.15 se puede observar un diagrama de ojos de como son las transiciones del Error Filtrado de este PLL en cada tiempo de bit, simulando diez mil bits en total.

Y por último en la Figura 4.16 se pueden observar las cien muestras iniciales entre la representación real de la señal de entrada y la de salida del DDS, bajo las mismas condiciones de simulación, donde se denota como la señal de salida sigue a la señal de entrada.

4.4.2. Diseño Final del PLL de tiempo discreto de señales complejas

Al diseñar nuestro PLL, obtuvimos que para las frecuencias de muestreo de 4 MHz, 5 MHz, 6 MHz, 7 MHz, 8 MHz, 21 MHz, 30 MHz y 60 MHz, con un Factor de Amortiguamiento de 2 y un Ancho de Banda de Ruido Equivalente de 1.5 MHz, se tienen los coeficientes proporcionales e integradores que se muestran en la Tabla 4.1. En dicha tabla, los valores de K_1 y K_2 representan las constantes proporcionales e integradoras, respectivamente, en orden ascendente de frecuencias de muestreo.

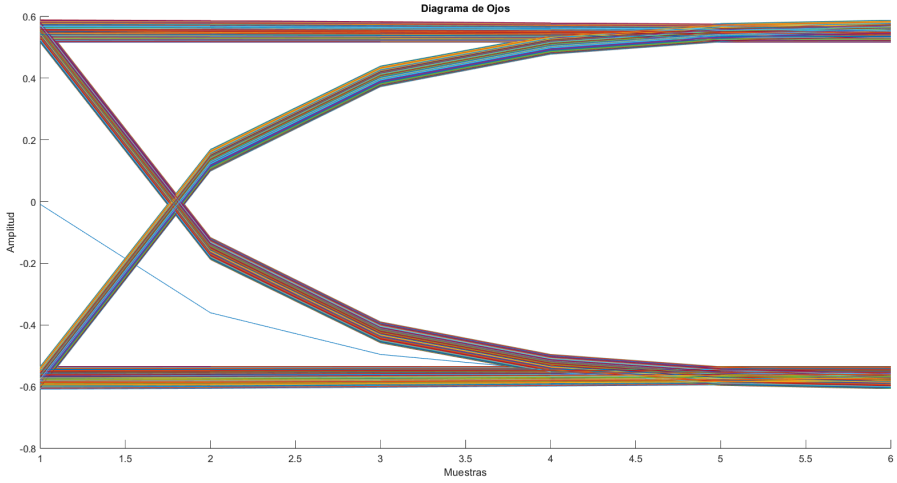

Figura 4.15: Diagrama de Ojo del Error de Fase Filtrado en cada tiempo de bit utilizando un PLL de tiempo discreto de señales complejas que muestrea una señal BFSK a una tasa de 6 MHz, con $B_n = 1{,}5MHz$ y $\zeta = 2$.

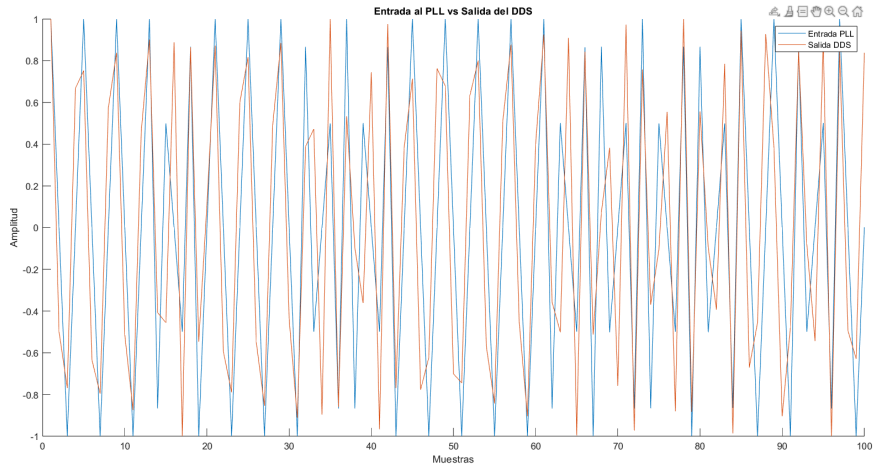

Figura 4.16: Parte real de la señal BFSK de entrada al PLL de señales exponenciales (curva azul) y la parte real de la salida del DDS (curva roja) muestreando a una tasa de 6 MHz, con $B_n = 1{,}5MHz$ y $\zeta = 2$.

Frecuencia de Muestreo (MHz)	K_1	K_2	Frecuencia Natural del DDS (MHz)
4	0.8127	0.0717	0
5	0.7127	0.0503	-1
6	0.6340	0.0373	2
7	0.5707	0.0288	0
8	0.5188	0.0229	2
21	0.2368	0.0040	-7.5
30	0.1720	0.0020	14
60	0.0899	0.0005	14

Tabla 4.1: Coeficientes K_1 y K_2 y Frecuencia Natural del DDS para diferentes frecuencias de muestreo.

Es importante señalar que, en la bibliografía consultada, no se encontró una relación explícita entre el ancho de banda de ruido equivalente seleccionado y la frecuencia de muestreo del lazo de seguimiento. No obstante, en la práctica se observó iterativamente que el lazo de seguimiento de fase (PLL) discreto para señales complejas presenta el mejor rendimiento cuando el ancho de banda de ruido equivalente se sitúa entre 1.5 MHz y 2 MHz, siendo 1.5 MHz el valor óptimo.

Por su parte, para todas las respuestas en frecuencia se tiene la misma frecuencia natural de los PLL, igual a $f_n = 225$ kHz, dado que es un parámetro que depende tanto del Factor de Amortiguamiento como del Ancho de Banda de Ruido de Lazo, y ambos son constantes para toda frecuencia de muestreo. Es hasta este valor f_n que el PLL posee una respuesta plana (*flat top*), lo cual se puede observar en la Figura 4.18 donde se muestran las respuestas de los distintos filtros con la frecuencia normalizada a la frecuencia natural del PLL.

Luego, a partir de estos coeficientes obtenemos las Respuestas en Frecuencia del Lazo de Seguimiento de Fase que se observan en la Figura 4.17.

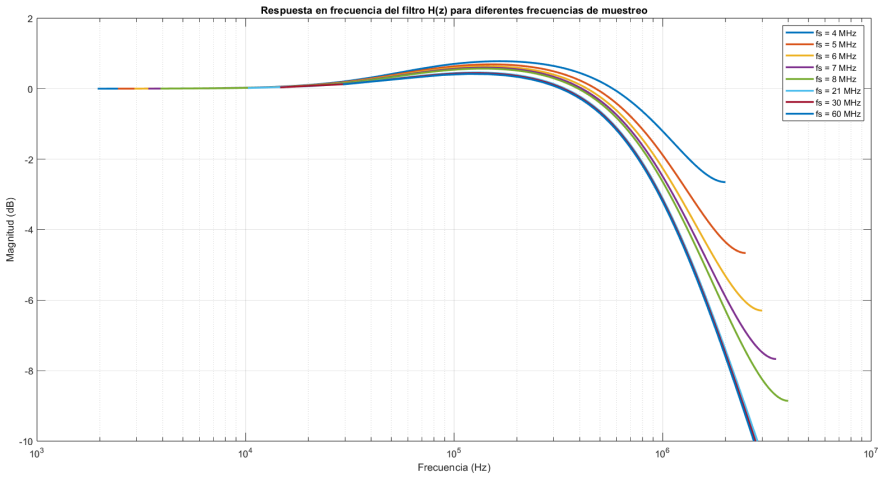

Figura 4.17: Respuesta de frecuencia del PLL de segundo orden con $\zeta = 2$, $B_n = 1{,}5$ MHz y, $f_s = 4$ MHz, 5 MHz, 6 MHz, 7 MHz, 8 MHz, 21 MHz, 30 MHz y 60 MHz.

El demodulador mediante Lazo de Seguimiento de Fase actúa como un filtro pasabajos sobre el espectro del error de fase. En particular, el componente integrador del filtro PI desempeña un rol fundamental en suavizar los errores acumulados, lo que atenúa las componentes de alta frecuencia del error, mejorando la estabilidad del sistema a expensas de una respuesta más lenta a las variaciones rápidas en la fase de la señal.

Este comportamiento de filtrado pasabajos se refleja en las Figuras 4.19 y 4.20, que muestran los espectros del Error de Fase obtenidos a la salida del detector de fase y del Error de Fase Filtrado a la salida del Filtro PI, para diez mil realizaciones por cada frecuencia de muestreo, a un E_b/N_0 constante de 10 dB, con mil bits aleatorios

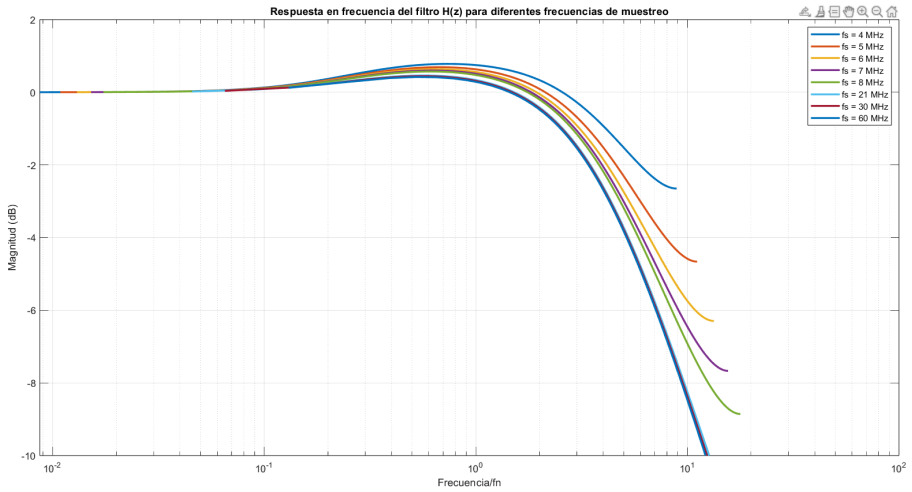

Figura 4.18: Respuesta de frecuencia del PLL de segundo orden normalizada a la frecuencia natural del PLL f_n con $\zeta = 2$, $B_n = 1{,}5$ MHz y, $f_s = 4$ MHz, 5 MHz, 6 MHz, 7 MHz, 8 MHz, 21 MHz, 30 MHz y 60 MHz.

modulados por cada realización. El término integrador del PLL filtra eficazmente las componentes de alta frecuencia, contribuyendo a mejorar la demodulación en presencia de ruido.

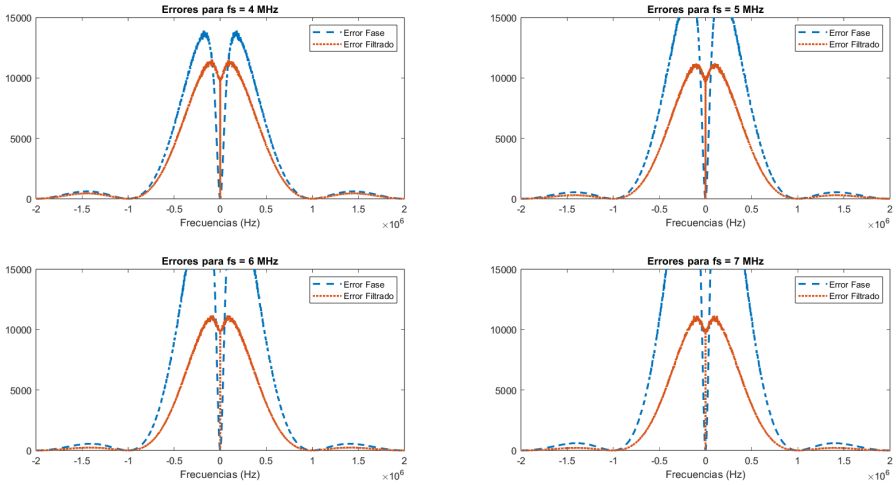

Figura 4.19: Espectros de Error de Fase y Error de Fase Filtrado de Receptores PLL-FSK que trabajan con frecuencias de muestreo iguales a $f_s = 4$ MHz, 5 MHz, 6 MHz y 7 MHz.

Por otra parte, se puede analizar los parámetros clave de este PLL. Se conoce que su Rango de Retención y su Rango de Atracción es infinito, por lo que este demodulador siempre se enganchará a la señal recibida. Por otra parte, su Rango de Separación es:

$$f_n = 225 \text{ kHz}$$

Por lo que:

$$\Delta f_{PO} = 1{,}8 \cdot 2{,}25 \cdot 10^5 \cdot (2+1) = 1{,}21 \text{ MHz}$$

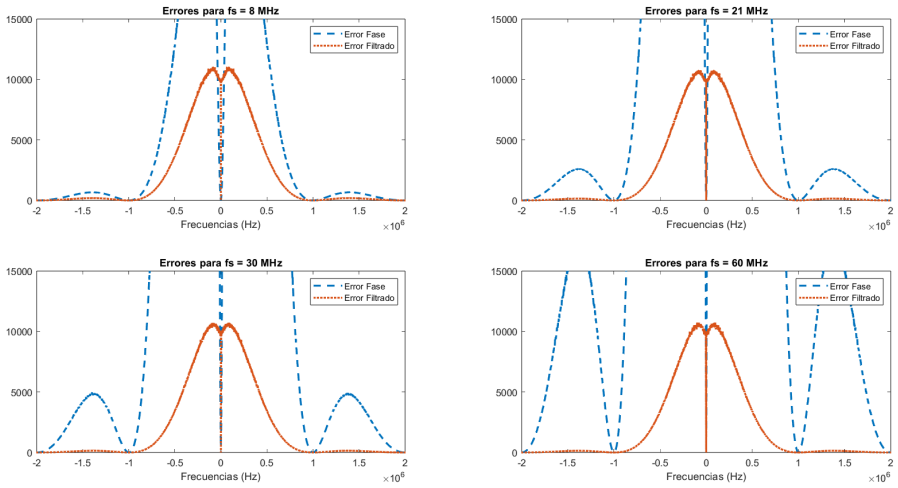

Figura 4.20: Espectros de Error de Fase y Error de Fase Filtrado de Receptores PLL-FSK que trabajan con frecuencias de muestreo iguales a f_s = 8 MHz, 21 MHz, 30 MHz y 60 MHz.

Por su parte, su Rango de Enganche es:

$$\Delta f_L = 2 \cdot 2 \cdot 2{,}2469 \cdot 10^5 = 900 \text{ kHz}$$

Como se puede observar en las figuras de los espectros de Error de Fase los parámetros del Rango Retención y del Rango de Separación se cumplen, dado que el contenido espectral del Error de Fase se encuentra dentro de estos límites.

Capítulo 5

Códigos Gold

5.1. Ruido Pseudoaleatorio (*Pseudorandom noise*)

En criptografía, el ruido pseudoaleatorio (PRN, por sus siglas en inglés) es una señal similar al ruido que satisface una o más de las pruebas estándar de aleatoriedad estadística. Aunque parece carecer de un patrón definido, el ruido pseudoaleatorio consiste en una secuencia determinística de valores que se repetirán después de su período.

En los dispositivos criptográficos, el patrón de ruido pseudoaleatorio está determinado por una clave (*key*, en inglés) y el período de repetición puede ser muy largo, incluso millones de dígitos.

En los sistemas de espectro ensanchado (*spread spectrum*, en inglés), el receptor correlaciona una señal generada localmente con la señal recibida. Dichos sistemas de espectro ensanchado requieren de un conjunto de uno o más "códigos" o "secuencias" tal que [14]:

- Tal como el ruido aleatorio, la secuencia local tenga una correlación muy baja con cualquier otra secuencia en el conjunto, o con la misma secuencia en un offset de tiempo significativamente diferente o con ruido térmico.
- A diferencia del ruido aleatorio, debe ser fácil generar exactamente la misma secuencia tanto en el transmisor como en el receptor.

Un código de pseudo-ruido (código PN) o código de pseudo-ruido aleatorio (código PRN) es aquel que tiene un espectro similar a una secuencia aleatoria de bits pero se genera de forma determinística. Las secuencias más utilizadas en los sistemas de espectro ensanchado de secuencia directa son las secuencias de longitud máxima, los códigos Gold, los códigos Kasami y los códigos Barker [14].

En este trabajo, nos enfocaremos en la implementación y uso de Códigos Gold para realizar la correlación en el receptor PLL-FSK, ya que, gracias a sus propiedades de

autocorrelación y correlación cruzada, son ideales para la detección y sincronización de señales en sistemas de comunicaciones.

5.2. Códigos Gold

En aplicaciones como la técnica de acceso múltiple por división de código (CDMA) y la navegación por satélite, es necesaria una gran cantidad de códigos con buenas propiedades de correlación cruzada. Las secuencias de código que tienen pequeñas correlaciones cruzadas limitadas son útiles cuando varios dispositivos transmiten en el mismo rango de frecuencias como sucede en el caso de estudio del presente trabajo. Y los códigos Gold resultan ser adecuados para esta aplicación, ya que se puede generar una gran cantidad de códigos con correlación controlada mediante un simple desplazamiento en el tiempo de dos pares preferidos de secuencias-m.

Qué son las secuencias-m y cual es el significado del término "pares preferidos" son conceptos que se explican a continuación.

5.2.1. Secuencias de longitud máxima (secuencias-m)

Las secuencias de longitud máxima (también llamadas secuencias-m o secuencias pseudoaleatorias) se construyen basándose en la teoría de campos de Galois. Las secuencias de longitud máxima se generan utilizando estructuras de Registros de Desplazamiento con Retroalimentación Lineal (LFSR, por sus siglas en inglés provenientes de Linear Feedback Shift Register) que implementan la recursividad lineal.

Un LFSR es un registro de desplazamiento en el cual la entrada es un bit proveniente de aplicar una función de transformación lineal a un estado anterior. El valor inicial se denomina **semilla** y, como la forma de operar del registro es determinística, la secuencia de valores producidos está completamente determinada por el estado actual o el estado anterior. La secuencia tiene un periodo de repetición, es decir que la secuencia vuelve a generarse y se repite indefinidamente.

Hay dos tipos de estructuras LFSR disponibles para la implementación:

- LFSR de Galois.
- LFSR de Fibonacci.

En este trabajo se realizó la implementación de un generador de secuencias-m basado en la arquitectura LFSR de Galois. La arquitectura básica de la misma para un polinomio generador de orden L en GF(2) se muestra en la Figura 5.1.

El valor entero L denota el número de elementos de retardo en la arquitectura LFSR y $g_0, g_1, ..., g_L$ representan los coeficientes del polinomio generador, donde el primer y

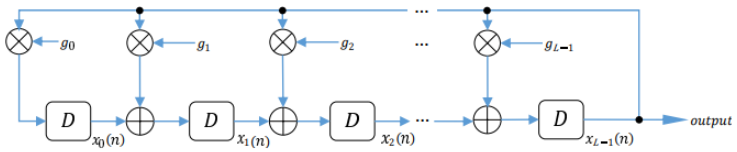

Figura 5.1: Arquitectura básica de un LFSR.

el último coeficiente suelen ser iguales a uno: $g_0 = g_L = 1$. El polinomio generador del LFSR dado es:

$$g(x) = g_0 + g_1 x + g_2 x^2 + \cdots + g_{L-1} x^{L-1} + g_L x^L \text{ (mod-2)}$$

donde $g_0, g_1, ..., g_{L-1} \in GF(2)$, por lo que toma valores binarios. Ahora bien, un LFSR se puede describir matemáticamente mediante la siguiente expresión matricial [15]:

$$x(n+1) = \mathbf{A}x(n) \tag{5.1}$$

con $n \geq 0$ y, donde $\mathbf{A}$ es la matriz de dimensión $L \times L$ dada por:

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & g_0 \\ 1 & 0 & 0 & \cdots & 0 & g_1 \\ 0 & 1 & 0 & \cdots & 0 & g_2 \\ 0 & 0 & 1 & \cdots & 0 & g_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & g_{L-1} \end{pmatrix} \tag{5.2}$$

y $x(n)$ es el vector columna de dimensión L con el estado inicial del registro de desplazamiento:

$$x(n) = [x_0(n), x_1(n), x_2(n), \cdots, x_{L-1}(n)]^T \tag{5.3}$$

5.2.2. Construcción de los códigos Gold

Las secuencias Gold pertenecen a la categoría de "códigos de producto" en donde dos **pares preferidos** de secuencias-m de la misma longitud se someten a una operación XOR (adición de módulo-2) para producir una secuencia Gold. Las dos secuencias-m deben mantener la misma relación de fase hasta que se realicen todas las adiciones. Un ligero cambio de fase, incluso en una de las secuencias-m, produce una secuencia Gold completamente diferente. Los códigos Gold no son máximos y, por lo tanto, tienen una propiedad de auto-correlación deficiente en comparación con la de las secuencias-m que se utilizan. Un Código Gold tiene una correlación de $2^N - 1$ consigo mismo, pero una correlación relativamente baja con otros códigos de la misma familia, es decir Códigos

Gold generados con los mismos pares preferidos pero con distinta semilla y que por lo tanto son de la misma periodicidad en la cual, la correlación cruzada máxima es $2^{\frac{(N+1)}{2}} + 1$. Por lo tanto, es fácil detectar el código mediante la correlación incluso ante el ruido que invierte algunos de los bits. Para nuestro código de 5 bits, la correlación es 31 consigo mismo y $\leq +7/-9$ con otros códigos de Gold.

En la Figura 5.2 se muestra una implementación típica del generador de códigos Gold. En esta implementación, dos registros de desplazamiento de retroalimentación lineal (LFSR), cada uno de longitud L, están configurados para generar dos secuencias-m diferentes.

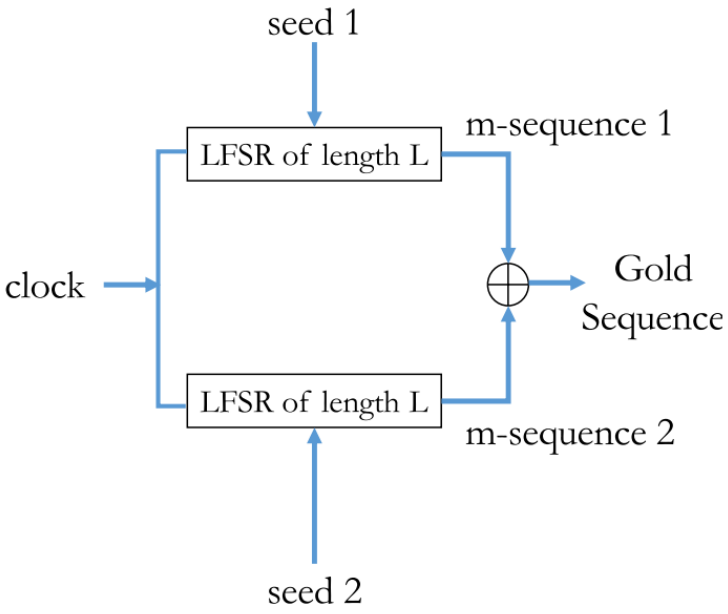

Figura 5.2: Arquitectura para generar codigos Gold.

