

**PROYECTO INTEGRADOR DE LA CARRERA DE
INGENIERÍA EN TELECOMUNICACIONES**

**CONFORMACIÓN DIGITAL DE HAZ PARA RECEPCIÓN DE
SEÑALES SATELITALES**

Lucas Mariano Grigolato
Estudiante

Dr. Santiago Hernandez
Director

Ing. Nicolás Catalano
Co-director

Miembros del Jurado
Ing. Roberto Costantini (INVAP - Instituto Balseiro)
Dr. Damián Dellavale Clara (CONICET - Instituto Balseiro)

23 de Noviembre de 2020

Departamento de Ingeniería en Telecomunicaciones
Comisión Nacional de Energía Atómica
Centro Atómico Bariloche

Instituto Balseiro
Universidad Nacional de Cuyo
Comisión Nacional de Energía Atómica
Argentina

A mi mamá
y a mis hermanos Fer y Palito,
máximos responsables de que haya podido llegar hasta acá.

Índice de siglas y acrónimos

- ACU: Arreglo Circular Uniforme.
- ALU: Arreglo Lineal Uniforme.
- ARU: Arreglo Rectangular Uniforme.
- BER: Bit Error Rate.
- DBF: Digital Beamforming/Beamformer - Conformación/Conformador Digital de Haz.
- DOA: Direction of Arrival - Dirección de arribo.
- DSP: Digital Signal Processor.
- FPGA: Field-Programmable Gate Array - Arreglo de compuertas **programables**.
- LEO: Low Earth Orbit - Baja órbita.
- MUSIC: Multiple Signal Classification.
- PL: Programmable Logic.
- PS: Processing System.
- QPSK: Quadrature Phase-Shift Keying.
- RF: RadioFrecuencia.
- RMSE: Root Mean Square Error.
- SDR: Software Defined Radio.
- SNR: Signal-to-Noise Ratio - Relación Señal a Ruido.
- SVD: Singular Value Decomposition.
- SVM: Support Vector Machines.
- TLS: Total Least Squares.

Índice de contenidos

Índice de siglas y acrónimos	v
Índice de contenidos	vii
Índice de figuras	ix
Índice de tablas	xiii
Resumen	xv
Abstract	xvii
1. Introducción	1
1.1. Motivación	1
1.2. Objetivos de proyecto	2
2. Conformación de haz	5
2.1. Introducción	5
2.2. Clasificación de conformadores de haz	9
2.3. Arreglos de antenas en fase	11
2.3.1. Tipos de arreglos	11
2.4. Definición del problema	15
3. Algoritmos de estimación de dirección de arribo.	17
3.1. Introducción	17
3.1.1. Modelo de datos	18
3.2. El algoritmo MUSIC	22
3.2.1. Algoritmo	23
3.3. El algoritmo ESPRIT	23
3.3.1. Algoritmo	27
3.4. Simulaciones	28
3.4.1. RMSE vs. SNR	29
3.4.2. RMSE vs. $\frac{\sigma_d}{d}$	30
3.4.3. RMSE vs. N° de muestras	30
3.4.4. Tiempo vs. M	31
3.4.5. Tiempo de ejecución de la SVD	33

4. Muestreo aleatorio	35
4.1. Introducción	35
4.2. Definición del problema	37
4.3. Muestreo Aleatorio	39
5. Estimación del número de señales recibidas	43
5.1. Introducción	43
5.2. Método de la máxima derivada primera	44
5.2.1. Resultados obtenidos	45
5.3. Clasificador binario mediante aprendizaje automático	46
5.3.1. Regresión Logística	46
5.3.2. Máquina de Vectores de Soporte	49
5.3.3. Resultados obtenidos	51
6. Diseño del sistema	57
6.1. Introducción	57
6.2. Diagrama de bloques del sistema	58
6.3. Muestreador aleatorio	59
6.3.1. Implementación en FPGA	59
6.4. Estimador de dirección de arribo	60
6.5. Conformador de haz	61
6.5.1. Implementación en FPGA	62
7. Integración en GNURadio	65
7.1. Introducción	65
7.2. El modelo equivalente banda base	66
7.3. Implementación de módulos	68
7.3.1. Tipos de datos en GNURadio	69
7.3.2. Callbacks	70
7.3.3. La librería VOLK	70
7.3.4. El emulador de ARU.	70
7.4. Simulaciones	71
7.4.1. El módulo gr-satellites	72
7.4.2. Resultados obtenidos	72
7.4.3. Requerimientos de procesamiento	72
8. Trabajo a futuro	75
8.1. Interfaz e integración con el sistema de adquisición	75
8.2. Carga del patrón de radiación del arreglo	76
8.3. Interferencias destructivas	76
8.4. Smart Beamforming	76
9. Conclusiones	77
A. Obtención de ángulos de arribo en ESPRIT	79
Bibliografía	81
Agradecimientos	83

Índice de figuras

1.1. Hardware donde se debe implementar el sistema a desarrollar.	2
2.1. Representación de un enlace punto a punto utilizando antenas parabólicas, indicando sus correspondientes patrones de radiación junto con sus lóbulos principales y secundarios.	5
2.2. Antenas parabólicas móviles de la estación terrena perteneciente al Centro Espacial Teófilo Tabanera [1].	6
2.3. Sistema de coordenadas horizontales para comunicaciones satelitales.	7
2.4. Frente de onda plano arribando a un arreglo lineal de antenas. Se indica en rojo el elemento de referencia, en el cual se considera que la señal arribante tiene fase nula.	7
2.5. Clasificación de conformadores de haz según cómo manipulan la señal recibida.	10
2.6. Arreglo lineal uniforme.	12
2.7. Arreglo circular uniforme.	13
2.8. Arreglo rectangular uniforme.	14
2.9. Esquema del problema a resolver.	15
3.1. Representación geométrica de la estimación de DOA mediante la intersección del conjunto \mathfrak{A} con el subespacio de señal para el caso de DOA unidimensional, con $D = 2$ y $M = 3$	19
3.2. Geometría del arreglo de antenas para el análisis del algoritmo ESPRIT. En azul y en rojo se separan los elementos pertenecientes a los subarreglos Z_x y Z_y , respectivamente.	24
3.3. Una posible elección de subarreglos para la aplicación del algoritmo ESPRIT en dos dimensiones utilizando un ARU.	26
3.4. Gráfico de comparación del RMSE en función de la SNR para los algoritmos MUSIC y ESPRIT.	29
3.5. Gráfica de \mathbf{P}_{MU} en el caso de una señal llegando al arreglo con dirección ($\theta = 45^\circ, \varphi = 30$) para distintos valores de SNR. Puede observarse cómo a medida que aumenta la SNR la “energía” del pseudoespectro se concentra cada vez más en la dirección de arriba real.	30
3.6. Gráfico de comparación del RMSE en función de los errores en la separación de elementos para los algoritmos MUSIC y ESPRIT.	31
3.7. Gráfica de comparación del RMSE en función de la cantidad de vectores de muestras para los algoritmos MUSIC y ESPRIT.	32
3.8. Gráfica de comparación del tiempo de ejecución de los algoritmos MUSIC y ESPRIT en función de la cantidad de elementos del ARU.	32
3.9. Gráfica de tiempo de ejecución del algoritmo SVD en función de la cantidad M de elementos del ARU y la cantidad N de vectores de muestra de la entrada.	34

3.10. Gráfica de tiempo de ejecución del algoritmo SVD en función de la cantidad N de vectores de muestras de la entrada con $M = 16$	34
4.1. Señal resultante al muestrear a $f_s = 1$ MHz un elemento de un arreglo de antenas al cual le llegan dos señales senoidales de frecuencias $f_A = 5$ kHz y $f_B = 100$ kHz.	36
4.2. Señal resultante al muestrear a $f_s = 1$ MHz un elemento de un arreglo de antenas al cual le llegan dos señales senoidales de frecuencias $f_A = 5$ kHz y $f_B = 100$ kHz escogiendo 25 muestras utilizando muestreo aleatorio.	37
4.3. Correlación cruzada entre $x(t)$ e $y(t)$ en función de T utilizando muestreo aleatorio con distintos valores p	41
4.4. Gráfica de comparación del RMSE en función de la ventana de observación T para distintos valores de p al recibir dos señales en distintas direcciones.	41
5.1. Distribución de valores singulares del subespacio de muestras ordenados de mayor a menor para el caso de dos señales arribando a un arreglo de 16 elementos. La línea azul indica la separación entre valores singulares correspondientes al subespacio de señal y al subespacio de ruido.	44
5.2. Derivada de la distribución de valores singulares mostrados en la Figura 5.1 ordenados de menor a mayor.	44
5.3. Esquema del estimador de cantidad de señales recibidas mediante el método de la máxima derivada visto como sistema para el caso de recepción de 4 señales. Se indican con λ_S aquellos valores singulares correspondientes al subespacio de señal y con λ_W aquellos que corresponden al subespacio de ruido.	45
5.4. Esquema de un clasificador utilizando aprendizaje automático visto como sistema.	46
5.5. Gráfica de la función sigmoide.	47
5.6. Ejemplo de frontera de decisión para un problema de clasificación binaria.	48
5.7. Cambios en la función de costo para SVM.	49
5.8. Frontera de decisión utilizando SVM.	50
5.9. Proyección de un dato $\bar{x}^{(i)}$ sobre el vector $\bar{\theta}$ para un clasificador lineal con $\theta_0 = 0$	50
5.10. Gráfica de valores singulares de distintas realizaciones de \mathbf{X} en función de las características definidas para el problema de clasificación mediante aprendizaje automático.	53
5.11. Frontera de decisión definida por el algoritmo SVM utilizando un kernel sigmoide	54
5.12. Frontera de decisión definida por el algoritmo SVM utilizando un kernel polinomial.	55
5.13. Frontera de decisión definida por el algoritmo SVM utilizando un kernel gaussiano.	55
6.1. Diagrama de bloques del sistema conformador de haz.	58
6.2. Representación en bloque del muestreador aleatorio con sus interfaces.	59
6.3. Diseño de bloques de la propuesta de implementación del muestreador aleatorio en la FPGA.	60
6.4. Representación como bloque del subsistema estimador de DOA con sus correspondientes interfaces.	61
6.5. Representación como bloque del subsistema conformador de haz con sus correspondientes interfaces.	62
6.6. Propuesta de implementación del subsistema conformador de haz en FPGA.	62
7.1. Esquema de un sistema de radio definida por software.	66
7.2.	67
7.3. Diagrama de bloques de un receptor de un sistema SDR.	68

7.4.	Tipos de datos disponibles en GNURadio con su correspondiente identificación de color.	69
7.5.	Conexiones entre bloques. GNURadio Companion indica con color rojo aquellas conexiones no permitidas.	69
7.6.	Representación como bloque del emulador de ARU para realizar las simulaciones del sistema conformador de haz.	70
7.7.	Simulación de transmisión QPSK.	71
7.8.	Diagrama de bloques utilizado para la medición de tasa de error de bit mediante la simulación de un enlace de comunicación con el satélite LilacSat-1.	72
7.9.	Curvas de BER en función de $\frac{E_b}{N_0}$ para distintos valores de errores en la separación de elementos del ARU.	73
8.1.	Propuesta de interfaz para la integración del sistema conformador de haz con el sistema de adquisición.	76

Índice de tablas

7.1. Utilización del procesador correspondiente a cada bloque del sistema conformador de haz.	73
--	----

Resumen

En el siguiente trabajo se realiza el análisis de distintas técnicas de conformación de haz, haciendo un hincapié inicial en el estudio de los algoritmos de estimación de dirección de arribo MUSIC y ESPRIT. Luego se introduce la técnica de muestreo aleatorio, la cual permite realizar estimaciones de los parámetros de las señales recibidas reduciendo por encima de dos órdenes de magnitud la cantidad de muestras necesarias con respecto al **muestreo ideal**. Seguido a esto se realiza el análisis de técnicas de estimación de cantidad de señales recibidas, introduciendo un método de estimación que utiliza el algoritmo de aprendizaje automático de máquinas de vectores de soporte. Por último se analizan propuestas de implementación del sistema conformador de haz en una placa de desarrollo CIAA-ACC con capacidad de distribuir funciones entre una FPGA y un **sistema de procesamiento**, para finalmente mostrar una implementación de todo el sistema diseñado utilizando GNURadio, validando su funcionamiento con simulaciones.

Palabras clave: INSTITUTO BALSEIRO, CONFORMACIÓN DE HAZ, ARREGLO DE ANTENAS EN FASE, ESPRIT, MUSIC, MUESTREO ALEATORIO, APRENDIZAJE AUTOMÁTICO, MÁQUINA DE VECTORES DE SOPORTE, GNURADIO, FPGA

Abstract

This is the title in English:

The thesis must reflect the work of the student, including the chosen methodology, the results and the conclusions that those results allow us to draw.

Keywords: INSTITUTO BALSEIRO, BEAMFORMING, PHASED ARRAY ANTENNA, ESPRIT, MUSIC, RANDOM SAMPLING, MACHINE LEARNING, SUPPORT VECTOR MACHINE, GNU-RADIO, FPGA

Capítulo 1

Introducción

“Mereces lo que sueñas.”

— Gustavo Cerati

1.1. Motivación

En las últimas décadas, el crecimiento exponencial de las redes de comunicaciones inalámbricas motivó el estudio y desarrollo de numerosas técnicas que permitieron una utilización cada vez más eficiente del espectro radioeléctrico. Gran parte de esta demanda se debió a la necesidad de poder contar con el servicio de internet en dispositivos móviles. En la actualidad, este servicio requiere que el usuario se encuentre dentro del área de cobertura de alguna de las antenas instaladas por el proveedor, sin embargo, en los últimos años, múltiples empresas como SpaceX, Facebook y Amazon [2] comenzaron a invertir en construir redes de satélites de baja órbita (LEO) buscando un cambio de paradigma en las comunicaciones satelitales.

En sus 60 años de desarrollo, las comunicaciones satelitales no lograron aún alcanzar la capacidad necesaria para brindar una infraestructura que permita proveer servicios de alta demanda, como es el caso de internet. Debido a esto, en la actualidad estas siguen consideradas como enlaces de comunicaciones de respaldo o de servicios de radiodifusión, como es el caso de la televisión satelital. Una de los motivos por los cuales las comunicaciones satelitales vieron limitado su crecimiento se debe al uso de órbitas geoestacionarias (GEO). Estas órbitas tienen la ventaja de tener una gran pisada pudiendo abastecer del servicio a una gran cantidad de usuarios, y además se encuentran ubicados en un punto fijo con respecto a la rotación de la Tierra, lo cual se transfiere en una mayor simplicidad en la recepción debido al uso de antenas estáticas. Sin embargo estos enlaces tienen una gran complicación debido a la distancia entre el transmisor y el receptor, las cuales se ubican por encima de los 35000 km. Esta distancia provoca una latencia mínima por camino de ida y vuelta que se ubica por encima de los 200 ms, lo cual hace que sea poco práctica su utilización para servicios de baja latencia.

La solución al problema de las órbitas GEO consiste en construir redes de satélites que se encuentren a una menor altura, utilizando órbitas LEO, cuyas distancias de enlace se encuentran típicamente entre los 200 km y los 2000 km. Estos enlaces proveen una latencia que se encuentra en el orden de los 10 ms, necesaria para implementar servicios de baja latencia. La complicación en estos enlaces viene dada por la pequeña pisada que tienen las antenas de los satélites, por ubicarse a una altura mucho menor que en el caso de los satélites GEO, y por el hecho de que, a diferencia de los GEOS, estos satélites no se encuentran fijos con respecto a la Tierra y, en cambio, la orbitan, lo cual requiere que el receptor debeat realizar un seguimiento de su trayectoria.

La manera típica de realizar la recepción de un satélite LEO consiste en utilizar antenas de gran ganancia, y por lo tanto gran tamaño, que apuntan mecánicamente al satélite durante su pasada. Esta técnica permite el seguimiento y, por ende, la recepción de un único satélite por antena. Sin embargo existen otras técnicas que permiten la recepción direccional simultánea de múltiples satélites utilizando una única antena estática, como es el caso de la técnica de *conformación de haz*, tema principal de estudio de este trabajo.

Las proyecciones indican que durante el resto de esta década decenas de miles de satélites LEO serán lanzados para brindar, no solo servicios de comunicaciones, sino, también, para realizar observaciones terrestres. Para dar un ejemplo, Starlink ya cuenta con habilitación para lanzar 12000 satélites. Esta predicción indica que la capacidad de dar soporte a satélites por parte de las estaciones terrenas deberá también aumentar. Por ende, poder utilizar una única antena estática para realizar la recepción simultánea de múltiples señales puede resultar en una alternativa que no solo permita reducir costos, sino que también puede permitir aumentar ganancias debido a la capacidad de brindar soporte simultáneo a más usuarios.

1.2. Objetivos de proyecto

El Departamento de Telecomunicaciones del Instituto Balseiro tiene en planes la construcción de una estación terrena satelital en el Centro Atómico Bariloche para la recepción de satélites LEO utilizando la técnica de conformación digital de haz. Para ello se solicita el estudio de esta tecnología para poder realizar una posterior implementación de un sistema conformador de haz que cumpla con los siguientes requerimientos:

- Ser capaz de estimar la dirección de arribo y la cantidad de señales satelitales recibidas a partir de las muestras obtenidas.
- Debe pensarse en un diseño orientado, en un principio, para la utilización de un arreglo de antenas rectangular uniforme de 16 elementos, pero con posibilidad de escalamiento.
- Deberá ser capaz de funcionar en una placa de desarrollo CIAA-ACC, mostrada en la Figura 1.1a, la cual recibe las señales de los elementos a través de una interfaz de adquisición AD9249, la cual se muestra en la Figura 1.1b.

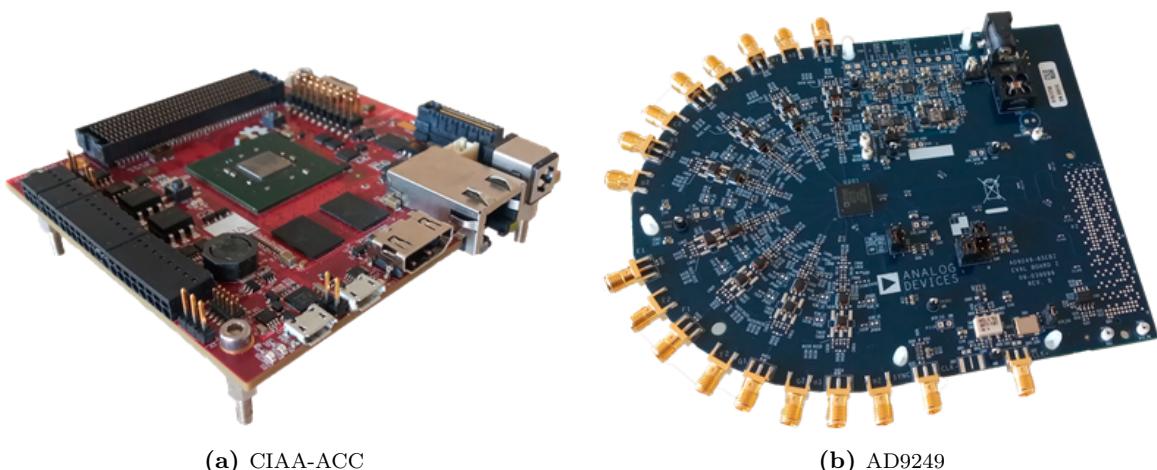


Figura 1.1: Hardware donde se debe implementar el sistema a desarrollar.

El objetivo de este proyecto es el de realizar el estudio de las distintas técnicas de conformación digital de haz para definir la manera óptima de realizar una implementación bajo los requerimientos indicados, validando mediante simulaciones las decisiones de diseño adoptadas.

Capítulo 2

Conformación de haz

“Never stopping. Never being satisfied. Never giving up. And if you keep pushing and keep moving forward, you’re gonna go to places you never even dreamed of.”

— Johnny Lawrence (Cobra Kai)

2.1. Introducción

Al momento de realizar una comunicación inalámbrica punto a punto uno de los aspectos que más contribuyen a la calidad del enlace es su dirección con respecto a la orientación de las antenas tanto del transmisor como del receptor. La capacidad de una antena de transformar en potencia eléctrica la energía recibida en forma de onda electromagnética en una cierta dirección viene caracterizada por su *patrón de radiación*. Este patrón caracteriza, también, el efecto recíproco, es decir, la capacidad de una antena de convertir en energía radiada en una dirección particular la potencia eléctrica con la cual se la alimenta. Es por esto que a la hora de diseñar un enlace inalámbrico se busca que la dirección de mayor radiación de la antena transmisora y de la receptora, es decir, el *lóbulo principal* del patrón de radiación, coincida con la dirección del enlace, como se muestra en la Figura 2.1, de manera tal de lograr la mayor eficiencia en la transmisión de energía, lo cual repercute en una mejor relación señal a ruido (*SNR*, por sus siglas en inglés) en la señal recibida.

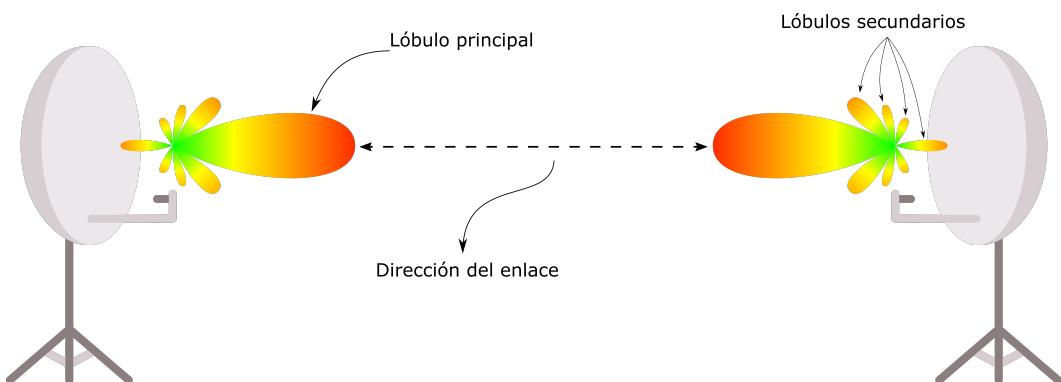


Figura 2.1: Representación de un enlace punto a punto utilizando antenas parabólicas, indicando sus correspondientes patrones de radiación junto con sus lóbulos principales y secundarios.

Lograr la mejor orientación de las antenas cuando el transmisor y el receptor se encuentran estáticos no conlleva mayores dificultades. Las complicaciones aparecen cuando uno de los dos o ambos se

encuentran en movimiento. En este caso la solución que permite aumentar la eficiencia del enlace implica que al menos una de las antenas dirija su patrón de radiación de manera tal de poder hacer un seguimiento del objetivo con el cual se desea establecer un enlace de comunicaciones. Esto se puede lograr utilizando antenas móviles, como es el caso de las antenas parabólicas de las estaciones terrenas que se comunican con satélites de baja y media órbita, como las que se muestran en la Figura 2.2.



Figura 2.2: Antenas parabólicas móviles de la estación terrena perteneciente al Centro Espacial Teófilo Tabanera [1].

Otra manera de lograr la orientación de los patrones de radiación de las antenas es mediante la técnica de *conformación de haz*, principal objeto de estudio de esta monografía. Esta técnica consiste en emular el comportamiento de una antena direccional sintetizando patrones de radiación arbitrarios usando antenas estáticas. Esto se consigue utilizando un *arreglo de antenas en fase*, el cual consiste en un conjunto de antenas, generalmente idénticas, dispuestas en una geometría particular y con la capacidad de poder variar la fase relativa de la señal transmitida entre elementos, de manera tal de poder generar interferencias constructivas en la dirección en la que se quiere orientar el haz y destructivas en las direcciones desde las cuales se están recibiendo interferencias [3].

La técnica de conformación de haz se puede aplicar a cualquier recepción o transmisión punto a punto de señales, sin embargo en este trabajo se orientará el estudio a la implementación de un conformador de haz para realizar comunicaciones con satélites de baja órbita. Para esto es necesario primero definir un sistema de coordenadas útil para describir la dirección de arriba de señales al momento de implementar una comunicación satelital. Este sistema es conocido por *coordenadas horizontales* y está definido por [4]:

- **Azimut:** es el ángulo tomado sobre el plano horizontal de la estación terrena midiendo desde el norte hacia la proyección de la dirección del satélite sobre el mismo plano en sentido horario. A lo largo de este trabajo se lo indicará con la letra griega φ .
- **Elevación:** es el ángulo formado entre la dirección del satélite y el plano horizontal. A lo largo

de este trabajo se lo indicará con la letra griega θ .

En la Figura 2.3 se muestra un esquema de este sistema de coordenadas.

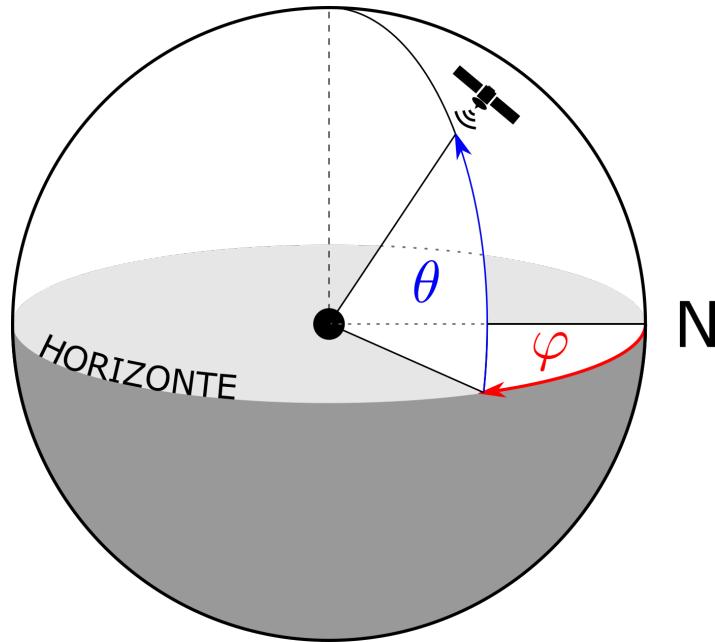


Figura 2.3: Sistema de coordenadas horizontales para comunicaciones satelitales.

Para comenzar a explicar en qué consiste la técnica de conformación de haz consideremos que tenemos un conjunto de antenas M idénticas dispuestas sobre una línea y equidistantes unas de otras, a las cuales les llega un frente de onda con un cierto ángulo θ con respecto a la vertical, como se muestra en la Figura 2.4.