Los valores iniciales de los LFSR están controlados por semillas diferentes que establecen los LFSR en estados distintos de cero. La fase de la secuencia-m cargada en un LFSR se puede controlar cambiando la semilla inicial, proporcionando así una opción para generar una nueva secuencia Gold. Existen numerosas opciones de secuencias Gold basadas en la configuración de secuencias-m de los LFSR y lo que se carga inicialmente en los dos LFSR. No todas las secuencias de Gold poseen buenas propiedades de correlación. Para tener una secuencia de Código Gold con tres magnitudes máximas de correlación cruzada valoradas, que sea a la vez limitada y uniforme, las secuencias-m generadas por los dos LFSR deben ser del tipo **preferido**.

Las configuraciones proporcionadas en la Figura 5.3 se pueden usar para generar códigos Gold que proporcionen propiedades de correlación óptimas [16]. Los gráficos de auto-correlación y correlación cruzada revelan que la secuencia del Código Gold no posee la excelente propiedad de auto-correlación como la de las secuencias-m indivi-

n	$P_i^n(x)$	*Generator polynomial	N
5	$P_1^5(x)$	$x^5 + x^2 + 1$	31
	$P_2^5(x)$	$x^5 + x^4 + x^3 + x^2 + 1$	
6	$P_1^6(x)$	$x^6 + x^5 + 1$	63
	$P_2^6(x)$	$x^6 + x^5 + x^4 + x + 1$	
7	$P_1^7(x)$	$x^7 + x^6 + 1$	127
	$P_2^7(x)$	$x^7 + x^4 + 1$	
8	$P_1^8(x)$	$x^8 + x^7 + x^6 + x + 1$	255
	$P_2^8(x)$	$x^8 + x^7 + x^5 + x^3 + 1$	
9	$P_1^9(x)$	$x^9 + x^5 + 1$	511
	$P_2^9(x)$	$x^9 + x^8 + x^7 + x^2 + 1$	
10	$P_1^{10}(x)$	$x^{10} + x^7 + 1$	1023
	$P_2^{10}(x)$	$x^{10} + x^9 + x^8 + x^5 + 1$	
11	$P_1^{11}(x)$	$x^{11} + x^9 + 1$	2047
	$P_2^{11}(x)$	$x^{11} + x^{10} + x^9 + x^7 + 1$	
12	$P_1^{12}(x)$	$x^{12} + x^{11} + x^{10} + x^4 + 1$	4095
	$P_2^{12}(x)$	$x^{12} + x^{11} + x^{10} + x^2 + 1$	

Figura 5.3: Polinomios para generar pares preferidos de secuencias-m.

duales, pero posee buenas propiedades de correlación cruzada en comparación con las secuencias-m individuales.

En resumen, lo que se realizó para generar un único Código Gold de la familia de un largo determinado fue:

- Definir los polinomios característicos de estado para los dos LFSR
- Indicar semilla para el segundo LFSR
- Utilizar siempre una semilla de tipo $[0, 0, \cdots, 0, 0, 1]$ para el primer LFSR

Por ejemplo, $GC(1 + x^2 + x^3 + x^4 + x^5, 1 + x^3 + x^5, 00011)$ para un período de largo N $= 31$.

Dentro de la familia de un Código Gold determinado, existen $2^N - 1$ Códigos Gold definidos por diferentes posibles semillas para el segundo LFSR exceptuando la semilla $[0, 0, \cdots, 0, 0, 0]$

Capítulo 6

Resultados

6.1. Generación de Códigos Gold y análisis de correlación

Se simularon los casos de Códigos Gold con periodo $N = 31$ y $N = 4095$. Para cada uno de los casos se realizó la auto-correlación del Código Gold generado con la semilla $x(n) = [0, 0, 0, \cdots, 1]$ y, posteriormente la correlación cruzada de este código con el resto de los códigos de la familia. En un primer caso, se realizó la correlación de una secuencia repetitiva de un Código Gold de largo N = 31 bits, cuya semilla es $[0, 0, 0, 0, 1]$ con el mismo Código Gold. Esto mismo se puede observar en la Figura 6.1 en la cual los picos de auto-correlación se repiten cada N = 31 bits de manera consecutiva 10 veces. De esta manera, se emula una comunicación en donde se transmite sobre un canal ideal (que no posee ruido aditivo) para poder detectar e identificar el Código Gold determinado que representará una etiqueta determinada dentro del sistema de monitoreo.

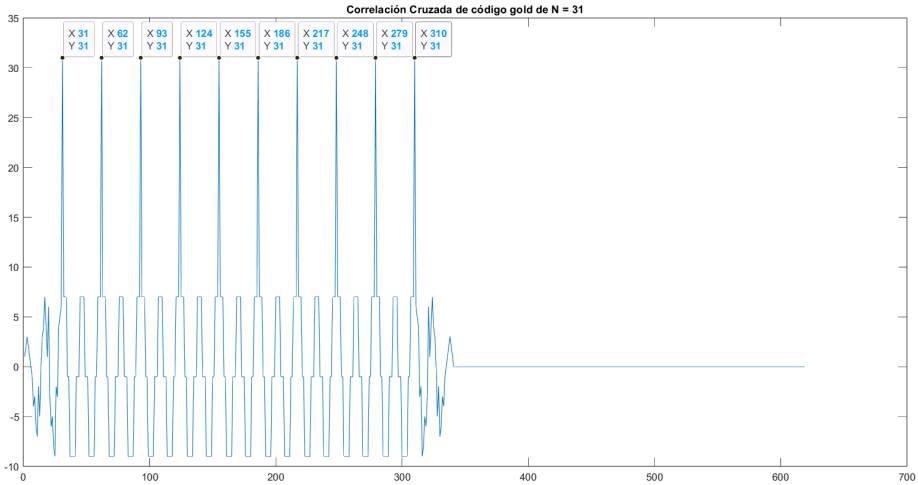

Figura 6.1: Auto-correlación de Código Gold de N = 31.

Por su parte, en la Figura 6.2 se puede observar un ejemplo de la correlación cruzada de este Código Gold con cualquier otro Código Gold de la misma familia.

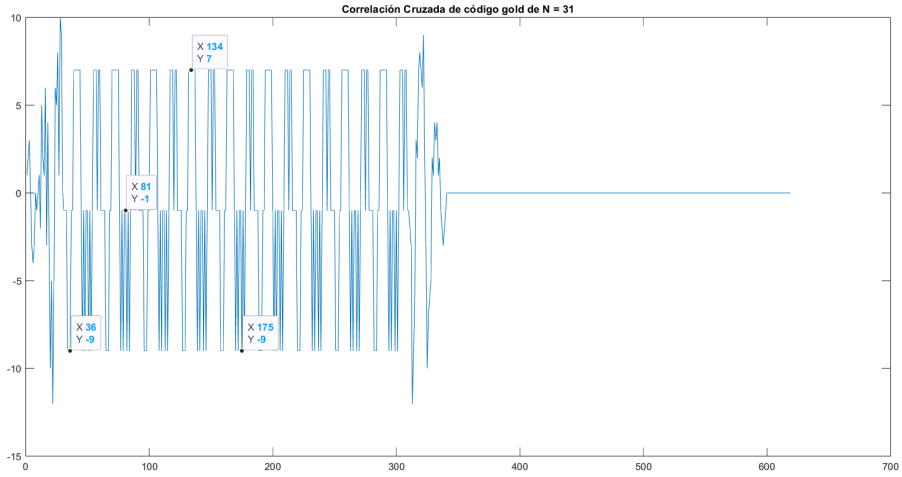

Figura 6.2: Correlación cruzada de códigos gold de N = 31.

Esto mismo se puede extrapolar, en un segundo caso, para el Código Gold de N = 4095 cuya semilla es [0 0 0 0 0 0 0 0 0 0 0 0 1]. En la Figura 6.3 se observa la auto-correlación de una secuencia repetitiva de 10 veces y, en la Figura 6.4 se puede observar la correlación cruzada con un segundo Código Gold de la misma familia, cuya semilla es $[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0]$.

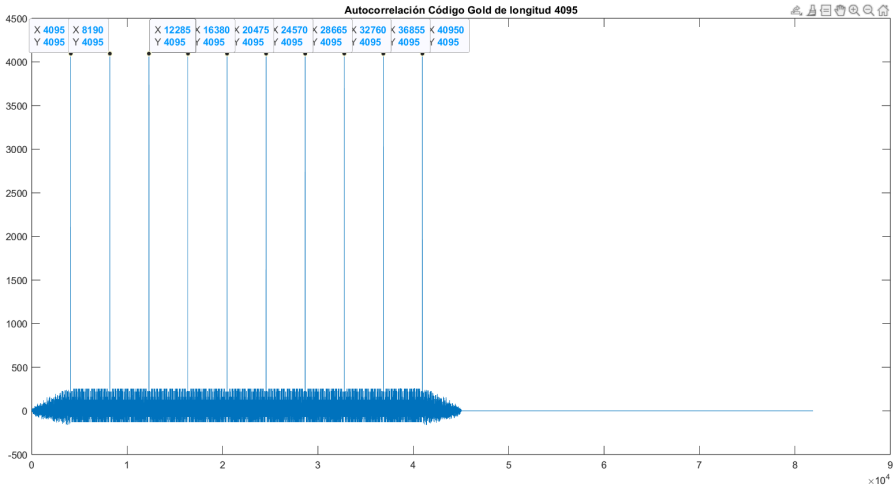

Figura 6.3: Auto-correlación de Código Gold de N = 4095.

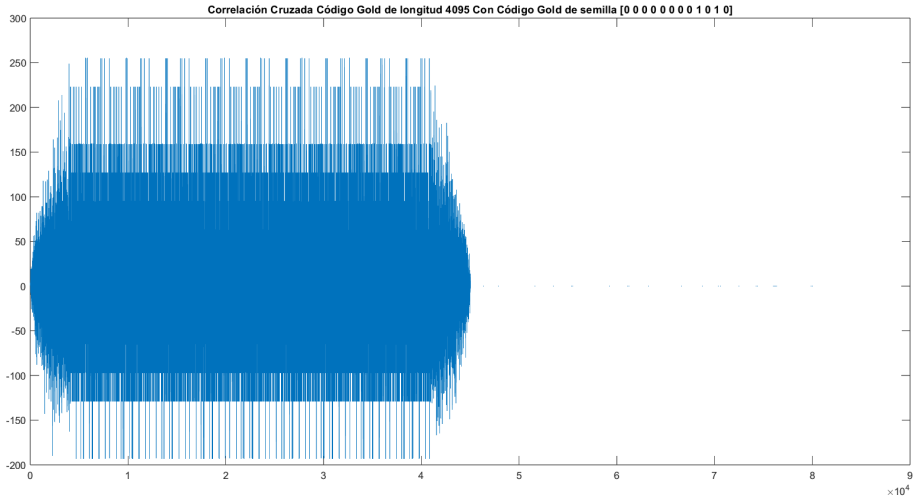

Figura 6.4: Correlación cruzada de códigos gold de N = 4095.

6.2. Análisis del desempeño de Tasa de Error de Bit (BER)

Se programó un código para estimar la Tasa de Error de Bit (BER) de los distintos demoduladores FSK. La metodología para estimar el BER consiste en: primero, se genera un conjunto inicial de mil bits aleatorios. A continuación, se modula la señal correspondiente a estos bits, se le suma ruido AWGN con la E_b/N_0 a simular, y luego se la somete a un proceso de demodulación utilizando los diferentes demoduladores descriptos con anterioridad. Tras cada demodulación, se calcula el número de bits erróneos comparando la señal demodulada con la original. Este proceso se repite en bloques de 10 iteraciones, donde los bits aleatorios se regeneran cada 10 ciclos para asegurar la variabilidad de la señal de prueba. El ciclo se mantiene hasta que se detectan al menos 100 errores de bits (o una cantidad definida previamente) para cada valor de la relación señal-ruido (E_b/N_0) evaluado. Al final, se calcula el BER dividiendo el número total de errores encontrados entre el número total de bits procesados para ese valor de E_b/N_0. Esta metodología garantiza una estimación precisa del BER, especialmente en escenarios con bajos niveles de ruido, donde los errores pueden ser poco frecuentes.

Utilizando una frecuencia de muestreo de 6 MHz, frente a los 434 MHz de la frecuencia portadora, se realizó el análisis para el rango de valores de E_b/N_0 comprendido entre los 0 dB y los 18 dB, con un paso igual a 2 dB entre mediciones, para los demoduladores mediante Lazo de Seguimiento de Fase y mediante Discriminador de Frecuencias dado que presentan comportamientos similares contra la curva teórica de Probabilidad de Error de Bit (P_{eb}) mediante detección no coherente, en el rango mencionado. En este caso se utilizó el demodulador PLL-FSK configurando su Ancho de Banda de Lazo a 1.5 MHz y su Factor de Amortiguamiento a 2. Los resultados se pueden observar en la Figura 6.5.

Por otra parte, también se determinó la curva de BER del demodulador mediante Correladores. Dado que presenta un mejor rendimiento a menores E_b/N_0 respecto de la teórica, se midió su rendimiento entre los -4 dB y los 14 dB, con un paso de 2 dB entre mediciones. El resultado se puede observar en la Figura 6.6, en la misma se puede ver como, en todo el rango elegido de valores de E_b/N_0, la curva de este demodulador se pega a la curva teórica. De manera que el rendimiento es excelente cuando el demodulador está diseñado tal que conoce las frecuencias F_1 y F_2 de operación, del mismo modo que lo hacía el Detector No Coherente a partir del cuál se derivó la Probabilidad de Error de Bit Teórica.

Cabe destacar que, en ambos casos, los rangos de E_b/N_0 fueron elegidos debido a que las simulaciones en los valores mencionados necesitan de números razonables de bits para calcular las Tasas de Error de Bit respectivas, es decir, para encontrar 100

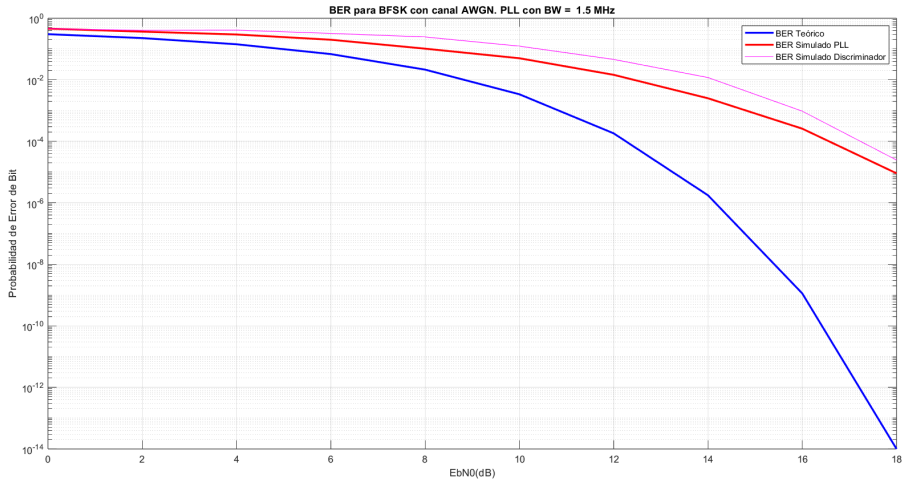

Figura 6.5: Comparación entre la curva teórica de Probabilidad de Error de Bit mediante detección No Coherente (azul) y estimación de Tasas de Error de Bit para demoduladores mediante PLL (roja) ($B_n = 1.5$ MHz y $\zeta = 2$) y mediante Discriminador de Frecuencias (rosa) utilizando una frecuencia de muestreo $f_s = 6$ MHz.

errores por cada E_b/N_0. Para valores mas grandes, el tiempo de cómputo se incrementa exponencialmente.

Adicionalmente, se puede evaluar el desempeño de los demoduladores variando la frecuencia de muestreo de la señal. Para el Demodulador PLL-FSK, se simularon frecuencias de muestreo de 4 MHz, 5 MHz, 6 MHz, 7 MHz, 8 MHz, 21 MHz, 30 MHz y 60 MHz. En cada uno de estos casos, se ajustó la frecuencia del DDS de acuerdo con el análisis espectral presentado en la sección 3.2.5. Dicho análisis permitió ubicar la frecuencia del DDS en los centros de los espectros de entrada de este receptor discreto, los cuales son las que se muestran en la Tabla 4.1. Además, para cada uno de los casos se ajustaron los coeficientes del filtro PI (K_1 y K_2) siguiendo las ecuaciones presentadas anteriormente y, manteniendo constante el Ancho de Banda de Ruido Equivalente (B_n) igual a 1.5 MHz y el Factor de Amortiguamiento igual a 2. Este ajuste se realizó para asegurar una comparación justa, ya que una mayor diferencia entre la frecuencia de la portadora y la frecuencia de referencia del DDS incrementa el tiempo de adquisición del PLL, lo que degrada la demodulación y, por consiguiente, incrementa la Tasa de Error de Bit (BER).

El BER se calculó en el rango de 0 dB a 16 dB, con un paso de 2 dB entre mediciones, para mantener un tiempo de simulación razonable. Los resultados obtenidos, mostrados en la Figura 6.7, evidencian un empeoramiento a medida que aumenta la frecuencia de muestreo.

Sospechamos que existe una mejora en el rendimiento del demodulador a medida que aumenta la razón entre la desviación de frecuencia y el ancho de banda de la primera zona de Nyquist, dado que se observa que una frecuencia de muestreo más baja implica un menor ancho de banda para el PLL, lo que puede ser beneficioso en entornos

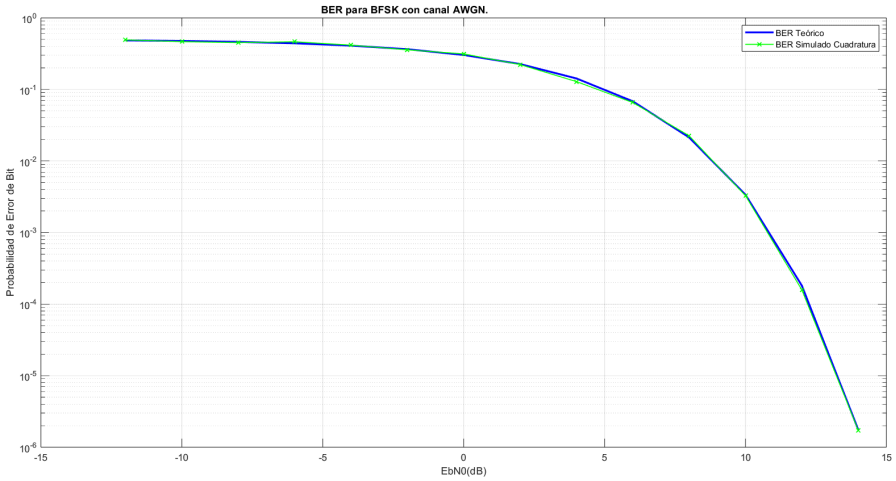

Figura 6.6: Comparación entre la curva teórica de Probabilidad de Error de Bit mediante detección No Coherente (azul) y estimación de Tasa de Error de Bit para demodulador mediante correladores (verde) utilizando una frecuencia de muestreo $f_s = 6$ MHz.

ruidosos. Un PLL con un ancho de banda más estrecho tiende a filtrar el ruido de manera más efectiva, favoreciendo una demodulación más precisa. En contraposición, un incremento en la frecuencia de muestreo expande el ancho de banda del sistema, lo que puede hacer al PLL más susceptible al ruido, disminuyendo el rendimiento en la demodulación. Un mayor ancho de banda en la primera zona de Nyquist obliga al DDS a seguir un rango más amplio de frecuencias, lo que aumenta el tiempo de adquisición del PLL y, consecuentemente, el BER. Finalmente, dado que se utiliza una única frecuencia de muestreo tanto para la señal recibida como para el error de fase en la salida del detector de fase, una reducción en el ancho de banda de la primera zona de Nyquist implica el ingreso de menor ruido dada la DEP de ruido constante al PLL y por lo tanto habrá un menor deterioro en el desempeño.

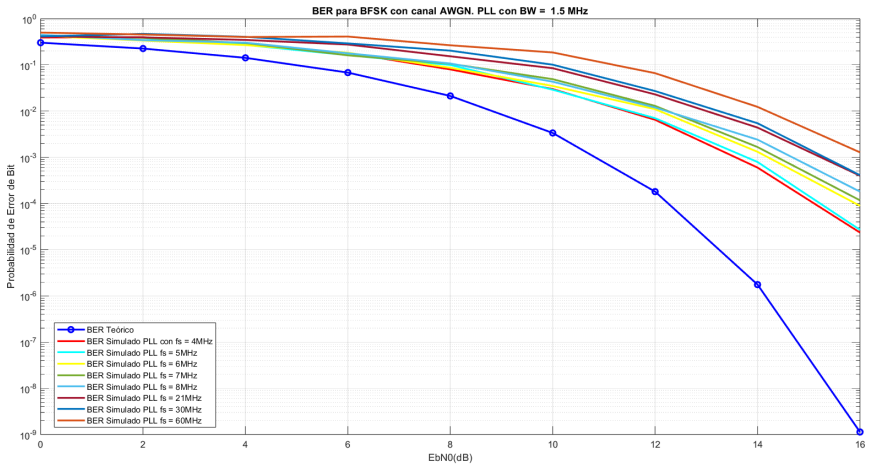

Figura 6.7: Comparación entre la curva teórica de Probabilidad de Error de Bit mediante detección No Coherente (azul) y la estimación de Tasa de Error de Bit para demodulador mediante PLL utilizando distintas frecuencias de muestreo.

Por su parte esto también se puede realizar para el demodulador mediante Discriminador de Frecuencias. En este caso lo simulamos para las frecuencias de muestreo iguales a 6 MHz, 8 MHz, 9 MHz, 10 MHz y, 12 MHz. Dado el tiempo de simulación razonable requerido, se calculó su BER en el rango entre 0 dB y 18 dB, con un paso de 2 dB entre mediciones. En la Figura 6.8 se pueden observar los resultados.

Posteriormente, en la Figura 6.9 se tiene una segunda visualización que consiste en cuatro subgráficas (una para cada frecuencia de muestreo) que muestran los umbrales de decisión y los promedios de señal para los bits '0' y '1'. En estas gráficas, se puede observar la separación entre los niveles de señal correspondientes a los diferentes bits y cómo el umbral de decisión se ubica entre ellos. Los promedios de señal para bits '0' se muestran en azul, para bits '1' en rojo, y el umbral en verde. Esta visualización ayuda a entender cómo el sistema discrimina entre los diferentes bits y cómo el ruido afecta esta discriminación.

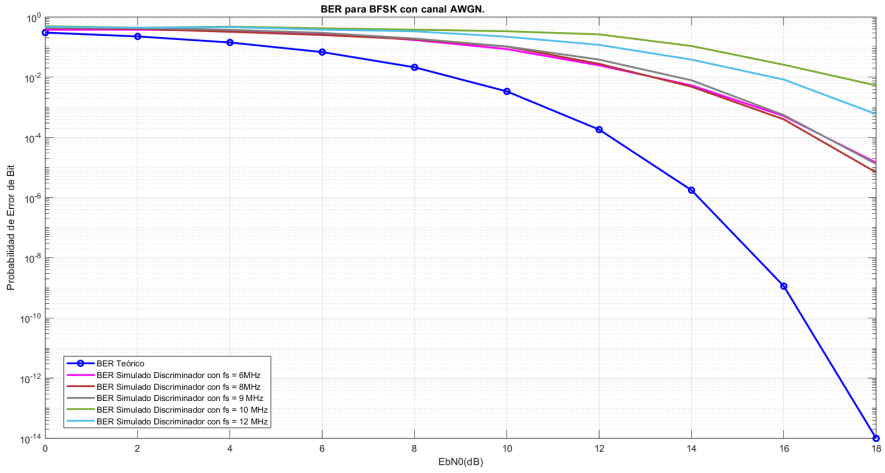

Figura 6.8: Comparación entre la curva teórica de Probabilidad de Error de Bit mediante detección No Coherente (azul) y estimación de Tasa de Error de Bit para demodulador mediante Discriminador de Frecuencias utilizando distintas frecuencias de muestreo.