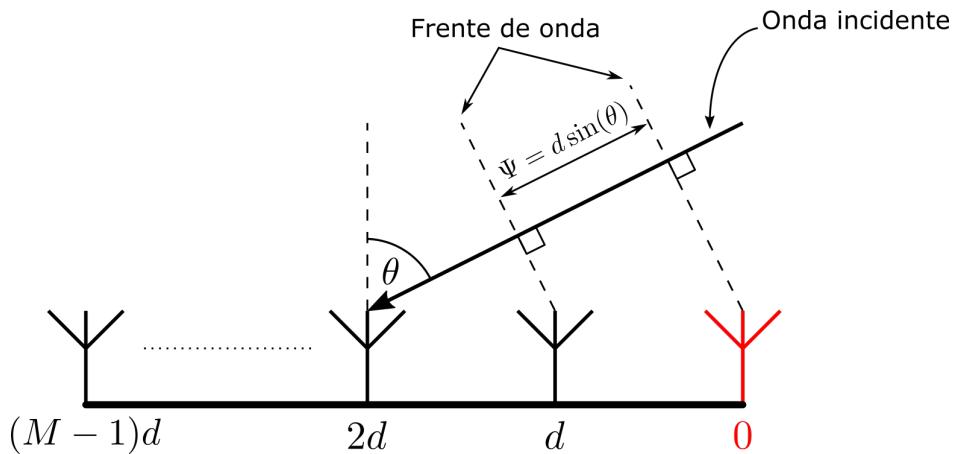


Figura 2.4: Frente de onda plano arribando a un arreglo lineal de antenas. Se indica en rojo el elemento de referencia, en el cual se considera que la señal arribante tiene fase nula.

Considerando que la señal proviene desde una distancia lejana (comparada con el tamaño del arreglo de antenas) podemos afirmar con gran precisión que el frente de onda que llega al arreglo de antenas es plano. La dirección de arribo de la señal, o *DOA* por sus siglas en inglés, en este caso definida únicamente por el ángulo θ , provoca que el frente de onda recorra distintas distancias al llegar a cada elemento del arreglo, y esa diferencia en las distancias se traduce en un desfasaje en la señal recibida por cada receptor. Por ende, teniendo en cuenta que la separación entre elementos del arreglo es la misma y que el frente de onda se lo puede considerar plano, y eligiendo convenientemente

al primer elemento del arreglo al cual llega el frente de onda como elemento de referencia, podemos obtener la diferencia de camino recorrido por la señal transmitida al llegar a cada receptor haciendo:

$$\Psi_m = m \cdot d \sin(\theta), \quad m = 0, 1, 2, \dots, (M - 1), \quad (2.1)$$

siendo d la distancia entre elementos y M la cantidad de elementos en el arreglo.

La forma de onda de una onda viajera en campo lejano recibida por un receptor puntual, considerando un transmisor también puntual e isotrópico, puede ser expresada por la magnitud de su campo eléctrico como [5]:

$$E(\bar{r}, t) = s(t) e^{j(\omega_p t - \bar{k} \cdot \bar{r})}, \quad (2.2)$$

siendo:

- \bar{r} : el vector que une al transmisor con el receptor,
- $s(t)$: la envolvente compleja de la señal transmitida en función del tiempo,
- $\omega_p = 2\pi f_p$: la frecuencia angular de la portadora,
- \bar{k} : el vector de onda, con $|\bar{k}| = \frac{2\pi}{\lambda_p}$,
- λ_p : la longitud de onda de la portadora.

Considerando el mismo arreglo de antenas de la Figura 2.4 podemos expresar la onda recibida por el m -ésimo elemento como:

$$E_m(\bar{r}_m, t) = s(t) e^{j(\omega_p t - \bar{k} \cdot \bar{r}_m)}, \quad m = 0, 1, 2, \dots, (M - 1), \quad (2.3)$$

siendo \bar{r}_m el vector que une el transmisor con el m -ésimo receptor. Es necesario aclarar que para que esta expresión sea válida es necesario que se cumpla la condición de que la señal $s(t)$ sea de banda angosta [5], lo cual significa que su ancho de banda debe ser al menos uno o dos órdenes de magnitud menor a la inversa del tiempo que le lleva al frente de onda propagarse desde el primer al último elemento del arreglo. Esto permite suponer que la envolvente de la onda transmitida no varía demasiado en el tiempo que le lleva al frente de onda alcanzar todos los elementos del arreglo, y en cambio se la puede considerar constante. Dicho de otra forma, puede considerarse que $s(t) \approx s(t - \tau_{\max})$, siendo τ_{\max} el tiempo que le lleva al frente de onda ir desde el primer al último elemento del arreglo.

Por conveniencia, para el resto del análisis se quitará el término $e^{j\omega_p t}$ correspondiente a la variación de la señal debido a la portadora para trabajar con la señal en banda base, la cual se denotará con la letra x . Si tomamos como referencia al primer elemento al cual le llega el frente de onda de la señal transmitida podemos considerar que la fase de la señal recibida por este es nula, y expresar al resto de las señales recibidas por los demás elementos en función de la señal de referencia como:

$$x_m(\theta, t) = s(t) e^{-j \cdot m \cdot k \cdot d \sin(\theta)}, \quad m = 0, 1, 2, \dots, (M - 1), \quad (2.4)$$

donde se observa que la señal recibida por el m -ésimo receptor difiere con respecto a las recibidas por el resto de los receptores únicamente en una fase, la cual además depende únicamente del ángulo de arribo θ y de la separación entre elementos d . Esto induce a pensar que si se conocen estos dos parámetros (la dirección de arribo y la geometría en la que están dispuestas las antenas receptoras) se podría conocer fácilmente las fases relativas entre todas las señales, pudiendo así corregirlas y sumarlas de manera tal de poder aumentar la relación señal a ruido en la señal recibida por el conjunto de antenas. A esta técnica se la conoce como *conformación de haz*.

Más allá de que este análisis se hizo teniendo en cuenta muchas suposiciones y únicamente para el caso de una disposición de elementos del arreglo de antenas en una única dimensión, más adelante se mostrará que este mismo análisis vale para casos generales, particularmente con distintas configuraciones de arreglos de antenas en dos dimensiones. También el mismo análisis se puede aplicar para la transmisión direccional de señales utilizando arreglos de antenas, sin embargo esto último no es motivo de estudio del presente trabajo.

A lo largo de este capítulo se desarrollará toda la teoría detrás de la conformación de haz, haciendo hincapié en los distintos tipos de técnicas que existen para su implementación, definiendo propiamente a los arreglos de antenas en fase y detallando los dos algoritmos de estimación de dirección de arribo que fueron de mayor importancia para la realización de este proyecto.

2.2. Clasificación de conformadores de haz

Según cómo se realice la implementación, los conformadores de haz pueden recibir distintas clasificaciones. Si se pone el foco en la manera de manipular las señales recibidas podemos distinguir entre conformadores de haz analógicos y digitales [6]. En el caso del conformador de haz analógico la señal que llega a cada elemento del arreglo de antenas pasa por un desfasador analógico que permite compensar las fases relativas de las señales recibidas cuando el frente de onda llega con una cierta dirección a cada elemento. Luego de esto, se combinan todas las señales con un combinador de potencias y solo se digitaliza la salida de este. Si se cuenta con M elementos en el arreglo, este método reduce la dimensión de la señal recibida de M a 1, reduciendo también así gran parte de la información recibida y solo permitiendo la recepción en una única dirección. Además de corregir las fases de las señales recibidas se puede también variar las ganancias de cada elemento de manera tal de generar interferencias destructivas en las direcciones donde queremos eliminar cualquier tipo de interferencia. Un esquema de este conformador se muestra en la Figura 2.5a, donde los pesos p_n son los números complejos que multiplican la señal recibida por cada elemento del arreglo para compensar las fases relativas y generar ganancias nulas en las direcciones de interferencias, de manera tal de poder así sintetizar el correspondiente patrón de radiación. Los pesos pueden ser representados en un vector \bar{p} de tamaño $M \times 1$ al cual se lo llama *arreglo de pesos*.

En el conformador digital de haz las señales son muestreadas y digitalizadas para luego alimentar a un procesador digital de señales, o *DSP* por sus siglas en inglés. En este caso se preserva la información disponible manteniendo todas las señales recibidas por cada elemento. De esta manera la dimensión de la señal recibida se mantiene, lo cual brinda una gran flexibilidad para operar con las muestras recibidas, permitiendo obtener grandes prestaciones que no se encuentran disponibles en su contraparte analógica. Algunas de ellas son la capacidad de rechazar automáticamente las interferencias o de estimar automáticamente la dirección de arribo de las señales de interés, la posibilidad de generar múltiples haces con un único conformador, lo cual permite recibir señales en múltiples direcciones de arribo, la capacidad de realizar una calibración de las antenas por procesamiento o la posibilidad de incluir inteligencia artificial en la conformación de haz, tema que se abordará con mayor detalle en la Sección 8.4.

Según la técnica utilizada, los conformadores de haces pueden también clasificarse en convencionales o adaptativos [7]. En el caso de los conformadores de haz convencionales el arreglo de pesos y fases relativas se encuentra fijos, lo cual no permite realizar una adaptación a cambios en el tiempo ya sea en la dirección de arribo de la señal u otros cambios que puedan ocurrir en las características del arreglo o en el medio de transmisión. En cambio, los conformadores adaptativos utilizan las propiedades estadísticas de la señal y del medio para variar los desfasajes y los pesos del filtro adaptativo y así poder hacer seguimiento de señales y mejorar la SNR.

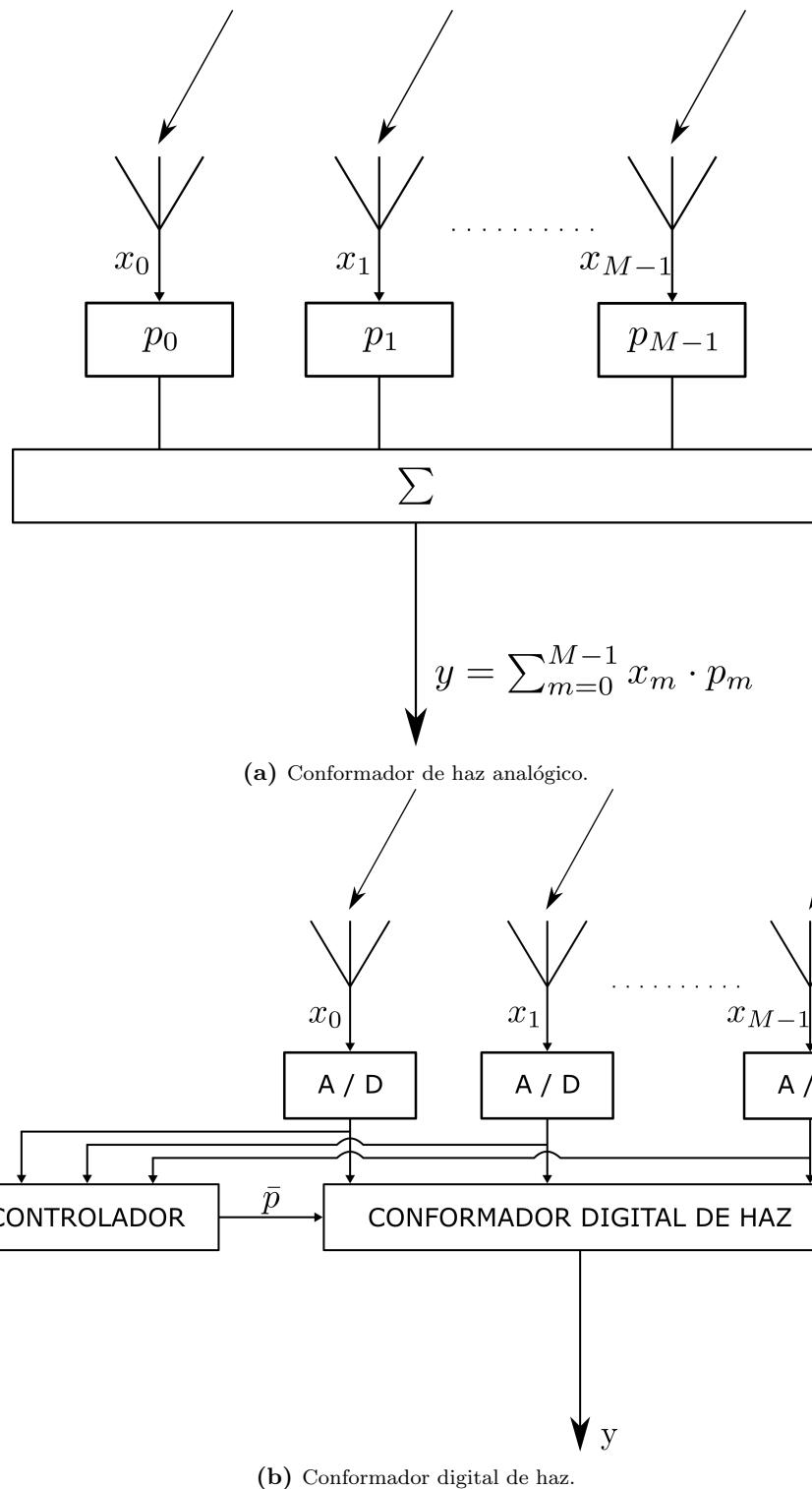


Figura 2.5: Clasificación de conformadores de haz según cómo manipulan la señal recibida.

A lo largo de este trabajo se estudiará la implementación de un conformador digital de haz adaptativo.

2.3. Arreglos de antenas en fase

Los arreglos de antenas en fase consisten en un conjunto de antenas estacionarias (elementos) dispuestas en una distribución unidimensional o bidimensional y que utilizan un control de variación de fases o retrasos temporales en cada elemento para escanear un haz en una dirección dada en el espacio y un control de amplitudes para dar forma al patrón de radiación [8]. Como ya se dijo, su principal uso se debe a la posibilidad que tienen de sintetizar un patrón de radiación direccional que puede ser dirigido electrónicamente.

La dimensión en la que están dispuestos los elementos del arreglo definen el direccionamiento que se le puede dar al haz sintetizado. En el caso de una disposición unidimensional solo se podrá dirigir el haz en función de la elevación, en cambio en disposiciones bidimensionales el haz se puede dirigir tanto en elevación como en azimut.

A primera vista, pareciera ser que este tipo de antenas solo ofrece ventajas comparadas con las antenas de apertura fija. La relación de compromiso está en la dificultad de fabricación, ya que los arreglos de antenas en fase tienen complicaciones que no existen en otras antenas, como la necesidad de que no existan desfasajes en las conexiones entre los elementos o el problema del acoplamiento mutuo entre antenas debido a la escasa separación que existe entre unas y otras.

2.3.1. Tipos de arreglos

A pesar de que los elementos de un arreglo de antenas en fase se pueden ubicar arbitrariamente, existen ciertas distribuciones que habilitan una utilización más simple de técnicas que son de gran utilidad al trabajar con la conformación de haz, como lo es la posibilidad de utilizar algoritmos de estimación de dirección de arribo. Generalizando la expresión de la Ecuación 2.4 para el caso de elementos no isotrópicos, y suponiendo que la respuesta en frecuencia de los mismos es plana en todo rango de frecuencias de interés, podemos ahora expresar a la señal recibida por cada elemento del arreglo como un vector definido como:

$$\bar{x}(\theta, t) = \bar{a}(\theta) \cdot s(t) + \bar{w}(t), \quad (2.5)$$

$$\bar{a}(\theta) = g(\theta) \cdot \begin{bmatrix} 1 & e^{-j\bar{k} \cdot \bar{r}_0} & \dots & e^{-j\bar{k} \cdot \bar{r}_{M-1}} \end{bmatrix}^T \quad (2.6)$$

donde $\bar{a}(\theta)$ es el *vector de apuntamiento*, el cual aplica los correspondientes desfasajes a las señales de cada elemento y afecta su amplitud según la ganancia que tenga el arreglo de antenas en fase en la dirección de arribo de la señal, $g(\theta)$ es la ganancia de cada elemento del arreglo en la dirección θ si consideramos que todos los elementos son idénticos, \bar{k} es el vector de onda del campo incidente y \bar{r}_m con $m = 0, 1, \dots, M - 1$ es el vector que va desde el origen de coordenadas hasta el elemento m -ésimo del arreglo. Como se puede ver, este vector de apuntamiento depende de la disposición del arreglo empleada. El vector $\bar{w}(t)$ es un vector de ruido aditivo que estará presente en cualquier implementación.

A continuación se mencionan las características de los tipos de arreglos más comunes.

Arreglo lineal uniforme

Dentro de las distribuciones regulares, el arreglo lineal uniforme (ALU) es la más simple de todas. Este arreglo consiste en disponer a los elementos colinealmente y separados a una misma distancia unos de otros, como se vio en el ejemplo de la Sección 2.1. Un esquema de este tipo de arreglos se muestra en la Figura 2.6. Al ser una geometría unidimensional solo permite direccionar el haz en la dimensión definida por el ángulo θ .

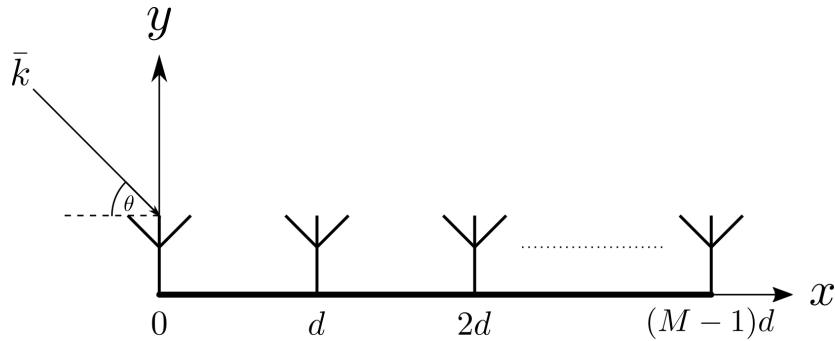


Figura 2.6: Arreglo lineal uniforme.

Si tomamos como el ángulo de arriba al indicado en la Figura 2.6, el vector de apuntamiento queda definido por:

$$\bar{a}_{ALU}(\theta) = g(\theta) \cdot \begin{bmatrix} 1 & e^{-jkd \cos \theta} & \dots & e^{-j(M-1)kd \cos \theta} \end{bmatrix}^T \quad (2.7)$$

donde $g(\theta)$ es la ganancia de cada elemento del arreglo en la dirección θ si consideramos que todos los elementos son idénticos.

Arreglo circular uniforme

Como su nombre lo indica, los arreglos circulares uniformes (ACU) consisten en disponer los elementos sobre una circunferencia, equidistante uno del otro, como se muestra en la Figura 2.7. Esta disposición bidimensional permite el escaneo tanto en elevación como en azimut de las señales arribantes, y tienen la ventaja de que, por su simetría, permiten que el patrón de radiación sintetizado pueda ser rotado azimutalmente sin sufrir variaciones en su forma [9].

Para el caso bidimensional podemos definir al vector de onda \bar{k} correspondiente a la señal que arriba al arreglo en función del ángulo de elevación y el azimut como:

$$\bar{k} = k \begin{pmatrix} \cos(\theta) \cdot \cos(\varphi) & \cos(\theta) \cdot \sin(\varphi) \end{pmatrix} \quad (2.8)$$

El vector \bar{r}_m considerando como origen de coordenadas el centro de la circunferencia que contiene al arreglo puede definirse como:

$$\bar{r}_m = R \begin{pmatrix} \cos\left(\frac{2\pi \cdot m}{M}\right) & \sin\left(\frac{2\pi \cdot m}{M}\right) \end{pmatrix} \quad (2.9)$$

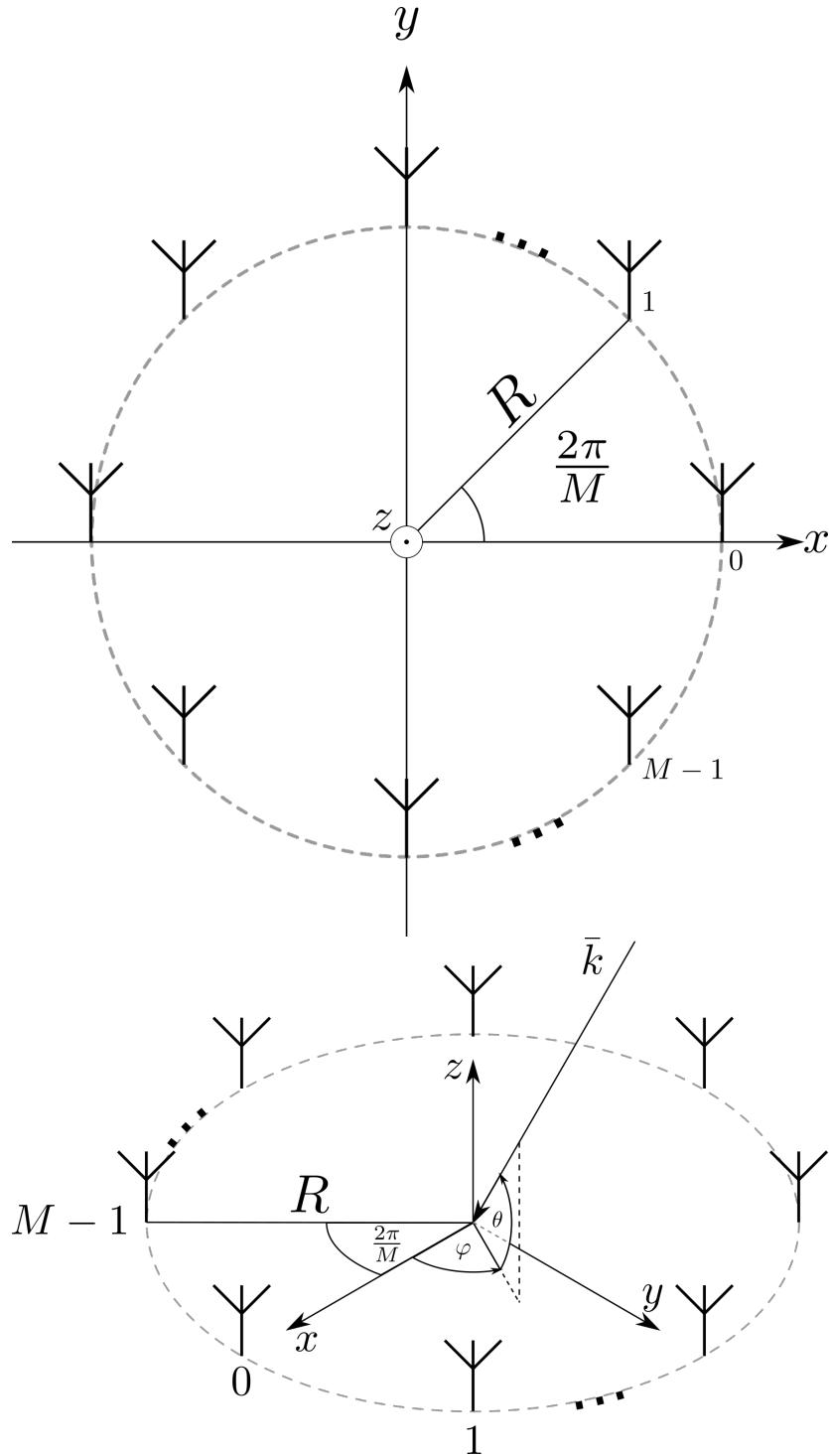
Entonces el vector de apuntamiento queda definido como:

$$\bar{a}_{ACU}(\theta, \varphi) = g(\theta, \varphi) \cdot \begin{bmatrix} e^{-jkR \cos \theta \cos \varphi} & \dots & e^{-jkR(\cos(\frac{2\pi \cdot m}{M}) \cos \theta \cdot \cos \varphi + \sin(\frac{2\pi \cdot m}{M}) \cos \theta \cdot \sin \varphi)} & \dots \end{bmatrix}^T \quad (2.10)$$

con $m = 0, 1, \dots, M - 1$. Debe notarse que ahora la directividad de la antena queda expresada en función de la elevación y del azimut.

Arreglo rectangular uniforme

El arreglo rectangular uniforme (ARU) es el tipo de arreglo de antenas en fase bidimensional más utilizado, debido a que permite contar con la mayor cantidad de elementos en un menor espacio, aumentando la *resolución* que se puede alcanzar tanto en elevación como azimut, definiendo la resolución como la posibilidad de distinguir dos fuentes de señal poco espaciadas [5]. Esta geometría consiste en ubicar a los elementos en una grilla rectangular, manteniendo una misma distancia d entre uno con su

**Figura 2.7:** Arreglo circular uniforme.

adyacente. Un esquema de este tipo de arreglos se muestra en la Figura 2.8.

El vector \bar{r}_{m_x, m_y} que une al elemento ubicado en las coordenadas $d \cdot (m_x, m_y)$ con el origen de coordenadas, el cual está ubicado en uno de los vértices del arreglo que se considera como elemento de referencia, puede definirse como:

$$\bar{r}_{m_x, m_y} = d \begin{pmatrix} m_x & m_y \end{pmatrix} \quad (2.11)$$

Debido a que en este caso tenemos una disposición bidimensional de elementos se debe elegir una

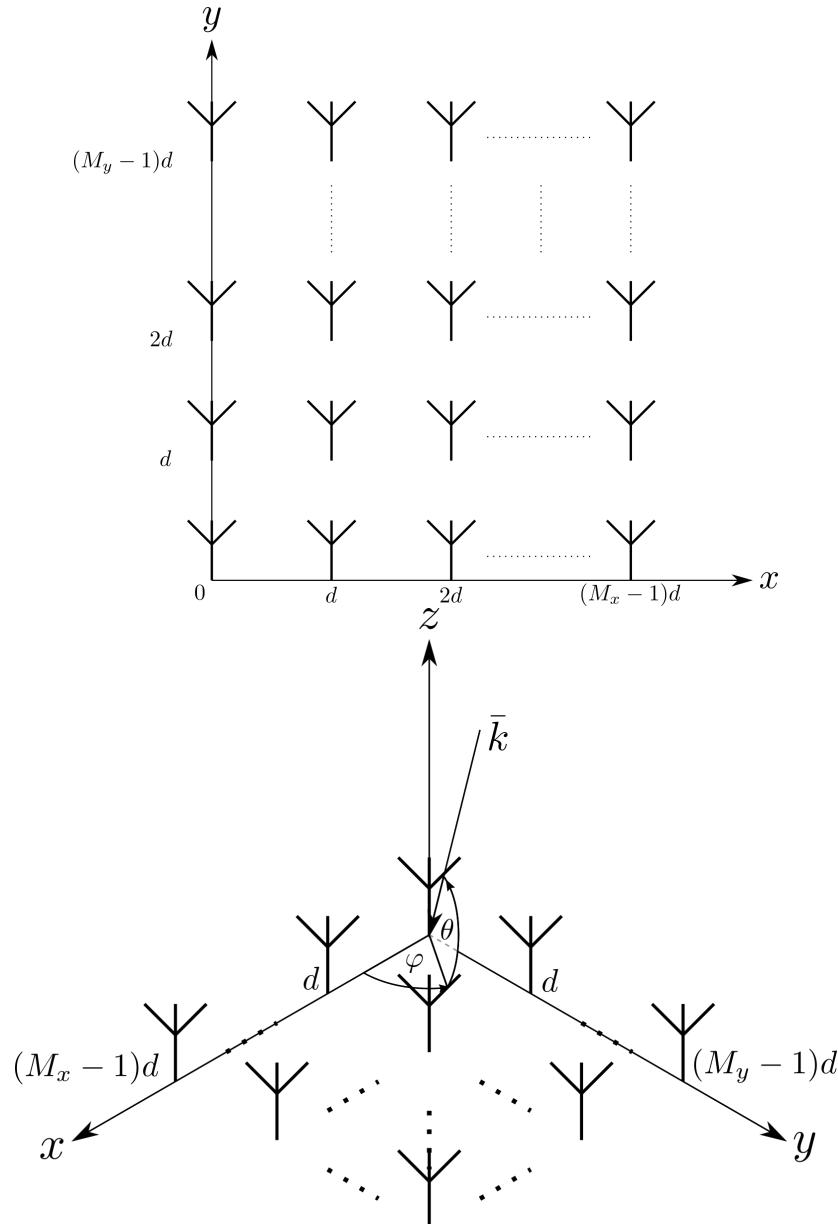


Figura 2.8: Arreglo rectangular uniforme.

convención para identificar a cada uno de ellos con una ubicación de un vector. La convención elegida es la de recorrer cada una de las columnas a lo largo del eje definido como y , de manera que el vector de muestras puede definirse como:

$$\bar{x}(t) = \begin{bmatrix} x_{0,0}(t) & \cdots & x_{0,M_y-1}(t) & \cdots & x_{M_x-1,0}(t) & \cdots & x_{M_x-1,M_y-1}(t) \end{bmatrix}^T \quad (2.12)$$

Teniendo en cuenta lo anterior y utilizando el vector de onda definido en la Ecuación 2.8, podemos escribir el correspondiente vector de apuntamiento como:

$$\bar{a}_{ARU}(\theta, \varphi) = g(\theta, \varphi) \cdot \begin{bmatrix} 1 & \cdots & e^{-jkd[(M_x-1)\cos\theta\cos\varphi + (M_y-1)\cos\theta\sin\varphi]} \end{bmatrix}^T \quad (2.13)$$

El arreglo rectangular uniforme es el tipo de arreglo más importante para el resto de este trabajo ya que es el elegido para realizar la implementación del sistema de conformación de haz que se busca implementar.

2.4. Definición del problema

Según los conceptos detallados a lo largo de este capítulo se puede definir el problema a resolver a partir de los objetivos planteados en la Sección 1.2 diciendo que se debe realizar la estimación de la dirección de arriba tanto en elevación (θ) como en azimut (φ) de una señal proveniente de un satélite LEO que es recibida por un ARU de 16 elementos dispuestos en una matriz de 4×4 para alimentar un conformador de haz digital adaptativo que entregue como salida dicha señal en tiempo real. Un esquema de este problema puede verse en la Figura 2.9.

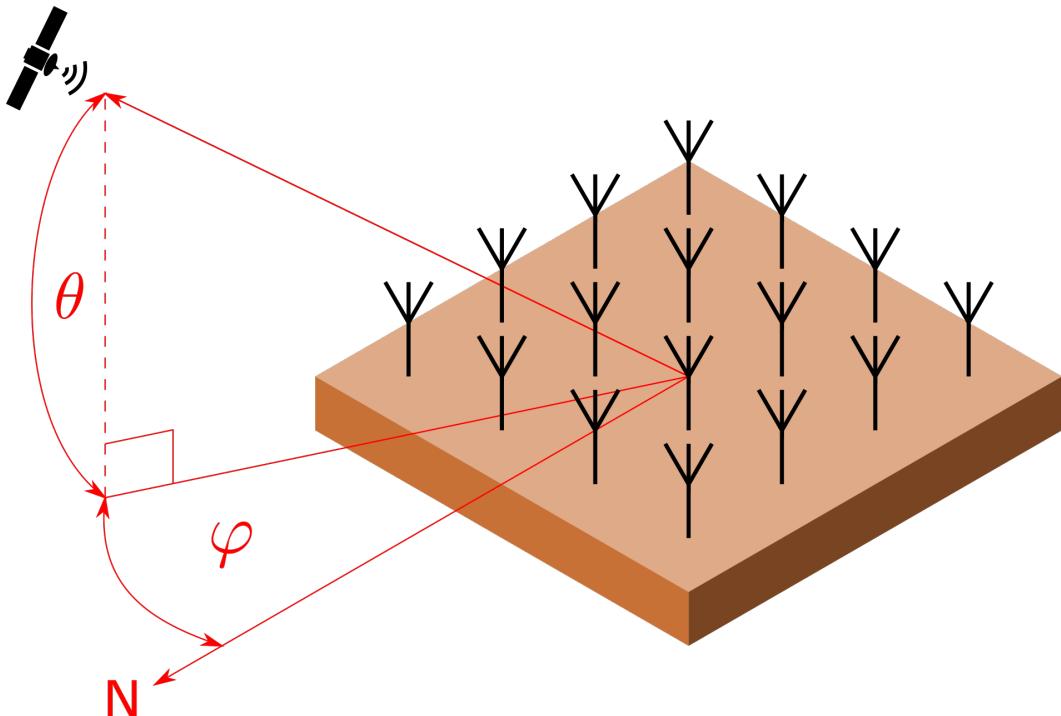


Figura 2.9: Esquema del problema a resolver.

Para realizar la conformación de haz se realizará la estimación de la dirección de arriba utilizando un algoritmo apropiado, el cual brindará esa información al conformador que se encargará de entregar la señal resultante. En la siguiente sección se hará un análisis de dos algoritmos de estimación de dirección de arriba, haciendo hincapié en las comparaciones de rendimiento y la fiabilidad entre ambos.

Capítulo 3

Algoritmos de estimación de dirección de arribo.

“Son nuestras elecciones, Harry, las que muestran lo que somos, mucho más que nuestras habilidades.”

— Albus Dumbledore

3.1. Introducción

Al momento de diseñar un conformador de haz adaptativo uno de los pasos más importantes es definir el algoritmo de estimación de dirección de arribo (DOA). Estos algoritmos explotan las propiedades estadísticas de las señales recibidas por los elementos del arreglo de antenas y entregan la dirección de arribo en las dimensiones correspondientes según el tipo de arreglo con el que se trabaje. El estudio de los algoritmos de estimación de DOA data de la Segunda Guerra Mundial con la aparición del conformador de Bartlett [5], el cual consiste en hacer un escaneo de potencia en el dominio de búsqueda identificando la dirección que maximiza la potencia de salida. Este algoritmo pertenece al grupo de algoritmos basados en el análisispectral, los cuales tienen la desventaja de que su resolución depende fuertemente del ancho del haz sintetizado y que requieren de una búsqueda en una o dos dimensiones para realizar la estimación, lo cual reduce su viabilidad si es que se los quiere utilizar para seguimientos de objetivos en tiempo real. Posteriormente, nuevos métodos fueron propuestos, mejorando así el desempeño en la detección y en la eficiencia, como son los métodos basados en la separación de subespacios de señal y ruido, los cuales se basan en explotar las propiedades geométricas del arreglo de antenas y tienen la ventaja de que su resolución no está limitada a la apertura del mismo. Dentro de este tipo de algoritmos de estimación de DOA, los algoritmos *Multiple Signal Classification (MUSIC)* y *Estimation of Signal Parameters via Rotational Invariance Techniques (ESPRIT)* son dos de los más importantes, y son, además, los que se estudiarán a lo largo de este capítulo.

La disponibilidad en la utilización de un algoritmo dependerá fuertemente del tipo de arreglo con el que se cuente para la implementación, es por esto que a la hora de elegir la manera de disponer los elementos del arreglo para realizar un conformador de haz adaptativo hay que tener en cuenta cuáles distribuciones son aquellas que brindan mejores opciones para la estimación de DOA. Para dar un ejemplo, el algoritmo *Root-MUSIC* tiene un costo computacional relativamente bajo para lograr la estimación, ya que su solución consiste simplemente en encontrar las raíces de un polinomio, pero solo funciona en arreglos lineales uniformes, por ende solo permite estimar la dirección de arribo en

una única dimensión.

Vale la pena aclarar que las técnicas utilizadas en este capítulo permiten, también, obtener información de otros parámetros de las señales arribantes, como pueden ser la frecuencia de portadora de las mismas o la cantidad de fuentes de señal que está recibiendo el arreglo de antenas, razón por la cual estos algoritmos resultan muy versátiles en el mundo del procesamiento estadístico de señales.

Para realizar el análisis de los algoritmos mencionados se deberá primero definir un modelo de datos común que refleje matemáticamente cómo pueden ser representadas las muestras obtenidas por el arreglo de antenas.

3.1.1. Modelo de datos

Antes de especificar el modelo de datos es necesario aclarar que para los próximos análisis se considerará que el medio de transmisión es lineal, y, por ende, vale el principio de superposición. Debido a esto, si consideramos que a un arreglo de M elementos está llegando un número D de señales desde distintas direcciones, la Ecuación 2.5 puede reescribirse como:

$$\bar{x}(t) = \sum_{d=0}^{D-1} \bar{a}(\theta_d, \varphi_d) \cdot s_d(t) + \bar{w}(t), \quad (3.1)$$

donde $\bar{a}(\theta_d, \varphi_d)$ con $d = 0, 1, \dots, D - 1$ es el vector de apuntamiento que corresponde a la señal $s_d(t)$ llegando al arreglo con dirección (θ_d, φ_d) , la cual es medida en el elemento de referencia. Esta ecuación puede reescribirse en forma matricial haciendo:

$$\bar{x}(t) = \mathbf{A}(\theta, \varphi) \bar{s}(t) + \bar{w}(t), \quad (3.2)$$

$$\mathbf{A}(\theta, \varphi) = \begin{bmatrix} \bar{a}(\theta_0, \varphi_0) & \bar{a}(\theta_1, \varphi_1) & \cdots & \bar{a}(\theta_{D-1}, \varphi_{D-1}) \end{bmatrix}_{(M \times D)} \quad (3.3)$$

donde $\bar{s}(t) = [s_0(t), s_1(t), \dots, s_{D-1}(t)]^T$.

Si tomamos una instantánea de las muestras del arreglo vemos que las mismas pueden representarse como un vector de números complejos de la siguiente manera:

$$\begin{aligned} \bar{x} &= \mathbf{A}(\theta, \varphi) \bar{s} + \bar{w}, \\ \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{M-1} \end{bmatrix}_{(M \times 1)} &= \begin{bmatrix} \bar{a}(\theta_0, \varphi_0) & \bar{a}(\theta_1, \varphi_1) & \cdots & \bar{a}(\theta_{D-1}, \varphi_{D-1}) \end{bmatrix}_{(M \times D)} \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{D-1} \end{bmatrix}_{(D \times 1)} + \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{M-1} \end{bmatrix}_{(M \times 1)} \end{aligned}$$

A partir de esto podemos ver que el vector de muestras \bar{x} pertenece a \mathbb{C}^M . Además, en ausencia de ruido, cada vector de muestras puede expresarse como combinación lineal de los vectores de \mathbf{A} , siendo los elementos de \bar{s} los coeficientes de esta combinación. Por ende, si $\bar{w} = \bar{0}$, las muestras estarán confinadas en un subespacio de dimensión D dentro de \mathbb{C}^M , generado por las columnas de \mathbf{A} , las cuales conforman la base del *subespacio de señal* \mathcal{S}_S [10]. Si definimos como \mathfrak{A} al conjunto que contiene a todos los posibles vectores de apuntamiento, para el caso en el que la dirección de arribo es bidimensional, estos vectores definirán una superficie con forma de “sábana” en \mathbb{C}^M , y en el caso unidimensional definirán una curva cerrada. Identificar cuáles de todos los vectores de \mathfrak{A} conforman la base del subespacio de señal corresponde a encontrar las intersecciones entre la superficie M -dimensional formada por \mathfrak{A} y el subespacio de señal [11]. En la Figura 3.1 se muestra una representación para el caso de estimación de DOA unidimensional, con $D = 2$ y $M = 3$. Si se supone que la función

que mapea las posibles direcciones de arriba (θ, φ) a elementos de \mathfrak{A} es inyectiva, encontrar los vectores de \mathbf{A} equivale a encontrar las direcciones de arriba de las D señales recibidas. Esto puede lograrse mediante un diseño apropiado del arreglo de antenas [10]. La dificultad radica ahora en definir \mathcal{S}_S a partir de las muestras obtenidas.

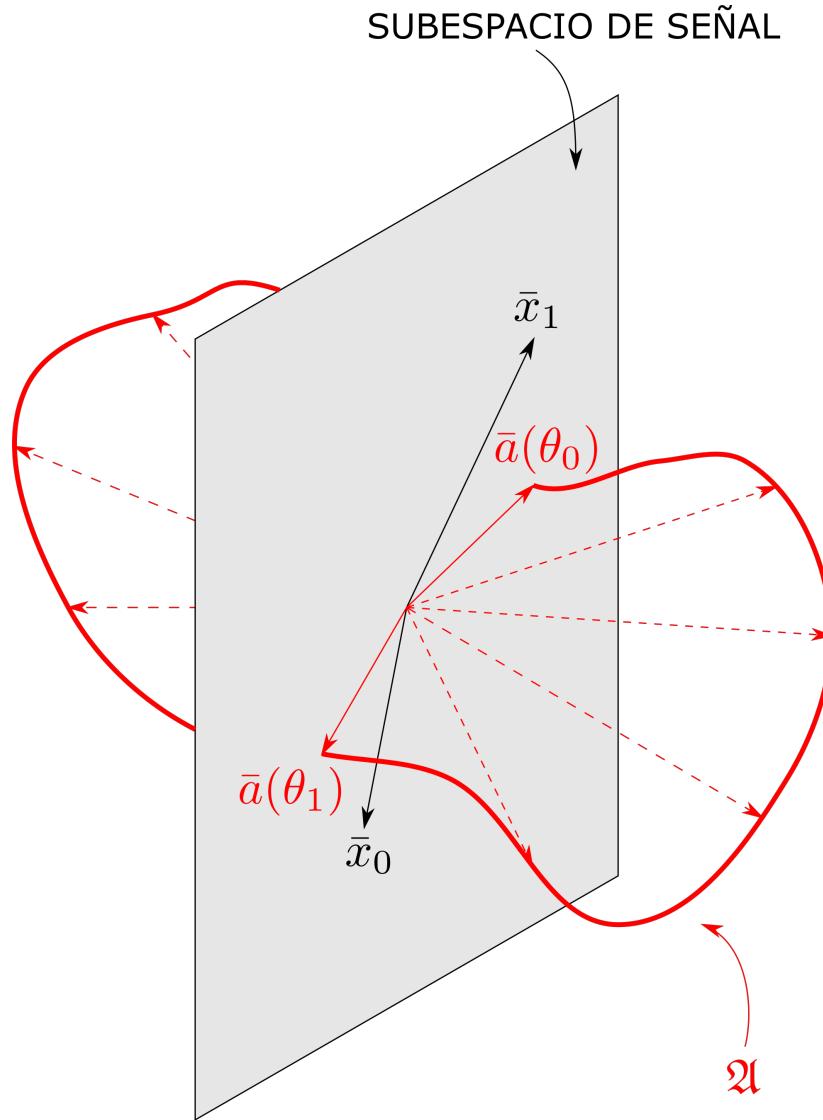


Figura 3.1: Representación geométrica de la estimación de DOA mediante la intersección del conjunto \mathfrak{A} con el subespacio de señal para el caso de DOA unidimensional, con $D = 2$ y $M = 3$.

La matriz de covarianza R_{xx}

Al muestrear digitalmente cada elemento del arreglo tendremos el equivalente a un vector de muestras por cada período de muestreo. Si se quiere representar a todas las muestras tomadas durante

N períodos de muestreo se puede escribir:

$$\mathbf{X}_{(M \times N)} = \mathbf{A}(\theta, \varphi)_{(M \times D)} \times \mathbf{S}_{(D \times N)} + \mathbf{W}_{(M \times N)} \quad (3.4)$$

$$\mathbf{X} = \begin{bmatrix} x_0^0 & x_0^1 & \cdots & x_0^{N-1} \\ x_1^0 & \ddots & & \vdots \\ \vdots & \ddots & & \vdots \\ x_{M-1}^0 & \cdots & \cdots & x_{M-1}^{N-1} \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} s_0^0 & s_0^1 & \cdots & s_0^{N-1} \\ s_1^0 & \ddots & & \vdots \\ \vdots & \ddots & & \vdots \\ s_{D-1}^0 & \cdots & \cdots & s_{D-1}^{N-1} \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} w_0^0 & w_0^1 & \cdots & w_0^{N-1} \\ w_1^0 & \ddots & & \vdots \\ \vdots & \ddots & & \vdots \\ w_{M-1}^0 & \cdots & \cdots & w_{M-1}^{N-1} \end{bmatrix}$$

Para simplificar la notación a partir de ahora se indicará a la matriz de vectores de apuntamiento simplemente como \mathbf{A} .

A partir de esto podemos encontrar la matriz de covarianza de las muestras haciendo:

$$\begin{aligned} \mathbf{R}_{\mathbf{XX}} &= \overline{\mathbf{XX}^H} = \overline{(\mathbf{AS} + \mathbf{W})(\mathbf{AS} + \mathbf{W})^H} = \overline{(\mathbf{AS} + \mathbf{W})(\mathbf{S}^H \mathbf{A}^H + \mathbf{W}^H)} \\ &= \overline{\mathbf{AS} \mathbf{S}^H \mathbf{A}^H} + \overline{\mathbf{AS} \mathbf{W}^H} + \overline{\mathbf{W} \mathbf{S}^H \mathbf{A}^H} + \overline{\mathbf{W} \mathbf{W}^H} \\ &= \mathbf{A} \mathbf{R}_{\mathbf{SS}} \mathbf{A}^H + \mathbf{R}_{\mathbf{WW}} = \mathbf{A} \mathbf{R}_{\mathbf{SS}} \mathbf{A}^H + \sigma_w^2 \mathbf{I}_M \end{aligned} \quad (3.5)$$

donde se supuso que las señales pueden ser modeladas como procesos estocásticos estacionarios y de media cero, y que el ruido es aditivo, blanco y gaussiano, de manera tal que la correlación entre la señal y el ruido es nula. La matriz $\mathbf{R}_{\mathbf{WW}}$ es la matriz de autocorrelación del ruido, siendo \mathbf{I}_M la matriz identidad de tamaño $M \times M$ y $\sigma_w^2 = \mathbf{E}\{|w[n]|^2\}$ el nivel de potencia de ruido. Es necesario aclarar que la operación \mathbf{XX}^H consiste en realizar el promedio entre las matrices obtenidas al multiplicar cada uno de los vectores columna de \mathbf{X} por su transpuesto conjugado, es decir:

$$\mathbf{R}_{\mathbf{XX}} = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=0}^{N-1} \bar{x}^n (\bar{x}^n)^H, \quad (3.6)$$

siendo \bar{x}^n la columna n -ésima de la matriz \mathbf{X} . En este análisis se considera la matriz de correlación teórica, la cual requiere de infinitas muestras para poder ser obtenida. En la práctica, la matriz de correlación $\mathbf{R}_{\mathbf{XX}}$ debe ser estimada a partir de las muestras obtenidas haciendo [12]:

$$\hat{\mathbf{R}}_{\mathbf{XX}} = \frac{1}{N} \mathbf{XX}^H \quad (3.7)$$

siendo N la cantidad de muestras tomadas en la ventana temporal de medición. En lo que resta del análisis se seguirá considerando la correlación teórica, debiendo el lector tomar las respectivas consideraciones.

Debido a que los vectores de \mathbf{A} definen la base del subespacio de señal, la matriz $\mathbf{A} \mathbf{R}_{\mathbf{SS}} \mathbf{A}^H$ de tamaño $M \times M$ tiene rango D , por lo que es de rango incompleto, a diferencia de $\mathbf{R}_{\mathbf{WW}}$ que es de rango completo. Si se considera que las señales que conforman la matriz \mathbf{S} no están correlacionadas entre sí, la matriz de autocorrelación de señal $\mathbf{R}_{\mathbf{SS}}$ es de la forma:

$$\mathbf{R}_{\mathbf{SS}} = \begin{bmatrix} |s_0|^2 & 0 & \cdots & 0 \\ 0 & |s_1|^2 & & \vdots \\ \vdots & \ddots & & \vdots \\ 0 & \cdots & \cdots & |s_{D-1}|^2 \end{bmatrix} \quad (3.8)$$

siendo cada elemento de la diagonal la potencia de cada una de las señales.

Si se aplica la descomposición en autovalores a la matriz $\mathbf{R}_{\mathbf{XX}}$, esta puede escribirse como [12]:

$$\mathbf{R}_{\mathbf{XX}} = \mathbf{E} \boldsymbol{\Lambda} \mathbf{E}^H, \quad (3.9)$$

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_0 & 0 & \cdots & 0 \\ 0 & \lambda_1 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \lambda_{M-1} \end{bmatrix}, \quad \mathbf{E} = [\bar{e}_0 \quad \bar{e}_1 \quad \cdots \quad \bar{e}_{M-1}]$$

donde $\lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_{M-1}$ son los autovalores de $\mathbf{R}_{\mathbf{XX}}$ en orden descendente y $\bar{e}_0, \bar{e}_1, \dots, \bar{e}_{M-1}$ sus correspondientes autovectores. Los D autovalores más grandes serán iguales a la suma de los autovalores de $\mathbf{A}\mathbf{R}_{\mathbf{SS}}\mathbf{A}^H$ más los autovalores de $\mathbf{R}_{\mathbf{WW}}$, los cuales son todos iguales. Los $M - D$ autovalores más chicos tendrán el mismo valor que los autovalores de $\mathbf{R}_{\mathbf{WW}}$, es decir [12]:

$$\begin{aligned} \lambda_m &= M \cdot |s_m|^2 + \sigma_w^2, & 0 \leq m \leq D-1 \\ \lambda_m &= \sigma_w^2, & D \leq m \leq M-1 \end{aligned} \quad (3.10)$$

Por ende, si separamos los autovalores y autovectores correspondientes a las señales y al ruido podemos realizar la siguiente descomposición:

$$\mathbf{R}_{\mathbf{XX}} = \mathbf{E}_{\mathbf{S}} \boldsymbol{\Lambda}_{\mathbf{S}} \mathbf{E}_{\mathbf{S}}^H + \sigma_w^2 \mathbf{E}_{\mathbf{W}} \mathbf{E}_{\mathbf{W}}^H \quad (3.11)$$

$$\boldsymbol{\Lambda}_{\mathbf{S}} = \begin{bmatrix} \lambda_0 & 0 & \cdots & 0 \\ 0 & \lambda_1 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \lambda_{D-1} \end{bmatrix}, \quad \mathbf{E}_{\mathbf{S}} = [\bar{e}_0 \quad \bar{e}_1 \quad \cdots \quad \bar{e}_{D-1}], \quad \mathbf{E}_{\mathbf{W}} = [\bar{e}_D \quad \bar{e}_{D+1} \quad \cdots \quad \bar{e}_{M-1}]$$

donde los vectores de $\mathbf{E}_{\mathbf{S}}$ conforman otra base del subespacio de señal \mathcal{S}_S y los vectores de $\mathbf{E}_{\mathbf{W}}$ conforman una base del *subespacio de ruido* \mathcal{S}_W .

Esto demuestra que a partir de una buena estimación de la matriz de covarianza de las muestras se pueden hallar las bases de los subespacios de señal y ruido, siempre y cuando la cantidad de elementos del arreglo de antenas M sea mayor a la cantidad de señales arribantes D , ya que **sino** el subespacio de las muestras sería de dimensión menor al subespacio de señal y no se podría realizar la separación con el subespacio de ruido. Para el caso práctico en el que la matriz de correlación de las muestras debe estimarse ocurrirá que los autovalores $\hat{\mathbf{R}}_{\mathbf{WW}}$ no serán iguales a σ_w^2 , y en los casos en los que la relación señal a ruido sea baja puede ser difícil diferenciar la frontera entre autovalores de ruido y autovalores de señal. Para encontrar esta frontera y así poder determinar la cantidad de señales arribantes para realizar la descomposición en subespacios de señal y ruido se puede recurrir a varios enfoques. Algunos de ellos se detallan en el Capítulo 5. Debido a que la matriz de autocorrelación muestral $\mathbf{R}_{\mathbf{XX}}$ tiene simetría Hermética, sus autovectores son ortogonales, lo cual implica que el subespacio de ruido y el subespacio de señal son ortogonales entre sí, característica de vital importancia para los algoritmos de estimación de DOA que se tratarán a continuación.

Descomposición en valores singulares (SVD)

Antes de comenzar el análisis de los algoritmos de estimación de arriba es conveniente mencionar otra manera de realizar la descomposición del subespacio muestral en subespacio de señal y ruido sin tener que calcular la descomposición en autovalores de la matriz de covarianza $\mathbf{R}_{\mathbf{XX}}$. La alternativa propuesta consiste en utilizar todo el conjunto de datos, es decir la matriz completa \mathbf{X} , y aplicarle la

descomposición en valores singulares (*SVD* por sus siglas en inglés), la cual es una generalización de la descomposición en autovalores para matrices que no son cuadradas. Esta técnica tiene la ventaja por sobre la anterior de que no eleva al cuadrado los elementos de \mathbf{X} , como sí ocurre cuando se calcula su matriz de covarianza, por ende es capaz de reducir errores de redondeo debidos a la representación utilizada al operar con matrices mal acondicionadas [10].

Si se tiene que la descomposición en valores singulares de \mathbf{X}/\sqrt{N} viene dada por $\mathbf{U}\Sigma\mathbf{V}^H$ se puede demostrar que esta descomposición genera el mismo subespacio que la descomposición por autovalores de la matriz de covarianza viendo que:

$$\frac{1}{N}\mathbf{XX}^H = \mathbf{U}\Sigma^2\mathbf{U}^H = \hat{\mathbf{R}}_{\mathbf{XX}} \quad (3.12)$$

dado que Σ es diagonal y real, y \mathbf{U} y \mathbf{V} son matrices unitarias. Por ende, los vectores singulares de la matriz \mathbf{U} son los autovectores de matriz de covarianza muestral $\hat{\mathbf{R}}_{\mathbf{XX}}$, luego generan el mismo espacio.