Existe un efecto umbral tanto para el demodulador mediante PLL como para el demodulador mediante Discriminador de Frecuencias dado que a partir de determinados valores de relación energía por bit sobre densidad espectral de potencia de ruido (E_b/N_0) estos empiezan a tener buenos rendimientos, que siguen mejorando a medida que se incrementa este valor. En el caso del PLL-FSK el umbral está en los 12 dB aproximadamente y para el caso del Discriminador en los 14 dB, teniendo en cuenta un BER = 10^{-2}, es decir, que encuentra 100 (cien) errores cada 10,000 (diez mil) bits analizados.

En el caso del demodulador mediante Correladores el efecto umbral no sucede. Esto último se puede comprobar al medir el rendimiento para el demodulador mediante Correladores a distintas frecuencias de muestreo en el rango de E_b/N_0 comprendido entre 0 y 14 dB con un paso igual a 2 dB entre mediciones. En la Figura 6.10 se mide

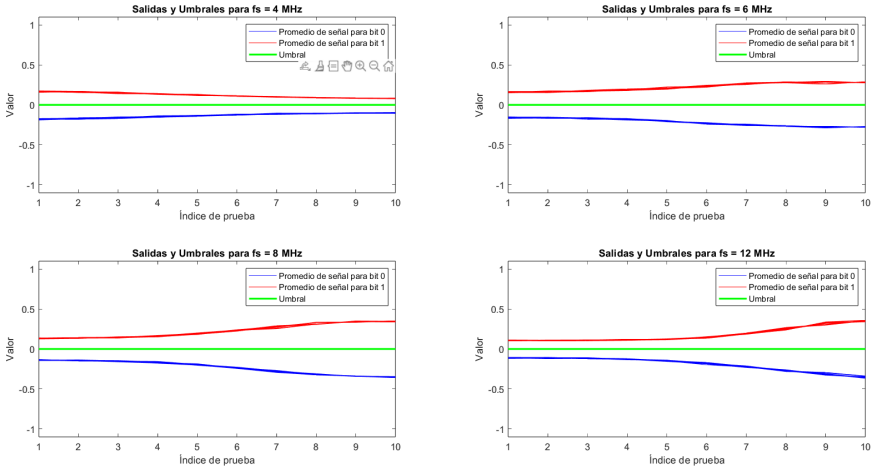

Figura 6.9: Visualización de los umbrales de decisión y promedios de señal para diferentes frecuencias de muestreo. Se muestran los niveles promedio para bits '0' (azul) y bits '1' (rojo), así como el umbral de decisión adaptativo (verde) que discrimina entre ambos niveles. Cada subgráfica corresponde a una frecuencia de muestreo diferente: 4 MHz, 6 MHz, 8 MHz y 12 MHz.

su rendimiento para $f_s = 4$ MHz, 5 MHz, 6 MHz, 7 MHz, 8 MHz, 9 MHz, 10 MHz, 12 MHz, 21 MHz, 30 MHz y, 60 MHz. Allí podemos observar cómo a distintas frecuencias de muestreo, su rendimiento es constante y se pega a la curva teórica de probabilidad de error de bit mediante detección no coherente, descripta en la Sección 3.2.1.

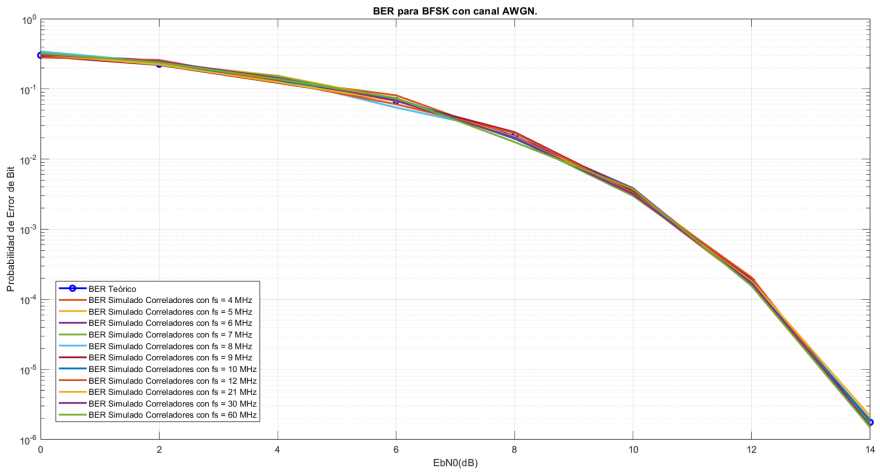

Figura 6.10: Comparación entre curva teórica de Probabilidad de Error de Bit mediante detección No Coherente (azul) y estimación de Tasa de Error de Bit para demodulador mediante Correladores utilizando distintas frecuencias de muestreo.

6.3. Análisis de desempeño de sincronismo

En el análisis de desempeño de sincronismo por parte del receptor, se simularon dos casos particulares en los que se determinó la muestra en la que se encuentra el pico de correlación entre la secuencia Gold, ya sea de largo 31 bits ó 4095 bits, y el Error Filtrado que se obtiene a la salida del Filtro PLL en nuestro demodulador PLL-FSK. Antes de realizar este análisis, vamos a realizar una observación que se debe tener en cuenta.

Error en un tiempo de muestra

En todos los casos que analizaremos más adelante sucederá que existe al menos un retardo en un tiempo de muestra en la sincronización. Este error se puede observar en la Figura 6.11, donde se compara la auto-correlación de un Código Gold de longitud 31, con la correlación entre ese mismo Código Gold y el Error de Fase Filtrado. Este último es obtenido en el proceso de demodulación de una señal recibida que contiene el mismo Código Gold, que fue transmitido por un canal canal ideal sin ruido aditivo.

Se puede concluir que ese error se debe al retardo de muestra introducido por el PLL en el Error de Fase Filtrado, tal como se observa en la Figura 6.12, cuando comparamos el Código Gold muestreado a la misma frecuencia del demodulador, contra el Error Filtrado obtenido a la salida del mismo demodulador.

Ahora bien, si adicionalmente realizamos un *zoom* sobre el pico de correlación, como el realizado en la Figura 6.13 se puede observar que existe una deformación de este pico. Esta deformación es introducida por el PLL, de manera posterior a la convolución entre el error de fase y la respuesta al impulso del lazo. Esto explica que a valores bajos de E_b/N_0, el máximo de correlación se encuentra una muestra adicional retrasada (la muestra 304), en el caso particular del Código Gold de largo 31.

Por otra parte, si realizamos el mismo experimento para el Código Gold de largo 4095, existe también un retardo de una muestra en sincronización introducido por el PLL, como se observa en la Figura 6.14. Además, si realizamos un *zoom* sobre el pico de correlación (Figura 6.15) se observa también cómo el PLL deforma este pico. Cabe destacar que aunque exista una deformación en el pico, no existirá un retardo de muestra adicional al ya existente, como sí sucede en el caso del Código Gold de largo 31. Esto se debe a que la magnitud del pico de correlación es mucho mayor en comparación con el anteriormente mencionado, debido a que las longitudes 4095 y 31, respectivamente, son las que determinan estas magnitudes.

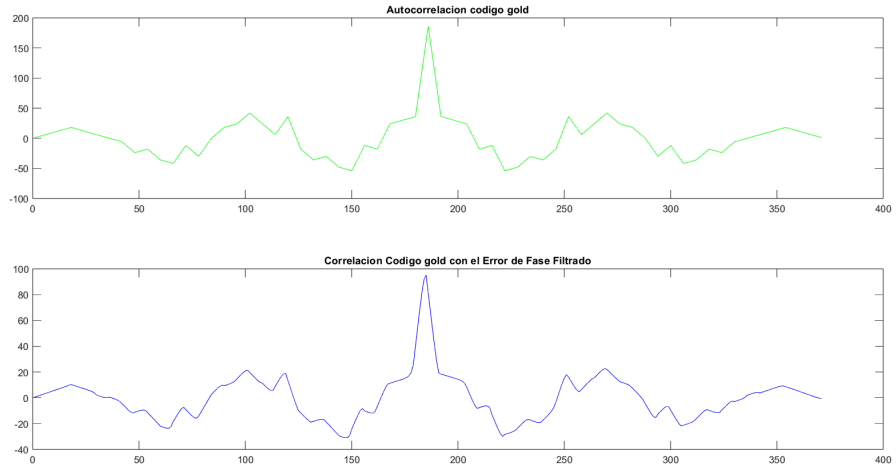

Figura 6.11: Auto-correlación de un Código Gold dado de N = 31 (arriba) y, correlación entre ese Código Gold dado y el Error de Fase Filtrado (abajo).

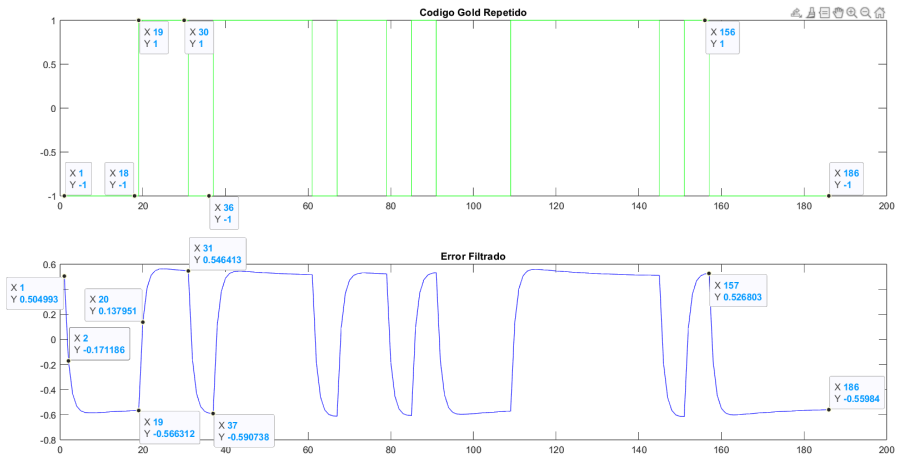

Figura 6.12: Código Gold muestreado a f_s = 6 MHz (arriba) y, Error de Fase Filtrado a la salida del Filtro de Lazo (abajo).

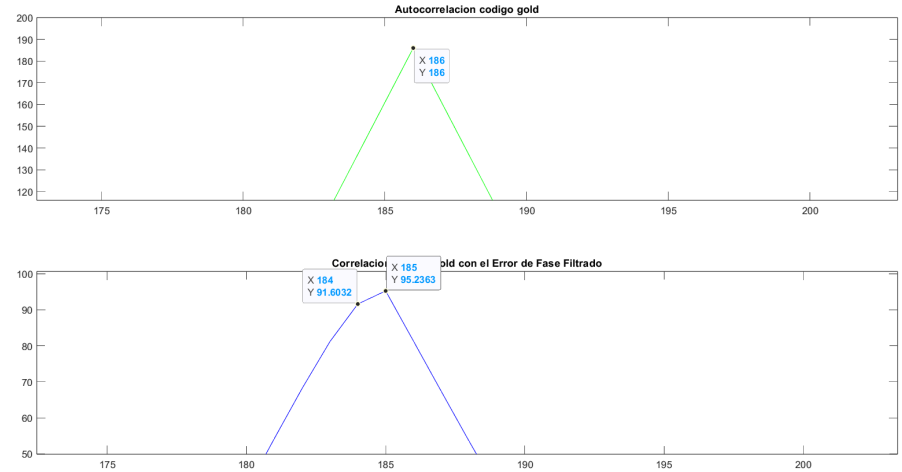

Figura 6.13: Pico de auto-correlación del Código Gold de N = 31 (arriba) y, Pico de correlación entre ese Código Gold dado y el Error de Fase Filtrado (abajo).

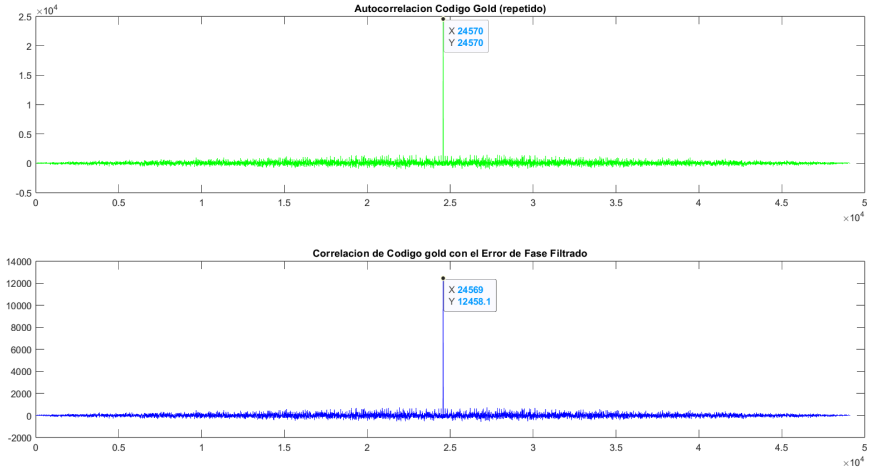

Figura 6.14: auto-correlación de un Código Gold dado de N = 4095 (arriba) y, correlación entre ese Código Gold dado y el Error de Fase Filtrado (abajo).

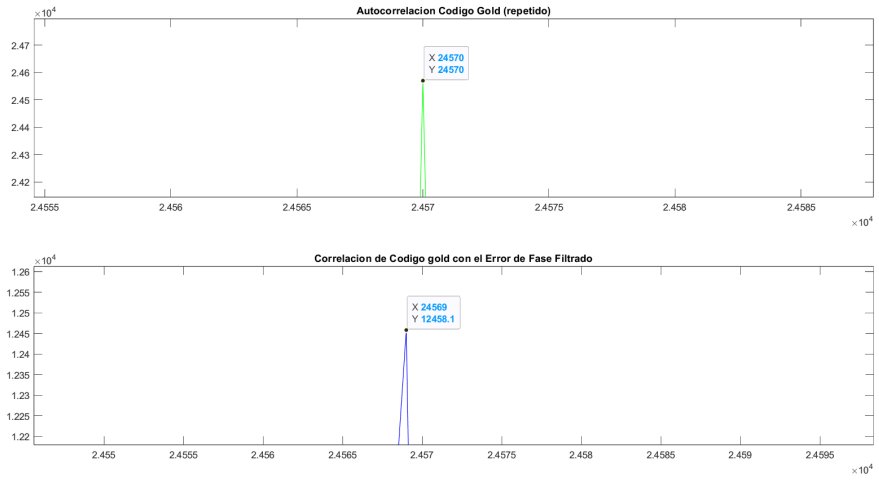

Figura 6.15: Pico de auto-correlación del Código Gold de N = 4095 (arriba) y, Pico de correlación entre ese Código Gold dado y el Error de Fase Filtrado (abajo).

Primer caso

En un primer caso se transmite a través de un canal AWGN una señal modulada desde el transmisor, que consiste de una cantidad de muestras sin señal alguna, también denominada "silencio". Concatenadamente se agrega un Código Gold modulado de un periodo particular y, posteriormente se repite la cantidad de muestras de silencio. A partir de ésto, determinamos en qué muestra se encuentra el pico de correlación entre dicha secuencia mencionada compuesta por un Código Gold y, una secuencia de referencia que corresponde a ese mismo Código Gold, para cada E_b/N_0 en el rango de simulación comprendido entre -4 dB y 20 dB, con un paso de 2 dB entre cada valor. Por cada E_b/N_0 este escenario se repite mil veces, es decir, mil picos observados por cada E_b/N_0, con el Factor de Amortiguamiento del PLL-FSK igual a 2, un Ancho de Banda de Ruido Equivalente igual a 1.5 MHz y una frecuencia de muestreo de 6 MHz. Luego, calculamos el porcentaje de veces que el pico de correlación se encuentra en la muestra correcta. Ésto se realiza, con el objetivo de, en términos de detección de la señal, lograr identificar la llegada de la señal y el *tag* determinado que se está transmitiendo en el canal ruidoso y, en términos de sincronización, para ubicar el momento exacto en que comienza la secuencia, lo cual es necesario para realizar el cálculo del TOA en el sistema ATLAS.

La cantidad de muestras de silencio al inicio y al final del Código Gold modulado, y que se le sumará ruido gaussiano al ser transmitido, será el equivalente a 20 bits en el caso del **Código Gold de 31 bits** de longitud. Ahora bien, a partir de todo lo mencionado ¿Cuál es la muestra correcta? Si se tiene en cuenta que la frecuencia de muestreo del Error Filtrado es igual a f_s y que el tiempo de bit de la señal digital moduladora es $T_b = 10^{-6}s$, entonces, la muestra correcta viene dada por la ecuación:

$$\textit{Muestra Correcta} = (\textit{Bits Iniciales Sin Señal} + \textit{Bits del Código Gold}) \cdot T_b \cdot f_s \text{ - } 1$$

El termino "-1" se debe a que el PLL al momento de realizar la demodulación le ingresa un retardo de muestra igual a 1 al Error Filtrado como se mencionó anteriormente. Con lo cual, si la frecuencia de muestreo es igual a 6 MHz y N = 31, la muestra correcta será $(20 + 31) \cdot 10^{-6} \cdot 6 \cdot 10^6 - 1 = 305$.

Con el fin de analizar el comportamiento del sincronismo obtenemos un histograma de la cantidad de veces que se encontró un pico máximo de correlación en cada muestra, para $E_b/N_0 = 6$dB, que se grafica en la Figura 6.16 y, en la Figura 6.17 se puede observar el mismo histograma pero para E_b/N_0 de 20 dB. En ambos se puede corroborar como el nivel de ruido y el retado del PLL afecta la estimación del máximo de correlación.

Luego, en la Figura 6.18 se puede observar el porcentaje de picos correctos para estos 1000 escenarios, es decir, para los 1000 picos analizados por cada E_b/N_0, si tenemos en cuenta que la muestra correcta se debe unicamente a la muestra 305. Sin embargo,

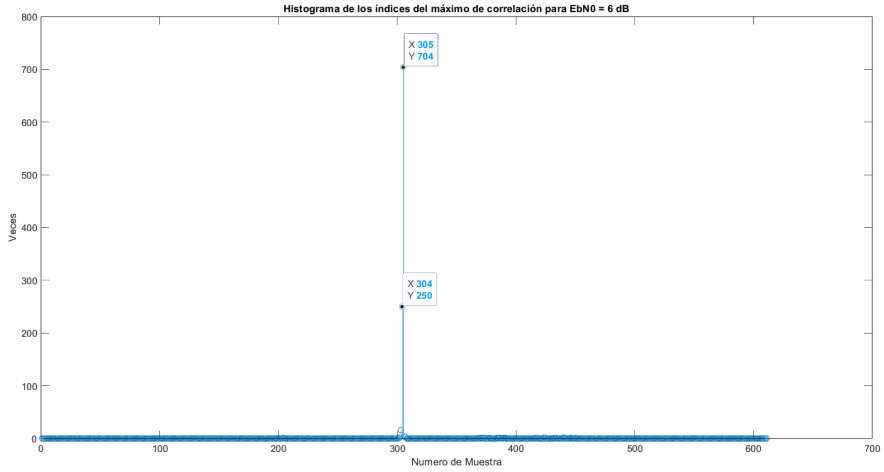

Figura 6.16: Histograma de los índices del máximo de correlación para EbN0 = 6 dB para Código Gold de N = 31 y, señal de silencio al principio y final.

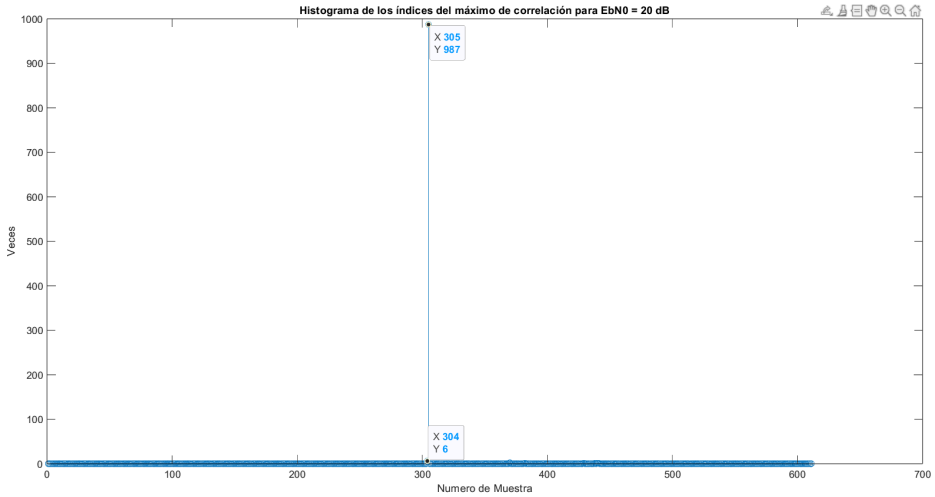

Figura 6.17: Histograma de los índices del máximo de correlación para EbN0 = 20 dB para Código Gold de N = 31 y, señal de silencio al principio y final.

para los dos valores de E_b/N_0 anteriores, existen muchos casos en los que los picos caen tanto en la muestra 305 como también en la muestra adyacente 304, hecho explicado anteriormente. Con lo cual, si adicionalmente, tenemos en cuenta que tanto la muestra 305 como la 304 son las muestras correctas y, calculamos de nuevo el porcentaje de picos correctos en mil escenarios, se obtienen los resultados observados en la Figura 6.19 en donde se obtiene una mejora en la curva respecto del caso anterior. Y es que si comparamos ambas curvas y si consideramos un 90 % de aciertos como un buen umbral de rendimiento, podemos decir que la sincronización del PLL que tiene en cuenta solo una muestra empieza a tener un buen rendimiento a partir de los 10 dB. En cambio, la sincronización del PLL que tiene en cuenta las dos muestras empieza a tener un buen rendimiento a partir de los 5 dB.

Por otra parte, para el **Código Gold de periodo 4095**, se realizaron mil (1000)

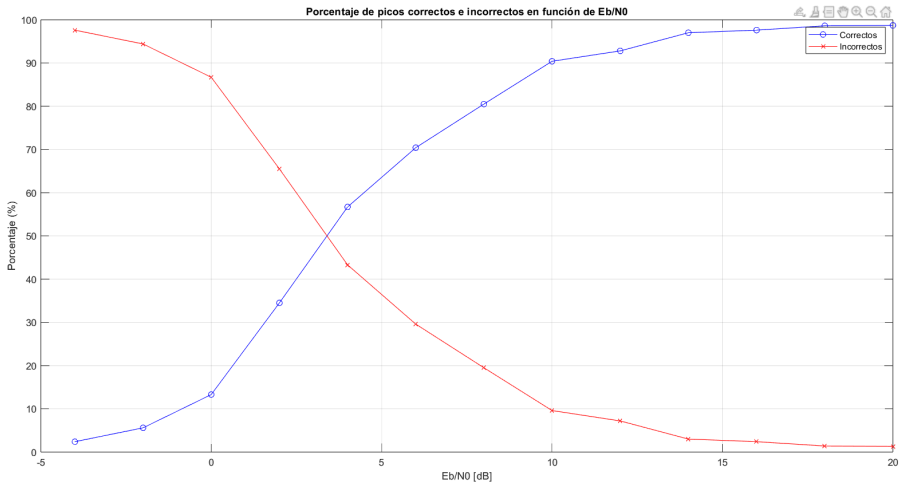

Figura 6.18: Porcentaje de picos de correlación en la muestra 305 en 1000 escenarios para Código Gold de N = 31 y, señal de silencio al principio y final.