3.2. El algoritmo MUSIC

Cuando las muestras obtenidas por el arreglo de antenas están contaminadas con ruido, y para el caso práctico en el que se cuenta con una cantidad N de muestras finita, los subespacios obtenidos mediante el método descripto en la sección anterior son estimaciones del subespacio de ruido y señal, es decir, $\hat{\mathcal{S}}_S$ y $\hat{\mathcal{S}}_W$. Al no ser los subespacios teóricos, $\hat{\mathcal{S}}_S$ tiene componentes del subespacio de ruido y viceversa, por ende $\hat{\mathcal{S}}_S$ es ortogonal a $\hat{\mathcal{S}}_S$. Esta cualidad de $\hat{\mathcal{S}}_S$ provoca que el método de encontrar las intersecciones entre \mathcal{S}_S y $\text{los elementos de } \mathfrak{A}$ no pueda aplicarse, debido a que $\hat{\mathcal{S}}_S \cap \mathfrak{A} = \bar{0}$, y de la misma manera no existen elementos de \mathfrak{A} que sean ortogonales a $\hat{\mathcal{S}}_W$ [10]. Sin embargo, aquellos elementos de \mathfrak{A} que se encuentren más cercanos a $\hat{\mathcal{S}}_S$ pueden ser considerados como los elementos que generan a \mathcal{S}_S y, por ende, los vectores que indican las direcciones de arribo de las D señales arribantes. La cuestión ahora es definir una noción de cercanía. En [11], Ralph O. Schmidt define una posible medida de distancia de un vector \bar{v} al subespacio de señal estimado $\hat{\mathcal{S}}_S$ como:

$$d^2 := \bar{v}^H \hat{\mathbf{E}}_{\mathbf{W}} \hat{\mathbf{E}}_{\mathbf{W}}^H \bar{v} \quad (3.13)$$

A partir de esta ecuación se pueden encontrar los elementos de \mathfrak{A} más cercanos al subespacio $\hat{\mathcal{S}}_S$ mediante una búsqueda en todos los elementos de \mathfrak{A} . Una forma gráfica de verlo es definiendo el “pseudoespectro”¹ $\mathbf{P}_{\mathbf{MU}}$ como [11]:

$$\mathbf{P}_{\mathbf{MU}}(\theta, \varphi) = \frac{1}{d^2} = \frac{1}{\bar{a}(\theta, \varphi)^H \hat{\mathbf{E}}_{\mathbf{W}} \hat{\mathbf{E}}_{\mathbf{W}}^H \bar{a}(\theta, \varphi)} \quad (3.14)$$

siendo $\bar{a}(\theta, \varphi)$ cualquier elemento de \mathfrak{A} . Como se dijo, esta distancia será mínima cuando el vector medido sea uno de los elementos de \mathfrak{A} que pertenecen a \mathcal{S}_S , por ende si se grafica este pseudoespectro en todo el dominio de direcciones de arribo, es decir, en el dominio bidimensional que contiene a todos los posibles ángulos de elevación y azimut, este pseudoespectro mostrará picos en los puntos donde $\bar{a}(\theta, \varphi)$ se acerca más al subespacio $\hat{\mathcal{S}}_S$. El algoritmo *Multiple Signal Classification* propuesto por Schmidt consiste en realizar una estimación de \mathcal{S}_S y \mathcal{S}_W para luego conformar el pseudoespectro $\mathbf{P}_{\mathbf{MU}}$ y finalmente encontrar los D picos máximos, los cuales indicarán las correspondientes direcciones de arribo de las D señales arribantes. Como se mostró, este algoritmo no requiere de ninguna disposición particular de los elementos del arreglo, lo cual permite ser utilizado en cualquier caso en el que la

¹El término “pseudoespectro” es utilizado debido a que la magnitud representada por la Ecuación 3.14 no expresa información sobre la potencia de una señal, sino que alude a una similitud entre su forma gráfica y los espectros de potencia.

forma del arreglo sea conocida. Debido a que este algoritmo requiere de una búsqueda en todo el dominio de DOA, la complejidad computacional puede ser muy alta según la cantidad de dimensiones que tenga la dirección de arriba y la resolución que se desee alcanzar en la detección de los picos. Algunos resultados con respecto a este asunto se muestran en la Sección 3.4.

3.2.1. Algoritmo

A continuación se detallan los pasos para implementar el algoritmo MUSIC:

1. Formar la matriz \mathbf{X} definida en la Ecuación 3.4 utilizando varias muestras temporales y armar $\hat{\mathbf{R}}_{\mathbf{XX}}$ como se muestra en la Ecuación 3.7 o descomponer la matriz \mathbf{X} usando SVD.
2. Hallar las matrices \mathbf{E} y Λ de la descomposición en autovalores de $\hat{\mathbf{R}}_{\mathbf{XX}}$ que se muestra en la Ecuación 3.9.
3. Estimar el número \hat{D} de señales recibidas. Las técnicas para lograr esto se mencionan en el Capítulo 5.
4. Armar la matriz \mathbf{E}_W como se muestra en la Ecuación 3.11.
5. Evaluar $\mathbf{P}_{MU}(\theta, \varphi)$ en función de θ y φ utilizando la Ecuación 3.14.
6. Elegir los \hat{D} picos máximos de $\mathbf{P}_{MU}(\theta, \varphi)$ y extraer sus coordenadas (θ, φ) correspondientes a las direcciones de arriba de las \hat{D} señales recibidas.

3.3. El algoritmo ESPRIT

En 1989 Richard Roy y Thomas Kailath se alejan del enfoque de cercanía al subespacio de señal planteado por Schmidt y desarrollan un nuevo algoritmo de estimación de parámetros de señales llamado *Estimation of Signal Parameters via Rotational Invariance Techniques (ESPRIT)*, el cual explota las invariancias en la geometría del arreglo de antenas.

Supongamos que tenemos un arreglo con una geometría arbitraria pero que está compuesta de elementos agrupados de a pares separados de manera traslacional por un vector $\bar{\Delta}$ como se muestra en la Figura 3.2 [10]. Los elementos apareados tienen el mismo patrón de radiación, pero entre pares no se requiere que sean iguales.

Bajo esta geometría, se puede considerar que el arreglo está compuesto por dos subarreglos, identificados en la figura por los elementos azules y los elementos rojos, los cuales conforman los subarreglos Z_x y Z_y , respectivamente. Estos subarreglos son idénticos entre sí, con la única diferencia de que uno está trasladado con respecto al otro por el vector $\bar{\Delta}$. Las señales recibidas por cada elemento del par i -ésimo pueden ser expresadas como:

$$\begin{aligned} x_i(t) &= \sum_{d=0}^{D-1} a_i(\theta_d, \varphi_d) \cdot s_d(t) + w_{x,i}(t) \\ y_i(t) &= \sum_{d=0}^{D-1} a_i(\theta_d, \varphi_d) \cdot e^{j\bar{k} \cdot \bar{\Delta}} \cdot s_d(t) + w_{y,i}(t), \end{aligned} \tag{3.15}$$

siendo $x_i(t)$ la señal recibida por el elemento perteneciente al subarreglo Z_x y dentro del i -ésimo par, $y_i(t)$ la señal recibida por el elemento perteneciente al subarreglo Z_y dentro del i -ésimo par y $a_i(\theta_d, \varphi_d)$ el elemento del vector de apuntamiento que corresponde al elemento de referencia dentro del par correspondiente (en este caso el perteneciente a Z_x) para una dirección de arriba (θ_d, φ_d) .

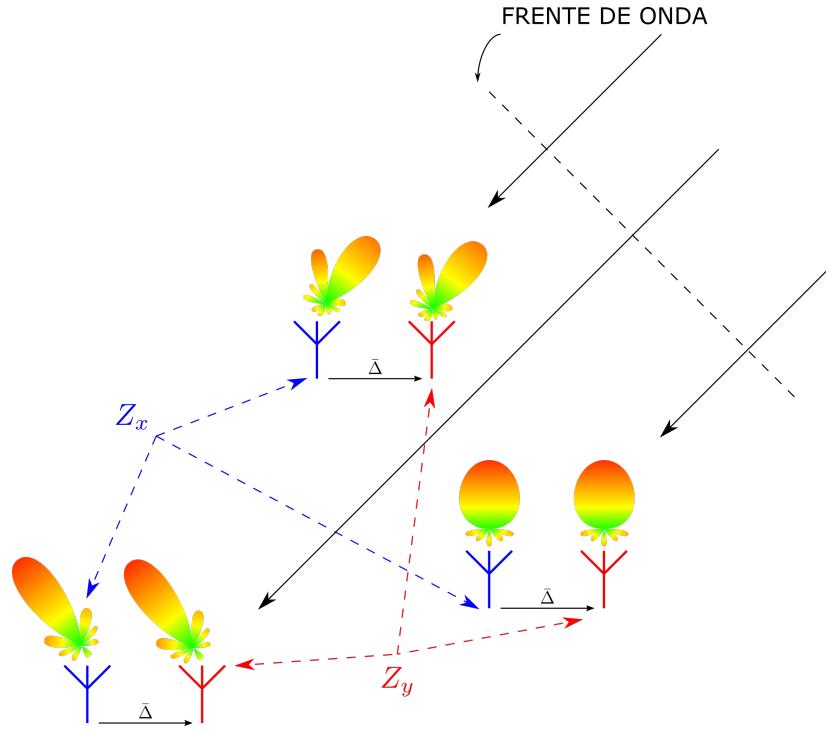


Figura 3.2: Geometría del arreglo de antenas para el análisis del algoritmo ESPRIT. En azul y en rojo se separan los elementos pertenecientes a los subarreglos Z_x y Z_y , respectivamente.

La dirección del vector de translación $\bar{\Delta}$ indica la dirección con respecto a la cual se podrá estimar el ángulo de arribo. Por ejemplo, si la translación se realiza sobre el eje x , solo se podrán estimar ángulos de arribo con respecto a este eje. Como consecuencia de esto, en el caso de la estimación bidimensional con ángulos de elevación y azimut se deberá contar con dos vectores de translación, uno en cada dimensión de interés.

Combinando las salidas de todos los elementos de cada subarreglo se pueden expresar a las señales recibidas como:

$$\begin{aligned}\bar{x}(t) &= \mathbf{A}s(t) + \bar{w}_x(t), \\ \bar{y}(t) &= \mathbf{A}\Phi s(t) + \bar{w}_y(t),\end{aligned}\tag{3.16}$$

donde \mathbf{A} es la matriz de apuntamiento correspondiente al subarreglo Z_x y Φ una matriz diagonal de tamaño $D \times D$ cuyos elementos están conformados por los desfasajes producidos entre pares de elementos para los D frentes de ondas recibidos, es decir:

$$\Phi = \begin{bmatrix} e^{j\bar{k}_0 \cdot \bar{\Delta}} & 0 & \dots & 0 \\ 0 & e^{j\bar{k}_1 \cdot \bar{\Delta}} & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \dots & \dots & e^{j\bar{k}_{D-1} \cdot \bar{\Delta}} \end{bmatrix} = \text{diag}\{\phi_0, \phi_1, \dots, \phi_{D-1}\}\tag{3.17}$$

siendo $\phi_d = e^{j\bar{k}_d \cdot \bar{\Delta}}$ con $d = 0, 1, \dots, D - 1$. La matriz Φ recibe el nombre de *operador de rotación de subespacio* [10]. Si se define al número de pares de elementos como M , de manera tal que el número

de elementos sea $2M$, y al vector de salida del arreglo entero como $\bar{z}(t)$ podemos escribirlo como:

$$\begin{aligned}\bar{z}(t) &= \begin{bmatrix} \bar{x}(t) \\ \bar{y}(t) \end{bmatrix} = \bar{\mathbf{A}}s(t) + \bar{w}(t), \\ \bar{\mathbf{A}} &= \begin{bmatrix} \mathbf{A} \\ \mathbf{A}\Phi \end{bmatrix}_{(2M \times D)}, \quad \bar{w}(t) = \begin{bmatrix} \bar{w}_x \\ \bar{w}_y \end{bmatrix}_{(2M \times 1)}\end{aligned}\quad (3.18)$$

Una consideración a tener en cuenta para el resto del análisis es que D tiene que ser menor a M , aunque posteriormente se mostrará que en realidad D puede ser como máximo menor a la cantidad total de elementos en el arreglo.

Los vectores de $\bar{\mathbf{A}}$ generan el subespacio de señal. Si se define la matriz \mathbf{Z} como la matriz conformada por los vectores obtenidos luego de muestrear N veces a todos los elementos del arreglo de antenas podemos expresarla de la siguiente manera:

$$\mathbf{Z} = \begin{bmatrix} \bar{z}_0 & \bar{z}_1 & \cdots & \bar{z}_{N-1} \end{bmatrix}_{(2M \times N)}, \quad (3.19)$$

donde \bar{z}_n con $n = 0, 1, \dots, N - 1$ es el vector de muestras tomadas en el instante n . Si se aplica la descomposición en subespacio de ruido y señal de las muestras contenidas en la matriz \mathbf{Z} , como se mostró en la Sección 3.1.1, puede verse que la matriz \mathbf{E}_S genera el mismo espacio $2M$ -dimensional que los vectores de $\bar{\mathbf{A}}$, ya que ambos generan el subespacio de señal. En consecuencia, debe existir una matriz de transformación invertible \mathbf{T} que mapee elementos de \mathbf{E}_S a $\bar{\mathbf{A}}$, es decir, existe \mathbf{T} tal que:

$$\mathbf{E}_S = \bar{\mathbf{A}}\mathbf{T} \quad (3.20)$$

Debido a esto la matriz \mathbf{E}_S puede descomponerse en dos submatrices \mathbf{E}_X y \mathbf{E}_Y de tamaño $M \times D$ de manera tal que:

$$\mathbf{E}_S = \begin{bmatrix} \mathbf{E}_X \\ \mathbf{E}_Y \end{bmatrix} = \begin{bmatrix} \mathbf{A}\mathbf{T} \\ \mathbf{A}\Phi\mathbf{T} \end{bmatrix} \quad (3.21)$$

de donde puede verse que

$$\text{Span}\{\mathbf{E}_X\} = \text{Span}\{\mathbf{E}_Y\} = \text{Span}\{\mathbf{A}\} \quad (3.22)$$

(donde $\text{Span}\{\mathbf{V}\}$ es el subespacio generado por \mathbf{V}), debido a que la rotación aplicada por Φ no modifica el subespacio generado [10]. Ahora se define la matriz \mathbf{E}_{XY} como:

$$\mathbf{E}_{XY} := [\mathbf{E}_X | \mathbf{E}_Y], \quad (3.23)$$

la cual tiene rango D debido a que \mathbf{E}_X y \mathbf{E}_Y tienen el mismo espacio de columnas. Debido a esto, debe existir una matriz $\mathbf{K} \in \mathbb{C}^{(2D \times D)}$ tal que:

$$\mathbf{0} = \mathbf{E}_{XY}\mathbf{K} = \mathbf{E}_X\mathbf{K}_X + \mathbf{E}_Y\mathbf{K}_Y \quad (3.24)$$

$$= \mathbf{AT}\mathbf{K}_X + \mathbf{A}\Phi\mathbf{T}\mathbf{K}_Y, \quad (3.25)$$

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_X \\ \mathbf{K}_Y \end{bmatrix},$$

es decir que \mathbf{K} genera el núcleo de $\mathbf{E}_{\mathbf{X}\mathbf{Y}}$. A partir de la Ecuación 3.24 puede escribirse:

$$\begin{aligned} -\mathbf{E}_{\mathbf{X}}\mathbf{K}_{\mathbf{X}} &= \mathbf{E}_{\mathbf{Y}}\mathbf{K}_{\mathbf{Y}} \\ \mathbf{E}_{\mathbf{X}}(-\mathbf{K}_{\mathbf{X}}\mathbf{K}_{\mathbf{Y}}^{-1}) &= \mathbf{E}_{\mathbf{Y}} \\ \mathbf{E}_{\mathbf{X}}\Psi &= \mathbf{E}_{\mathbf{Y}} \end{aligned} \quad (3.26)$$

siendo $\Psi := -\mathbf{K}_{\mathbf{X}}\mathbf{K}_{\mathbf{Y}}^{-1}$ la matriz de transformación de $\mathbf{E}_{\mathbf{X}}$ a $\mathbf{E}_{\mathbf{Y}}$. A partir de esta definición se puede reescribir la Ecuación 3.25 de la siguiente manera:

$$\mathbf{A}\mathbf{T}\Psi = \mathbf{A}\Phi\mathbf{T} \rightarrow \mathbf{A}\mathbf{T}\Psi\mathbf{T}^{-1} = \mathbf{A}\Phi \quad (3.27)$$

Suponiendo que \mathbf{A} es de rango completo se puede escribir:

$$\mathbf{T}\Psi\mathbf{T}^{-1} = \Phi, \quad (3.28)$$

por ende, los autovalores de Ψ son los elementos de la diagonal de Φ , y los vectores de \mathbf{T} son los autovectores de Φ . Debido a esto, obteniendo una estimación de Ψ se puede obtener una estimación de la matriz Φ y a partir de ahí los ángulos de arriba.

Los subarreglos no tienen por qué ser conjuntos disjuntos, la posibilidad de armar subarreglos con elementos en común está permitida en el algoritmo ESPRIT y es una técnica que permite reducir la cantidad de elementos necesaria para poder implementarlo. Si contamos con un arreglo de antenas en fase de geometría ARU se deberá aplicar el algoritmo ESPRIT en dos dimensiones distintas para obtener los ángulos de elevación y azimut de la dirección de arriba. La opción lógica es elegir subarreglos separados en el eje x y en el eje y , como se muestra en la Figura 3.3.

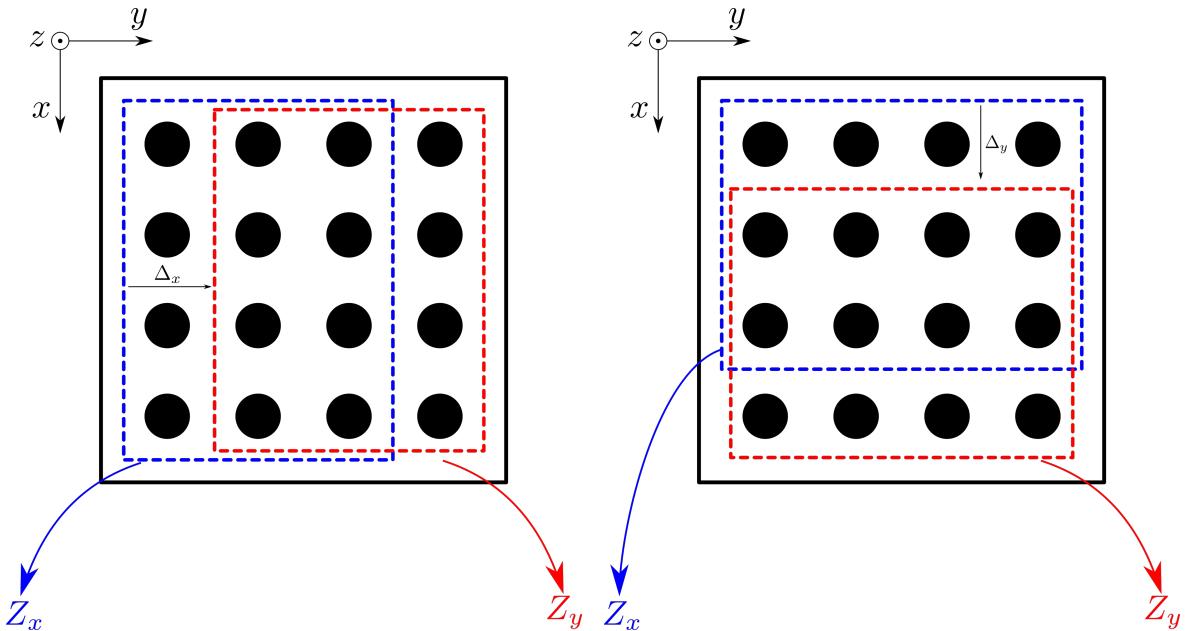


Figura 3.3: Una posible elección de subarreglos para la aplicación del algoritmo ESPRIT en dos dimensiones utilizando un ARU.

Teniendo una geometría como la mostrada, luego de utilizar ESPRIT en las dos dimensiones indicadas se obtendrán dos matrices Φ_x y Φ_y de tamaño $D \times 1$, los cuales permiten obtener los

ángulos de arribo haciendo:

$$\theta_d = \arccos \left(\sqrt{\frac{(\angle \phi_{x,d})^2 + (\angle \phi_{y,d})^2}{(k \cdot \delta)^2}}, \right) \quad (3.29)$$

$$\varphi_d = \arctan \left(\frac{\angle \phi_{y,d}}{\angle \phi_{x,d}} \right) \quad (3.30)$$

con $d = 0, 1, \dots, D - 1$ y siendo δ la separación entre elementos. La deducción de estas expresiones se muestra en el Apéndice A.

Estimación del operador de rotación de subespacio

Al igual que como ocurre en MUSIC, al no poder operar en la práctica con infinitas muestras, la matriz \mathbf{E}_S obtenida mediante SVD aplicada a la matriz \mathbf{Z} o por descomposición en autovalores de la estimación de la matriz de covarianza $\hat{\mathbf{R}}_{ZZ}$ no es la teórica, sino una estimación de ella, la cual se denota como $\hat{\mathbf{E}}_S$. Por ende, $\hat{\mathbf{E}}_S$ no genera el subespacio de señal y $\text{Span}\{\hat{\mathbf{E}}_S\} \neq \text{Span}\{\bar{\mathbf{A}}\}$. Además $\text{Span}\{\hat{\mathbf{E}}_X\} \neq \text{Span}\{\hat{\mathbf{E}}_Y\}$, y debido a esto la matriz Ψ no puede encontrarse de la manera que se mostró en la Ecuación 3.26. Es por esto que debe elegirse un criterio para obtener una correcta estimación de $\hat{\Psi}$ para luego estimar $\hat{\Phi}$, y ese criterio es el de *mínimos cuadrados totales* (*TLS* por sus siglas en inglés). Este criterio consiste en reemplazar la matriz nula en la Ecuación 3.24 por una matriz de errores cuya norma de Frobenius debe minimizarse [10]. La solución a este problema primero consiste en calcular la SVD de la matriz $\hat{\mathbf{E}}_{XY}$ definida en la Ecuación 3.23:

$$\hat{\mathbf{E}}_{XY} = \mathbf{U} \Sigma \mathbf{V} \quad (3.31)$$

A partir de esto se trabaja con la matriz $\mathbf{V} \in \mathbb{C}^{2D \times 2D}$, la cual se partitiona en 4 matrices de tamaño $D \times D$ de la siguiente manera:

$$\mathbf{V} = \begin{bmatrix} \mathbf{V}_{00} & \mathbf{V}_{01} \\ \mathbf{V}_{10} & \mathbf{V}_{11} \end{bmatrix} \quad (3.32)$$

Luego la solución TLS viene dada por:

$$\hat{\Psi}_{TLS} = -\mathbf{V}_{01} \mathbf{V}_{11}^{-1} \quad (3.33)$$

cuyos autovalores representan a los elementos $\hat{\phi}_d$ con $d = 0, 1, \dots, D - 1$ de la matriz $\hat{\Phi}$.

Finalmente, la dirección de arribo estimada $(\hat{\theta}_d, \hat{\phi}_d)$ se obtiene mediante las Ecuaciones 3.29 y 3.30.

3.3.1. Algoritmo

A continuación se detallan los pasos para implementar el algoritmo ESPRIT:

1. Separar el arreglo de antenas en dos subarreglos iguales pero trasladados por un vector $\bar{\Delta}$ uno del otro.
2. Formar la matriz \mathbf{Z} definida en la Ecuación 3.19 utilizando varias muestras temporales.
3. Realizar la descomposición en autovalores de la matriz $\hat{\mathbf{R}}_{ZZ}$ obtenida con la Ecuación 3.7 o realizar la SVD sobre la matriz \mathbf{Z} quedándose únicamente con los valores singulares y los vectores singulares izquierdos \mathbf{U} .

4. Estimar el número \hat{D} de señales recibidas. Las técnicas para lograr esto se mencionan en el Capítulo 5.
5. Elegir los \hat{D} autovectores de la descomposición de $\hat{\mathbf{R}}_{\mathbf{ZZ}}$ correspondientes a los autovalores más grandes, o los vectores singulares de \mathbf{U} correspondientes a los valores singulares más grandes de la SVD de \mathbf{Z} para formar la matriz $\hat{\mathbf{E}}_{\mathbf{S}}$ como se indica en la Ecuación 3.11.
6. Separar la matriz $\hat{\mathbf{E}}_{\mathbf{S}}$ en $\hat{\mathbf{E}}_{\mathbf{X}}$ y $\hat{\mathbf{E}}_{\mathbf{Y}}$ como se muestra en la Ecuación 3.21.
7. Armar la matriz $\hat{\mathbf{E}}_{\mathbf{XY}}$ como se muestra en la Ecuación 3.23 y obtener su SVD quedándose únicamente con los vectores singulares derechos \mathbf{V} .
8. Partitionar la matriz \mathbf{V} en 4 submatrices de tamaño $D \times D$ como se muestra en la Ecuación 3.32.
9. Obtener $\hat{\Psi}_{\text{TLS}}$ como se indica en la Ecuación 3.33.
10. Obtener los elementos de la matriz $\hat{\Phi}$ obteniendo los autovalores de $\hat{\Psi}_{\text{TLS}}$.
11. Obtener los ángulos de arribo $(\hat{\theta}_d, \hat{\phi}_d)$ a partir de las Ecuaciones 3.29 y 3.30.

3.4. Simulaciones

Utilizando el modelo de muestras que se detalló en la Sección 3.1.1 y los algoritmos descriptos en las Secciones 3.2.1 y 3.3.1, se implementaron ambas técnicas de estimación de dirección de arribo, comparándolas con respecto al error obtenido en la estimación y el tiempo de ejecución de cada una. En todas las simulaciones realizadas a continuación se utilizó el método de descomposición en valores singulares para la obtención de los estimadores de subespacios de señal y ruido, y no se realizó la estimación de cantidad de señales arribantes, fijando este número en 1. Además, en ambos algoritmos se utilizó una técnica de muestreo aleatorio para mejorar la eficiencia en las estimaciones, la cual se detalla en el Capítulo 4. Finalmente, se consideró que el arreglo de antenas es de tipo ARU de tamaño $M_x \times M_y$ y que la separación entre elementos es de media longitud de onda de la portadora. Para generar las muestras simuladas se desarrolló un algoritmo que a partir de una muestra de una señal generaba el vector de muestras correspondientes a cada elemento del arreglo, simulando el comportamiento de la señal llegando al mismo con una determinada DOA según el análisis descripto en la Sección 2.3.1 para el caso de un arreglo rectangular uniforme. Como señal de prueba se utilizó la captura de un beacon del satélite GOMX-1 [13].

En esta sección se utilizarán los siguientes símbolos para identificar a los parámetros de las simulaciones realizadas:

- M_x : número de elementos del ARU en la dirección x .
- M_y : número de elementos del ARU en la dirección y .
- M : número total de elementos en el ARU.
- $\mathfrak{R}_{\text{MUSIC}}$: resolución del dominio de búsqueda del algoritmo MUSIC tanto en azimut como en elevación.
- N : número de vectores de muestras con los que se alimentó a cada algoritmo.
- $\frac{\sigma_d}{d}$: desvío estándar del error en la separación de elementos con respecto a la distancia nominal d .

3.4.1. RMSE vs. SNR

Para realizar esta simulación se promediaron 10 realizaciones por cada valor de SNR. En cada realización se alteraba aleatoriamente la DOA y las muestras elegidas por el algoritmo de muestreo aleatorio. Para el caso del algoritmo MUSIC, el dominio de búsqueda se generó de manera tal de poder contar con una resolución en la estimación de $\Re_{\text{MUSIC}} = 0,5^\circ$ tanto en elevación como en azimut. La métrica de error utilizada en este análisis es la raíz cuadrada del error cuadrático medio (RMSE), la cual se obtiene de la siguiente manera:

$$\text{RMSE}(\hat{\theta}, \hat{\phi}) = \sqrt{\frac{1}{2N} \sum_{i=0}^{N-1} \left((\hat{\theta}_i - \theta_i)^2 + (\hat{\phi}_i - \varphi_i)^2 \right)} \quad (3.34)$$

siendo N el número de realizaciones. Los parámetros utilizados en esta simulación son:

- $M_x = 4$
- $M_y = 4$
- $N = 1500$ vectores de muestras
- $\frac{\sigma_d}{d} = 0$
- $\Re_{\text{MUSIC}} = 0,5^\circ$

Los resultados obtenidos se muestran en la Figura 3.4.

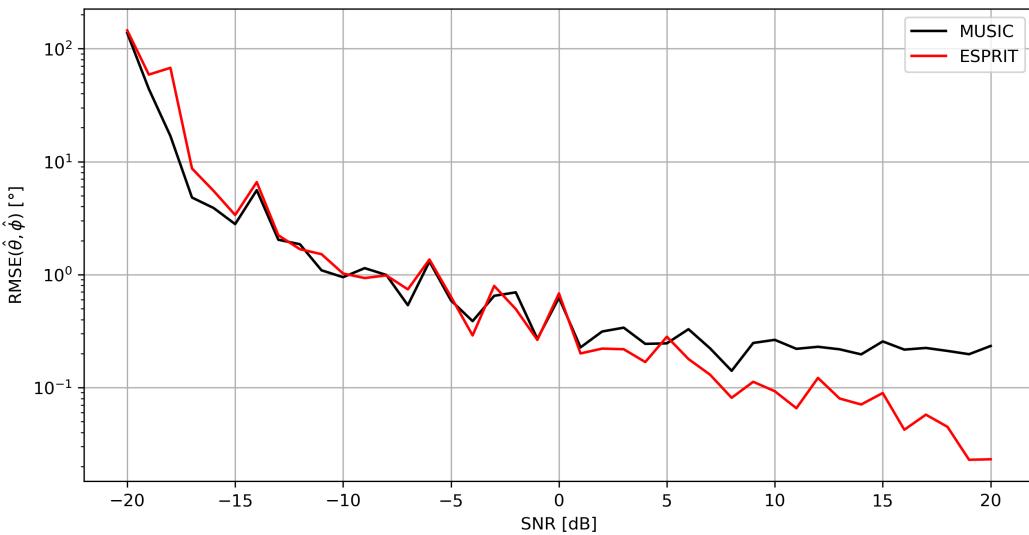


Figura 3.4: Gráfico de comparación del RMSE en función de la SNR para los algoritmos MUSIC y ESPRIT.

Como puede apreciarse, ambos algoritmos se desempeñan de manera similar en el rango de SNR evaluado, alcanzando errores menores al grado para valores de SNR a partir de -5 dB. También se puede ver que para valores de SNR mayores a 0 dB MUSIC comienza a verse limitado por su resolución (la cual puede mejorarse a costa de un mayor requerimiento de procesamiento), y ESPRIT continúa mejorando la estimación alcanzando errores menores a $0,1^\circ$.

Como dato adicional en la Figura 3.5 se muestra cómo varía el pseudoespectro de MUSIC para el caso de una señal arribando con una DOA ($\theta = 45^\circ, \varphi = 30$) para distintos valores de SNR.

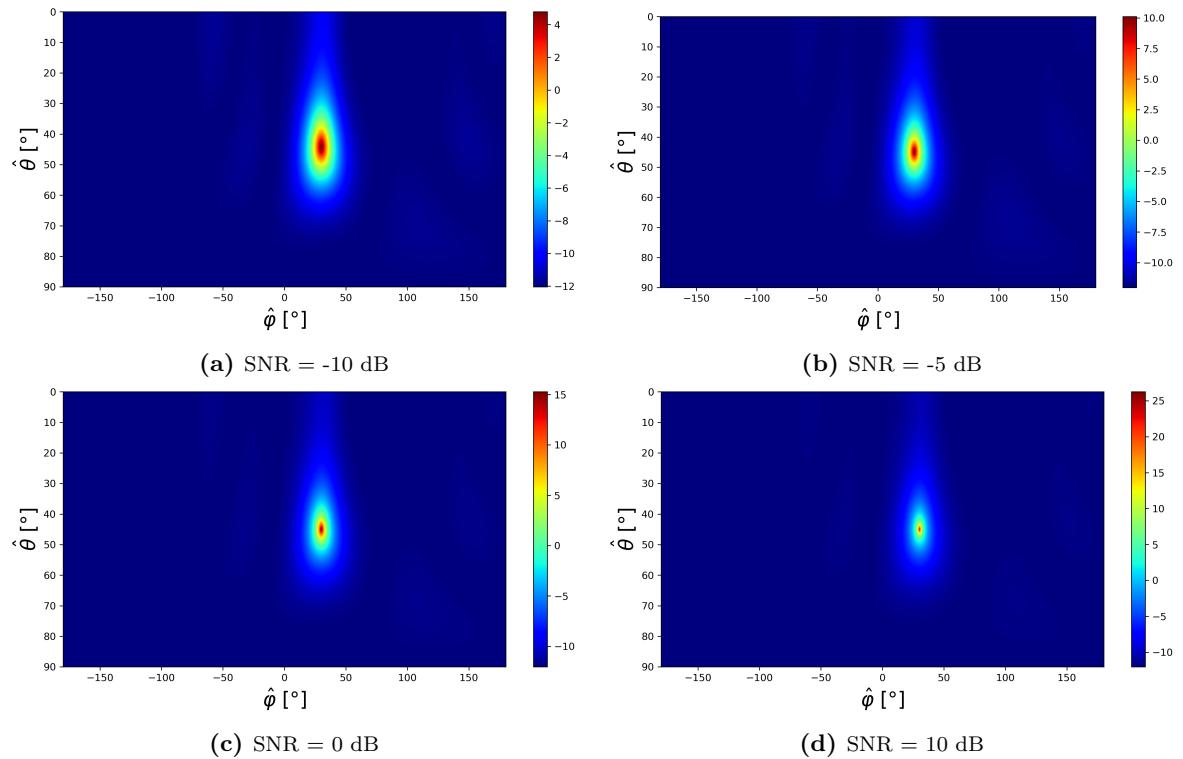


Figura 3.5: Gráfica de \mathbf{P}_{MU} en el caso de una señal llegando al arreglo con dirección $(\theta = 45^\circ, \varphi = 30)$ para distintos valores de SNR. Puede observarse cómo a medida que aumenta la SNR la “energía” del pseudoespectro se concentra cada vez más en la dirección de arriba real.

3.4.2. RMSE vs. $\frac{\sigma_d}{d}$

En este apartado se evalúan ambos algoritmos en función de errores aleatorios en la separación de elementos. Para hacer esto se suman en cada elemento fases aleatorias en ambas dimensiones del arreglo, de manera tal que se simulen errores gaussianos en la ubicación de elementos, ubicando el centro de la gaussiana en el centro de cada uno y tomando como desvió estndar σ_d de la distribución distintas fracciones de la distancia entre elementos d . Los parámetros utilizados en esta simulación son:

- SNR = 10 dB
- $M_x = 4$
- $M_y = 4$
- $N = 1500$ vectores de muestras
- $\Re_{\text{MUSIC}} = 0,5^\circ$

Los resultados que se muestran en la Figura 3.6 se obtuvieron promediando 10 realizaciones por cada valor de $\frac{\sigma_d}{d}$. Nuevamente, ambos algoritmos evolucionan de manera similar, manteniendo errores en el orden del grado para desviaciones del orden de entre 5 % y 10 % en la separación entre elementos.

3.4.3. RMSE vs. N° de muestras

En esta comparación se analiza la variación del RMSE en función de las muestras utilizadas para realizar la estimación con ambos algoritmos. Para esto se repitieron 10 realizaciones por cada valor de N , promediándolas. Los parámetros de esta simulación son:

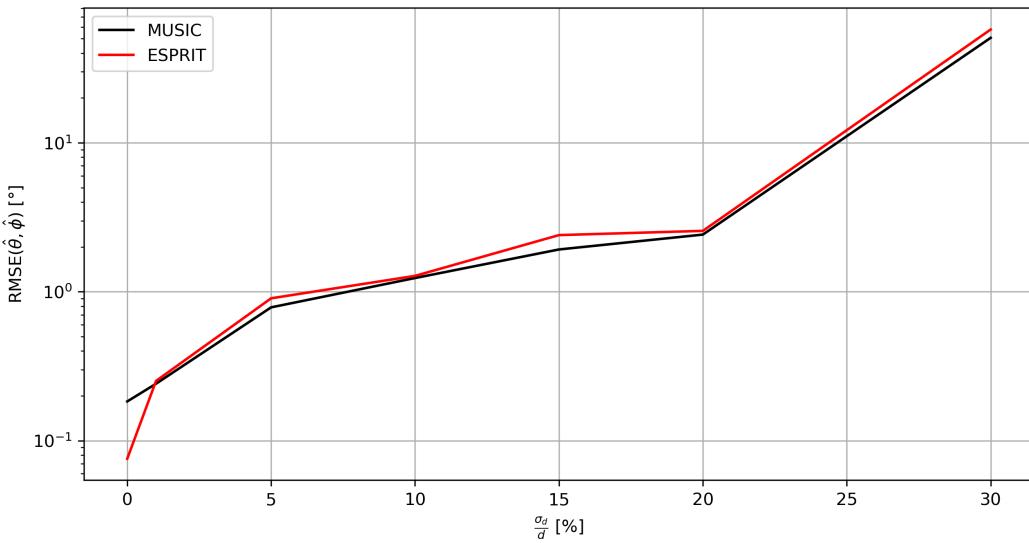


Figura 3.6: Gráfico de comparación del RMSE en función de los errores en la separación de elementos para los algoritmos MUSIC y ESPRIT.

- SNR = 10 dB
- $M_x = 4$
- $M_y = 4$
- $\frac{\sigma_d}{d} = 0$
- $\Re_{\text{MUSIC}} = 0,5^\circ$

Los resultados obtenidos se muestran en la Figura 3.7. Como puede verse, ambos algoritmos mejoran su desempeño cuantas más muestras reciben debido a la mejor estimación que se logra de los subespacios de señal y de ruido, siguiendo una curva similar en ambos casos. Para el caso de gran cantidad de muestras, MUSIC muestra una asíntota horizontal, producto de su limitada resolución. Sin embargo, para esta SNR elegida, se alcanzan errores menores a la décima del grado utilizando una cantidad de muestras del orden de 10^3 , lo cual computacionalmente no conlleva grandes costos como se mostrará posteriormente.

3.4.4. Tiempo vs. M

En esta sección se comparan los tiempos de ejecución de ambos algoritmos en función de la cantidad de elementos del arreglo. Además, en el caso de MUSIC también se analiza cuánto varían las curvas de tiempo para distintos valores de resoluciones alcanzadas. Para este análisis no se considera el tiempo que se tarda en realizar la SVD debido a que es una tarea común a ambos algoritmos, en cambio este análisis se realiza en la siguiente sección. Por ende, en el caso de MUSIC sólo es evaluado el tiempo que lleva realizar los pasos 5 y 6 del algoritmo desarrollado en la Sección 3.2.1 y en el caso de ESPRIT solo se evalúan los pasos 6 a 11 descriptos en la Sección 3.3.1. Los resultados se muestran en la Figura 3.8. Los parámetros definidos para esta simulación son:

- SNR = 10 dB
- $\frac{\sigma_d}{d} = 0$
- $N = 1500$ vectores de muestras

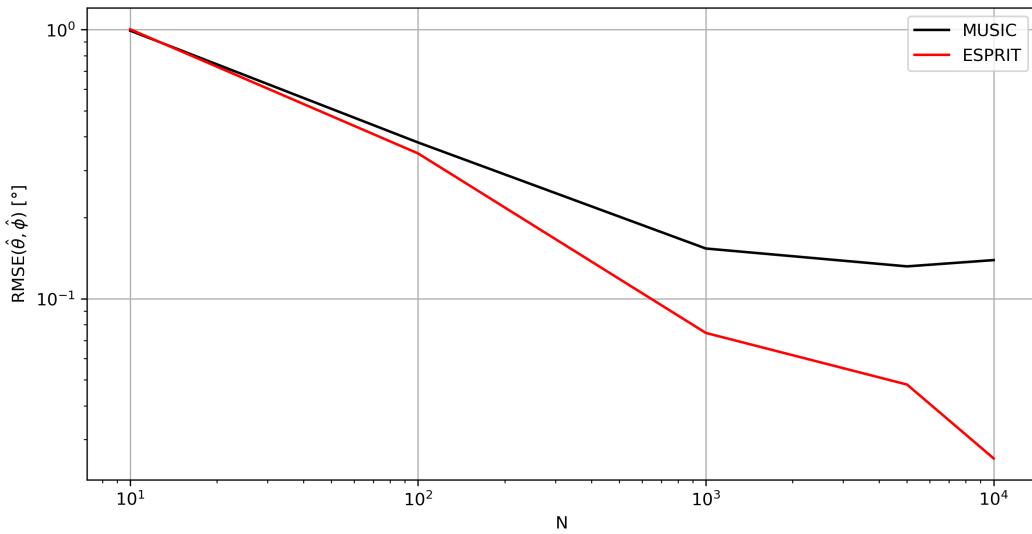


Figura 3.7: Gráfica de comparación del RMSE en función de la cantidad de vectores de muestras para los algoritmos MUSIC y ESPRIT.

- $\Re_{\text{MUSIC}} = 0,5^\circ$

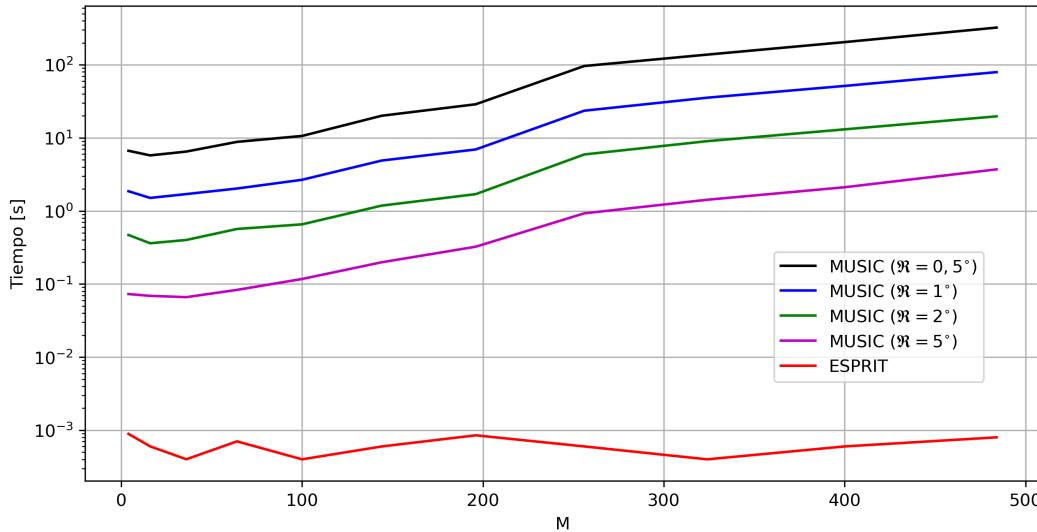


Figura 3.8: Gráfica de comparación del tiempo de ejecución de los algoritmos MUSIC y ESPRIT en función de la cantidad de elementos del ARU.

Como puede verse, para todos los valores de cantidad de elementos analizados, el algoritmo MUSIC resultó ser mucho menos eficiente que ESPRIT, obteniendo tiempos de al menos dos órdenes de magnitud mayor para $M < 100$ y tres órdenes mayor para $M > 300$ para una resolución en la estimación de $\Re_{\text{MUSIC}} = 5^\circ$. Además, si se analiza la curva de tiempos de MUSIC para una resolución de $\Re_{\text{MUSIC}} = 0,5^\circ$, la cual brinda un nivel de error semejante a ESPRIT como se mostró en la Sección 3.4.1, el tiempo de ejecución supera al ESPRIT por más de 5 órdenes de magnitud. Esto provoca que esta implementación del MUSIC sea ineficaz para aplicaciones de tiempo real como la que se desea desarrollar en este proyecto. El motivo del incremento del costo computacional del MUSIC con respecto a M se debe al cálculo de la Ecuación 3.14, ya que todas las matrices que entran en la multiplicación tienen M filas, y la matriz $\hat{\mathbf{E}}\mathbf{W}$ tiene mayor cantidad de columnas cuanto mayor sea M , lo que provoca que la cantidad de operaciones a realizar crezca rápidamente a medida que aumenta

M . En cambio el algoritmo ESPRIT reduce la dimensión del problema separando $\hat{\mathbf{E}}_S$ en $\hat{\mathbf{E}}_X$ y $\hat{\mathbf{E}}_Y$ para armar $\hat{\mathbf{E}}_{XY}$ la cual como máximo puede tener una cantidad de filas de $M - 1$. Esta matriz solo es utilizada cuando se le aplica la descomposición en valores singulares, momento en el cual se comienza a operar con matrices de tamaño $D \times D$, siendo $D < M$. La complejidad de ambos algoritmos aumenta con el aumento en la cantidad de señales a estimar.

Si se analizan las variaciones de tiempo entre distintas resoluciones de la solución brindada por MUSIC el motivo del crecimiento del costo a medida que se aumenta la resolución radica en el dominio de búsqueda en el que se debe calcular la Ecuación 3.14, ya que una mayor resolución implica mayores puntos del dominio donde se desea saber el valor de $\mathbf{P}_{MU}(\theta, \varphi)$. El dominio de búsqueda puede reducirse a partir de la utilización de información previa sobre las direcciones de arriba estimadas, sin embargo la gran eficiencia de ESPRIT no motivó la realización de ese estudio para este trabajo.

3.4.5. Tiempo de ejecución de la SVD

En esta sección se analiza el costo computacional del algoritmo SVD implementado dentro de la librería NumPy [14] en función de la cantidad de elementos del arreglo M y la cantidad de vectores de muestras con el que se alimenta a algoritmo N . Debido a que en la simulación en función de M vamos a estar variando la cantidad de filas de la matriz \mathbf{X} definida en la Ecuación 3.4 y en la simulación en función de N vamos a estar variando las columnas se decidió que la variable que quede fija en cada una tenga el mismo valor, lo cual se logró fijando la cantidad de elementos del arreglo en 100 para la simulación en función de N y fijando la cantidad de vectores de muestras en 100 para la simulación en función de M . Por ende, los parámetros utilizados en estas simulaciones son los siguientes:

- SNR = 10 dB
- $M_x = 10$ (solo para el análisis en función de N)
- $M_y = 10$ (solo para el análisis en función de N)
- $N = 100$ vectores de muestras (solo para el análisis en función de M)
- $\frac{\sigma_d}{d} = 0$

obteniendo los resultados que se indican en la Figura 3.9.

Como puede observarse, ambas simulaciones evolucionan de manera similar en todos los rangos, lo cual indica que este algoritmo no muestra preferencias con respecto a las dimensiones de la matriz de entrada. Siendo que para la aplicación de este proyecto se espera una cantidad fija de 16 elementos en el arreglo la gráfica que más interesa en este momento es la del tiempo de cómputo de la SVD en función de N , ya que indica a partir de qué cantidad de muestras este algoritmo complica la conformación de un sistema que opere en tiempo real. Es por esto que en la Figura 3.10 se muestra una nueva simulación del tiempo de cómputo de la SVD en función de N pero fijando M en 16 elementos.

Aquí puede verse que, para una cantidad de 16 elementos en el arreglo, la SVD puede realizarse en tiempos menores a la décima de segundo para una cantidad de vectores de muestras del orden de $N = 10^3$, y tiempos menores a la centésima de segundo con una cantidad de vectores del orden de $N = 500$. A partir de esta gráfica y la gráfica de la Figura 3.7 puede escogerse una tolerancia al error y un tiempo de ejecución para un cierto valor de SNR y así luego escoger la cantidad de muestras con las cuales se va a trabajar en la implementación real.

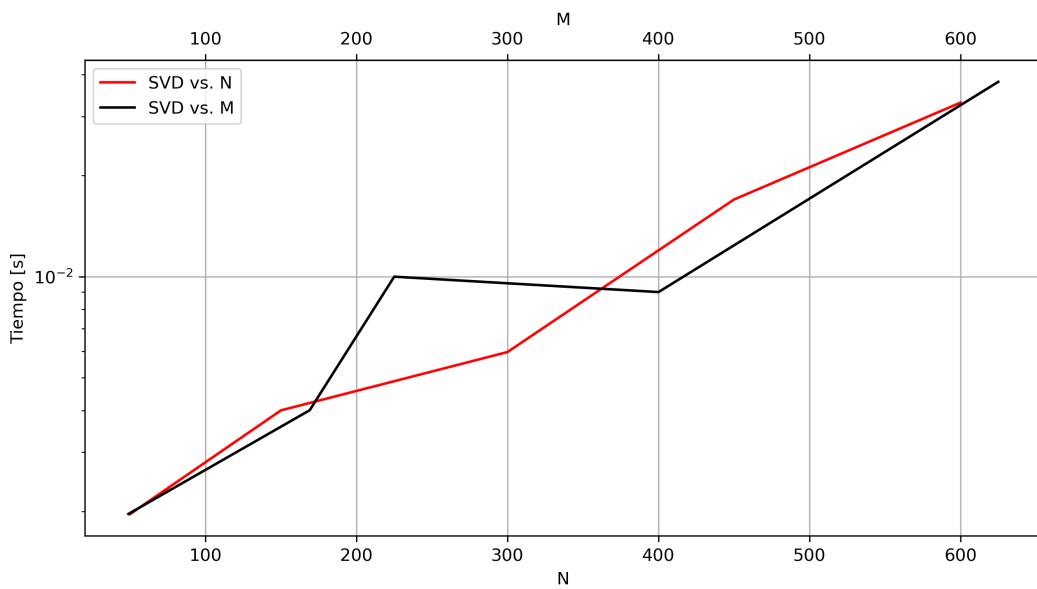


Figura 3.9: Gráfica de tiempo de ejecución del algoritmo SVD en función de la cantidad M de elementos del ARU y la cantidad N de vectores de muestra.

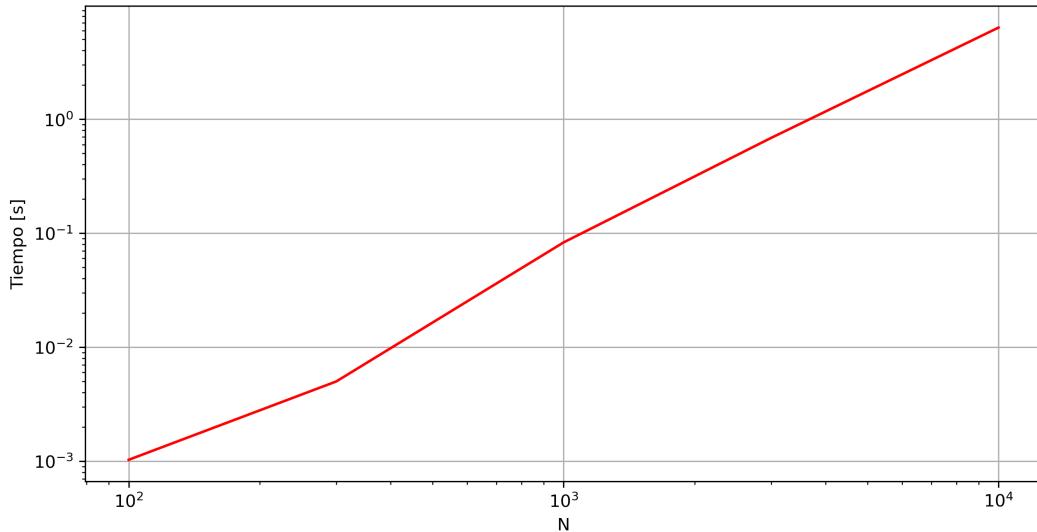


Figura 3.10: Gráfica de tiempo de ejecución del algoritmo SVD en función de la cantidad N de vectores de muestra de la entrada con $M = 16$.

Capítulo 4

Muestreo aleatorio

“Debo esa variedad casi atroz a una institución que otras repúblicas ignoran o que obra en ellas de modo imperfecto y secreto: la lotería.”

— Jorge Luis Borges

4.1. Introducción

Durante el desarrollo de las simulaciones de prueba de los algoritmos de estimación de DOA se observó que para el caso en el que se tenía más de una señal arribando al arreglo con anchos de banda $BW \ll f_s$ (siendo f_s la frecuencia de muestreo), **en el caso en el en el que existieran ventanas de tiempo en el que estas señales se encontraban muy correlacionadas** no se podía realizar una correcta estimación de la dirección de arribo de ambas señales a menos que se tomara una ventana de tiempo mayor con una cantidad de muestras que hacían impracticable su uso en tiempo real. Para explicar esto se tomará como ejemplo el caso en el que las señales arribantes son senoidales puras de baja frecuencia comparadas a la frecuencia de muestreo.

Suponiendo el caso en el que dos señales senoidales $s_A(t)$ y $s_B(t)$ de frecuencias $f_A = 5$ kHz y $f_B = 7$ kHz respectivamente arriban a un arreglo de antenas en fase cuyos elementos son muestreados a una frecuencia $f_s = 1$ MHz en ausencia de ruido, se puede representar la señal recibida en un elemento del arreglo como se muestra en la Figura 4.1a, en donde se grafican las primeras 200 muestras de la misma, las cuales son suficientes para muestrear al menos un período de ambas señales. Si se quisiese aplicar algunas de las técnicas de estimación de DOA que se mostraron en el Capítulo 3, esta cantidad de muestras debería ser suficiente para poder distinguir una señal de la otra y así poder lograr una buena estimación de la matriz de autocorrelación de señal \mathbf{R}_{SS} definida en la Ecuación 3.8, la cual se forma bajo el supuesto de que las señales recibidas no están correlacionadas. Si se quisiese reducir el número de muestras con las que se desea operar con el objetivo de minimizar los costos computacionales se podrían tomar solo las primeras 25 muestras y tendríamos la representación que se muestra en la Figura 4.1b. En esta gráfica puede observarse que la ventana de tiempo utilizada para tomar las 25 muestras con las que se desea representar la suma de ambas señales no es suficiente como para darles tiempo a que varíen de manera tal que puedan brindar la información necesaria para poder separarlas una de otra, debido a sus bajas frecuencias con respecto a la frecuencia de muestreo. En este intervalo de tiempo ambas señales se encuentran fuertemente correlacionadas y esta condición no permite una correcta estimación de la matriz \mathbf{R}_{SS} , necesaria para el funcionamiento de los algoritmos de estimación de DOA.