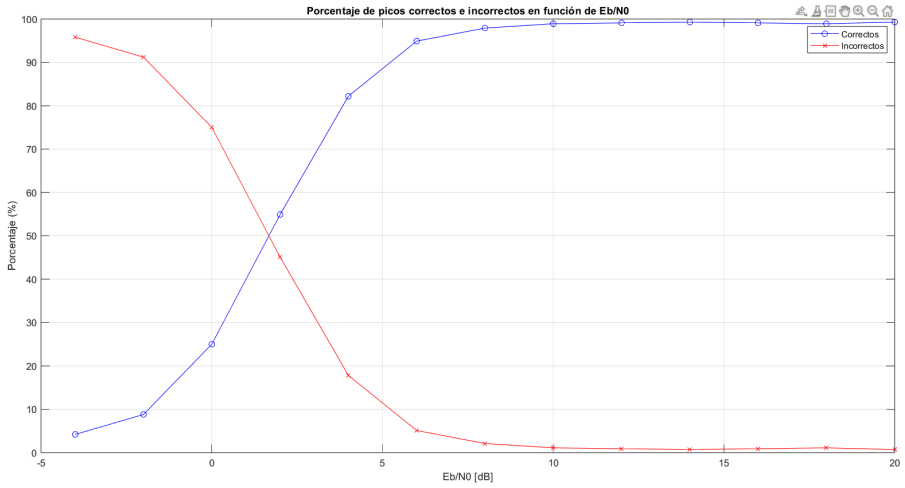

Figura 6.19: Porcentaje de picos de correlación en las muestras 304 y 305 en 1000 escenarios para Código Gold de N = 31 y, señal de silencio al principio y final.

escenarios por cada E_b/N_0, es decir, mil picos observados por cada E_b/N_0, con un Factor de Amortiguamiento del PLL-FSK igual a 2, un Ancho de Banda de Lazo igual 1.5 MHz y una frecuencia de muestreo igual a 6 MHz. La señal estaba compuesta por un Código Gold modulado con el equivalente a 200 tiempos de bits de silencio (sin señal) al principio y 200 tiempos de bits idénticos al final del Código Gold modulado, es decir, ambos a los costados del mismo. Con estos parámetros, la muestra correcta se calculó como $(200 + 4095) \cdot (6 \cdot 10^6) \cdot (\frac{1}{10^6}) - 1 = 25{,}769$, con lo cual, es la número 25,769 dentro del conjunto de muestras de la señal demodulada por el demodulador PLL-FSK.

En los histogramas para distintos niveles de E_b/N_0, se calcula la cantidad de veces en las que se determinó el pico de correlación en las muestras dadas por el eje de abscisas para $E_b/N_0 = 0$ dB (Figura 6.20), para $E_b/N_0 = 4$ dB (Figura 6.21) y, para

$E_b/N_0 = 20$ dB (Figura 6.22). Como se puede observar en todos los casos existe un retardo de un tiempo de una muestra como se analizó en la sección anterior. Por último, en la Figura 6.23 se pueden observar los resultados del porcentaje de picos correctos e incorrectos para el rango comprendido entre -4 dB y 20 dB con un paso igual a 2 dB entre mediciones. En esta última figura, se puede observar que la sincronización para este caso particular empieza a funcionar correctamente a partir de los 2 dB. Sí tomamos como umbral óptimo el 90 % de aciertos. Esto significa una mejora sustancial respecto del caso del Código Gold de largo 31, por lo que se puede concluir que a mayor longitud de Código Gold, mejor será la sincronización del receptor BFSK.

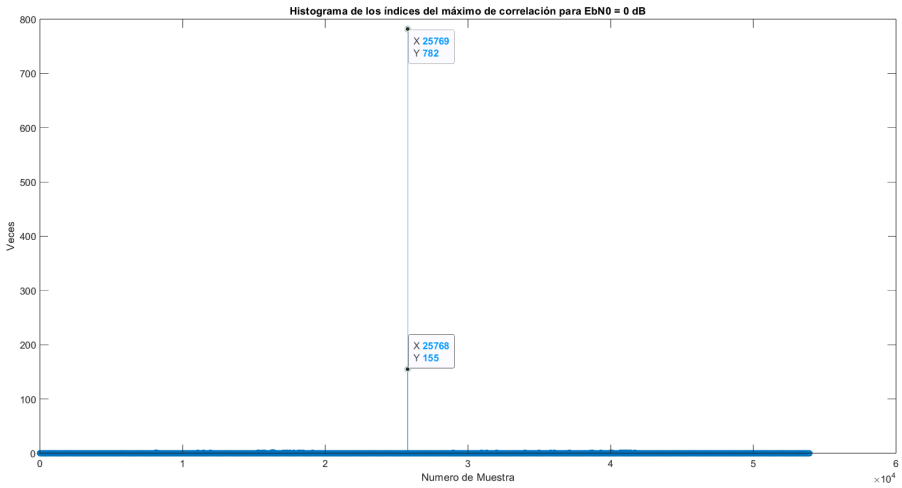

Figura 6.20: Histograma de los índices del máximo de correlación para EbN0 = 0 dB para Código Gold de N = 4095 y, señal de silencio al principio y final.

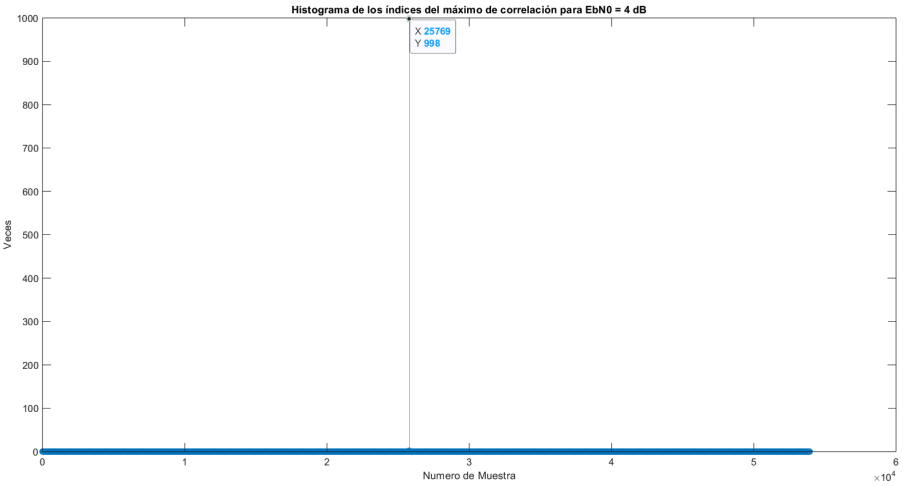

Figura 6.21: Histograma de los índices del máximo de correlación para EbN0 = 4 dB para Código Gold de N = 4095 y, señal de silencio al principio y final.

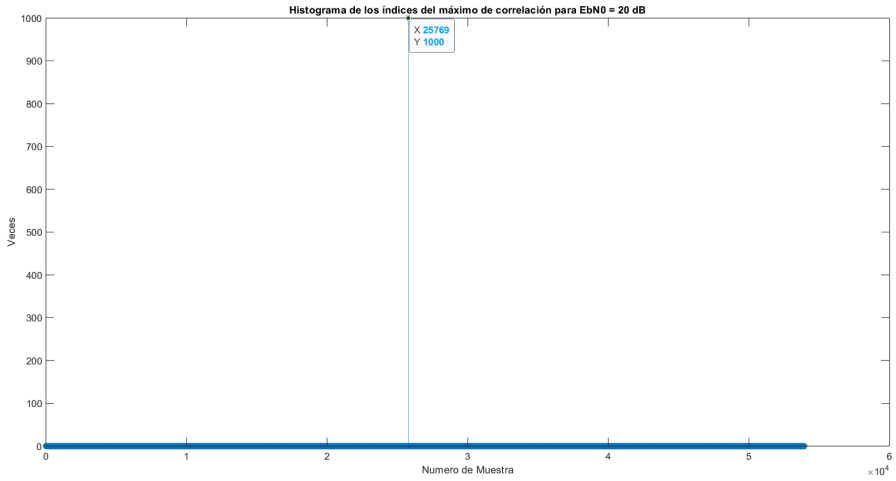

Figura 6.22: Histograma de los índices del máximo de correlación para EbN0 = 20 dB para Código Gold de N = 4095 y, señal de silencio al principio y final.

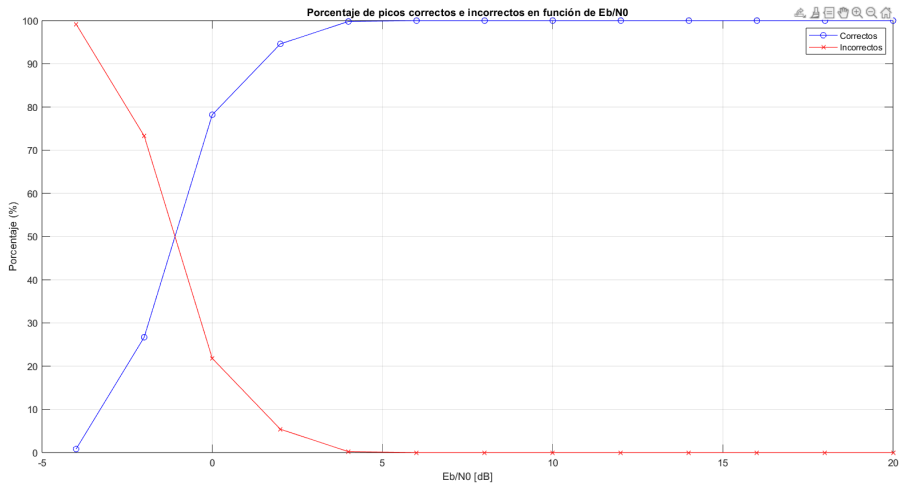

Figura 6.23: Porcentaje de picos de correlación en la muestra 25769 en 1000 escenarios para Código Gold de N = 4095 y, señal de silencio al principio y final.

Segundo Caso

En el segundo caso, de forma similar al primer caso, se envía una señal que fue modulada a partir de una señal mensaje, que consiste de una cantidad de bits determinados que alternan 1 (unos) y 0 (ceros) de manera consecutiva. Posteriormente se concatena con un Código Gold de un periodo particular y, por último se vuelve a concatenar dicha secuencia de unos y ceros del mismo largo. Repetimos los mismos dos escenarios del primer caso. En la Figura 6.24 se puede observar el histograma que determina la cantidad de veces que se encontró un pico máximo de correlación entre el error filtrado y el **Código Gold de longitud 31**, con un adicional de veinte bits compuestos de los ceros y unos consecutivos mencionados, en el número de muestra determinada por el eje de abscisas, para un E_b/N_0 de 6 dB, siendo la muestra correcta la número 305. En la Figura 6.25 se puede observar el mismo histograma pero para un E_b/N_0 de 20 dB. En ambas figuras, se pueden observar los retardos introducidos por el PLL. Luego, en la Figura 6.26 se ve el porcentaje de aciertos y errores en la ubicación del pico de correlación.

Por último, si se tiene en cuenta también la muestra 304 en combinación con la 305, en la Figura 6.27 se ve el porcentaje de aciertos y errores en la ubicación del pico de correlación. Si comparamos está última grafica de porcentajes con el caso que solo tiene en cuenta a la muestra 305 como correcta, se nota una mejora sustancial. En efecto, la sincronización empieza a funcionar correctamente a partir de los 2 dB, frente a los 8 dB necesarios en el caso anterior, teniendo en cuenta un umbral de 90 % de aciertos. Por último, también se observan mejoras frente a los casos con silencio al inicio y al final, ya que esta secuencia alternada, presente en el preámbulo de la señal de interés, permite al PLL-FSK ajustarse a la frecuencia y fase de la señal, lo que conlleva un mejor enganche del mismo y una mejora en el rendimiento de sincronización.

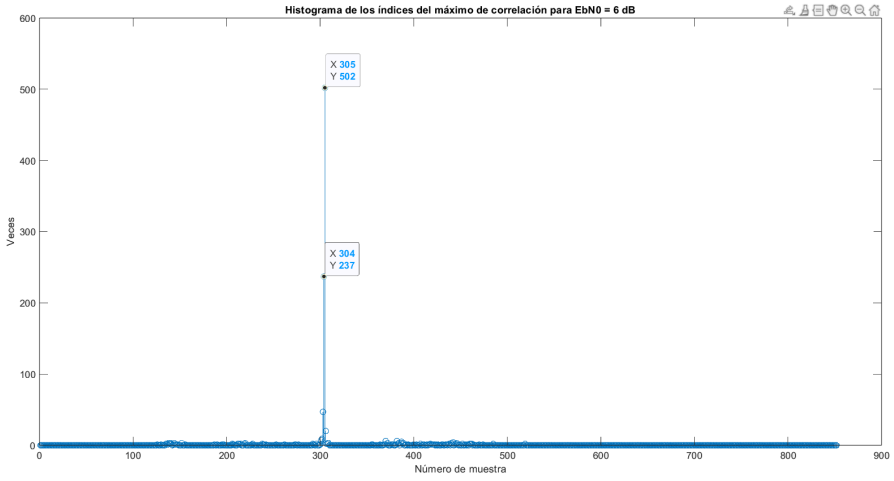

Figura 6.24: Histograma de los índices del máximo de correlación para EbN0 = 6 dB para Código Gold de N = 31 y, señal de unos y ceros alternados al principio y final.

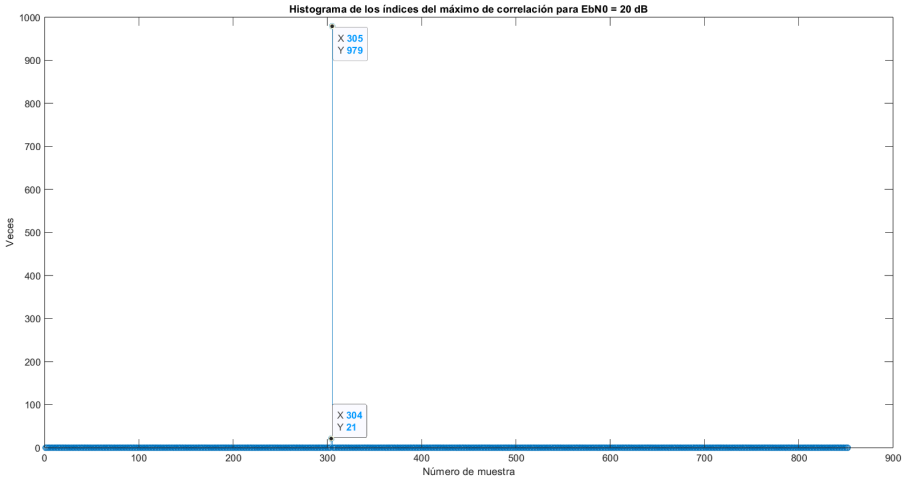

Figura 6.25: Histograma de los índices del máximo de correlación para EbN0 = 20 dB para Código Gold de N = 31 y, señal de unos y ceros alternados al principio y final.

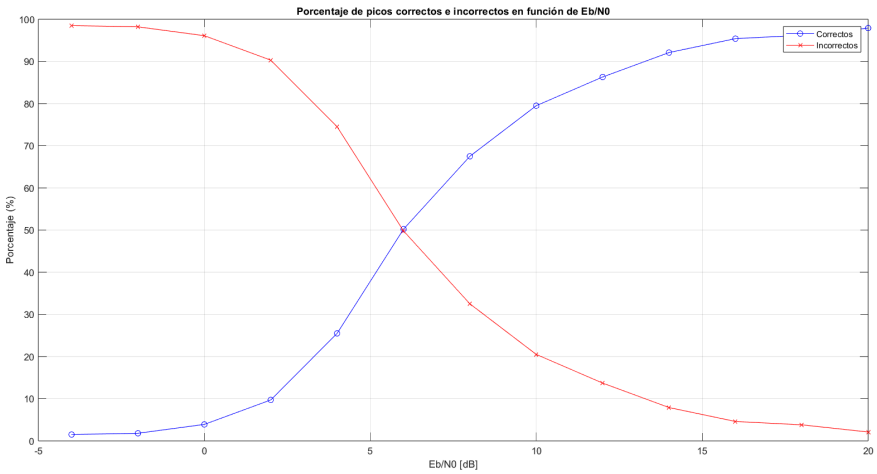

Figura 6.26: Porcentaje de picos de correlación en la muestra 305 en 1000 escenarios para Código Gold de N = 31 y, señal de unos y ceros alternados al principio y final.

Posteriormente se hizo lo mismo para el **Código Gold de longitud 4095**, es decir para un Código Gold modulado de longitud 4095, con doscientos unos y ceros consecutivos modulados y concatenados al inicio de la señal y al final de la señal a transmitir. En la Figura 6.28 se puede observar el histograma que determina la cantidad de veces que se encontró un pico máximo de correlación en la muestra determinada para un E_b/N_0 de 4 dB y, en la Figura 6.29 se puede observar el mismo histograma para E_b/N_0 igual a 20 dB. En éstos se ve que aparece el retardo en un tiempo de muestra. El porcentaje de aciertos y errores en la ubicación del pico de correlación entre el error filtrado y el Código Gold se muestra en la Figura 6.30. Allí se ve que la sincronización para este caso empieza a funcionar correctamente a partir de los 2 dB de E_b/N_0, por lo que observamos un comportamiento similar al caso del Código Gold de misma longitud pero con bits de silencio al principio y final.

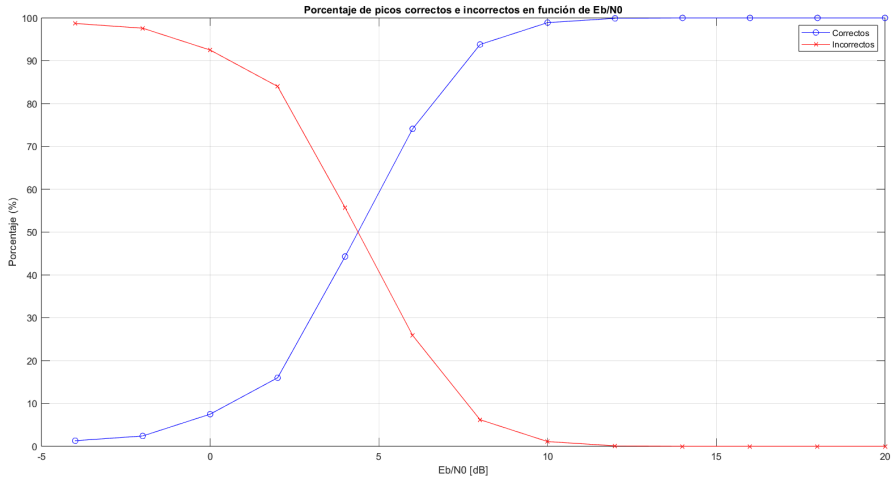

Figura 6.27: Porcentaje de picos de correlación en las muestras 304 y 305 en 1000 escenarios para Código Gold de N = 31 y, señal de unos y ceros alternados al principio y final.

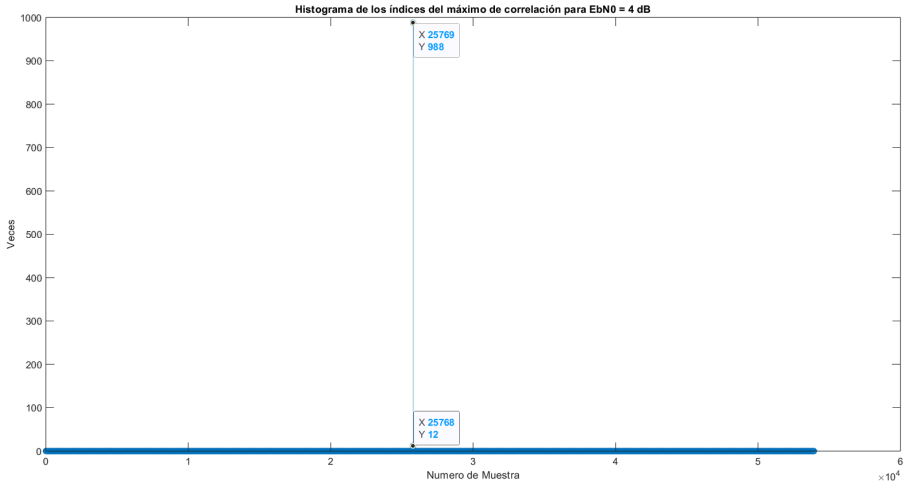

Figura 6.28: Histograma de los índices del máximo de correlación para EbN0 = 4 dB para Código Gold de N = 4095 y, señal de unos y ceros alternados al principio y final.

Cabe aclarar que en todos los casos se utilizó el Error de Fase Filtrado para sincronizar, sin pasar por una etapa de decisión dura en la que se demodula a partir de ésta señal.

Por otra parte, el uso del Error Filtrado trae como ventaja un buen funcionamiento en escenarios con bajos niveles de E_b/N_0 donde la señal es débil dado que, como se observaron en las gráficas, se tiene una correcta sincronización en el caso del Código Gold de longitud 4095 a partir de los 2 dB de E_b/N_0 y que la demodulación según las gráficas de BER observadas a esta tasa de muestreo posee un funcionamiento correcto a partir de los 12 dB (BER = 10^{-2}), lo cual permite una sincronización más robusta frente al caso en el que la demodulación falle.

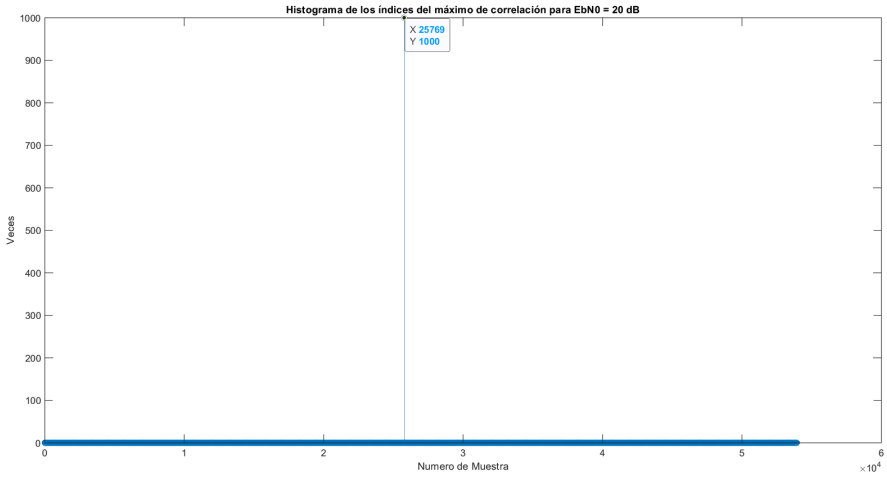

Figura 6.29: Histograma de los índices del máximo de correlación para EbN0 = 20 dB para Código Gold de N = 4095 y, señal de unos y ceros alternados al principio y final.