Una solución a este problema es reducir la frecuencia de muestreo, de manera tal de obtener mayor

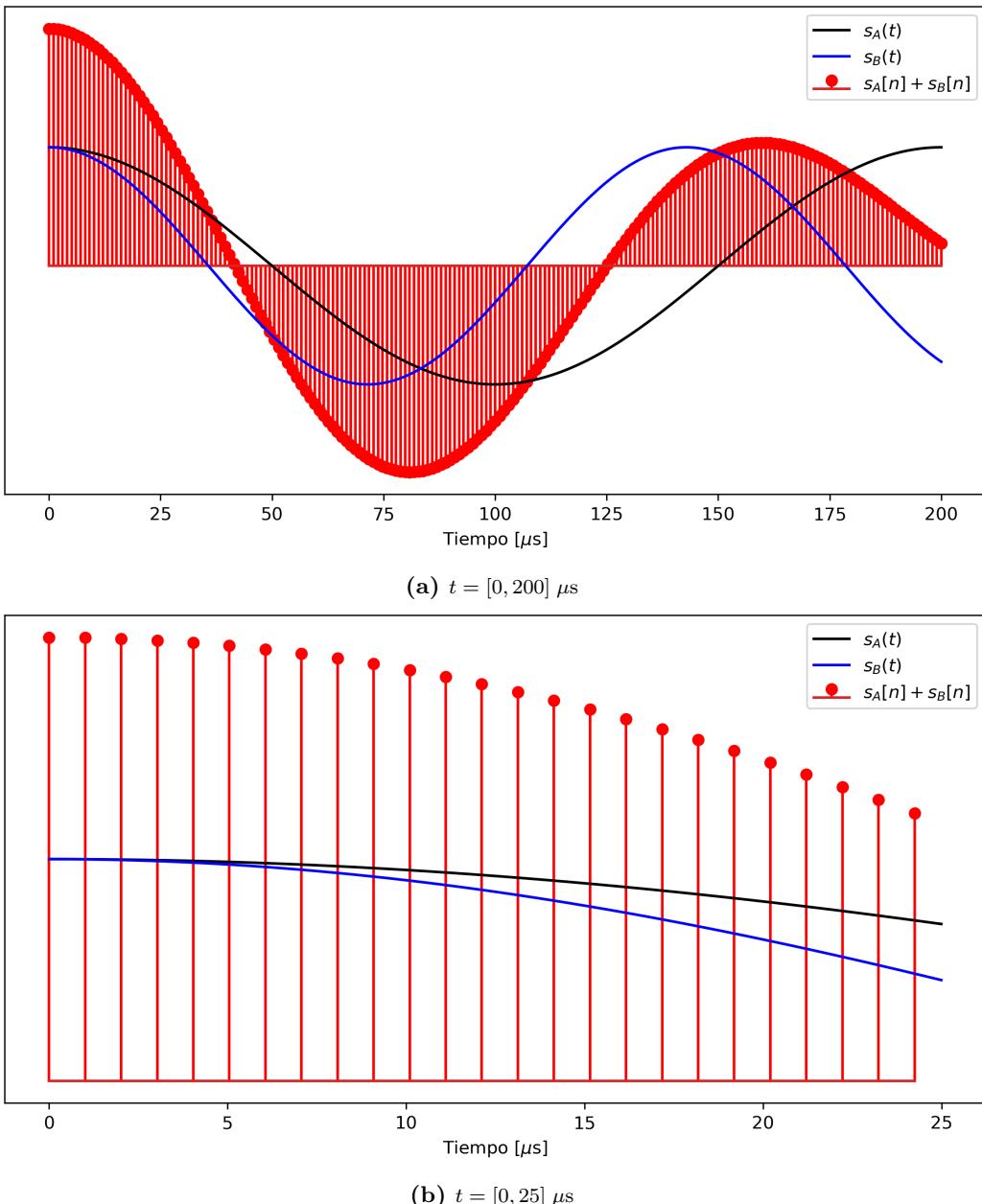


Figura 4.1: Señal resultante al muestrear a $f_s = 1$ MHz un elemento de un arreglo de antenas al cual le llegan dos señales senoidales de frecuencias $f_A = 5$ kHz y $f_B = 100$ kHz.

información sobre la variación de la señal resultante manteniendo la cantidad de muestras con las que se desea trabajar, pero este enfoque reduce el ancho de banda de las señales que se pueden detectar y reconstruir, debido a lo que enuncia el teorema de Nyquist [15]. Otro enfoque que permite resolver este problema consiste en muestrear la señal recibida utilizando la máxima frecuencia disponible, pero eligiendo aleatoriamente los instantes de muestreo, como se muestra en la Figura 4.2, en la cual se tomaron 25 muestras aleatoriamente a lo largo de $200\ \mu\text{s}$. Esta gráfica induce a pensar que estas 25 muestras tomadas en un intervalo de tiempo más separado contienen más información que las muestras de la Figura 4.1b, y al mismo tiempo se mantiene la frecuencia de muestreo, **lo cual permite aplicar el mismo método a señales de mayor ancho de banda**.

A lo largo de este capítulo se esbozará una justificación teórica de esta propuesta para luego indicar con simulaciones las mejoras alcanzadas.

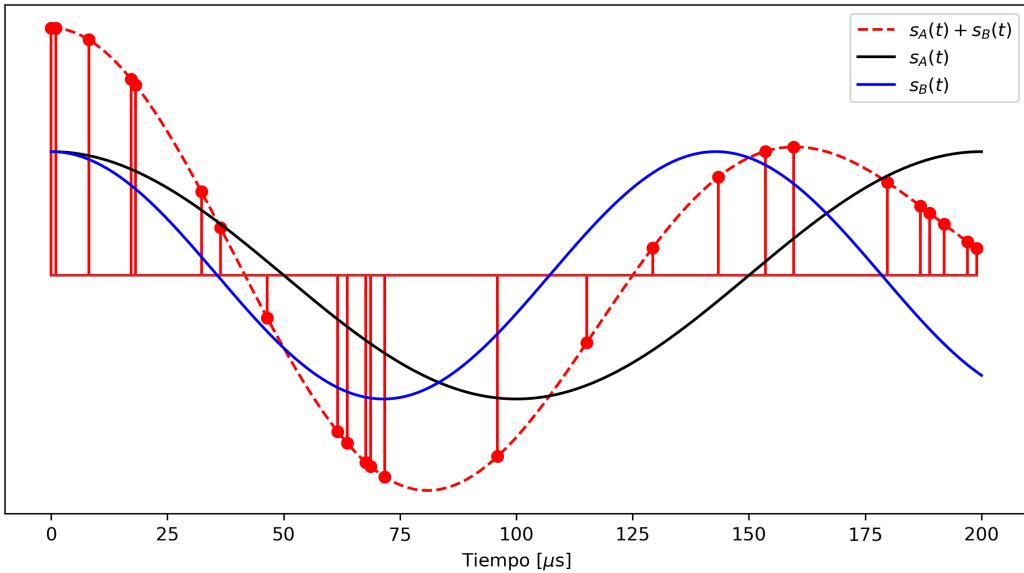


Figura 4.2: Señal resultante al muestrear a $f_s = 1$ MHz un elemento de un arreglo de antenas al cual le llegan dos señales senoidales de frecuencias $f_A = 5$ kHz y $f_B = 100$ kHz escogiendo 25 muestras utilizando muestreo aleatorio.

4.2. Definición del problema

Para resolver el problema de distinguir dos señales dadas $f(t)$ y $g(t)$ puede definirse una buena métrica que viene de la noción geométrica de considerar qué tan “alienadas” se encuentra una con respecto a la otra. Esta noción es de especial interés cuando se puede considerar que se trabaja con espacios vectoriales (como lo es el espacio de señal), ya que dicha métrica puede determinarse obteniendo el producto interno entre las señales, el cual está definido como:

$$\langle f, g \rangle = \int_T f(t)g^*(t)dt, \quad (4.1)$$

siendo T el intervalo temporal en el cual se realiza la integración. Si multiplicamos esta ecuación por T^{-1} tenemos (en el contexto adecuado) un estimador de la correlación cruzada entre dos procesos estocásticos f y g , ya que:

$$\frac{1}{T} \int_T f(t)g^*(t)dt = \mathbf{E}[f(t), g(t)] = \mathbf{R}_{f,g} \quad (4.2)$$

De aquí en adelante nos atendremos a la correlación como medida de distinguibilidad de las señales con las que trabajamos.

En lo que sigue vamos a restringir el análisis a señales centradas alrededor de una portadora de frecuencia $\omega_c = 2\pi f_c$ y que pueden considerarse de banda angosta, es decir, que su ancho de banda $BW \ll \omega_c$. Luego, para señales de la forma $x(t) = \text{Re}\{a(t)e^{j\omega_c t}\}$, podemos decir que la envolvente compleja $a(t) = \rho(t)e^{j\phi(t)}$ es tal que verifica que $\rho(t) \approx \text{cte.}$ para intervalos de tiempo un orden de magnitud mayores al período de la portadora.

Como ejemplo para motivar el resto del análisis, se consideran dos señales $x(t)$ e $y(t)$ de banda angosta, donde $x(t)$ es una referencia nominal contra la que se quiere comparar una señal $y(t)$ recibida. En particular:

$$\begin{aligned} x(t) &= \text{Re}\{a(t)e^{j\omega_c t}\} \approx a \cos(\omega_c t) \\ y(t) &= \text{Re}\{b(t)e^{j\omega_c t+\phi(t)}\} \approx b \cos(\omega_c t + \phi(t)) \end{aligned} \quad (4.3)$$

La correlación cruzada entre $x(t)$ e $y(t)$ viene dada por:

$$\hat{\mathbf{R}}_{xy} = \frac{1}{T} \int_0^T ab \cos(\omega_c t) \cos(\omega_c t + \phi(t)) dt \quad (4.4)$$

donde $\phi(t)$ puede ser un desfasaje constante o tener un comportamiento arbitrario en función del tiempo. Se analiza el caso en el que:

$$\phi(t) \approx \phi_0 + \delta\omega t, \quad (4.5)$$

con ϕ_0 y $\delta\omega$ constantes. Debe notarse que para señales de banda angosta la aproximación de la Ecuación 4.5 puede resultar bastante general incluso para tiempos que representen varios períodos de la portadora, y en cuyo caso también valdrá que $\delta\omega \ll \omega_c$. Para tener una intuición del problema que se está planteando, $\phi(t)$ podría estar representando el corrimiento Doppler de una señal recibida respecto a los valores nominales de frecuencia de la portadora.

Sin pérdida de generalidad, se considera $\phi_0 = 0$, en cuyo caso, la Ecuación 4.4 resulta ser:

$$\hat{\mathbf{R}}_{xy} = \frac{1}{2} [\text{sinc}(2(2f_c + \delta f)T) + \text{sinc}(2\delta f T)] \approx \frac{1}{2} [\text{sinc}(4f_c T) + \text{sinc}(2\delta f T)] \quad (4.6)$$

con $\delta f = \delta\omega/(2\pi)$ y habiendo usado la suposición $\delta f \ll f_c$ para la aproximación. Si se dispone de un tiempo de observación T lo suficientemente grande, la Ecuación 4.6 dará que la correlación cruzada tiende a cero, y esta es la magnitud que representa la relación de interés entre las dos señales. Si se considera adicionalmente que $|\text{sinc}(x)| \leq \pi^{-1} \frac{1}{|x|}$ podemos deducir que la condición para obtener la correlación cruzada entre $x(t)$ e $y(t)$ con un error de a lo sumo ϵ debe ser:

$$T > \frac{1}{2\pi\delta f\epsilon} \quad (4.7)$$

donde nuevamente se supuso $\delta f \ll f_0$.

Si ahora se consideran señales de tiempo discreto que representan a las señales de interés muestreadas a una cierta tasa de muestreo $f_s = \frac{1}{T_s}$, entonces se puede aproximar la Ecuación 4.4 haciendo:

$$\hat{\mathbf{R}}_{xy} := T_s \sum_{n=0}^{(N-1)T_s} x(nT_s)y(nT_s) \quad (4.8)$$

con $T = nT_s$.

A esta altura ya se está en condiciones de enunciar el problema que motiva el presente capítulo: por un lado, se desea que la Ecuación 4.8 sea una buena aproximación de 4.6 para un error ϵ suficientemente pequeño, y por otro, se quiere que la cantidad de muestras N sea lo más pequeña posible para disminuir la carga computacional.

Antes de enunciar la proposición que motiva el presente estudio, se enumeran algunas alternativas *naive* que fueron brevemente mencionadas en la Sección 4.1 para intentar dar una solución al problema planteado:

- Dado un cierto T_s fijo que hace que la Ecuación 4.8 pueda considerarse una buena aproximación de la Ecuación 4.6 se podría escoger un $N_1 < N$, donde $NT_s = T$, pero en este caso no se puede cumplir con condición de la Ecuación 4.7. Más aún, de la forma misma que adopta la correlación cruzada, se puede ver que si se achica mucho el número de muestras consideradas, se puede llegar a valores de correlación cruzada que indiquen una considerable correlación entre las señales comparadas, y eso es exactamente lo opuesto a lo que debe manifestarse.
- Si se fija el intervalo de integración T se garantiza que se verifica la Ecuación 4.7 en tiempo

continuo, pero para pasar a discreto, si se desea achicar N aparece la obligación de aumentar T_s . De este modo la aproximación de la Ecuación 4.8 a la Ecuación 4.6 es el factor limitante, y con la disminución del número de muestras rápidamente se pierde convergencia al valor buscado. Para visualizar esto de un modo intuitivo se puede pensar lo siguiente: si se disminuye la frecuencia de muestreo f_s al punto de dejar de cumplir con el teorema de muestreo de Nyquist [15], luego se manifestarán en el espectro de la señal discretizada alias correspondientes a frecuencias menores. Como se vio de la Ecuación 4.6 y la condición de la Ecuación 4.7, cuanto menores sean las frecuencias involucradas, el tiempo $T = NT_s$ necesario para poder distinguirlas (es decir, que la correlación cruzada converja a menos del error ϵ) será mayor.

Como se vio, disminuir la cantidad de muestras ya sea achicando el intervalo de integración numérica o, si se deja este último fijo, aumentando el tiempo entre muestras, no conduce a los resultados esperados. Intuitivamente, si se quiere achicar la cantidad de muestras pero conservando el intervalo total de observación y al mismo tiempo sin perder información que ocurre a escalas temporales pequeñas, no parece haber alternativa a utilizar algún tipo de distribución de tiempos entre muestras que no sea uniforme. En particular, como a priori se desea trabajar con señales genéricas, lo que se propone es que esta distribución siga algún tipo de ley aleatoria que no favorezca **ningún escenario** en particular. Esto lleva a enunciar el algoritmo de *muestreo aleatorio* que se desarrolla en la próxima sección.

Como última observación, cabe destacar que al quitar muestras ocurre que porciones de las señales que se encontraban correlacionadas seguirán correlacionadas mientras que otras que estaban decorrelacionadas no lo estarán más, es por esto que el análisis llevado a cabo se para en este eslabón débil para motivar el método.

4.3. Muestreo Aleatorio

Teniendo en mente la intuición que surgió de la motivación de la sección anterior, se considera lo que sucede con una versión en tiempo discreto de la señal $z(t) = x(t)y(t)$. Se considera como señal muestreada a:

$$z_m(t) = z(t)s_m(t), \quad (4.9)$$

donde

$$s_m(t) = \left(\sum_{n \in \mathbb{Z}} \delta(t - nT_s) \right) m(t) \quad (4.10)$$

y $m(t)$ es una *máscara de muestreo*. Si, por ejemplo, $m(t) = 1$ para todo t , luego se tiene que $z_m(t)$ se corresponde a $z(t)$ muestreada idealmente y la correlación de la Ecuación 4.8 se puede calcular como:

$$\mathbf{R}_{xy} = \frac{T_s}{T} \int_0^T z_m(t) dt \quad (4.11)$$

de forma exacta. Sin embargo, como ya se adelantó, interesa utilizar una forma de muestreo aleatorio que consiste en definir la máscara $m(t)$ como una realización del proceso estocástico definido por:

$$M(t) = \sum_{n \in \mathbb{Z}} \Pi \left(\frac{t - nT_s}{T_s} \right) \cdot c_n \quad (4.12)$$

siendo

$$\Pi(t) = \begin{cases} 1 & t \in [0, 1) \\ 0 & \text{c.o.c.} \end{cases}, \quad (4.13)$$

y c_n un proceso aleatorio discreto tal que $c_n = 1$ con probabilidad p , $c_n = 0$ con probabilidad $1-p$ y c_n es independiente de c_m para todo $n \neq m$. Nótese que para $p = 1$ se está en el caso de muestreo ideal, y para $p = 0$ no se utiliza ninguna muestra. El valor esperado de muestras utilizando este enfoque es pT/T_s .

La clave está en notar que la correlación que se busca aproximar, es decir la indicada por la Ecuación 4.11, se corresponde a calcular numéricamente el valor medio temporal de la señal $z(t)$ a partir de un instante inicial $t = 0$. Es decir, interesa específicamente lo que ocurre con la componente de continua de dicha señal. Por otro lado, dada una realización $m(t)$ del proceso $M(t)$ para una cierta probabilidad p , se puede reescribir:

$$m(t) = \mu_m + m_0(t), \quad (4.14)$$

con μ_m siendo una constante que representa el valor medio de $m(t)$ y cuyo valor esperado es p , y $m_0(t) = m(t) - \mu_m$, siendo $m_0(t)$ una realización del proceso $M(t) - p$, es decir, un proceso con las mismas características que $M(T)$ pero de media nula.

Ahora se está en condiciones de calcular la Ecuación 4.11 que comprende el resultado principal de esta propuesta. Esto es:

$$\frac{T_s}{T} \int_0^T z_m(t) dt = \mu_m \mathbf{R}_{xy} + \frac{T_s}{T} \int_0^T z(t) m_0(t) dt \quad (4.15)$$

El primer término es la correlación que se busca determinar a menos de un factor μ_m conocido, y el segundo término representa el “ruido” que se agrega por la utilización del método de muestreo aleatorio propuesto. Para mayor claridad, si se multiplica por μ_m^{-1} ambos lados de la igualdad se tiene:

$$\mu_m^{-1} \left(\frac{T_s}{T} \int_0^T z_m(t) dt \right) = \mathbf{R}_{xy} + \mu_m^{-1} \left(\frac{T_s}{T} \int_0^T z(t) m_0(t) dt \right), \quad (4.16)$$

lo que deja en evidencia que al disminuir el valor de μ_m y, por lo tanto, usar menos muestras, también se amplifica el valor del ruido (ya que $0 < \mu_m \leq 1$). También se puede estimar con gran precisión cuánto vale ese ruido en la determinación de \mathbf{R}_{xy} . Suponiendo que $T_s \ll T$ y que $z(t)$ es independiente de $m_0(t)$, se puede mostrar fácilmente que:

$$\left| \int_0^T z(t) m_0(t) dt \right| \leq \frac{M_z T_s}{2} \quad (4.17)$$

con $M_z = \max_{0 \leq t \leq T} (|z(t)|)$, de modo que el ruido en la estimación de la correlación \mathbf{R}_{xy} por el método de muestreo aleatorio queda acotado por:

$$\mu_m^{-1} \frac{M_z}{2} \frac{T_s^2}{T} \quad (4.18)$$

A modo de ejemplo, en la Figura 4.3 se presenta el cálculo numérico de la correlación \mathbf{R}_{xy} para señales $x(t)$ e $y(t)$ como las presentadas en la Sección 4.2 para $\delta_\omega = \frac{\omega_0}{10}$ y utilizando muestreos aleatorios con distintos valores de p . Las funciones de correlación están calculadas para T variando entre 0 y un valor suficientemente grande. Como se puede ver, tanto para p grandes como pequeños la aproximación es muy buena.

Finalmente en la Figura 4.4 se muestra una simulación que se realizó emulando la recepción de beacons de los satélites GOMX-1 y AISTECHSAT-3 en distintas direcciones y comparando el error obtenido en la estimación utilizando muestreo ideal ($p = 1$) con el obtenido utilizando muestreo aleatorio para distintos valores de p . Se puede observar que hasta para $p = 0, 1$, es decir, utilizando

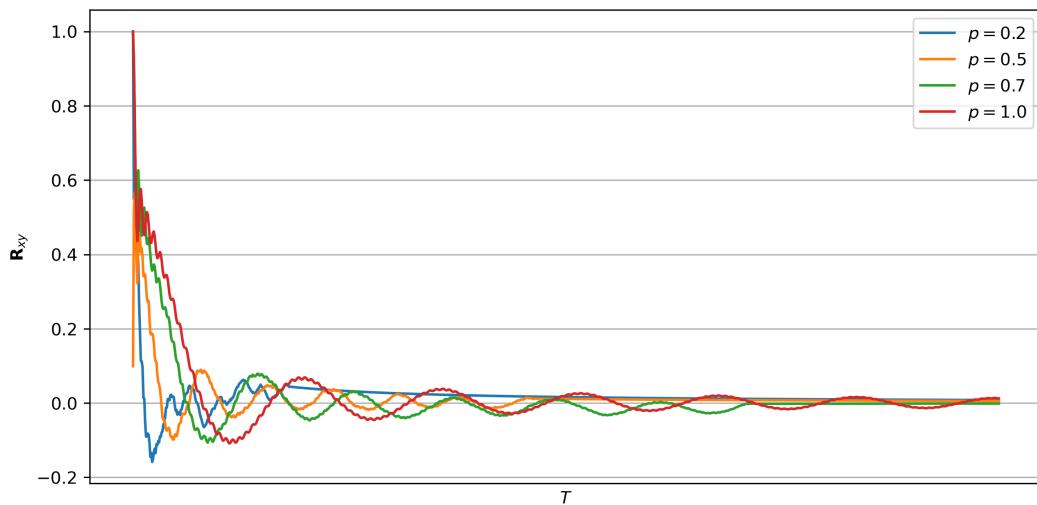


Figura 4.3: Correlación cruzada entre $x(t)$ e $y(t)$ en función de T utilizando muestreo aleatorio con distintos valores p .

el 10 % de las muestras totales, el error obtenido se encuentra en el mismo orden que para el caso de $p = 1$. Para el caso de $p = 0,01$ el error sube un orden de magnitud con respecto al muestreo secuencial. También se observa cómo el error disminuye a medida que aumenta la ventana de observación T . En la práctica habrá que analizar esta relación de compromiso eligiendo valores de p y T que generen un error tolerable y al mismo tiempo permitan tener un tamaño de muestras que puedan ser operables en tiempo real.

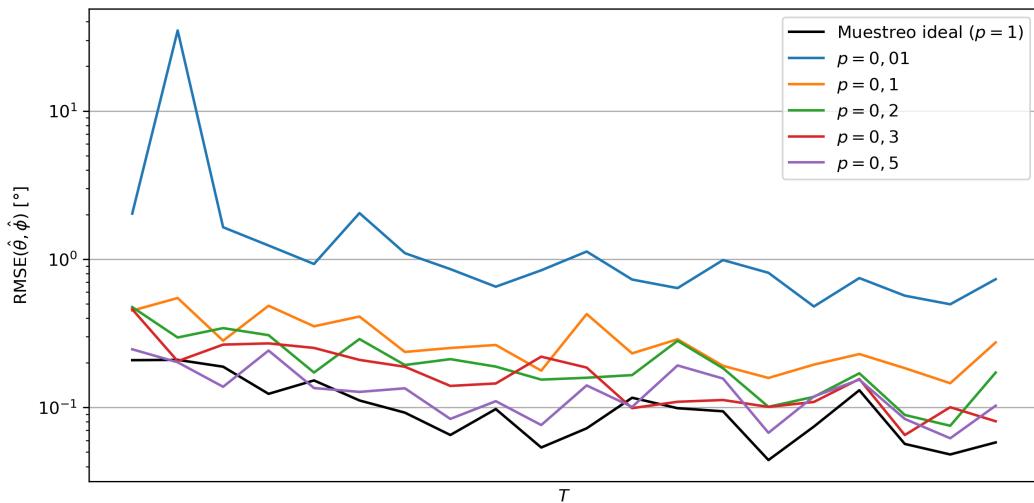


Figura 4.4: Gráfica de comparación del RMSE en función de la ventana de observación T para distintos valores de p al recibir dos señales en distintas direcciones.

Capítulo 5

Estimación del número de señales recibidas

“Comin’ at you from every side...”

— Mike Shinoda

5.1. Introducción

Una gran capacidad que tienen los sistemas de conformación de haz digitales frente a los analógicos es la de poder realizar la conformación de más de una señal simultáneamente utilizando un único sistema. Además, la versión adaptativa de este tipo de conformadores de haz permiten no solo ajustarse automáticamente a cambios en las direcciones de arribo de las señales sino, además, poder estimar qué cantidad de señales están siendo sensadas por los elementos del arreglo de antenas. Esta magnitud es de extrema importancia en todos los algoritmos de estimación de dirección de arribo ya que define todas las dimensiones de las matrices con las que se debe operar, comenzando por las matrices $\hat{\mathbf{E}}_S$ y $\hat{\mathbf{E}}_N$ definidas en la Ecuación 3.11, que son las matrices conformadas por las estimaciones de los autovectores de los subespacios de señal y ruido respectivamente.

Como se indicó en la Sección 3.1.1 el problema de estimar la cantidad de señales recibidas consiste en separar los **valores singulares de la SVM de la matriz \mathbf{X}** definida en la Ecuación 3.4 entre aquellos de mayor valor, los cuales corresponden a los valores singulares del subespacio de señal, y aquellos más chicos, que corresponden al subespacio de ruido. En el caso ideal en el que ambos subespacios pueden ser estimados perfectamente esta tarea no conlleva mayor dificultad debido a que los valores singulares del subespacio de ruido tienen todos la misma magnitud, por ende es fácil separarlos del resto. En la práctica, cuando lo que se tiene son estimaciones de estos subespacios, esta característica no se observa y las magnitudes de los valores singulares del subespacio de ruido pueden diferir entre ellos.

En la Figura 5.1 se muestra una simulación en la que se obtuvieron los valores singulares de una matriz de muestras \mathbf{X} armada a partir de la recepción hecha por un ARU de $M = 16$ elementos de dos señales en direcciones distintas con una SNR de 10 dB cada una. Como puede verse, en este caso se puede determinar fácilmente un umbral a partir del cual cualquier valor singular que esté por encima sea considerado como correspondiente al subespacio de señal. Sin embargo, la práctica demuestra que este umbral no es constante sino que varía según las características de la señal recibida, como lo son la SNR de las mismas, la cantidad de señales a detectar y cuán correlacionadas se encuentran entre

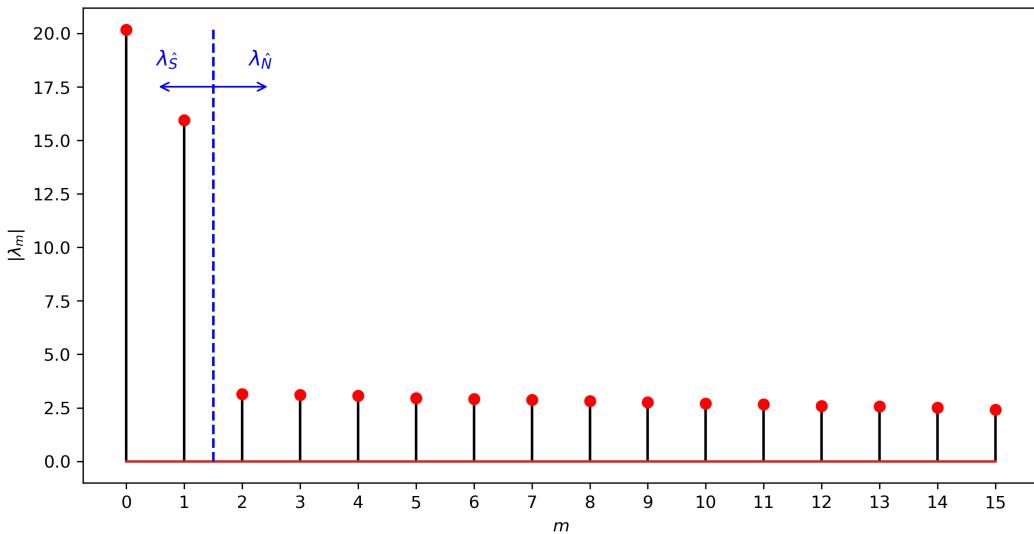


Figura 5.1: Distribución de valores singulares del subespacio de muestras ordenados de mayor a menor para el caso de dos señales arribando a un arreglo de 16 elementos. La línea azul indica la separación entre valores singulares correspondientes al subespacio de señal y al subespacio de ruido.

ellas, por ende esta propuesta no es aplicable en general.

En este capítulo se presentan y comparan dos métodos de estimación de número de señales recibidas que se basan en el análisis de los valores singulares del subespacio de muestras.

5.2. Método de la máxima derivada primera

El método de la máxima derivada primera es un método sencillo de comprender e implementar, y se basa en la premisa de que, aunque en la práctica los valores singulares del subespacio de ruido son distintos entre sí, cuanto mejor sea su estimación menor va a ser la diferencia entre ellos. Esto puede apreciarse en la Figura 5.1, donde se ve que los valores singulares pertenecientes al subespacio de ruido están contenidos en un rango de valores muy pequeños. Si los valores singulares son ordenados de menor a mayor y se aplica la derivada a la distribución obtenida se obtiene la curva de la Figura 5.2. A partir

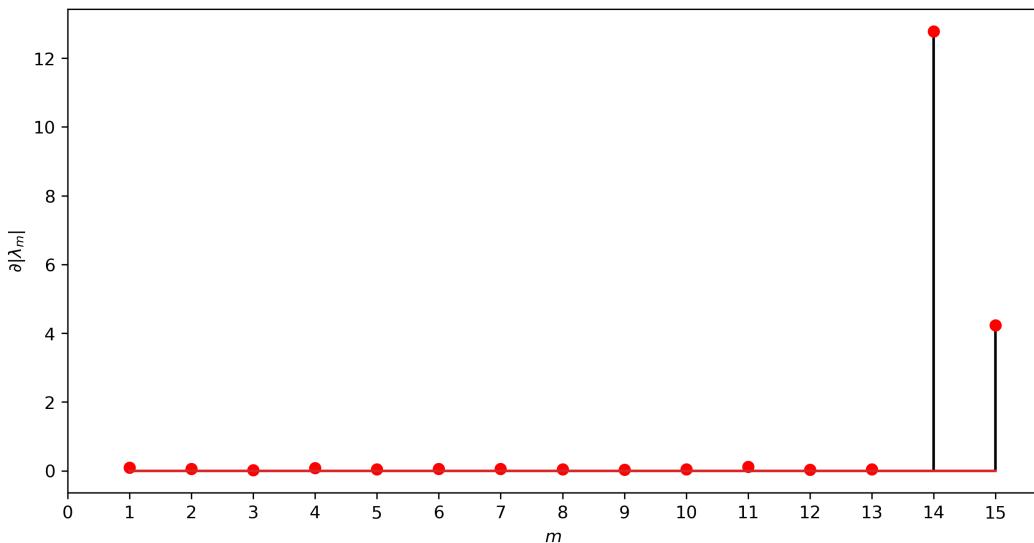


Figura 5.2: Derivada de la distribución de valores singulares mostrados en la Figura 5.1 ordenados de menor a mayor.

de esta figura se puede observar que la clasificación de valores singulares puede hacerse fácilmente considerando todos los valores singulares cuyos índices son menores al índice correspondiente al valor máximo de la derivada de la distribución.

Esta técnica no solo requiere de una buena estimación del subespacio de ruido sino que requiere que las señales sensadas tengan niveles de potencia semejantes, ya que si la diferencia entre ellas es muy grande es posible que la derivada máxima ocurra dentro del rango de valores singulares de señal y se terminen detectando menos señales que las que existen en realidad. Esta suposición es muy fuerte y en la práctica se ve que, según con qué tipo de señales se esté trabajando, es probable que no se cumpla siempre.

5.2.1. Resultados obtenidos

En base a la técnica que se detalló en esta sección se implementó el algoritmo de estimación de cantidad de señales recibidas utilizando el método de la máxima derivada. Para obtener un conjunto de datos de prueba se simularon 100 iteraciones de recepción simultánea de 4 señales con un ARU de 16 elementos variando aleatoriamente los siguientes parámetros:

- Modulación: BPSK, QPSK, DQPSK o 8PSK
- $\theta \sim U(0^\circ; 90^\circ)$
- $\varphi \sim U(-180^\circ; 180^\circ)$
- Voltaje de ruido $W_0 \sim U(0,01 \text{ V}; 1 \text{ V})$
- Error de separación entre elementos $\frac{\sigma_d}{d} \sim U(0\%; 5\%)$

A partir de estas simulaciones se obtuvieron 51376 valores singulares, de los cuales el 25 % correspondían a valores singulares de señal y el resto a valores singulares de ruido. Como por cada descomposición en valores singulares que toma lugar en cada estimación de DOA se obtienen $M = 16$ valores singulares (siendo M la cantidad de elementos en el arreglo), el clasificador recibe como entrada un vector de M valores singulares obtenidos a partir de la misma SVD y devuelve un vector binario de M elementos, en el cual se indican con un 1 aquellas posiciones que corresponden a valores singulares de señal y con 0 a aquellas que corresponden a valores singulares de ruido, como se muestra en la Figura 5.3.

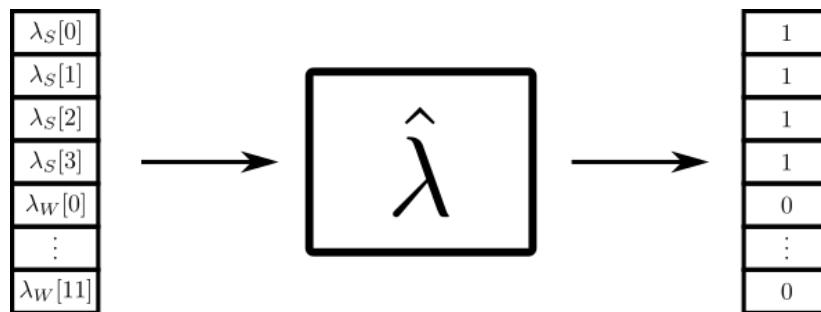


Figura 5.3: Esquema del estimador de cantidad de señales recibidas mediante el método de la máxima derivada visto como sistema para el caso de recepción de 4 señales. Se indican con λ_S aquellos valores singulares correspondientes al subespacio de señal y con λ_W aquellos que corresponden al subespacio de ruido.

Con esta simulación se compararon los vectores entregados por el algoritmo con los datos conocidos de los valores singulares clasificados para obtener una cuenta de los errores en la estimación y así obtener una medición de la precisión del algoritmo, la cual fue del 86,87 %.

5.3. Clasificador binario mediante aprendizaje automático

El problema de determinar si un valor singular corresponde al subespacio de ruido o al subespacio de señal **entra** dentro de lo que en el campo del *aprendizaje automático* o *machine learning* se conoce como “problema de clasificación binaria”.

En este tipo de problemas se intenta predecir el valor de una variable y la cual se sabe que puede tomar dos valores, 0 y 1, que en este caso específico corresponden a la característica de si cada valor singular es un valor singular de ruido o de señal respectivamente, es decir:

$$y = \begin{cases} 0 & \text{si corresponde a } \mathcal{S}_W \\ 1 & \text{si corresponde a } \mathcal{S}_S \end{cases} \quad (5.1)$$

Esta predicción debe realizarse a partir de una entrada \bar{x} , la cual para esta aplicación en particular va a estar conformada por los parámetros que caracterizan a cada valor singular que se desea clasificar, y una hipótesis $h_{\bar{\theta}}(\bar{x})$ la cual no es más que una función que al recibir una entrada \bar{x} entrega a la salida una predicción de la probabilidad de que y sea igual a 1 para dicha entrada y para una elección de coeficientes representados en los elementos de un vector $\bar{\theta}$. Luego, configurando el umbral correspondiente, puede considerarse como $y = 1$ a valores de probabilidad por encima de 0,5 y viceversa, y de esta forma se tiene el sistema clasificador que se indica en la Figura 5.4.

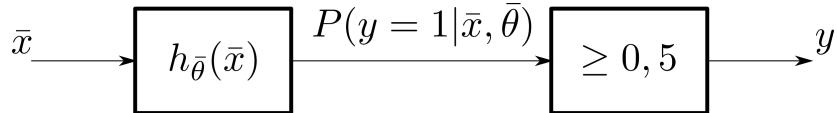


Figura 5.4: Esquema de un clasificador utilizando aprendizaje automático visto como sistema.

Existen varios tipos de algoritmos clasificadores, los cuales cada uno define su propia función de hipótesis $h_{\bar{\theta}}(\bar{x})$. A lo largo de este capítulo se analizarán dos de estos tipos; en primer lugar se explicará la técnica de *Regresión Logística* para dar una introducción a los conceptos relacionados con el aprendizaje automático, y, finalmente, se analizará el algoritmo conocido como *Máquina de Vectores de Soporte*, el cual es el algoritmo que finalmente se escogió para la implementación de este proyecto.

5.3.1. Regresión Logística

Debido a que la salida del clasificador es una variable que se encuentra contenida dentro del rango $[0, 1]$ hay que definir una función para $h_{\bar{\theta}}(\bar{x})$ que cumpla con esta característica. El algoritmo de Regresión Logística define la siguiente función de hipótesis [16]:

$$h_{\bar{\theta}}(\bar{x}) = g(\bar{\theta}^T \bar{x}), \quad (5.2)$$

$$g(z) = \frac{1}{1 + e^{-z}}, \quad (5.3)$$

donde $g(z)$ es conocida como la *función sigmoide* o *función lógistica*, cuya forma se muestra en la Figura 5.5, y $\bar{\theta}$ es un vector de coeficientes a determinar. Como se ve, esta función tiende a 1 para $z \rightarrow \infty$ y a 0 para $z \rightarrow -\infty$, y para $z = 0$ vale 0,5.

El objetivo de todos los algoritmos de aprendizaje automático es el de ajustar los elementos del vector $\bar{\theta}$ de manera tal que la hipótesis genere una frontera de decisión que permita clasificar con la mayor precisión posible un conjunto de datos representados como vectores \bar{x} . Por ejemplo, si tenemos

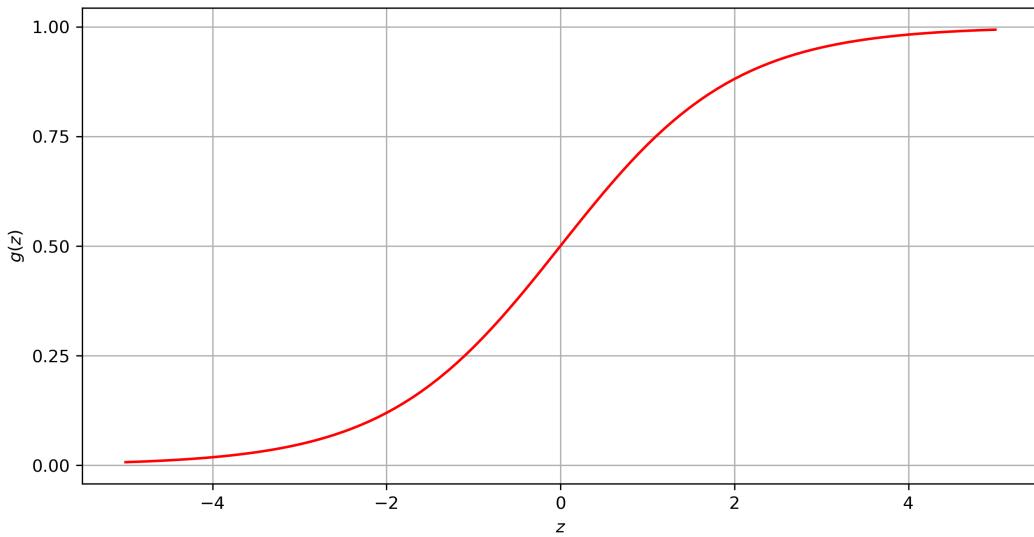


Figura 5.5: Gráfica de la función sigmoide.

el conjunto de datos que se muestra en la Figura 5.6 se pueden definir los vectores:

$$\bar{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}, \quad \bar{\theta} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix},$$

y de esta manera se tiene una función de hipótesis definida por:

$$h_{\bar{\theta}}(\bar{x}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2) = g(-1 + x_1^2 + x_2^2)$$

Debido a lo que se ve en la Figura 5.5, la función sigmoide devuelve valores por encima de 0,5 para entradas mayores a 0. Por esto, para encontrar la frontera de decisión basta con ver los puntos del argumento de $g(\bar{\theta}^T \bar{x})$ tal que:

$$\bar{\theta}^T \bar{x} = -1 + x_1^2 + x_2^2 = 0,$$

y esos puntos son los que definen la frontera circular que se muestra en la Figura 5.6.

Vale la pena notar que en el ejemplo de la Figura 5.6 los datos a clasificar dependen de dos parámetros: x_1 y x_2 . Sin embargo, el vector \bar{x} no contiene únicamente estos dos parámetros sino que agrega nuevas *características*, como lo son un término independiente $x_0 = 1$ y dos parámetros más que dependen de los originales: x_1^2 y x_2^2 . Este método de agregar características polinomiales o términos rectangulares permite lograr fronteras de decisión más complejas que ajusten mejor a un determinado conjunto de datos, aunque pueden provocar un “sobreajuste” de los parámetros que degrada el comportamiento del clasificador al trabajar con datos que se encuentren fuera del conjunto de entrenamiento.

Como se dijo, el objetivo del aprendizaje automático consiste en ajustar los valores de $\bar{\theta}$ de manera tal de encontrar una frontera de decisión óptima. La manera de realizar este ajuste es entrenando al algoritmo a partir de un conjunto de datos llamado *conjunto de entrenamiento*, de manera tal que, a medida de que el algoritmo itere, el error en la clasificación se reduzca. Para esto hay que definir una métrica de ese error, la cual es conocida como *función de costo*. Para el algoritmo de regresión logística y teniendo un conjunto de datos de tamaño M y una cantidad de características N , la función de

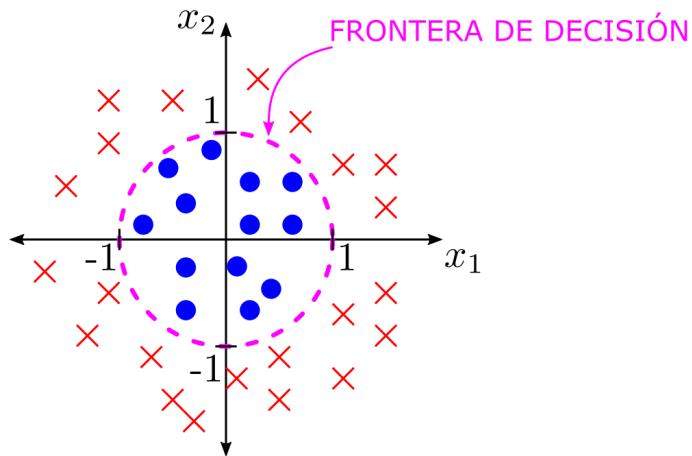


Figura 5.6: Ejemplo de frontera de decisión para un problema de clasificación binaria.

costo está definida por [16]:

$$J(\bar{\theta}, \lambda) = -\frac{1}{M} \sum_{i=0}^{M-1} \left[y^{(i)} \log(h_{\bar{\theta}}(\bar{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\bar{\theta}}(\bar{x}^{(i)})) \right] + \frac{\lambda}{2M} \sum_{j=1}^{N-1} \theta_j^2, \quad (5.4)$$

donde λ es el *parámetro de regularización* que permite suavizar la salida de la función de hipótesis para reducir el sobreajuste. Este parámetro también debe ser ajustado mediante entrenamiento. Finalmente, el problema a resolver consiste en encontrar los valores de $\bar{\theta}$ y λ que minimicen la función de costo, es decir:

$$\min_{\bar{\theta}, \lambda} J(\bar{\theta}, \lambda) \quad (5.5)$$

Para minimizar la función de costo en regresión logística el procedimiento que se utiliza es el método de *descenso de gradiente*, el cual es un algoritmo iterativo que permite acercarse con una cierta velocidad al mínimo de una función derivable. Utilizando esta técnica, la actualización de parámetros por cada iteración viene dada por:

$$\begin{aligned} \theta_0 &:= \theta_0 - \frac{\alpha}{M} \sum_{i=0}^{M-1} (h_{\bar{\theta}}(\bar{x}^{(i)}) - y^{(i)}) x_0^{(i)} \\ \theta_j &:= \theta_j - \frac{\alpha}{M} \left[\sum_{i=0}^{M-1} (h_{\bar{\theta}}(\bar{x}^{(i)}) - y^{(i)}) x_j^{(i)} + \lambda \theta_j \right] \quad \text{para } j = 1, \dots, N-1, \end{aligned} \quad (5.6)$$

donde α es el *coeficiente de aprendizaje* el cual se encarga de definir el tamaño de los “pasos” que se dan cuesta abajo en cada iteración del algoritmo. Si este α se elige muy grande es posible que el algoritmo nunca converja, y si se elige muy pequeño puede que tarde mucho tiempo en converger. Es necesario ver que con este método no se actualiza el valor del parámetro λ . La manera de elegir un valor óptimo para este parámetro consiste en entrenar el algoritmo utilizando distintos valores de λ y luego observar el rendimiento del clasificador utilizando datos que se encuentran fuera del conjunto de entrenamiento, eligiendo, finalmente, el valor de λ que mejor ajuste este nuevo conjunto de datos, el cual es llamado *conjunto de validación*.

5.3.2. Máquina de Vectores de Soporte

Uno de los algoritmos de clasificación más utilizados tanto en la industria como en la academia es el conocido como *Máquina de Vectores de Soporte* (o *SVM*, por sus siglas en inglés), el cual tiene la capacidad de poder entregar una mejor elección de una frontera de decisión a un menor costo comparado con el algoritmo de Regresión Logística. El algoritmo SVM toma la función de costos de la Ecuación 5.4, y la simplifica de la siguiente manera:

$$J(\bar{\theta}, C) = C \sum_{i=0}^{M-1} \left[y^{(i)} \text{cost}_1(\bar{\theta}^T \bar{x}^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\bar{\theta}^T \bar{x}^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^{N-1} \theta_j^2 \quad (5.7)$$

donde $\text{cost}_0(z)$ y $\text{cost}_1(z)$ son las funciones que se indican en la Figura 5.7 y $C = \frac{1}{\lambda}$.

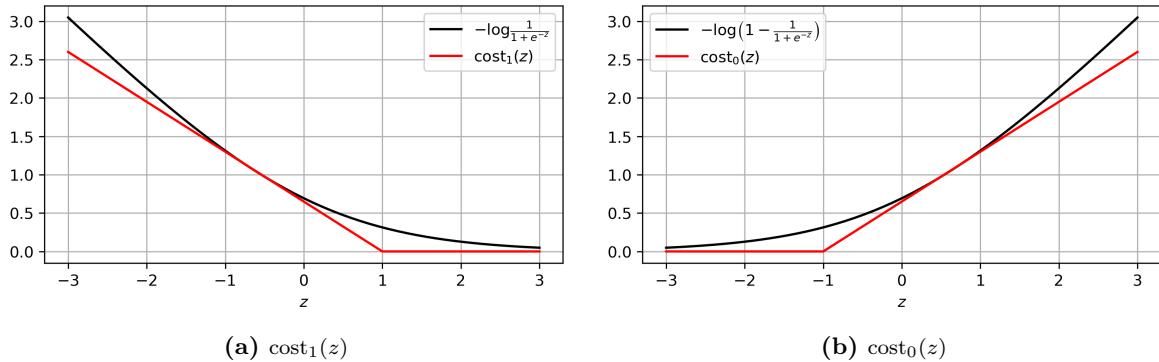


Figura 5.7: Cambios en la función de costo para SVM.

Otra diferencia que tiene SVM con respecto a Regresión Logística es que la función de hipótesis en este caso no entrega una probabilidad sino directamente la predicción de y , es decir:

$$h_{\bar{\theta}}(\bar{x}) = \begin{cases} 0 & \text{si } \bar{\theta}^T \bar{x} < 0 \\ 1 & \text{si } \bar{\theta}^T \bar{x} \geq 0 \end{cases} \quad (5.8)$$

Si se analizan las funciones $\text{cost}_0(z)$ y $\text{cost}_1(z)$ que se muestran en la Figura 5.7 puede verse que la condición para que el primer término de la función de costos definida en la Ecuación 5.7 se anule es:

$$\begin{aligned} \bar{\theta}^T \bar{x}^{(i)} &\geq 1 && \text{si } y^{(i)} = 1, \\ \bar{\theta}^T \bar{x}^{(i)} &\leq -1 && \text{si } y^{(i)} = 0. \end{aligned} \quad (5.9)$$

Cuando se resuelve el problema de optimización de minimizar la Ecuación 5.7 resulta que la frontera de decisión que se obtiene utilizando SVM maximiza la distancia entre ella y los datos más cercanos a ella de cada una de las dos clases. Para dar un ejemplo de esto se muestra el conjunto de entrenamiento de la Figura 5.8. Aquí vemos que en este caso de clasificación lineal pueden generarse infinitas rectas que logren una perfecta clasificación de este conjunto de datos, como las indicadas por las fronteras de color verde, sin embargo la frontera definida por SVM induce a pensar que va a lograr un mejor trabajo clasificando datos nuevos.

De la Figura 5.8 se observa también que la frontera de decisión generada por SVM es la que maximiza el margen de separación entre datos rojos y azules. La razón por la cual ocurre esto tiene que ver con la condición definida en la Ecuación 5.9 y la función de costos de la Ecuación 5.7 ya que

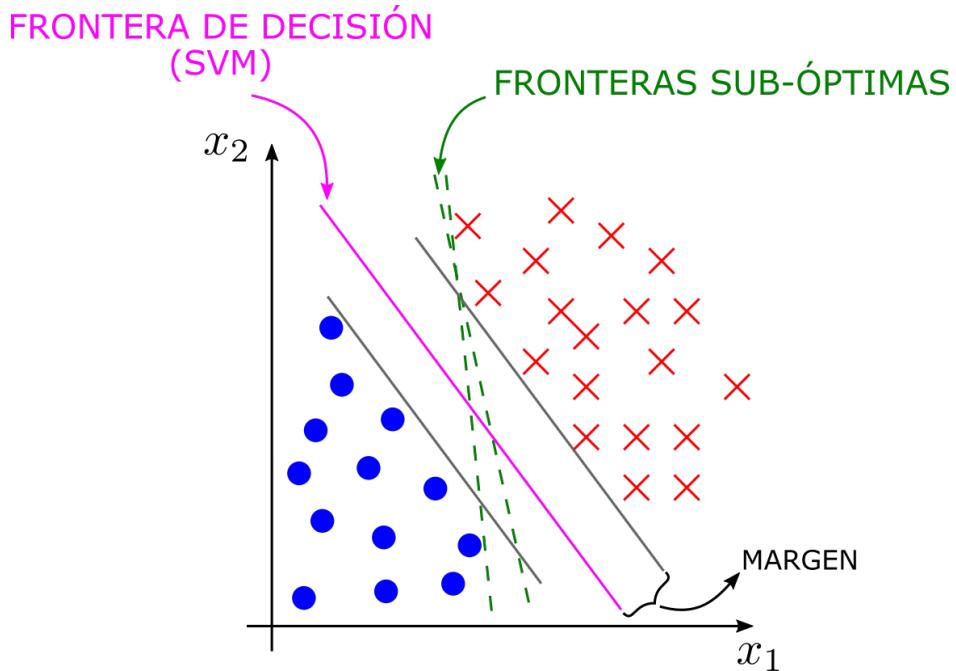


Figura 5.8: Frontera de decisión utilizando SVM.

cumpliendo esas condiciones tenemos que la función de costo queda definida por:

$$J(\bar{\theta}) = \frac{1}{2} \sum_{j=1}^{N-1} \theta_j^2 = \frac{1}{2} \|\bar{\theta}\|^2. \quad (5.10)$$

Además, las condiciones de la Ecuación 5.9 pueden reescribirse como:

$$\begin{aligned} p^{(i)} \cdot \|\bar{\theta}\| &\geq 1 & \text{si } y^{(i)} = 1, \\ p^{(i)} \cdot \|\bar{\theta}\| &\leq -1 & \text{si } y^{(i)} = 0, \end{aligned} \quad (5.11)$$

donde $p^{(i)}$ es la proyección del vector $\bar{x}^{(i)}$ sobre el vector $\bar{\theta}$, o, lo que es lo mismo, la distancia del dato $\bar{x}^{(i)}$ a la frontera de decisión, debido a que la frontera de decisión es perpendicular a $\bar{\theta}$. Esto puede verse en el ejemplo de la Figura 5.9. Si algún dato $\bar{x}^{(i)}$ se encontrara sobre la frontera de decisión

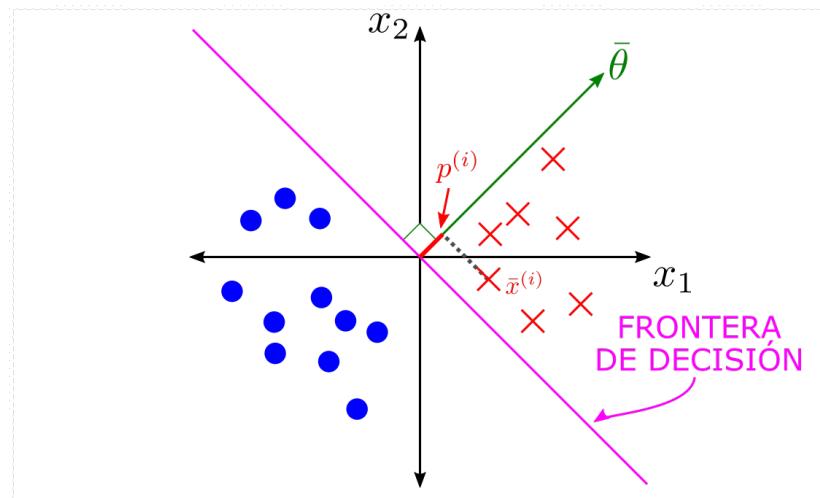


Figura 5.9: Proyección de un dato $\bar{x}^{(i)}$ sobre el vector $\bar{\theta}$ para un clasificador lineal con $\theta_0 = 0$.

estaría ubicado numéricamente en el salto de la función de hipótesis definida en la Ecuación 5.8, lo que significa que para puntos definidos por vectores $\bar{x}^{(i)}$ que se encuentren dentro de la frontera de decisión, el producto escalar $\bar{\theta}^T \bar{x}$ vale 0, lo que indica que todos los puntos dentro de la frontera de decisión son ortogonales a $\bar{\theta}$.