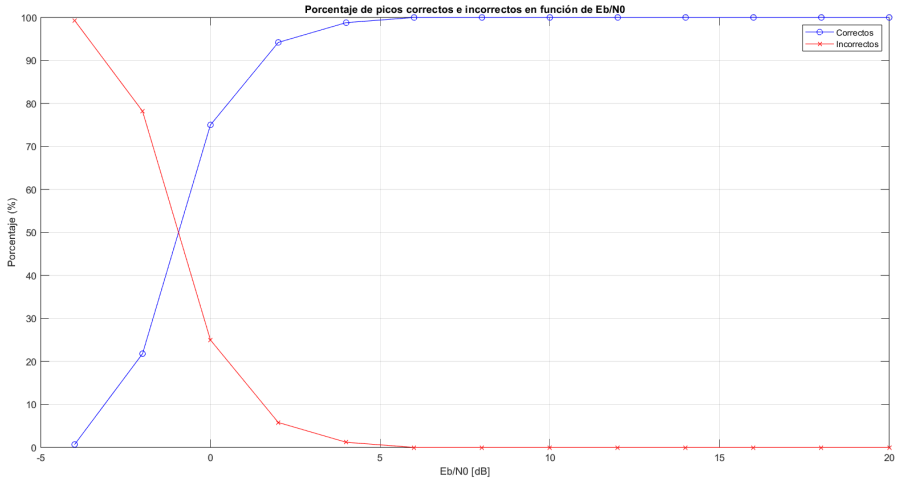

Figura 6.30: Porcentaje de picos de correlación en la muestra 25769 en 1000 escenarios para Código Gold de N = 4095 y, señal de unos y ceros alternados al principio y final.

Capítulo 7

Conclusiones

En este trabajo, partiendo de la motivación de implementación de estaciones receptoras para monitoreo de movimiento animal basadas en el existente sistema ATLAS, nos enfocamos en el análisis exhaustivo de los diferentes métodos de demodulación y sincronismo de señales FSK transmitidas a través de un canal AWGN. Este estudio detallado de las diferentes técnicas, entre las cuales se ahondó particularmente en el Lazo de Seguimiento de Fase (PLL), sienta las bases para la implementación futura de las mencionadas estaciones.

En el contexto del sistema ATLAS y del monitoreo de fauna silvestre, los resultados obtenidos demuestran que la selección del método de demodulación tiene implicaciones cruciales para el rendimiento global del sistema. Se han identificado las siguientes conclusiones principales:

- El demodulador mediante correladores presenta el mejor desempeño, logrando el resultado de la curva teórica de probabilidad de error para modulación BFSK no coherente, pero requiere de un conocimiento previo de las frecuencias de operación.
- Los discriminadores de frecuencia han mostrado una mayor flexibilidad operativa, siendo que no requiere un conocimiento previo de las frecuencias de operación.
- La implementación de lazos de seguimiento de fase (PLL) ha evidenciado un equilibrio favorable entre precisión y adaptabilidad, ofreciendo un rendimiento robusto en presencia de ruido, siendo que no requiere un conocimiento previo de las frecuencias de operación.

Estas conclusiones tienen implicancias significativas para el futuro del monitoreo de fauna silvestre, contribuyendo a:

- Mejorar la precisión en la localización de especies en áreas donde los sistemas GNSS tradicionales presentan limitaciones.

- Reducir los errores de detección y seguimiento, especialmente en condiciones ambientales desafiantes.
- Facilitar la implementación de sistemas de seguimiento más económicos y accesibles para proyectos de conservación.

En lo que respecta al análisis del PLL, los resultados obtenidos validan las decisiones de diseño adoptadas y demuestran la versatilidad y eficacia de este dispositivo, particularmente en su implementación digital como demodulador FSK. Las simulaciones realizadas con el PLL complejo diseñado, han proporcionado resultados notables:

- La elección de un Factor de Amortiguamiento de 2 y un Ancho de Banda de Ruido Equivalente de 1.5 MHz ha demostrado ser óptima para el seguimiento de señales BFSK en las condiciones de operación especificadas.
- El sistema exhibe un rendimiento robusto, manteniendo un rango de enganche de 900 kHz y un rango de separación de 1.21 MHz.
- El diseño implementado muestra una notable capacidad para manejar eficientemente el ruido de fase mientras mantiene inmunidad a las variaciones de amplitud en la señal recibida.
- La respuesta plana del sistema hasta la frecuencia natural de 225 kHz garantiza un procesamiento uniforme de las componentes espectrales relevantes de la señal de Error de Fase.

La implementación del filtro de lazo tipo PI (Proporcional más Integrador) ha demostrado ser una elección acertada, proporcionando el balance necesario entre la respuesta transitoria y el rechazo al ruido.

En cuanto al estudio de los demoduladores FSK en la recepción de señales moduladas con códigos Gold, se han establecido las siguientes conclusiones fundamentales:

- Los códigos Gold demuestran ser una solución efectiva para la medición del TOA, proporcionando secuencias con buenas propiedades de correlación y permitiendo la generación de grandes familias de códigos.
- Los demoduladores PLL-FSK y discriminador de frecuencias muestran un efecto umbral, requiriendo una E_b/N_0 mínima de 12 dB y 14 dB respectivamente, con los parámetros de diseño que se utilizaron, para alcanzar un BER de 10^{-2}.
- El rendimiento de los demoduladores PLL-FSK y del discriminador de frecuencias se deteriora al aumentar la frecuencia de muestreo, mientras que el demodulador por correladores mantiene un rendimiento constante.

Respecto de la sincronización:

- La utilización del error de fase filtrado para la sincronización mediante correlación resulta más robusta que la demodulación directa, especialmente en escenarios de bajo E_b/N_0.
- Los códigos Gold de mayor longitud (4095 bits) proporcionan una sincronización más efectiva que los de menor longitud (31 bits), requiriendo una E_b/N_0 de solo 2 dB para un funcionamiento correcto.
- La inclusión de secuencias de preámbulo con alternancia de bits mejora el rendimiento de la sincronización en comparación con períodos de silencio a medida que los Códigos Gold son de menor longitud, lo cual se debe a que permite un mejor enganche del PLL.

El análisis exhaustivo de los diferentes métodos de demodulación aplicados al sistema ATLAS, la evaluación de demoduladores FSK en la recepción de señales moduladas a través de un canal AWGN a distintos niveles de ruido y el estudio detallado del Lazo de Seguimiento de Fase (PLL) ha proporcionado una comprensión profunda y multifacética de soluciones en el campo del monitoreo de fauna silvestre y las comunicaciones digitales modernas.

Para futuras investigaciones, se recomiendan las siguientes líneas de trabajo:

- Explorar la implementación de estos sistemas en hardware reconfigurable de matriz de compuertas lógicas programables o FPGA.
- Investigar la posibilidad de implementar sistemas híbridos que combinen las ventajas de diferentes métodos de demodulación.
- Investigar técnicas avanzadas de sincronización que puedan mejorar aún más el rendimiento en escenarios de bajo E_b/N_0.
- Desarrollar los algoritmos de triangulación necesarios para estimar la ubicación de los *tags*.

Bibliografía

[1] Weiser, A. W., Orchan, Y., Nathan, R., Charter, M., Weiss, A. J., Toledo, S. Characterizing the Accuracy of a Self-Synchronized Reverse-GPS Wildlife Localization System. En: 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN 2016 - Proceedings. Institute of Electrical and Electronics Engineers Inc., 2016. 7

[2] Beardsworth, C. E., Gobbens, E., van Maarseveen, F., Denissen, B., Dekinga, A., Nathan, R., *et al.* Validating ATLAS: A regional-scale high-throughput tracking system. *Methods in Ecology and Evolution*, **13**, 1990–2004, 9 2022. 7

[3] Bijleveld, A. I., van Maarseveen, F., Denissen, B., Dekinga, A., Penning, E., Ersoy, S., *et al.* WATLAS: high-throughput and real-time tracking of many small birds in the Dutch Wadden Sea. *Animal Biotelemetry*, **10**, 36, 12 2022. 7

[4] Beardsworth, C. E., Whiteside, M. A., Capstick, L. A., Laker, P. R., Langley, E. J. G., Nathan, R., *et al.* Spatial cognitive ability is associated with transitory movement speed but not straightness during the early stages of exploration. *Royal Society Open Science*, **8**, rsos.201758, 3 2021. 7

[5] Toledo, S. A Guide to ATLAS, Vildehaye, and Kamadata, 2023. URL https://groups.google.com/g/atlas-vildehaye-users/. 8

[6] CC1310 CC1310 SimpleLink™ Ultra-Low-Power Sub-1 GHz Wireless MCU, 2015. URL https://www.ti.com/lit/ds/symlink/cc1310.pdf. 8

[7] Leshchenko, A., Toledo, S. Modulation and Signal-Processing Tradeoffs for Reverse-GPS Wildlife Localization Systems. En: 2018 European Navigation Conference (ENC), págs. 154–165. IEEE, 2018. 9, 10

[8] Tomasi, W. Sistemas de Comunicaciones Electrónicas. Pearson Educacioén, 2003. 11, 20

[9] Sklar, B., Ray, P. K. Digital Communications, Fundamentals and Applications, Second Edition. 13, 15

[10] Bria, O. N. John G. Proakis and Dimitris G. Manolakis, Digital signal processing. Principles, algorithms, and aplications, tomo 1. 1999. URL https://journal.info.unlp.edu.ar/JCST/article/view/1034. 21

[11] Hopper, R. RF Sampling: Aliasing Can Be Your Friend, 2023. URL https://www.ti.com/document-viewer/lit/html/SSZTCF7. 21

[12] Rice, M. Digital communications: a discrete-time approach. 2009. 26, 28, 33

[13] Stephens, D. R. Phase-Locked Loops for Wireless Communications. Springer US, 1998. 34

[14] Pseudorandom noise. URL https://en.wikipedia.org/wiki/Pseudorandom_noise. 45

[15] Mathuranathan, V. Wireless Communication Systems in Matlab, Second Edition. 2020. URL https://www.gaussianwaves.com/. 47

[16] Wojuola, O. Cross-correlation index and multiple-access performance of spreading codes for a wireless system. En: 2019 URSI Asia-Pacific Radio Science Conference (AP-RASC), págs. 1–4. IEEE, 2019. 48

[17] Toledo, S. Location Estimation from the Ground Up. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2020. URL https://doi.org/10.1137/1.9781611976298.

[18] Banerjee, D. PLL Performance, Simulation, and Design, 5th Edition, 2017.

[19] Toledo, S., Shohami, D., Schiffner, I., Lourie, E., Orchan, Y., Bartan, Y., *et al.* Cognitive map-based navigation in wild bats revealed by a new high-throughput tracking system, 2020. URL https://www.science.org.

[20] Gupte, P. R., Beardsworth, C. E., Spiegel, O., Lourie, E., Toledo, S., Nathan, R., *et al.* A guide to pre-processing high-throughput animal tracking data. *Journal of Animal Ecology*, **91**, 287–307, 2 2022.

[21] Ingle, V. K., Proakis, J. G. Digital Signal Processing Using MATLAB ® Third Edition. 2010.

[22] Best, R. E. Phase-Locked Loops. McGraw-Hill Companies, Inc., 2007.

[23] Qasim, C. Phase Locked Loop (PLL) in a Software Defined Radio (SDR). URL https://wirelesspi.com/phase-locked-loop-pll-in-a-software-defined-radio-sdr/.

[24] Inc., T. M. Matlab version: 9.13.0 (r2022b), 2022. URL https://www.mathworks.com.

[25] Maximum length sequence. URL https://en.wikipedia.org/wiki/Maximum_length_sequence.

[26] Gold code. URL https://en.wikipedia.org/wiki/Gold_code.

Apéndice A

Código MATLAB

A.1. Modulador BFSK

```
function senial_modulada = Modulador_FSK(senial_mensaje, fc, fdev, fs, Tb)
%% Modulador FSK en MATLAB
% Esta función genera una señal modulada en frecuencia (FSK) a partir de
% una señal de mensaje binaria.
%
% Parámetros de entrada:
% - senial_mensaje: Vector que contiene la señal de mensaje binaria (1 y 0).
% - fc: Frecuencia de la portadora en Hz.
% - fdev: Desviación de frecuencia para la modulación FSK.
% - fs: Frecuencia de muestreo en Hz.
% - Tb: Duración de cada bit en segundos.
%
% Parámetro de salida:
% - senial_modulada: Señal FSK modulada con las características
% especificadas.

%% Constantes
Ts = 1/fs;                    % Período de muestreo
varianza_fase = 1e-9;          % Ruido de fase introducido a la señal

%% Tiempo para la señal modulada
t = 0:Ts:(length(senial_mensaje))*Tb - Ts;  % Vector de tiempo con duración
% proporcional al mensaje
%% Generación de la señal portadora
fase_inicial = pi/4;                        % Fase inicial de la portadora
```

```
ruido_fase = sqrt(varianza_fase) * randn(1, length(t));  % Ruido de fase
% agregado
%% Mapeo de la señal de mensaje binaria
% Los valores de 0 en el mensaje se mapean a -1 para representar cambios en
% la frecuencia
mensaje_mapeado = senial_mensaje;
mensaje_mapeado(mensaje_mapeado == 0) = -1;

%% Modulación de la señal FSK
L = Tb * fs;  % Número de muestras por cada bit
coeficiente = kron(mensaje_mapeado, ones(1, L));  % Repetición de cada bit
% para ajustarlo al tiempo
senial_modulada = exp(1j * (2 * pi * (fc + coeficiente * fdev) .* t +
fase_inicial + ruido_fase));  % Señal FSK modulada
end
```

A.2. Demodulador PLL-FSK

```
function [salida_dds, error_filtrado, senial_demodulada] = ...
Demodulador_FSK_PLL(senial_modulada, fvco, factor_amortiguamiento,
ancho_banda, fs, Tb)
% Demodulación FSK utilizando un PLL (Phase-Locked Loop)
% senial_modulada - Señal modulada en FSK que se recibe
% fvco - Frecuencia del Frecuencia del sintetizador digital
% directo (DDS) en Hz
% factor_amortiguamiento - Factor de amortiguamiento (Damping Factor)
% del PLL
% ancho_banda - Ancho de banda del PLL en Hz
% fs - Frecuencia de muestreo en Hz
% Tb - Duración de cada bit en segundos
% salida_dds - Señal de Salida del DDS
% error_filtrado - Error de fase filtrado
% senial_demodulada - Bits de datos después de la demodulación

%% Constantes del PLL
theta_n = ancho_banda / (fs * (factor_amortiguamiento + 1/(4 * ...
factor_amortiguamiento))); % Ancho de banda normalizado del PLL
```

```
Kp = 4 * factor_amortiguamiento * theta_n / (1 + 2 * ...
factor_amortiguamiento * theta_n + theta_n^2);  % Ganancia proporcional
Ki = 4 * theta_n^2 / (1 + 2 * factor_amortiguamiento * theta_n + ...
theta_n^2);  % Ganancia integradora

%% Preinicializar vectores
salida_proporcional = 0;  % Salida de la rama proporcional del filtro
% de lazo
salida_integrador = 0;     % Salida de la rama integradora del filtro
% de lazo
error_fase = zeros(1, length(senial_modulada)); % Estimación de error
% de fase
error_filtrado = zeros(1, length(senial_modulada)); % Salida del error
% de fase filtrado
dds_arg = zeros(1, length(senial_modulada)); % Salida del filtro de lazo
salida_dds = zeros(1, length(senial_modulada)); % Salida del DDS

%% Demodulación FSK
senial_demodulada = [];

%% PLL (Phase-Locked Loop)
for i = 1:length(senial_modulada)
%% Salida DDS
salida_dds(i) = exp(1j * dds_arg(i));  % Salida del oscilador controlado
% por fase

%% Detector de Fase
prod_dds = senial_modulada(i) * conj(salida_dds(i));  % Producto entre
% señal modulada y
% el conjugado
% de la salida DDS
error_fase(i) = angle(prod_dds);  % Error de fase

%% Filtro del bucle
salida_proporcional = Kp * error_fase(i);  % Componente proporcional del
% filtro
salida_integrador = Ki * error_fase(i) + salida_integrador; % Componente
% integradora
% del filtro
```

```
error_filtrado(i) = salida_proporcional + salida_integrador;  % Error de
% fase
% filtrado

%% DDS (Direct Digital Synthesis)
dds_arg(i+1) = error_filtrado(i) + 2 * pi * (fvco / fs) + dds_arg(i);
% Actualización del argumento DDS

end

%% Obtener los bits demodulados
mean_val = mean(error_filtrado);  % Calcular la media del error filtrado
paso = Tb * fs;  % Número de muestras por bit

indice_muestra_inicio = 1;  % Índice de inicio para cada bit
indice_muestra_fin = paso;  % Índice de fin para cada bit

while indice_muestra_fin <= length(senial_modulada)
mean_bit = mean(error_filtrado(indice_muestra_inicio:indice_muestra_fin));
% Media del error filtrado en el segmento actual

if mean_bit > mean_val
senial_demodulada(end+1) = 1;  % Asignar bit 1 si la media del segmento
% es mayor que la media global
else
senial_demodulada(end+1) = 0;  % Asignar bit 0 si la media del segmento
% es menor o igual a la media global
end

indice_muestra_inicio = indice_muestra_inicio + paso;  % Avanzar el índice
% de inicio al
% siguiente segmento
% de bit
indice_muestra_fin = indice_muestra_fin + paso;  % Avanzar el índice de
% fin al siguiente segmento
% de bit
end
error_filtrado = error_filtrado - mean_val;  % Ajustar el error filtrado
% restando la media global
```

```
end
```

A.3. Demodulador Mediante Discriminador de Frecuencias

```
function [senial_demodulada] = Demodulador_FSK_Discriminador( ...
senial_modulada, fs, Tb)
% DEMODULADOR_FSK_DISCRIMINADOR: Demodulación de una señal FSK
% utilizando un discriminador de frecuencias.
%
% Entradas:
% senial_modulada - Señal modulada en FSK recibida (vector complejo)
% fs              - Frecuencia de muestreo en Hz
% Tb              - Duración de cada bit en segundos
%
% Salida:
% senial_demodulada - Vector binario con los bits de datos después
% de la demodulación

% Limitador en amplitud de la señal recibida
senial_modulada = exp(1i * angle(senial_modulada));

% Derivada discreta de la fase para estimar la frecuencia instantánea
xd = [0, diff(senial_modulada) * fs];

% Obtener la envolvente de la señal derivada
xd = abs(xd);

% Restar la media y normalizar la señal
yd = xd - mean(xd);
yd = yd / max(abs(yd));
mean_val = mean(yd);

%% Filtro pasa-bajos
Nyquist_freq = fs / 2;
cutoff_freq = 1.5e6;
normalized_cutoff = cutoff_freq / Nyquist_freq;
```

```
filter_order = 100;
b = fir1(filter_order, normalized_cutoff);

% Aplicar el filtro FIR a la señal
senial_filtrada = filtfilt(b, 1, yd);
yd = senial_filtrada;

%% Demodulación de la señal
senial_demodulada = [];
indice_muestra_inicio = 1;
indice_muestra_fin = Tb * fs;

% Iterar sobre segmentos de la señal de acuerdo a la duración de
% cada bit
while indice_muestra_fin <= length(yd)
% Calcular la media del segmento actual
mean_bit = mean(yd(indice_muestra_inicio:indice_muestra_fin));

% Decisión de bit: si la media es mayor que el valor medio, se
% asigna un 1, de lo contrario, un 0
if mean_bit > mean_val
senial_demodulada(end+1) = 1;
else
senial_demodulada(end+1) = 0;
end

% Avanzar a los siguientes índices para el próximo bit
indice_muestra_inicio = indice_muestra_inicio + fs * Tb;
indice_muestra_fin = indice_muestra_fin + fs * Tb;
end
end
```

A.4. Demodulación mediante Correladores

```
function senial_demodulada = Demodulador_FSK_Cuadratura( ...
senial_modulada, fc, fdev, Tb, fs)
% Demodulación no coherente de una señal modulada en BFSK
% senial_modulada - Señal modulada en BFSK en el receptor
```

```
% fc - Frecuencia central de la portadora en Hertz
% fdev - Desviación de frecuencias desde Fc
% Tb - Duración de cada bit en segundos
% fs - Frecuencia de muestreo para simulación en tiempo discreto
% senial_demodulada - Bits de datos después de la demodulación

Ts = 1/fs;  % Período de muestreo
t = 0:Ts:(length(senial_modulada)-1)*Ts;  % Base de tiempo
F1 = (fc + fdev);  % Frecuencia F1
F2 = (fc - fdev);  % Frecuencia F2
L = Tb * fs;  % Número de muestras por bit

% Definir dos funciones base
p1 = 1 * exp(-1j * (2 * pi * F1 * t));  % Onda portadora para F1
p2 = 1 * exp(-1j * (2 * pi * F2 * t));  % Onda portadora para F2

% Multiplicar e integrar desde 0 hasta Tb
r1 = conv(senial_modulada .* p1, ones(1, L));  % Correlación con p1
r2 = conv(senial_modulada .* p2, ones(1, L));  % Correlación con p2

% Muestrear en cada instante de muestreo
r1 = r1(L:L:end);
r2 = r2(L:L:end);

% Cálculo de la energía
x = real(r1).^2 + imag(r1).^2;
y = real(r2).^2 + imag(r2).^2;

% Comparar y decidir
senial_demodulada = (x - y) > 0;
end
```

A.5. BER PLL-FSK

```
function BER_simulado_PLL = BER_PLL( ...
senial_mensaje, senial_modulada, fs, df, fvco, Bn, Tb, EbN0_dB)
% BER_PLL: Simulación de la Tasa de Error de Bit (BER) utilizando un
```

```
% demodulador FSK con PLL.
%
% Entradas:
% senial_mensaje   - Señal original de mensaje (vector binario)
% senial_modulada  - Señal modulada en FSK (vector complejo)
% fs               - Frecuencia de muestreo en Hz
% df               - Desviación de frecuencia en Hz
% fvco             - Frecuencia del sintetizador digital directo
%                    (DDS) en Hz
% Bn               - Ancho de Banda del PLL
% Tb               - Duración de cada bit en segundos
% EbN0_dB          - Relación de energía por bit a ruido espectral
%                    (en dB)
%
% Salida:
% BER_simulado_PLL - Vector con la tasa de error de bit simulada
%                    para cada valor de $E_b/N_0$

delay_bits = 0.2 * length(senial_mensaje);
% Ignora el primer 20% para que el PLL logré engancharse
% Conversión de $E_b/N_0$ de dB a lineal
EbN0 = 10 .^ (EbN0_dB / 10);
BER_simulado_PLL = zeros(size(EbN0_dB));

% Cálculo de la energía por bit
Eb = (Tb * fs) * sum(abs(senial_modulada).^2) / ...
length(senial_modulada);

% Cálculo de BER para cada valor de $E_b/N_0$
for k = 1:length(EbN0_dB)
N0 = Eb / EbN0(k);
ruido = sqrt(N0 / 2) * (randn(1, length(senial_modulada)) + ...
1i * randn(1, length(senial_modulada)));

% Señal de entrada con ruido añadido
senial_entrada = senial_modulada + ruido;

% Demodulación utilizando el demodulador PLL
[~, ~, senial_demodulada_PLL] = ...
```

```
Demodulador_FSK_PLL(senial_entrada, fvco, df, Bn, fs, Tb);

% Cálculo de la tasa de error de bit (BER)
BER_simulado_PLL(k) = sum(senial_mensaje(delay_bits:end) ~= ...
senial_demodulada_PLL(delay_bits:end)) / ...
(length(senial_mensaje) - delay_bits);
end
end
```

A.6. BER Discriminador

```
function BER_simulado_discriminador = BER_Discriminador( ...
senial_mensaje, senial_modulada, Tb, fs, EbN0_dB)
% BER_DISCRIMINADOR: Simulación de la Tasa de Error de Bit (BER)
% utilizando un demodulador FSK.
%
% Entradas:
% senial_mensaje   - Señal original de mensaje (vector binario)
% senial_modulada  - Señal modulada en FSK (vector complejo)
% Tb               - Duración de cada bit en segundos
% fs               - Frecuencia de muestreo en Hz
% EbN0_dB         - Relación de energía por bit a ruido espectral
%                    (en dB)
%
% Salida:
% BER_simulado_discriminador - Vector con la tasa de error de bit
%                              simulada para cada valor de $E_b/N_0$

% Cálculo del retraso inicial para los bits
delay_bits = 0.2 * length(senial_mensaje);  % Ignora el primer 20%

% Conversión de $E_b/N_0$ de dB a lineal
EbN0 = 10 .^ (EbN0_dB / 10);
BER_simulado_discriminador = zeros(size(EbN0_dB));

% Cálculo de la energía por bit
Eb = (Tb * fs) * sum(abs(senial_modulada).^2) / ...
```

```
length(senial_modulada);

% Cálculo de BER para cada valor de $E_b/N_0$
for k = 1:length(EbN0_dB)
N0 = Eb / EbN0(k);

% Generación de ruido AWGN
ruido = sqrt(N0 / 2) * (randn(1, length(senial_modulada)) + ...
1i * randn(1, length(senial_modulada)));

% Señal de entrada con ruido añadido
senial_entrada = senial_modulada + ruido;

% Demodulación utilizando el demodulador FSK
senial_demodulada_discriminador = ...
Demodulador_FSK_Discriminador_2(senial_entrada, fs, Tb);

% Cálculo de la tasa de error de bit (BER)
BER_simulado_discriminador(k) = sum(senial_mensaje(delay_bits:end) ...
~= senial_demodulada_discriminador(delay_bits:end)) / ...
(length(senial_mensaje) - delay_bits);
end
end
```

A.7. BER Correladores

```
function BER_simulado_cuadratura = BER_Cuadratura(senial_mensaje, ...
senial_modulada, fc, fdev, Tb, fs, EbN0_dB)
% BER_CUADRATURA: Simulación de la Tasa de Error de Bit (BER)
% utilizando un demodulador FSK en cuadratura.
%
% Entradas:
% senial_mensaje   - Señal original de mensaje (vector binario)
% senial_modulada  - Señal modulada en FSK (vector complejo)
% fc               - Frecuencia portadora en Hz
% fdev             - Desviación de frecuencia en Hz
% Tb               - Duración de cada bit en segundos
```

```
% fs                - Frecuencia de muestreo en Hz
% EbN0_dB          - Relación de energía por bit a ruido espectral
%                    (en dB)
%
% Salida:
% BER_simulado_cuadratura - Vector con la tasa de error de bit
%                           simulada para cada valor de $E_b/N_0$


delay_bits = 0.2 * length(senial_mensaje);  % Ignora el primer 20%

% Conversión de $E_b/N_0$ de dB a lineal
EbN0 = 10 .^ (EbN0_dB / 10);
BER_simulado_cuadratura = zeros(size(EbN0_dB));

% Cálculo de la energía por bit
Eb = (Tb * fs) * sum(abs(senial_modulada).^2) / ...
length(senial_modulada);

% Cálculo de BER para cada valor de $E_b/N_0$
for k = 1:length(EbN0_dB)
N0 = Eb / EbN0(k);

% Generación de ruido AWGN
ruido = sqrt(N0 / 2) * (randn(1, length(senial_modulada)) + ...
1i * randn(1, length(senial_modulada)));

% Señal de entrada con ruido añadido
senial_entrada = senial_modulada + ruido;

% Demodulación utilizando el demodulador en cuadratura
senial_demodulada_cuadratura = ...
Demodulador_FSK_Cuadratura(senial_entrada, fc, fdev, Tb, fs);

% Cálculo de la tasa de error de bit (BER)
BER_simulado_cuadratura(k) = sum(senial_mensaje(delay_bits:end) ...
~= senial_demodulada_cuadratura(delay_bits:end)) / ...
(length(senial_mensaje) - delay_bits);
end
```

```
end
```

A.8. Estimación de BER para los tres Demoduladores

```
%% Inicialización de Parámetros
fc = 434e6;      % Frecuencia portadora en Hz (434 MHz)
fdev = 500e3;    % Desviación máxima de frecuencia en Hz (500 kHz),
% utilizada para la modulación FSK
df = 2;          % Factor de amortiguamiento del PLL

Bn = 1.5e6;      % Ancho de banda del PLL en Hz (1.5 MHz)

fs1 = 6e6;       % Frecuencia de muestreo 1 en Hz (6 MHz)

fvco1 = 2e6;     % Frecuencia del DDS en Hz (2 MHz)
num_bits = 1000; % Número de bits a simular en cada mensaje transmitido
NminErr = 100;   % Número mínimo de errores requeridos antes de detener la
% simulación para un valor de $E_b/N_0$

Tb = 1e-6;       % Tiempo de bit en segundos (1 microsegundo)

%% Rango de $E_b/N_0$ en dB
EbN0_dB = 0:4:12;          % Rango de $E_b/N_0$ en dB
EbN0 = 10.^(EbN0_dB/10);   % Conversión del $E_b/N_0$ de dB a valor lineal

%% Cálculo de BER Teórico para BFSK en canal AWGN
BER_teorico = 0.5 * exp(-EbN0 / 2);   % Fórmula de BER teórico para BFSK

%% Inicialización de matrices para almacenar los BER simulados
BER_simulado_PLL_1 = zeros(size(EbN0_dB));
BER_simulado_cuadratura_1 = zeros(size(EbN0_dB));
BER_simulado_discriminador_1 = zeros(size(EbN0_dB));

nErrTot_PLL = zeros(size(EbN0_dB));
```

```
nErrTot_cuadratura = zeros(size(EbN0_dB));
nErrTot_discriminador = zeros(size(EbN0_dB));

nTot_PLL = zeros(size(EbN0_dB));
nTot_Cuadratura = zeros(size(EbN0_dB));
nTot_Discriminador = zeros(size(EbN0_dB));

%% Simulación para cada valor de $E_b/N_0$
for k = 1:length(EbN0_dB)
EbN0_actual = EbN0(k);

% Generación de la señal mensaje inicial (secuencia binaria aleatoria)
senial_mensaje = randi([0,1], 1, num_bits);

% Modulación FSK de la señal
senial_modulada = Modulador_FSK(senial_mensaje, fc, fdev, fs1, Tb);

% Contadores de ciclos para cada método de demodulación
count_PLL = 0;
count_Disc = 0;
count_Cuad = 0;

% Realizar simulación hasta obtener NminErr errores para cada método
while nErrTot_PLL(k) < NminErr || ...
nErrTot_discriminador(k) < NminErr || ...
nErrTot_cuadratura(k) < NminErr

if nErrTot_PLL(k) < NminErr
if mod(count_PLL, 10) == 0
senial_mensaje = randi([0,1], 1, num_bits);
senial_modulada = Modulador_FSK(senial_mensaje, fc, fdev, ...
fs1, Tb);
end

% Cálculo de BER simulado para PLL
BER_simulado_PLL = BER_PLL(senial_mensaje, senial_modulada, ...
fs1, df, fvco1, Bn, Tb, EbN0_dB(k));
nErrTot_PLL(k) = nErrTot_PLL(k) + BER_simulado_PLL * num_bits;
nTot_PLL(k) = nTot_PLL(k) + num_bits;
```

```
count_PLL = count_PLL + 1;
end

if nErrTot_discriminador(k) < NminErr
if mod(count_Disc, 10) == 0
senial_mensaje = randi([0,1], 1, num_bits);
senial_modulada = Modulador_FSK(senial_mensaje, fc, fdev, ...
fs1, Tb);
end

% Cálculo de BER simulado para el discriminador
BER_simulado_discriminador = BER_Discriminador(senial_mensaje, ...
senial_modulada, Tb, fs1, ...
EbN0_dB(k));
nErrTot_discriminador(k) = nErrTot_discriminador(k) + ...
BER_simulado_discriminador * ...
num_bits;
nTot_Discriminador(k) = nTot_Discriminador(k) + num_bits;

count_Disc = count_Disc + 1;
end

if nErrTot_cuadratura(k) < NminErr
if mod(count_Cuad, 10) == 0
senial_mensaje = randi([0,1], 1, num_bits);
senial_modulada = Modulador_FSK(senial_mensaje, fc, fdev, ...
fs1, Tb);
end

% Cálculo de BER simulado para demodulador en cuadratura
BER_simulado_cuadratura = BER_Cuadratura(senial_mensaje, ...
senial_modulada, fc, fdev, Tb, ...
fs1, EbN0_dB(k));
nErrTot_cuadratura(k) = nErrTot_cuadratura(k) + ...
BER_simulado_cuadratura * num_bits;
nTot_Cuadratura(k) = nTot_Cuadratura(k) + num_bits;

count_Cuad = count_Cuad + 1;
```

```
end
end

% Cálculo final del BER para cada método en este valor de $E_b/N_0$
BER_simulado_PLL_1(k) = nErrTot_PLL(k) / nTot_PLL(k);
BER_simulado_cuadratura_1(k) = nErrTot_cuadratura(k) / nTot_Cuadratura(k);
BER_simulado_discriminador_1(k) = nErrTot_discriminador(k) / ...
nTot_Discriminador(k);
end

%% Gráfica: SNR vs Probabilidad de Error
figure(1);
semilogy(EbN0_dB, BER_teorico, 'b-', 'LineWidth', 2);
hold on;
semilogy(EbN0_dB, BER_simulado_PLL_1, 'r-', 'LineWidth', 2);
semilogy(EbN0_dB, BER_simulado_discriminador_1, 'm--', 'LineWidth', 2);
semilogy(EbN0_dB, BER_simulado_cuadratura_1, 'g-x', 'LineWidth', 2);
hold off;

legend('BER Teórico', 'BER Simulado PLL', 'BER Simulado Discriminador', ...
'BER Simulado Cuadratura');
xlabel('$E_b/N_0$ (dB)');
ylabel('Probabilidad de Error de Bit');
title(['BER para BFSK con canal AWGN. PLL con BW = ', ...
num2str(Bn/1e6), ' MHz']);
grid on;
```

A.9. Espectros de la primera zona de Nyquist para distintas f_s

```
%% Inicialización de Parámetros
fc = 434e6; % Frecuencia de portadora de la señal FSK (434 MHz)
fdev = 500e3; % Desviación máxima de frecuencia para la modulación FSK
num_bits = 1e3; % Número de bits a transmitir en cada realización
num_realizaciones = 1e4; % Número de realizaciones para promedio
```

```
% Vector con diferentes frecuencias de muestreo (en Hz)
frecuencias_muestreo = [4 5 6 7 8 21 30 60] * 1e6;

EbN0_dB = 10;
EbN0 = 10^(EbN0_dB/10); % Conversión de dB a valor lineal

%% Inicialización de Resultados
% Inicialización de variables para almacenar resultados del espectro
resultados_spectro = cell(length(frecuencias_muestreo), 1);
resultados_ruido = cell(length(frecuencias_muestreo), 1);
frecuencias = cell(length(frecuencias_muestreo), 1);

% Bucle a través de cada frecuencia de muestreo
for idx_fs = 1:length(frecuencias_muestreo)
fs = frecuencias_muestreo(idx_fs); % Frecuencia de muestreo actual
Ts = 1/fs; % Período de muestreo correspondiente

% Definición del tiempo de bit (1 µs)
Tb = 1e-6;
% Definición del tiempo total de simulación basado en número de bits
tiempo_total = 0:Ts:(num_bits)*Tb-Ts;

% Inicialización de acumuladores de espectro modulado y de ruido
espectro_modulado = zeros(1, length(tiempo_total));
espectro_ruido = zeros(1, length(tiempo_total));

tic % Inicia el cronómetro

% Bucle de realizaciones para promediar resultados
for realizacion = 1:num_realizaciones
% Generación de bits aleatorios (0s y 1s)
bits_aleatorios = randi([0, 1], 1, num_bits);

% Modulación FSK de los bits aleatorios
senial_modulada = Modulador_FSK(bits_aleatorios, fc, ...
fdev, fs, Tb);

% Cálculo de la potencia de la señal modulada
```

```
potencia_senial = sum(abs(senial_modulada).^2) / ...
numel(senial_modulada);

% Generación de ruido Gaussiano blanco complejo (AWGN)
ruido = sqrt(Tb / Ts) * sqrt(potencia_senial / EbN0) * ...
(randn(1, numel(senial_modulada)) + ...
1j * randn(1, numel(senial_modulada))) / sqrt(2);

% Acumulación del espectro de la señal modulada y del ruido
espectro_modulado = espectro_modulado + ...
abs(fftshift(fft(senial_modulada))).^2;
espectro_ruido = espectro_ruido + abs(fftshift(fft(ruido))).^2;
end

toc % Detiene el cronómetro

% Promedio del espectro acumulado de la señal modulada y del ruido
resultados_spectro{idx_fs} = espectro_modulado / num_realizaciones;
resultados_ruido{idx_fs} = espectro_ruido / num_realizaciones;

% Definición del vector de frecuencias para el gráfico
frecuencias{idx_fs} = [-1/Ts/2:1/Ts/length(resultados_spectro{idx_fs}): ...
1/Ts/2];
frecuencias{idx_fs} = frecuencias{idx_fs}(1:end-1);
end

for idx_fs = 1:length(frecuencias_muestreo)
if mod(idx_fs, 4) == 1
figure('units', 'normalized', 'outerposition', [0 0 1 1])
end

subplot(2, 2, mod(idx_fs - 1, 4) + 1)
plot(frecuencias{idx_fs}, resultados_spectro{idx_fs}, ...
frecuencias{idx_fs}, resultados_ruido{idx_fs}, 'LineWidth', 2)
xlabel('Frecuencias (Hz)')
title(['Espectro para fs = ', num2str(frecuencias_muestreo(idx_fs) / ...
1e6), ' MHz']);

% Ajustar el rango de frecuencias para fs específicos
```

```
if idx_fs == 4
xlim([-3.5e6, 3.5e6]); % Limita el eje X para fs de 8 MHz
end
if idx_fs == 6
xlim([-10.5e6, 10.5e6]); % Limita el eje X para fs de 21 MHz
end
end
```

A.10. Espectros Error de Fase para distintas f_s

```
%% Inicialización de Parámetros
fc = 434e6; % Frecuencia de la portadora en Hz (434 MHz).
fdev = 500e3; % Desviación máxima de frecuencia en Hz (500 kHz).
df = 2; % Factor de amortiguamiento del PLL.
Bn = 1.5e6; % Ancho de banda del PLL en Hz (1.5 MHz).
num_bits = 1e3; % Número de bits a transmitir.
num_realizaciones = 1e4; % Número de realizaciones para promediar.

frecuencias_muestreo = [4 5 6 7 8 21 30 60] * 1e6; % Vector de frecuencias
EbN0_dB = 10; % Relación $E_b/N_0$ en dB (10 dB).
EbN0 = 10^(EbN0_dB/10); % Conversión de $E_b/N_0$ de dB a valor lineal.

%% Inicialización de Resultados
% Se crean celdas para almacenar los resultados de los espectros.
resultados_spectro = cell(length(frecuencias_muestreo), 1);
resultados_ruido = cell(length(frecuencias_muestreo), 1);
resultados_error_fase = cell(length(frecuencias_muestreo), 1);
resultados_error_filtrado = cell(length(frecuencias_muestreo), 1);
frecuencias = cell(length(frecuencias_muestreo), 1); % Frecuencias para
% graficar.

% Iteración sobre las frecuencias de muestreo
for idx_fs = 1:length(frecuencias_muestreo)
fs = frecuencias_muestreo(idx_fs); % Frecuencia de muestreo actual.
Ts = 1/fs; % Periodo de muestreo (inverso de la frecuencia de muestreo).
```

```
% Tiempo de bit y tiempo total
Tb = 1e-6; % Tiempo de bit (1 microsegundo).
tiempo_total = 0:Ts:(num_bits)*Tb-Ts; % Vector de tiempo de transmisión.

% Inicialización de variables de espectro.
espectro_modulado = zeros(1, length(tiempo_total));
espectro_ruido = zeros(1, length(tiempo_total));
espectro_error_fase = zeros(1, length(tiempo_total));
espectro_error_filtrado = zeros(1, length(tiempo_total));

tic  % Inicio del temporizador.

for realizacion = 1:num_realizaciones
% Generación de bits aleatorios (0 o 1).
bits_aleatorios = randi([0, 1], 1, num_bits);

% Modulación FSK
senial_modulada = Modulador_FSK(bits_aleatorios, fc, ...
fdev, fs, Tb); % Modulación de bits en FSK.

% Cálculo de la potencia de la señal modulada.
potencia_senial = sum(abs(senial_modulada).^2) / ...
numel(senial_modulada);

% Generación de ruido aditivo blanco gaussiano (AWGN).
ruido = sqrt(Tb / Ts) * sqrt(potencia_senial / EbN0) * ...
(randn(1, numel(senial_modulada)) + 1j * ...
randn(1, numel(senial_modulada))) / sqrt(2);

% Demodulación FSK usando un PLL
[~, error_fase, error_filtrado, ~] = ...
Demodulador_FSK_PLL(senial_modulada, fc, df, Bn, ...
fs, Tb);

% Acumula los espectros de la señal modulada, ruido, y errores.
espectro_modulado = espectro_modulado + abs(fftshift(fft( ...
senial_modulada))).^2;
espectro_ruido = espectro_ruido + abs(fftshift(fft(ruido))).^2;
espectro_error_fase = espectro_error_fase + abs(fftshift(fft( ...
```

```
error_fase))).^2;
espectro_error_filtrado = espectro_error_filtrado + abs( ...
fftshift(fft(error_filtrado))).^2;
end

toc  % Fin del temporizador.

% Promedio de los espectros acumulados a través de realizaciones.
resultados_spectro{idx_fs} = espectro_modulado / num_realizaciones;
resultados_ruido{idx_fs} = espectro_ruido / num_realizaciones;
resultados_error_fase{idx_fs} = espectro_error_fase / ...
num_realizaciones;
resultados_error_filtrado{idx_fs} = espectro_error_filtrado / ...
num_realizaciones;

% Definición de frecuencias para el gráfico (espectro).
frecuencias{idx_fs} = [-1/Ts/2:1/Ts/length(resultados_spectro{ ...
 idx_fs}):1/Ts/2]; % Vector de frecuencias.
frecuencias{idx_fs} = frecuencias{idx_fs}(1:end-1); % Ajuste de long.
end

%% Gráfico de Error de Fase y Error Filtrado
for idx_fs = 1:length(frecuencias_muestreo)
% Cada 4 frecuencias, se genera una nueva figura de gráficos.
if mod(idx_fs, 4) == 1
figure('units', 'normalized', 'outerposition', [0 0 1 1])
end

% Gráfico de errores de fase y error filtrado.
subplot(2, 2, mod(idx_fs - 1, 4) + 1)
plot(frecuencias{idx_fs}, resultados_error_fase{idx_fs}, '--', ...
'LineWidth', 2) % Espectro del error de fase.
hold on
plot(frecuencias{idx_fs}, resultados_error_filtrado{idx_fs}, ':', ...
'LineWidth', 2) % Espectro del error filtrado.
xlabel('Frecuencias (Hz)') % Etiqueta del eje x.
title(['Errores para fs = ', num2str(frecuencias_muestreo(idx_fs) ...
/ 1e6), ' MHz']); % Título del gráfico.
xlim([-2e6,2e6]) % Limita el eje x entre -2 MHz y 2 MHz.
```

```
ylim([0 1.5e4]) % Limita el eje y entre 0 y 15000.
legend('Error Fase', 'Error Filtrado') % Añade leyenda.
end
```

A.11. Respuestas en Frecuencias para distintas Frecuencias de Muestreo

```
% Parámetros de entrada
fc = 434e6;                 % Frecuencia de la portadora en Hz (434 MHz).
fdev = 500e3;               % Desviación de frecuencia máxima en Hz.
factor_amortiguamiento = 2; % Factor de amortiguamiento para el PLL.
ancho_banda = 1.5e6;        % Ancho de banda del PLL en Hz (1.5 MHz).
Tb = 1e-6;                  % Duración de un bit en segundos (1 microsegundo).
num_bits = 1e3;             % Número de bits a transmitir.
frecuencias_muestreo = [4 5 6 7 8 21 30 60] * 1e6;  % Frecuencias de muestreo.



% Prealocación de matrices para almacenar respuestas y valores de Kp y Ki
H_z_all = [];           % Matriz para almacenar las respuestas H(z).
f_Hz_all = [];          % Matriz para almacenar las frecuencias H(z).
kp_all = zeros(1, length(frecuencias_muestreo));  % Vector Kp.
ki_all = zeros(1, length(frecuencias_muestreo));  % Vector Ki.

% Bucle sobre cada frecuencia de muestreo
for i = 1:length(frecuencias_muestreo)
fs = frecuencias_muestreo(i);  % Frecuencia de muestreo actual.

% Modular la señal en FSK
senial_modulada = Modulador_FSK(senial_mensaje, fc, fdev, fs, Tb);

% Calcular constantes del PLL
theta_n = ancho_banda / (fs * (factor_amortiguamiento + ...
1/(4 * factor_amortiguamiento)));
Kp = 4 * factor_amortiguamiento * theta_n / ...
(1 + 2 * factor_amortiguamiento * theta_n + theta_n^2);
```

```
Ki = 4 * theta_n^2 / ...
(1 + 2 * factor_amortiguamiento * theta_n + theta_n^2);

% Almacenar valores de Kp y Ki
kp_all(i) = Kp;      % Guarda Kp.
ki_all(i) = Ki;      % Guarda Ki.

% wn
wn = theta_n * fs;   % Frecuencia natural del PLL.

% Coeficientes del numerador y denominador de H(z)
numerador_Hz = [0, (Kp + Ki), -1 * Kp];
denominador_Hz = [1, -2 * (1 - (1/2) * (Kp + Ki)), (1 - Kp)];

% Calcular respuesta en frecuencia
[H_z, f_Hz] = freqz(numerador_Hz, denominador_Hz, 1024, fs);

% Almacenar los resultados
H_z_all = [H_z_all, 20 * log10(abs(H_z))];
f_Hz_all = [f_Hz_all, f_Hz];
end

% Calculo de la frecuencia natural fn
wn = 2 * ancho_banda / ...
(factor_amortiguamiento + 1 / (4 * factor_amortiguamiento));
fn = wn / (2 * pi);   % Frecuencia fn en Hz.

% Graficar todas las respuestas en la misma gráfica
figure;
for i = 1:length(frecuencias_muestreo)
semilogx(f_Hz_all(:, i), H_z_all(:, i), 'LineWidth', 2, ...
'DisplayName', sprintf('fs = %.0f MHz', ...
frecuencias_muestreo(i) / 1e6));
hold on;
ylim([-10, 2]);  % Límites del eje y.
end

% Etiquetas del gráfico
title('Respuesta en frecuencia del filtro H(z) para diferentes fs');
```

```
xlabel('Frecuencia');
ylabel('Magnitud (dB)');
grid on;
legend show;

% Mostrar los valores de Kp y Ki
disp('Valores de Kp para cada frecuencia de muestreo:');
disp(kp_all);

disp('Valores de Ki para cada frecuencia de muestreo:');
disp(ki_all);
```

A.12. Generador de Código Gold

```
function [codigo_gold] = generador_codigo_gold(polinomio1, polinomio2, ...
semilla1, semilla2)
% Implementación del generador de Código Gold
% polinomio1 - polinomio preferido 1 para LFSR1, organizado como
%              [g0 g1 g2 ... gL-1]
% polinomio2 - polinomio preferido 2 para LFSR2, organizado como
%              [g0 g1 g2 ... gL-1]
% semilla1   - semilla inicial para LFSR1 [x0 x1 x2 ... xL-1]
% semilla2   - semilla inicial para LFSR2 [x0 x1 x2 ... xL-1]
% codigo_gold - Código Gold generado
% La función devuelve la secuencia para un solo período

polinomio1 = polinomio1(:);
estado1 = semilla1(:); % Reformular como vectores columna
% a polinomios y semillas
polinomio2 = polinomio2(:);
estado2 = semilla2(:);

if length(polinomio1) ~= length(polinomio2) && length(estado1) ~= ...
length(estado2)
error('Error: Longitudes distintas entre los polinomios o semillas');
end
```

```
% Construcción de la matriz de transición del LFSR
orden_polinomio = length(polinomio1) - 1; % Orden del polinomio
matriz_A0 = [zeros(1, orden_polinomio-1); eye(orden_polinomio-1)];
polinomio1 = polinomio1(1:end-1);
polinomio2 = polinomio2(1:end-1);
matriz_transicion1 = [matriz_A0 polinomio1]; % Matriz de transición LFSR1
matriz_transicion2 = [matriz_A0 polinomio2]; % Matriz de transición LFSR2

periodo = 2^orden_polinomio - 1; % Período de las secuencias de longitud
% máxima
codigo_gold = zeros(1, periodo); % Array para almacenar la salida
for i = 1:periodo % Repetir para cada período de reloj
codigo_gold(i) = mod(estado1(end) + estado2(end), 2); % XOR LFSR1 y 2
estado1 = mod(matriz_transicion1 * estado1, 2); % Ecuación del LFSR1
estado2 = mod(matriz_transicion2 * estado2, 2); % Ecuación del LFSR2
end
end
```

A.13. Correlación Cruzada para Código Gold de 31 bits en un canal sin ruido

```
%% Configurar algunas variables
palabras_datos = 10;  % Número de veces que se repetirá el Código Gold
M = 5;  % Orden del polinomio generador, determina la longitud de la
% secuencia LFSR
N = 2^M - 1;  % Período del Código Gold (longitud de la secuencia Gold)

% Pares preferidos de polinomios generadores para los registros de
% desplazamiento lineal
G1 = [1 1 1 1 0 1];  % Polinomio generador G1
G2 = [1 0 0 1 0 1];  % Polinomio generador G2

% Semilla inicial para el primer registro de desplazamiento
X1 = [0 0 0 0 1];  % Semilla para G1
```

```
% Semillas para el segundo registro de desplazamiento, generadas en
% binario para todas las combinaciones
X2 = dec2bin(1:2^M-1)' - '0';  % Convierte las combinaciones binarias a
% números
X2 = transpose(X2);  % Transpone las filas para que coincidan con
% las columnas

% Inicializa la matriz para almacenar los códigos Gold generados
y = zeros(N,N);  % Matriz de tamaño N x N para almacenar cada código
% Gold generado

% Generar códigos Gold para cada semilla de X2
for i = 1:N
g = generador_codigo_gold(G1, G2, X1, X2(i,:));  % Genera el
% Código Gold
% con las semillas
% G1, G2, X1 y
% X2(i)
y(i,1:N) = g;  % Almacena el Código Gold generado en la matriz 'y'
end

y(y == 0) = -1;  % Convierte los ceros en -1 para utilizar una
% representación bipolar de los códigos Gold

%% Crear la secuencia de datos para la transmisión (TX)
RX = zeros(N, N*palabras_datos);  % Inicializa la matriz de
% transmisión 'RX' con tamaño
% N x N*palabras_datos

% Repetir la secuencia de Código Gold para simular la transmisión de datos
for i = 1:N
TX = [];  % Inicializa la variable para la secuencia transmitida
TX = y(i,1:N);  % Toma el Código Gold correspondiente a la fila 'i'
for j = 1:palabras_datos-1
TX = [TX y(i,1:N)];  % Repite el Código Gold 'palabras_datos'
% veces
end
RX(i,1:N*palabras_datos) = TX;  % Almacena la secuencia transmitida
```

```
% en 'RX'
end

%% Enviar la señal a través del canal
% RX se utiliza directamente ya que no se modela ningún canal con ruido


% Calcular la correlación cruzada de cada Código Gold con la primera
% secuencia generada
for i = 1:N
corr = [];  % Inicializa la variable para almacenar la correlación
corr = xcorr(y(1,1:N), RX(i,1:N*palabras_datos));  % Calcula la
% correlación
% cruzada entre
% el primer
% Código Gold
% y el i-ésimo
longitud_correlacion = length(corr);  % Obtiene la longitud de
% la correlación
correlacion(i, 1:longitud_correlacion) = corr;  % Almacena la
% correlación en
% la matriz
% 'correlacion'
end

%% Graficar la salida
% Graficar la correlación cruzada de cada Código Gold generado
for i = 1:N
figure(1);  % Crear una nueva figura
plot(correlacion(i,1:longitud_correlacion));  % Graficar la
% correlación cruzada
title('Correlación Cruzada de Código Gold de N = 31')  % Título
% de la figura
pause(3);  % Pausar 3 segundos antes de cerrar la figura
close(figure(1));  % Cerrar la figura
pause(0.1);  % Pausar brevemente antes de la siguiente iteración
end
```

A.14. Correlación Cruzada para Código Gold de 4095 bits en un canal sin ruido

```
%% Configurar algunas variables
palabras_datos = 10;  % Número de palabras de datos a transmitir
cantidad_escenarios = 10;  % Número de códigos Gold a generar y analizar

M = 12;  % Orden del polinomio, determina la longitud del registro de
% desplazamiento (LFSR)
N = 2^M - 1;  % Período del Código Gold, N = 4095 para M = 12

% Pares preferidos de polinomios generadores para los registros de
% desplazamiento
G1 = [1 1 1 0 0 0 0 0 1 0 0 0 1];  % Polinomio generador G1
G2 = [1 1 1 0 0 0 0 0 0 0 1 0 1];  % Polinomio generador G2

% Semilla inicial para el primer registro de desplazamiento
X1 = [0 0 0 0 0 0 0 0 0 0 0 0 1];  % Semilla para G1

% Semillas para el segundo registro de desplazamiento, generadas en
% binario para todas las combinaciones
X2 = dec2bin(1:2^M-1)' - '0';  % Convierte las combinaciones binarias a
% números (1 a 4095)
X2 = transpose(X2);  % Transpone las filas para que coincidan con
% las columnas

% Inicializa la matriz para almacenar los códigos Gold generados
y = zeros(cantidad_escenarios, N);  % Matriz de tamaño
% cantidad_escenarios x N
% para almacenar cada código
% Gold generado

% Generar códigos Gold para cada semilla de X2
for i = 1:cantidad_escenarios
g = generador_codigo_gold(G1, G2, X1, X2(i,:));  % Genera el
% Código Gold con
% las semillas G1,
```

```
% G2, X1 y X2(i)
y(i, 1:N) = g;  % Almacena el Código Gold generado en la
% matriz 'y'
end

y(y == 0) = -1;  % Convierte los ceros en -1 para utilizar una
% representación bipolar de los códigos Gold

%% Crear la secuencia de datos para la transmisión (TX)
RX = zeros(cantidad_escenarios, N * palabras_datos);  % Inicializa
% la matriz de
% recepción 'RX'
% con tamaño
% cantidad_escenarios
% x (N *
% palabras_datos)

% Repetir la secuencia de Código Gold para simular la transmisión de
% datos
for i = 1:cantidad_escenarios
TX = [];  % Inicializa la variable para la secuencia transmitida
TX = y(i, 1:N);  % Toma el Código Gold correspondiente a la
% fila 'i'
for j = 1:palabras_datos - 1
TX = [TX y(i, 1:N)];  % Repite el Código Gold 'palabras_datos'
% veces
end
RX(i, 1:N * palabras_datos) = TX;  % Almacena la secuencia
% transmitida en 'RX'
end

%% Enviar la señal a través del canal
% RX se utiliza directamente ya que no se modela ningún canal con ruido



% Inicializa la matriz para almacenar la auto-correlación

for i = 1:cantidad_escenarios
corr = [];  % Inicializa la variable para almacenar la correlación
```

```
corr = xcorr(y(1, 1:N), RX(i, 1:N * palabras_datos));  %
% Calcula la
% correlación
% cruzada entre
% el primer
% Código Gold
% y el i-ésimo
longitud_correlacion = length(corr);  % Obtiene la longitud de
% la correlación
correlacion(i, 1:longitud_correlacion) = corr;  % Almacena la
% correlación en
% la matriz
% 'correlacion'
end

%% Graficar la salida
% Graficar la correlación cruzada de diferentes códigos Gold generados
figure, plot(correlacion(1, 1:longitud_correlacion));
title('auto-correlación Código Gold de longitud 4095');

figure, plot(correlacion(5, 1:longitud_correlacion));  % Graficar
% la correlación
% cruzada para
% el quinto
% Código Gold
title('Correlación Cruzada Código Gold de longitud 4095 Con Código Gold '
'de semilla [0 0 1 0 1]');

figure, plot(correlacion(10, 1:longitud_correlacion));  % Graficar
% la correlación
% cruzada para el
% décimo código
% Gold
title('Correlación Cruzada Código Gold de longitud 4095 Con Código Gold '
'de semilla [0 0 0 0 0 0 0 0 1 0 1 0]');  % Título de la figura

cantidad_graficos = 31;
for i = 1:cantidad_graficos
figure(1);  % Crear una nueva figura
```

```
plot(correlacion(i, 1:longitud_correlacion));
title('Correlación Cruzada Código Gold de longitud 4095 Con Código Gold');
if i == 1
pause(10);
else
pause(10);
end
close(figure(1));
pause(0.1);
end
```

A.15. Correlación entre Error Filtrado y Código Gold de 31 bits en un canal ideal

```
%% Inicialización de Parámetros
fc = 434e6;  % Frecuencia de la portadora en Hz (434 MHz).
fdev = 500e3;  % Desviación máxima de frecuencia en Hz (500 kHz).
df = 2;  % Factor de amortiguamiento del PLL.
Bn = 1.5e6;  % Ancho de banda del PLL en Hz (1.5 MHz).
fs = 6e6;  % Frecuencia de muestreo en Hz (6 MHz).
Ts = 1/fs;  % Período de muestreo en segundos (1/fs).
fvco = 2e6;  % Frecuencia del VCO en Hz (2 MHz).
num_bits = 1e3;  % Número de bits a transmitir (1000 bits).

%% Tiempo de bit y tiempo total
Tb = 1e-6;  % Duración de un bit en segundos (1 microsegundo).
tbit = 0:Ts:Tb-Ts;  % Vector de tiempo para un bit, con pasos de Ts.

%% Generación de la Señal Mensaje: Secuencias Gold
G1 = [1 1 1 1 0 1];  % Polinomio generador G1 para la secuencia Gold.
G2 = [1 0 0 1 0 1];  % Polinomio generador G2 para la secuencia Gold.
X1 = [0 0 0 0 1];  % Estado inicial del registro de desplazamiento X1.
X2 = [0 0 0 0 1];  % Estado inicial del registro de desplazamiento X2.
codigo_gold = generador_codigo_gold(G1, G2, X1, X2);
% Genera la secuencia Gold usando los polinomios y estados iniciales.
```

```
%% Código Gold repetido
nro_repeticiones = ones(1, length(tbit));
% Vector de repeticiones para la secuencia Gold.
gold_repetido = kron(codigo_gold, nro_repeticiones);
% Repite la secuencia Gold para adaptarse a la duración de un bit.
gold_repetido(gold_repetido == 0) = -1;
% Convierte los ceros en -1 (formato bipolar).
gold_repetido_corr = xcorr(gold_repetido);
% Calcula la auto-correlación del Código Gold repetido.

%% Crear la secuencia de datos para la transmisión (TX)
senial_modulada = Modulador_FSK(codigo_gold, fc, fdev, fs, Tb);
% Modula la secuencia Gold usando modulación FSK.

%% Rango de EbN0 y cálculo de Eb
EbN0_dB = 0;
% Valor de $E_b/N_0$ (energía por bit a densidad espectral de ruido) en dB.
EbN0 = 10.^(EbN0_dB/10);
% Convierte $E_b/N_0$ de dB a escala lineal.
Eb = (Tb*fs)*sum(abs(senial_modulada).^2)/(length(senial_modulada));
% Calcula la energía por bit Eb basado en la señal modulada.

%% Demodulación FSK
for k = 1:length(EbN0_dB)  % Itera sobre cada valor de $E_b/N_0$.
N0 = Eb/EbN0(k);
% Calcula N0 (densidad espectral de potencia de ruido).
ruido = 0;
% Canal ideal.
senial_entrada = senial_modulada;
% Señal de entrada, sin ruido agregado.

% Demodulador PLL
[~, error_filtrado, ~] = Demodulador_FSK_PLL(senial_entrada, fvco, ...
df, Bn, fs, Tb);
% Aplica demodulación FSK usando un PLL, obteniendo el error filtrado.
end

%% Correlación error_fase con gold repetido
```

```
gold_error_filtrado_corr = xcorr(gold_repetido, error_filtrado);
% Calcula la correlación entre el Código Gold repetido y el error de fase.

%% Gráfico 1: Correlación gold filtrado vs auto-correlación gold
figure,
subplot(2,1,1)
plot(gold_repetido_corr, 'g');
% Gráfica la auto-correlación del Código Gold repetido.
title('Auto-correlación Código Gold');

subplot(2,1,2)
plot(gold_error_filtrado_corr, 'b')
% Gráfica la correlación entre el Código Gold y el error de fase.
title('Correlación Código Gold con Error de Fase Filtrado');


%% Gráfico 2: Gold Repetido y Fase Filtrada
figure,
subplot(2,1,1)
stairs(gold_repetido,'g');
% Gráfica el Código Gold repetido.
title('Código Gold Repetido');

subplot(2,1,2)
plot(error_filtrado, 'b')
% Gráfica el error de fase filtrado.
title('Error Filtrado');
```

A.16. Correlación entre Error Filtrado y Código Gold de 4095 bits en un canal ideal

```
%% Inicialización de Parámetros
fc = 434e6;  % Frecuencia de la portadora en Hz (434 MHz).
fdev = 500e3;  % Desviación máxima de frecuencia en Hz (500 kHz).
df = 2;  % Factor de amortiguamiento del PLL.
```

```
Bn = 1.5e6;  % Ancho de banda del PLL en Hz (1.5 MHz).
fs = 6e6;  % Frecuencia de muestreo en Hz (6 MHz).
Ts = 1/fs;  % Período de muestreo
fvco = 2e6;  % Frecuencia del VCO en Hz (2 MHz).
num_bits = 1e3;  % Número de bits a transmitir (1000 bits).

%% Tiempo de bit y tiempo total
Tb = 1e-6;  % Duración de un bit en segundos (1 microsegundo).
br = 1/Tb;  % Tasa de bits
tbit = 0:Ts:Tb-Ts;  % Tiempo para la duración de un bit.
t = 0:Ts:(num_bits)*Tb-Ts;  % Tiempo total para transmitir 'num_bits'.

%% Inicialización de variables
bits = zeros(1, num_bits);  % Vector de bits de longitud 'num_bits'.

%% Generación de la Señal Mensaje: Secuencias Gold
M = 12;  % Orden del polinomio generador para la secuencia Gold.
N = 2^M-1;  % Período del Código Gold basado en el orden del polinomio.
G1 = [1 1 1 0 0 0 0 0 1 0 0 0 1];  % Polinomio generador G1.
G2 = [1 1 1 0 0 0 0 0 0 0 1 0 1];  % Polinomio generador G2.
X1 = [0 0 0 0 0 0 0 0 0 0 0 0 1];  % Estado inicial del registro G1.
X2 = [0 0 0 0 0 0 0 0 0 0 0 0 1];  % Estado inicial del registro G2.

codigo_gold = generador_codigo_gold(G1, G2, X1, X2);  % Generar secuencia.

%% Código Gold repetido
nro_repeticiones = ones(1, length(tbit));  % Vector de repeticiones.
gold_repetido = kron(codigo_gold, nro_repeticiones);  % Repite Código Gold.
gold_repetido(gold_repetido == 0) = -1;  % Mapea ceros del código a -1.
gold_repetido_corr = xcorr(gold_repetido);  % auto-correlación del código.

%% Crear la secuencia de datos para la transmisión (TX)
senial_modulada = Modulador_FSK(codigo_gold, fc, fdev, fs, Tb);  % Modulación FSK.

%% Rango de EbN0
EbN0_dB = 0;  % Valor de $E_b/N_0$ en dB.
EbN0 = 10.^(EbN0_dB/10);  % Conversión de dB a escala lineal.

%% Cálculo de EbN0
```

```
Eb = (Tb*fs)*sum(abs(senial_modulada).^2)/(length(senial_modulada));

%% Demodulación FSK
for k = 1:length(EbN0_dB)
N0 = Eb/EbN0(k);  % Densidad espectral de potencia de ruido.
ruido = 0  % Ruido AWGN.
senial_entrada  = senial_modulada;  % Señal sin ruido añadido.
[~, error_filtrado, ~] = Demodulador_FSK_PLL(senial_entrada, ...
fvco, df, Bn, fs, Tb);  % Demodulación FSK-PLL.
end

%% Correlación del error de fase con el Código Gold repetido
gold_error_filtrado_corr = xcorr(gold_repetido, error_filtrado);

%% Gráfico 1: Correlación entre Código Gold y error de fase
figure,
subplot(2,1,1)
plot(gold_repetido_corr,'g');
title('Autocorrelacion Código Gold (repetido)');
subplot(2,1,2)
plot(gold_error_filtrado_corr, 'b');
title('Correlacion Código Gold y Error de Fase Filtrado');
```

A.17. Sincronización con Gold 31 Bits sin señal al principio y final

```
%% Inicialización de Parametros
fc = 434e6;
fdev = 500e3;
df = 2;
Bn = 1.5e6;

fs = 6e6;
```

```
Ts = 1/fs;
fvco = 2e6;

num_bits = 1e3;

%% Tiempo de bit y tiempo total
Tb = 1e-6;
br = 1/Tb; % Tasa de bits

tbit = 0:Ts:Tb-Ts;
t = 0:Ts:(num_bits)*Tb-Ts;

%% Generación de la Señal Mensaje: Secuencias Gold
cantidad_palabras = 1;

M = 12; % Orden del polinomio, Longitud del LFSR
N = 2^M-1; % Período del Código Gold
% Pares Preferidos
G1 = [1 1 1 1 0 1];
G2 = [1 0 0 1 0 1];
X1 = [0 0 0 0 1];
X2 = [0 0 0 0 1];
codigo_gold = generador_codigo_gold(G1, G2, X1, X2); % Generar Código Gold

%% Código Gold repetido
nro_repeticiones = ones(1,length(tbit));
gold_repetido = kron(codigo_gold, nro_repeticiones);
gold_repetido(gold_repetido == 0) = -1;

%% Crear la secuencia de datos para la transmisión (TX)
gold_modulado = Modulador_FSK(codigo_gold, fc, fdev, fs, Tb);
bits_sin_senial = 20;
muestras_sin_senial = round(bits_sin_senial*Tb*fs);
senial_modulada = [zeros(1, muestras_sin_senial), gold_modulado, ...
zeros(1, muestras_sin_senial)];

%% Parametros del Histograma
muestras_por_bit = length(tbit);
periodo_bits = 2*bits_sin_senial + N;
```

```
muestras_por_periodo = 2 * periodo_bits * muestras_por_bit - 1;

%% Rango de EbN0
EbN0_dB = -4:2:20; % Rango de valores EbN0 a simular en escala de dB
EbN0 = 10.^(EbN0_dB/10);

%% Cálculo de EbN0
Eb = (Tb*fs)*sum(abs(senial_modulada).^2)/(length(senial_modulada));

cantidad_escenarios = 1000;
muestra_correcta = [round(Tb*fs*51)-2, round(Tb*fs*51)-1];


% Inicialización de los vectores de porcentaje
porcentaje_correctos = zeros(1, length(EbN0_dB));
porcentaje_incorrectos = zeros(1, length(EbN0_dB));

%% Demodulación FSK
for k = 1:length(EbN0_dB)
contador_maximos = zeros(1, muestras_por_periodo);
correctos = 0;
incorrectos = 0;

for j = 1:cantidad_escenarios
N0 = Eb/EbN0(k);
ruido = sqrt(N0/2)*(randn(1,length(senial_modulada)) + ...
1j*randn(1,length(senial_modulada)));
senial_entrada  = senial_modulada + ruido;
% Demodulador PLL
[salida_dds, error_filtrado, ~] = Demodulador_FSK_PLL(...
senial_entrada, fvco, df, Bn, fs, Tb);

%% Correlacion del error de fase filtrado con el Código Gold repetido
correlacion_gold_repetido = xcorr(gold_repetido, error_filtrado);

%% Número de periodos completos en la señal de correlacion
[~, max_idx] = max(correlacion_gold_repetido);

% Almacenar el índice relativo dentro del periodo de 51 bits
```

```
contador_maximos(max_idx) = contador_maximos(max_idx) + 1;

% Contabilizar si es un máximo correcto o incorrecto
if max_idx == muestra_correcta(1) || max_idx == muestra_correcta(2)
correctos = correctos + 1;
else
incorrectos = incorrectos + 1;
end
end

%% Cálculo de los porcentajes
total_maximos = correctos + incorrectos;
porcentaje_correctos(k) = (correctos / total_maximos) * 100;
porcentaje_incorrectos(k) = (incorrectos / total_maximos) * 100;

%% Graficar el histograma de las posiciones de los máximos
figure, stem(1:muestras_por_periodo, contador_maximos);
title(['Histograma de los índices del máximo de correlación para ', ...
'EbN0 = ', num2str(EbN0_dB(k)), ' dB']);
xlabel('Numero de Muestra');
ylabel('Veces');
end

%% Graficar los porcentajes de correctos e incorrectos
figure;
plot(EbN0_dB, porcentaje_correctos, 'b-o');
hold on;
plot(EbN0_dB, porcentaje_incorrectos, 'r-x');
xlabel('$E_b/N_0$ [dB]');
ylabel('Porcentaje (%)');
title('Porcentaje de picos correctos e incorrectos en función de $E_b/N_0$');
legend('Correctos', 'Incorrectos');
grid on;
```

A.18. Sincronización con Gold 31 Bits con secuencia de unos y menos unos alternados al principio y final

```
% Inicialización de Parámetros
fc = 434e6;
fdev = 500e3;
df = 2;
Bn = 1.5e6;
fs = 6e6;
Ts = 1/fs;
fvco = 2e6;
num_bits = 1e3;

% Tiempo de bit y tiempo total
Tb = 1e-6;
tbit = 0:Ts:Tb-Ts;
t = 0:Ts:(num_bits)*Tb-Ts;

% Generación de la Señal Mensaje: Secuencias Gold
G1 = [1 1 1 1 0 1];
G2 = [1 0 0 1 0 1];
X1 = [0 0 0 0 1];
X2 = [0 0 0 0 1];
codigo_gold = generador_codigo_gold(G1,G2,X1,X2);

% Código Gold repetido
nro_repeticiones = ones(1,length(tbit));
gold_repetido = kron(codigo_gold, nro_repeticiones);
gold_repetido(gold_repetido == 0) = -1;

% Crear la secuencia de datos para la transmisión (TX)
gold_modulado = Modulador_FSK(codigo_gold, fc, fdev, fs, Tb);
bits_sin_senial = 20;
muestras_sin_senial = floor(bits_sin_senial*Tb*fs) + 1;
bits_alternados = repmat([1 0], 1, muestras_sin_senial/2);
gold_gap_modulado = [bits_alternados, gold_modulado, bits_alternados];
```

```
senial_modulada = gold_gap_modulado;
senial_mensaje = [bits_alternados, codigo_gold, bits_alternados];

% Parametros del Histograma
muestras_por_bit = length(tbit);
periodo_bits = 71;
muestras_por_periodo = 2 * periodo_bits * muestras_por_bit;

% Rango de EbN0
EbN0_dB = -4:2:20;
EbN0 = 10.^(EbN0_dB/10);

% Cálculo de EbN0
Eb = (Tb * fs) * sum(abs(senial_modulada).^2) / length(senial_modulada);
cantidad_escenarios = 1000;
muestra_correcta = [round(Tb * fs * 51) - 2, round(Tb * fs * 51) - 1];

% Inicialización de vectores de porcentaje
porcentaje_correctos = zeros(1, length(EbN0_dB));
porcentaje_incorrectos = zeros(1, length(EbN0_dB));

% Demodulación FSK
for k = 1:length(EbN0_dB)
contador_maximos = zeros(1, muestras_por_periodo);
correctos = 0;
incorrectos = 0;

for j = 1:cantidad_escenarios
N0 = Eb / EbN0(k);
ruido = sqrt(N0/2) * (randn(1, length(senial_modulada)) + ...
 1j * randn(1, length(senial_modulada)));
senial_entrada  = senial_modulada + ruido;

% Demodulador PLL
[salida_dds, error_filtrado, senial_demodulada] = ...
 Demodulador_FSK_PLL(senial_entrada, fvco, df, Bn, fs, Tb);

% Correlacion del error de fase filtrado con el Código Gold repetido
correlacion_gold_repetido = xcorr(gold_repetido, error_filtrado);
```

```
[~, max_idx] = max(correlacion_gold_repetido);

% Almacenar el índice relativo dentro del periodo de 51 bits
contador_maximos(max_idx) = contador_maximos(max_idx) + 1;

% Contabilizar si es un máximo correcto o incorrecto
if max_idx == muestra_correcta(1) || max_idx == muestra_correcta(2)
correctos = correctos + 1;
else
incorrectos = incorrectos + 1;
end

end

% Cálculo de los porcentajes
total_maximos = correctos + incorrectos;
porcentaje_correctos(k) = (correctos / total_maximos) * 100;
porcentaje_incorrectos(k) = (incorrectos / total_maximos) * 100;

% Graficar el histograma de las posiciones de los máximos
figure, stem(1:muestras_por_periodo, contador_maximos);
title(['Histograma de los índices del máximo de correlación para EbN0 = ', ...
 num2str(EbN0_dB(k)), ' dB']);
xlabel('Índice de muestra dentro del periodo de 51 bits');
ylabel('Veces');
end

%% Graficar los porcentajes de correctos e incorrectos
figure;
plot(EbN0_dB, porcentaje_correctos, 'b-o');
hold on;
plot(EbN0_dB, porcentaje_incorrectos, 'r-x');
xlabel('$E_b/N_0$ [dB]');
ylabel('Porcentaje (%)');
title('Porcentaje de picos correctos e incorrectos en función de $E_b/N_0$');
legend('Correctos', 'Incorrectos');
grid on;
```

A.19. Sincronización con Gold 4095 Bits sin señal al principio y final

```
%% Inicialización de Parámetros
fc = 434e6; % Frecuencia de la portadora, 434 MHz.
fdev = 500e3; % Desviación de frecuencia para la modulación FSK.
df = 2; % Factor de amortiguamiento del PLL (Phase-Locked Loop).
Bn = 1.5e6; % Ancho de banda de ruido del PLL, 1.5 MHz.

fs = 6e6; % Frecuencia de muestreo
Ts = 1/fs; % Periodo de muestreo.
fvco = 2e6; % Frecuencia del DDS.

num_bits = 1e3; % Número de bits en la secuencia de datos.

%% Tiempo de bit y tiempo total
Tb = 1e-6; % Duración de un bit, 1 microsegundo.
br = 1/Tb; % Tasa de bits
tbit = 0:Ts:Tb-Ts; % Vector de tiempo para un bit.
t = 0:Ts:(num_bits)*Tb-Ts; % Vector de tiempo total para los datos.

%% Generación de la Señal Mensaje: Secuencias Gold
cantidad_palabras = 1; % Cantidad de códigos a generar.

M = 12; % Orden del polinomio del LFSR
N = 2^M-1; % Período del Código Gold, basado en el orden del LFSR.
% Pares preferidos para generar el Código Gold.
G1 = [1 1 1 0 0 0 0 0 1 0 0 0 1]; % Polinomio de conexión del LFSR.
G2 = [1 1 1 0 0 0 0 0 0 0 1 0 1]; % Polinomio de conexión del segundo LFSR.
% Semillas iniciales para los registros de corrimiento.
X1 = [0 0 0 0 0 0 0 0 0 0 0 0 1]; % Semilla del primer LFSR.
X2 = [0 0 0 0 0 0 0 0 0 0 0 0 1]; % Semilla del segundo LFSR.

codigo_gold = generador_codigo_gold(G1,G2,X1,X2); % Genera el Código Gold.

%% Código Gold repetido
nro_repeticiones = ones(1,length(tbit)); % Repetición por bit.
```

```
gold_repetido = kron(codigo_gold, nro_repeticiones); % Repite el código.
gold_repetido(gold_repetido == 0) = -1; % Transforma ceros en -1.

%% Crear la secuencia de datos para la transmisión (TX)
gold_modulado = Modulador_FSK(codigo_gold, fc, fdev, fs, Tb); % Modulación FSK.
bits_sin_senial = 200; % Bits de silencio antes y después de la señal.
muestras_sin_senial = round(bits_sin_senial*Tb*fs); % Muestras de silencio.
senial_modulada = [zeros(1, muestras_sin_senial), gold_modulado, ...
zeros(1, muestras_sin_senial)]; % Señal completa.

%% Parámetros del Histograma
muestras_por_bit = length(tbit); % Número de muestras por bit.
periodo_bits = round(2*bits_sin_senial + N); % Total de bits.
muestras_por_periodo = 2*periodo_bits*muestras_por_bit - 1;

%% Rango de EbN0
EbN0_dB = -4:2:20; % Rango de $E_b/N_0$ en dB.
EbN0 = 10.^(EbN0_dB/10); % Conversión de dB a escala lineal.

%% Cálculo de $E_b/N_0$
Eb = (Tb*fs)*sum(abs(senial_modulada).^2)/(length(senial_modulada));

cantidad_escenarios = 1000; % Cantidad de escenarios de ruido.
muestra_correcta = round((bits_sin_senial + N)*fs*Tb); % Muestra correcta.

porcentaje_correctos = zeros(1, length(EbN0_dB));
porcentaje_incorrectos = zeros(1, length(EbN0_dB));

%% Demodulación FSK
for k = 1:length(EbN0_dB)
contador_maximos = zeros(1, muestras_por_periodo);
correctos = 0; % Picos correctos.
incorrectos = 0; % Picos incorrectos.

for j = 1:cantidad_escenarios
N0 = Eb/EbN0(k); % Densidad espectral de ruido.
ruido = sqrt(N0/2)*(randn(1,length(senial_modulada)) + ...
1j*randn(1,length(senial_modulada))); % Ruido AWGN.
senial_entrada  = senial_modulada + ruido; % Señal recibida.
```

```
% Demodulación FSK usando PLL.
[salida_dds, error_filtrado, ~] = Demodulador_FSK_PLL(...
senial_entrada, fvco, df, Bn, fs, Tb);

%% Correlación del error de fase filtrado
correlacion_gold_repetido = xcorr(gold_repetido, ...
error_filtrado);

%% Número de periodos completos en la correlación
[~, max_idx] = max(correlacion_gold_repetido); % Índice máximo.

contador_maximos(max_idx) = contador_maximos(max_idx) + 1;

if max_idx == muestra_correcta(1)
correctos = correctos + 1; % Pico correcto.
else
incorrectos = incorrectos + 1; % Pico incorrecto.
end
end

%% Cálculo de los porcentajes
total_maximos = correctos + incorrectos;
porcentaje_correctos(k) = (correctos / total_maximos) * 100;
porcentaje_incorrectos(k) = (incorrectos / total_maximos) * 100;

%% Graficar el histograma
figure, stem(1:muestras_por_periodo, contador_maximos);
title(['Histograma de los índices del máximo de correlación ', ...
'para EbN0 = ', num2str(EbN0_dB(k)), ' dB']);
xlabel('Numero de Muestra');
ylabel('Veces');
end

%% Graficar los porcentajes de correctos e incorrectos
figure;
plot(EbN0_dB, porcentaje_correctos, 'b-o');
hold on;
plot(EbN0_dB, porcentaje_incorrectos, 'r-x');
```

```
xlabel('$E_b/N_0$ [dB]');
ylabel('Porcentaje (%)');
title('Porcentaje de picos correctos e incorrectos en función de $E_b/N_0$');
legend('Correctos', 'Incorrectos');
grid on;
```

A.20. Sincronización con Gold 4095 Bits con secuencia de unos y menos unos alternados al principio y al final

```
%% Inicialización de Parametros
fc = 434e6;  % Frecuencia de la portadora (434 MHz)
fdev = 500e3;  % Desviación de frecuencia de la modulación FSK (500 kHz)
df = 2;  % Factor de amortiguamiento del PLL
Bn = 1.5e6;  % Ancho de banda del bucle del PLL (1.5 MHz)

fs = 8e6;  % Frecuencia de muestreo (8 MHz)
Ts = 1/fs;  % Periodo de muestreo
fvco = 2e6;  % Frecuencia del oscilador controlado por tensión (2 MHz)

num_bits = 1e3;  % Número de bits de la señal de mensaje

%% Tiempo de bit y tiempo total
Tb = 1e-6;  % Duración de cada bit (1 microsegundo)
br = 1/Tb;  % Tasa de bits

tbit = 0:Ts:Tb-Ts;  % Vector de tiempo para un bit
t = 0:Ts:(num_bits)*Tb-Ts;  % Vector de tiempo para la señal (1000 bits)

%% Generación de la Señal Mensaje: Secuencias Gold
cantidad_palabras = 1;  % Cantidad de secuencias Gold

M = 12;  % Orden del polinomio, longitud del LFSR
N = 2^M-1;  % Período del Código Gold (4095 bits)
% Pares Preferidos
```

```
G1 = [1 1 1 0 0 0 0 0 1 0 0 0 1];  % Polinomio generador 1
G2 = [1 1 1 0 0 0 0 0 0 0 1 0 1];  % Polinomio generador 2
% Semillas
X1 = [0 0 0 0 0 0 0 0 0 0 0 0 1];  % Semilla para el registro 1
X2 = [0 0 0 0 0 0 0 0 0 0 0 0 1];  % Semilla para el registro 2

codigo_gold = generador_codigo_gold(G1,G2,X1,X2);  % Generar Código Gold

%% Código Gold repetido
nro_repeticiones = ones(1,length(tbit));  % Repetir el código por bit
gold_repetido = kron(codigo_gold, nro_repeticiones);  % Repetición
gold_repetido(gold_repetido == 0) = -1;  % Ceros a -1

%% Crear la secuencia de datos para la transmisión (TX)
bits_sin_senial = 200;  % Bits sin señal para inicialización
muestras_sin_senial = bits_sin_senial*Tb*fs;  % Muestras sin señal
bits_alternados = repmat([1 0], 1, 100);  % Bits de unos
%y ceros alternados
senial_mensaje = [bits_alternados,codigo_gold,bits_alternados];
senial_modulada = Modulador_FSK(senial_mensaje, fc, fdev, fs, Tb);

%% Parametros del Histograma
muestras_por_bit = length(tbit);  % Muestras por bit
periodo_bits = 4095 + 400;  % Total de bits
muestras_por_periodo = 2 * periodo_bits * muestras_por_bit - 1;
% Muestras totales

%% Rango de EbN0
EbN0_dB = -4:2:20;  % Rango de $E_b/N_0$ a simular, en dB
EbN0 = 10.^(EbN0_dB/10);  % Convertir $E_b/N_0$ a escala lineal

%% Cálculo de EbN0
Eb = (Tb*fs)*sum(abs(senial_modulada).^2)/(length(senial_modulada));

cantidad_escenarios = 10;  % Número de escenarios a simular
muestra_correcta = 25769;  % Muestra correcta esperada

% Inicialización de los vectores de porcentaje
porcentaje_correctos = zeros(1, length(EbN0_dB));
```

```
porcentaje_incorrectos = zeros(1, length(EbN0_dB));

%% Demodulación FSK
for k = 1:length(EbN0_dB)  % Bucle sobre los valores de $E_b/N_0$
contador_maximos = zeros(1, muestras_por_periodo);
correctos = 0;  % Contador de correlaciones correctas
incorrectos = 0;  % Contador de correlaciones incorrectas

for j = 1:cantidad_escenarios
N0 = Eb/EbN0(k);
ruido = sqrt(N0/2)*(randn(1,length(senial_modulada)) + ...
1j*randn(1,length(senial_modulada)));  % Ruido complejo
senial_entrada = senial_modulada + ruido;  % Señal con ruido
% Demodulador PLL
[salida_dds, error_filtrado, ~] = Demodulador_FSK_PLL(...
senial_entrada, fvco, df, Bn, fs, Tb);

%% Correlacion del error de fase filtrado con el Código Gold repetido
correlacion_gold_repetido = xcorr(gold_repetido, ...
error_filtrado);  % Correlación

%% Número de periodos completos en la señal de correlacion
[~, max_idx] = max(correlacion_gold_repetido);  % Índice máximo

% Almacenar el índice relativo dentro del periodo de bits
contador_maximos(max_idx) = contador_maximos(max_idx) + 1;

% Contabilizar si es un máximo correcto o incorrecto
if max_idx == muestra_correcta  % Verificar si el máximo es correcto
correctos = correctos + 1;  % Incrementar correctos
else
incorrectos = incorrectos + 1;  % Incrementar incorrectos
end
end

%% Cálculo de los porcentajes
total_maximos = correctos + incorrectos;
porcentaje_correctos(k) = (correctos / total_maximos) * 100;
porcentaje_incorrectos(k) = (incorrectos / total_maximos) * 100;
```

```
%% Graficar el histograma de las posiciones de los máximos
figure, stem(1:muestras_por_periodo, contador_maximos);
title(['Histograma de los índices del máximo de correlación para ' ...
'EbN0 = ', num2str(EbN0_dB(k)), ' dB']);
xlabel('Numero de Muestra');
ylabel('Veces');
end

%% Graficar los porcentajes de correctos e incorrectos
figure;
plot(EbN0_dB, porcentaje_correctos, 'b-o');  % Porcentaje correctos
hold on;
plot(EbN0_dB, porcentaje_incorrectos, 'r-x');  % Porcentaje incorrectos
xlabel('$E_b/N_0$ [dB]');
ylabel('Porcentaje (%)');
title('Porcentaje de picos correctos e incorrectos');
```

A.21. Obtención de los umbrales de decisión para el Demodulador BFSK mediante Discriminador de Frecuencias

```
%% Inicialización de Parámetros

f_portadora = 434e6;
f_desviacion = 1e6; % Desviación máxima de frecuencia (1 MHz)

numero_bits = 1e3;
frecuencias_muestreo = [4,6,8,12]*1e6; % Frecuencias de muestreo
EbN0_dB = 0:2:18; % Rango de valores Eb/N0 a simular en escala de dB
EbN0_lineal = 10.^(EbN0_dB/10);
num_errores_minimos = 100;
num_pruebas_minimas = 30;

% Inicialización de matrices para almacenar resultados
```

```
num_errores_total = zeros(length(frecuencias_muestreo), length(EbN0_lineal));
num_bits_total = zeros(length(frecuencias_muestreo), length(EbN0_lineal));

% Matrices para almacenar información de umbrales y salidas
umbrales_pruebas = zeros(length(frecuencias_muestreo), length(EbN0_lineal),...
num_pruebas_minimas);
promedios_senal0_pruebas = zeros(length(frecuencias_muestreo),...
length(EbN0_lineal), num_pruebas_minimas);
promedios_senal1_pruebas = zeros(length(frecuencias_muestreo),...
length(EbN0_lineal), num_pruebas_minimas);

%% Simulación

for indice_fs = 1:length(frecuencias_muestreo)
fs = frecuencias_muestreo(indice_fs);
Ts = 1/fs;
Tb = 1e-6; % Tiempo de bit
t = 0:Ts:(numero_bits)*Tb-Ts;

for indice_EbN0 = 1:length(EbN0_lineal)
EbN0 = EbN0_lineal(indice_EbN0);
contador = 0;
contador_EbN0 = 0;

tic
while ((num_errores_total(indice_fs, indice_EbN0) < num_errores_minimos) || ...
(contador_EbN0 < num_pruebas_minimas))

if (mod(contador,10) == 0)
bits = randi([0,1],1, numero_bits);
coeficientes = bits;
coeficientes(coeficientes == 0) = -1;
fase_inicial = 2*pi*rand(1,numero_bits);
coeficientes_tiempo = zeros(size(t));
fase_inicial_tiempo = zeros(size(t));

for m = 0:numero_bits-1
coeficientes_tiempo((t >= m*Tb-eps) & (t < (m+1)*Tb+eps)) = ...
coeficientes(m+1);
```

```
fase_inicial_tiempo((t >= m*Tb-eps) & (t < (m+1)*Tb+eps)) = ...
fase_inicial(m+1);
end

amplitud = 1;
senal_modulada = amplitud * exp(1j*(2*pi*(f_portadora + ...
(coeficientes_tiempo * f_desviacion)).*t + ...
fase_inicial_tiempo));

potencia_senal_modulada = sum(abs(senal_modulada).^2) / ...
numel(senal_modulada);
end

ruido = sqrt(Tb/Ts) * sqrt(potencia_senal_modulada/EbN0) * ...
(randn(1,numel(senal_modulada)) + 1j * randn(1,numel(senal_modulada))) / sqrt(2);
senal_recibida = senal_modulada + ruido;

[bits_demodulados, umbral, promedios] = ...
Demodulador_FSK_Discriminador_2(senal_recibida, fs, Tb);

contador_EbN0 = contador_EbN0 + 1;

if contador_EbN0 <= num_pruebas_minimas
umbrales_pruebas(indice_fs, indice_EbN0, contador_EbN0) = umbral;
promedios_senal0_pruebas(indice_fs, indice_EbN0, contador_EbN0) = promedios(1);
promedios_senal1_pruebas(indice_fs, indice_EbN0, contador_EbN0) =  promedios(2);
end

num_errores_total(indice_fs, indice_EbN0) = ...
num_errores_total(indice_fs, indice_EbN0) + sum(bits ~= ... bits_demodulados);
num_bits_total(indice_fs, indice_EbN0) = ...
num_bits_total(indice_fs, indice_EbN0) + numero_bits;

contador = contador + 1;
end
toc
end
end
```

```
%% Cálculo de probabilidades de error


prob_error_estimada = num_errores_total ./ num_bits_total;

figure('units','normalized','outerposition',[0 0 1 1])
semilogy(EbN0_dB, prob_error_teorica, EbN0_dB, prob_error_teorica2, ...
EbN0_dB, prob_error_estimada(1,:),'*--', EbN0_dB, prob_error_estimada(2,:),'*--',
EbN0_dB, prob_error_estimada(3,:),'*--', EbN0_dB, prob_error_estimada(4,:),'*--',
'LineWidth',2, 'MarkerSize',15);

legend('Teórica 1', 'Teórica 2','Estimada - 4MHz','Estimada - 6MHz',...
'Estimada - 8 MHz','Estimada - 12MHz');

xlabel('Eb/N0 (dB)');
ylabel('Probabilidad de error de bit (BER)');
title('Comparación de BER teórica y estimada ...
para diferentes frecuencias de muestreo');
grid on;

%% Gráfica de umbrales y salidas

figure('units','normalized','outerposition',[0 0 1 1]);

for indice_fs = 1:4
subplot(2, 2, indice_fs)
p0 = plot(squeeze(promedios_senal0_pruebas(indice_fs,:,:)), '-b');
hold on;

for i = 2:length(p0)
p0(i).Annotation.LegendInformation.IconDisplayStyle = 'off';
end
set(p0(1), 'DisplayName', 'Promedio de señal para bit 0');

p1 = plot(squeeze(promedios_senal1_pruebas(indice_fs,:,:)), '-r');
for i = 2:length(p1)
p1(i).Annotation.LegendInformation.IconDisplayStyle = 'off';
end
set(p1(1), 'DisplayName', 'Promedio de señal para bit 1');
```

```
u = plot(squeeze(umbrales_pruebas(indice_fs,:,:)), '-g', 'LineWidth', 2);
for i = 2:length(u)
u(i).Annotation.LegendInformation.IconDisplayStyle = 'off';
end
set(u(1), 'DisplayName', 'Umbral');

legend();
ylim([-1.1, 1.1]);
title(['Salidas y Umbrales para fs = ', ...

num2str(frecuencias_muestreo(indice_fs)/1e6), ' MHz']);
xlabel('Índice de prueba');
ylabel('Valor');
end

% Función del demodulador
function [senial_demodulada, umbral, promedios] = ...
Demodulador_FSK_Discriminador_2(senial_modulada, fs, Tb)

senial_modulada = exp(1i*angle(senial_modulada));
xd = [0, diff(senial_modulada) .* fs];

xd = abs(xd);
yd = xd - mean(xd);
yd = yd / max(abs(yd));
umbral = mean(yd);

nyquist_freq = fs / 2;
cutoff_freq = 1.5e6;
normalized_cutoff = cutoff_freq / nyquist_freq;
filter_order = 100;
b = fir1(filter_order, normalized_cutoff);
senial_filtrada = filtfilt(b, 1, yd);
yd = senial_filtrada;

senial_demodulada = [];
mean_bit = [];
g = 1;
```

```
indice_muestra_inicio = 1;
indice_muestra_fin = round(Tb * fs);

while indice_muestra_fin <= length(yd)
mean_bit(g) = mean(yd(indice_muestra_inicio:indice_muestra_fin));
if mean_bit(g) > umbral
senial_demodulada(end+1) = 1;
else
senial_demodulada(end+1) = 0
```