A partir de aquí se observa que al minimizar la Ecuación 5.10 se minimiza la norma de $\bar{\theta}$, por ende para cumplir las condiciones de la Ecuación 5.11 al minimizarse $\|\bar{\theta}\|$ debe maximizarse $p^{(i)}$, lo que significa maximizar la distancia entre los datos del conjunto de entrenamiento y la frontera de decisión.

Finalmente, SVM implementa una nueva manera de definir las características del vector \bar{x} distinta a la manera polinomial que se vio en Regresión Logística. Teniendo un conjunto de M datos de entrenamiento se define el vector de características de un dato $\bar{f}^{(i)} \in \mathbb{R}^{M \times 1}$ para $i = 0, 1, \dots, M - 1$ como:

$$\bar{f}^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ \vdots \\ f_{M-1}^{(i)} \end{bmatrix}_{(M \times 1)}, \quad (5.12)$$

donde $f_j^{(i)} = k(\bar{x}^{(i)}, \bar{x}^{(j)})$ es una función que mide la “semejanza” entre el dato $\bar{x}^{(i)}$ y el dato $\bar{x}^{(j)}$, la cual es llamada *función kernel*. Ahora la función de hipótesis queda definida por:

$$h_{\bar{\theta}}(\bar{f}) = \begin{cases} 0 & \text{si } \bar{\theta}^T \bar{f} < 0 \\ 1 & \text{si } \bar{\theta}^T \bar{f} \geq 0 \end{cases} \quad (5.13)$$

Debe notarse que ahora la dimensión del vector $\bar{\theta}$ es igual a la cantidad de datos del conjunto de entrenamiento, entonces la función de costo de la Ecuación 5.4 puede reescribirse como:

$$J(\bar{\theta}, C) = C \sum_{i=0}^{M-1} \left[y^{(i)} \text{cost}_1(\bar{\theta}^T \bar{f}^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\bar{\theta}^T \bar{f}^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^{M-1} \theta_j^2 \quad (5.14)$$

Existen múltiples maneras de definir la función kernel, donde cada una se ajusta mejor a ciertos tipos de distribuciones de datos. Algunos de los tipos de kernels más comunes son [17]:

- **Kernel Lineal** := $k_{\text{lineal}}(\bar{x}^{(i)}, \bar{x}^{(j)}) = \langle \bar{x}^{(i)}, \bar{x}^{(j)} \rangle$
- **Kernel Gaussiano** := $k_{\text{gaussiano}}(\bar{x}^{(i)}, \bar{x}^{(j)}) = -\gamma \|\bar{x}^{(i)} - \bar{x}^{(j)}\|^2$, con $\gamma > 0$
- **Kernel Polinomial** := $k_{\text{polinomial}}(\bar{x}^{(i)}, \bar{x}^{(j)}) = (\gamma \langle \bar{x}^{(i)}, \bar{x}^{(j)} \rangle + r)^d$, siendo d el grado del polinomio y r un término independiente.
- **Kernel Sigmoide** := $k_{\text{sigmoide}}(\bar{x}^{(i)}, \bar{x}^{(j)}) = \tanh(\gamma \langle \bar{x}^{(i)}, \bar{x}^{(j)} \rangle + r)$

En la actualidad existen múltiples librerías con implementaciones optimizadas de algoritmos de SVM para distintos tipos de kernels escritos en una gran variedad de lenguajes, como lo son la librería **scikit-learn** [17] en Python o **LIBSVM** [18] en C++, lo cual hace que la implementación de estos algoritmos puedan realizarse de forma muy rápida.

5.3.3. Resultados obtenidos

En esta sección se muestran los resultados obtenidos luego de aplicar las técnicas de estimación de cantidad de señales recibidas mediante aprendizaje automático. En este caso solo se realizó el análisis del algoritmo SVM debido a su simplicidad de implementación con respecto a Regresión Logística.

Para poder operar con estas técnicas primero deben elegirse las características que definen a cada valor singular a clasificar. La primer característica a definir es la magnitud del mismo, ya que cuanto más grande sea, mayor va a ser la probabilidad de que sea un valor singular correspondiente al subespacio de señal. Sin embargo esta característica no es suficiente, ya que, según cómo sean las propiedades de las señales recibidas, el piso de ruido puede variar de manera tal de que lo que antes era una magnitud que correspondía a un valor singular de señal en otro caso puede corresponder a un valor singular de ruido. Por ende hay que definir una nueva característica que tenga en cuenta la relación de magnitudes entre los valores singulares de ruido y de señal en cada descomposición de valores singulares. Según lo que se definió en la Ecuación 3.10, en la SVD de una matriz \mathbf{X} definida en la Ecuación 3.4 los valores singulares máximos y mínimos vienen dados por:

$$\lambda_{\max} = M \cdot |s_{\max}|^2 + \sigma_w^2, \quad (5.15)$$

$$\lambda_{\min} = \sigma_w^2. \quad (5.16)$$

Por ende, a partir de estos valores puede definirse una noción de SNR haciendo:

$$\hat{\text{SNR}} = \frac{\lambda_{\max} - \lambda_{\min}}{M \cdot \lambda_{\min}} = \frac{|s_{\max}|^2}{\sigma_w^2} \quad (5.17)$$

Hay que tener en cuenta que esta definición no indica una SNR per se, ya que λ_{\max} solo contiene información de la señal que arriba con mayor potencia e ignora a las otras, pero esta diferencia de magnitudes permite aportar información de utilidad que permita realizar la clasificación de los valores singulares. Además, siendo que se trabaja con una cantidad de muestras finitas, los valores singulares no serán iguales a lo que se mostró en la Ecuación 3.10 y ocurrirá que los valores singulares de ruido contendrán información de las señales y viceversa. Sin embargo, la práctica demuestra que es una buena métrica que puede usarse como característica. A partir de estas dos características ahora puede definirse un vector de entrada para el clasificador de la siguiente manera:

$$\bar{x}^{(i)} = \begin{bmatrix} \lambda^{(i)} \\ \hat{\text{SNR}}^{(i)} \end{bmatrix} \quad (5.18)$$

con $i = 0, 1, \dots, M - 1$, y siendo M el tamaño del conjunto de datos.

Utilizando la misma base de valores singulares que se generó en la Sección 5.2.1 y en función de la definición de la Ecuación 5.18, puede realizarse la gráfica de la Figura 5.10, en donde se representan 2000 valores singulares correspondientes al subespacio de ruido y 2000 valores singulares correspondientes al subespacio de señal elegidos al azar en función de las características definidas. Como puede verse en esta gráfica, existen dos zonas muy separadas que pueden utilizarse para definir una frontera de decisión. Los valores singulares de ruido se ubican sobre el margen izquierdo de la gráfica, la cual corresponde a valores singulares de magnitud pequeña, y los autovalores de señal se extienden por toda la gráfica hacia la derecha, tomando diferentes valores según la potencia de la señal que representan.

Antes de comenzar con el entrenamiento del algoritmo es necesario preparar los datos para que las iteraciones minimicen la función de costo de la manera más rápida. Para lograr esto se utilizará la técnica de *escalamiento de características* [16], que consiste en normalizar los rangos de variación de los valores de las características del conjunto de entrenamiento para que ambos varíen en una misma

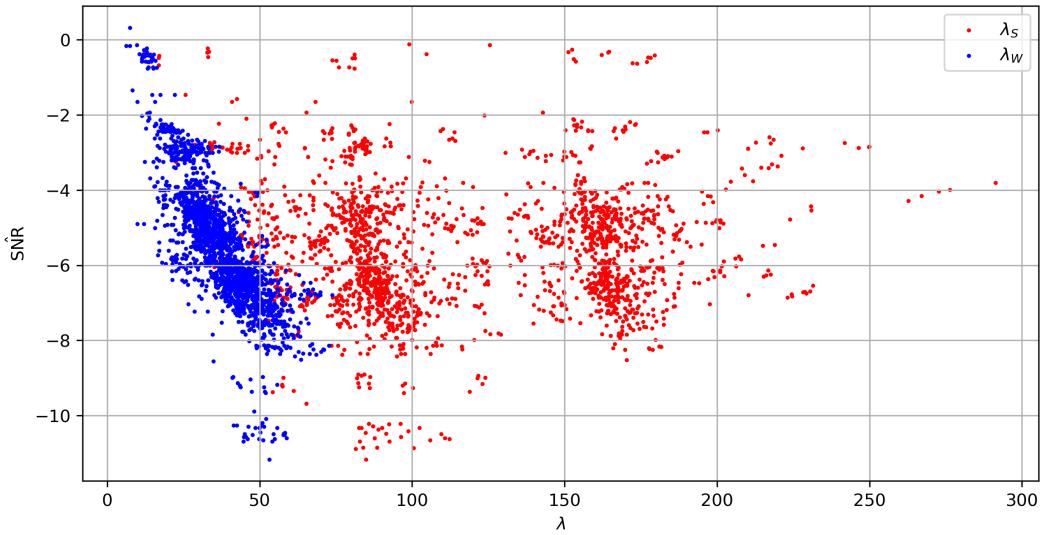


Figura 5.10: Gráfica de valores singulares de distintas realizaciones de \mathbf{X} en función de las características definidas para el problema de clasificación mediante aprendizaje automático.

escala. Para eso se procede definiendo las siguientes magnitudes:

$$\begin{aligned}\mu_\lambda &= \frac{1}{M_{tr}} \sum_{i=0}^{M_{tr}-1} \lambda^{(i)}, & \mu_{\hat{SNR}} &= \frac{1}{M_{tr}} \sum_{i=0}^{M_{tr}-1} \hat{SNR}^{(i)} \\ \sigma_\lambda &= \max(\bar{\lambda}) - \min(\bar{\lambda}), & \sigma_{\hat{SNR}} &= \max(\bar{\hat{SNR}}) - \min(\bar{\hat{SNR}}),\end{aligned}\quad (5.19)$$

donde μ_λ es el valor medio de las magnitudes de todos los valores singulares en el conjunto de entrenamiento, $\mu_{\hat{SNR}}$ es el valor medio de la \hat{SNR} de todos los valores singulares del conjunto de entrenamiento como se definió en la Ecuación 5.17, σ_λ es la diferencia entre la magnitud del valor singular máximo y mínimo del conjunto de entrenamiento, $\sigma_{\hat{SNR}}$ es la diferencia entre el valor de \hat{SNR} máximo y mínimo de todo el conjunto de entrenamiento, y M_{tr} es el tamaño del conjunto de entrenamiento. Luego de definir estas variables se afecta a cada vector $\bar{x}^{(i)}$ del conjunto total de datos de la siguiente manera:

$$\begin{aligned}\bar{x}_{fs}^{(i)} &= \Sigma \cdot (\bar{x}^{(i)} - \bar{\mu}) = \begin{bmatrix} \frac{\lambda^{(i)} - \mu_\lambda}{\sigma_\lambda} \\ \frac{\hat{SNR}^{(i)} - \mu_{\hat{SNR}}}{\sigma_{\hat{SNR}}} \end{bmatrix} \\ \Sigma &= \begin{bmatrix} \frac{1}{\sigma_\lambda} & 0 \\ 0 & \frac{1}{\sigma_{\hat{SNR}}} \end{bmatrix}, \quad \bar{\mu} = \begin{bmatrix} \mu_\lambda \\ \mu_{\hat{SNR}} \end{bmatrix}\end{aligned}\quad (5.20)$$

Luego de acondicionar los datos se definieron los conjuntos de entrenamiento, validación y prueba de la siguiente manera:

- **Conjunto de entrenamiento:** 60 % del conjunto total de datos. Es el utilizado para obtener los elementos del vector de parámetros $\bar{\theta}$ mediante entrenamiento del algoritmo.
- **Conjunto de validación:** 20 % del conjunto total de datos. Se utiliza para ajustar los coeficientes y términos independientes propios de cada kernel.
- **Conjunto de prueba:** 20 % del conjunto total de datos. Se utiliza para medir la precisión del algoritmo.

Utilizando los conjuntos de entrenamiento y validación se entrenó el algoritmo SVM utilizando distintos kernels, obteniendo los resultados que se muestran a continuación.

Kernel Sigmoide

Luego de entrenar al algoritmo utilizando este kernel se llegó a la elección de parámetros $C = 30$, $\gamma = 0,3$ y $r = 0,01$. A partir de estos parámetros y la obtención del vector θ mediante entrenamiento del algoritmo se obtuvo la frontera de decisión que se muestra en la Figura 5.11.

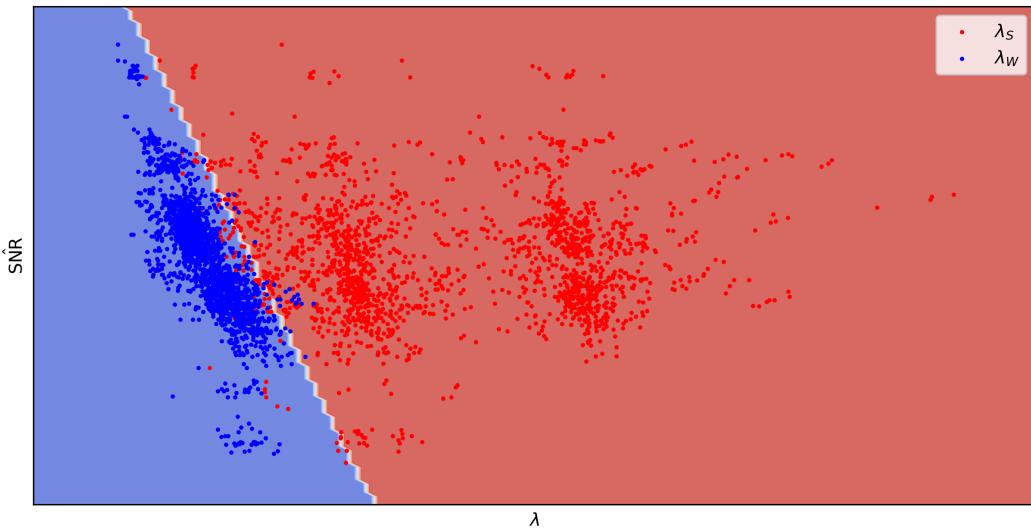


Figura 5.11: Frontera de decisión definida por el algoritmo SVM utilizando un kernel sigmoide

Comparando las características conocidas del conjunto de datos se evaluó la precisión del clasificador utilizándolo en distintos conjuntos de prueba obteniendo los siguientes resultados:

- **Conjunto de prueba:** 96,70 % de precisión.
- **Conjunto de validación:** 96,60 % de precisión.
- **Conjunto de entrenamiento:** 96,71 % de precisión.
- **Todos los datos:** 96,69 % de precisión.

Kernel Polinomial

Realizando numerosas iteraciones se optimizó el algoritmo SVM utilizando un kernel polinomial con los parámetros $C = 0,03$, $d = 4$, $\gamma = 50$, $r = 1$. En la Figura 5.12 se muestra la frontera de decisión obtenida con este clasificador. Como puede observarse, esta frontera no pareciera mostrar un resultado adecuado en la zona inferior izquierda e inferior derecha de la gráfica, esto se debe a la falta de valores singulares en esa zona que ayuden a aportar información al algoritmo.

- **Conjunto de prueba:** 96,85 % de precisión.
- **Conjunto de validación:** 97,15 % de precisión.
- **Conjunto de entrenamiento:** 96,92 % de precisión.
- **Todos los datos:** 96,95 % de precisión.

Kernel Gaussiano

Luego de entrenar al algoritmo se fijaron los parámetros $C = 50$ y $\gamma = 30$. En la Figura 5.13 se indica la frontera de decisión definida por este kernel junto con el conjunto de datos clasificado.

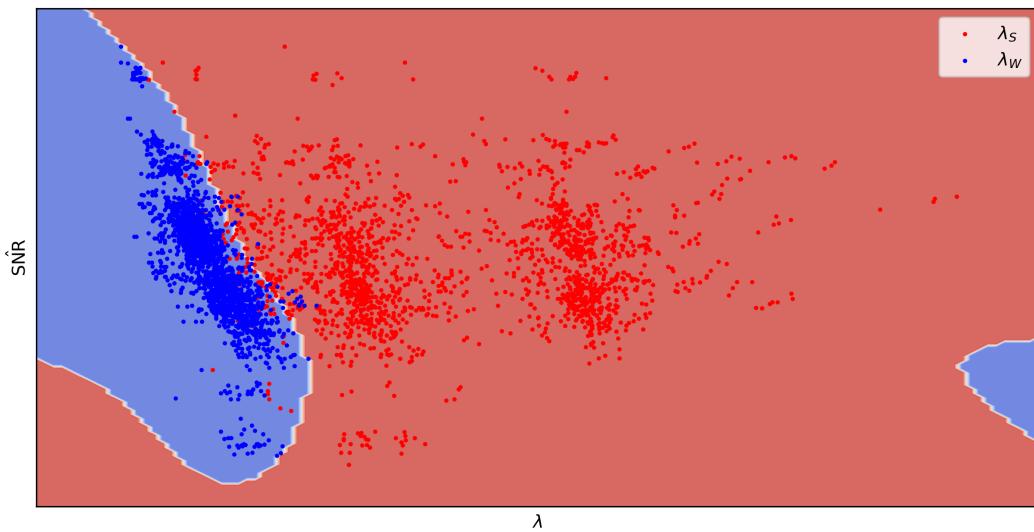


Figura 5.12: Frontera de decisión definida por el algoritmo SVM utilizando un kernel polinomial.

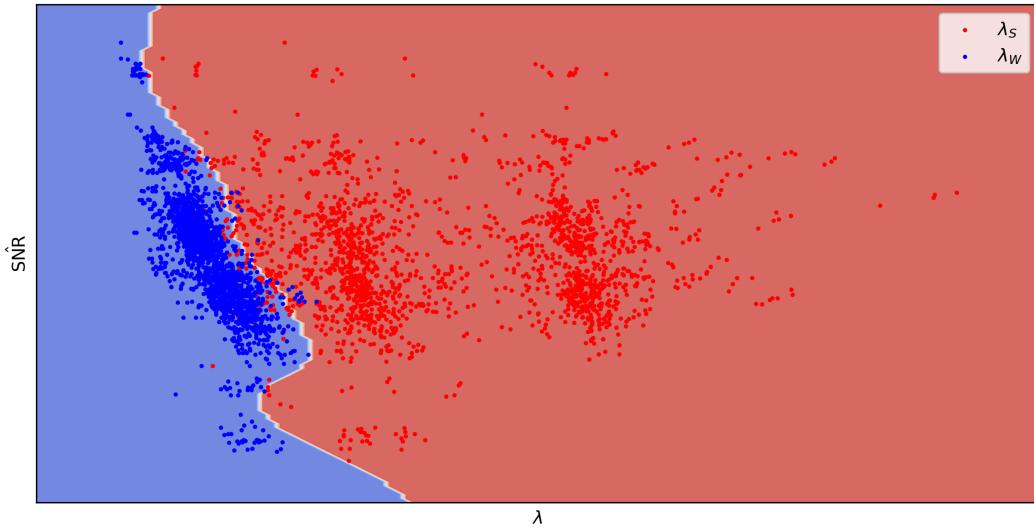


Figura 5.13: Frontera de decisión definida por el algoritmo SVM utilizando un kernel gaussiano.

Contando los errores producidos al clasificar los distintos conjuntos de datos se obtuvieron las siguientes medidas de precisión:

- **Conjunto de prueba:** 96,92 % de precisión.
- **Conjunto de validación:** 97,25 % de precisión.
- **Conjunto de entrenamiento:** 97,12 % de precisión.
- **Todos los datos:** 97,11 % de precisión.

Con las simulaciones realizadas puede concluirse que este es el kernel que alcanzó la mayor precisión. Sin embargo, el entrenamiento del algoritmo con datos simulados no es suficiente para llevar este clasificador a una implementación real. En ese caso lo correcto será volver a entrenar al mismo utilizando mediciones reales de las señales satelitales que se desea recibir.

Capítulo 6

Diseño del sistema

“Los ingenieros hacemos realidad las utopías de los físicos.”

— Julio Benítez

6.1. Introducción

A esta altura ya se cuenta con los conocimientos teóricos sobre todas las tareas que debe cumplir el sistema conformador de haz. En particular este debe ser capaz de:

- Muestrear aleatoriamente con el algoritmo detallado en el Capítulo 4 los vectores de muestras provenientes del sistema de adquisición encargado de muestrear al ARU.
- Estimar la cantidad de señales recibidas utilizando el algoritmo SVM analizado en la Sección 5.3.2.
- Estimar la dirección de arriba de cada una de las señales recibidas utilizando el algoritmo ESPRIT que se detalló en la Sección 3.3.
- Con la información obtenida de la estimación de cantidad de señales y estimación de direcciones de arriba realizar la conformación de haz de cada una de las señales detectadas.

Este sistema correrá en una placa de desarrollo CIAA-ACC, la cual cuenta con un SoC¹ Xilinx Zynq-7030 [19], el cual integra, por un lado, *lógica programable* (PL) bajo la forma de un chip FPGA de la familia Kintex-7, y por otro, un *sistema de procesamiento* (PS) con dos microprocesadores ARM Cortex-A9. Esto brinda la posibilidad de implementar bloques en distintas arquitecturas pudiendo así explotar las virtudes de cada una, como lo es la posibilidad de optimizar cálculos paralelizables en la PL o la posibilidad de utilizar librerías o lenguajes de alto nivel en el PS. Teniendo esto en cuenta se realizará un diagrama de bloques del sistema en su totalidad para luego hacer hincapié en cada uno de los subsistemas, analizando qué ventajas ofrece su implementación en distintas arquitecturas. Este sistema, además, compartirá los recursos de la PL y del PS con un sistema de adquisición encargado de obtener las muestras digitales simultáneamente de los 16 elementos del ARU. Esta información será de importancia para definir la ubicación y las interfaces de los subsistemas desarrollados en este proyecto y el sistema de adquisición.

A lo largo de este capítulo se utilizará el siguiente esquema de colores para identificar los tipos de datos de las interfaces de los distintos sistemas:

¹System on a chip, circuito integrado que concentra una gran variedad de recursos en un solo chip.

- █ Datos complejos.
- █ Vectores o matrices de datos complejos.
- █ Vectores o matrices de datos reales.
- █ Datos enteros.

6.2. Diagrama de bloques del sistema

Conociendo las tareas que debe cumplir el sistema a diseñar se propone el desglose en subsistemas que se muestra en el diagrama de bloques de la Figura 6.1. El sistema conformador de haz recibe desde un sistema de adquisición matrices de muestras complejas \mathbf{X} de tamaño $M \times N$, como se definió en la Ecuación 3.4, donde M es la cantidad de elementos en el ARU y N es la cantidad de veces que se muestreó al arreglo de antenas en un determinado tiempo. Este sistema está conformado por tres subsistemas que se encargan de realizar el muestreo aleatorio de la matriz de entrada, la estimación de DOA y cantidad de señales recibidas y finalmente la conformación de las señales detectadas, las cuales son entregadas como muestras complejas en las D salidas del sistema, siendo D la cantidad de señales detectadas.

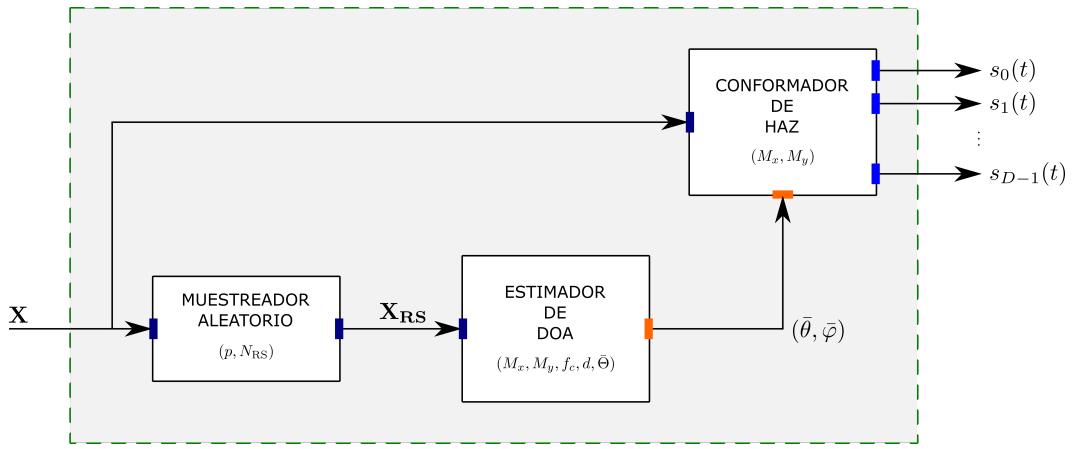


Figura 6.1: Diagrama de bloques del sistema conformador de haz.

Antes de operar el sistema debe ser configurado con los siguientes parámetros:

- M_x : cantidad de elementos del arreglo de antenas en la dirección x .
- M_y : cantidad de elementos del arreglo de antenas en la dirección y .
- f_c : frecuencia nominal de portadora de las señales que se esperan recibir.
- d : distancia de separación entre elementos.
- p : probabilidad de tomar un vector de muestras en el muestreo aleatorio.
- $\bar{\Theta}$: vector de parámetros del clasificador SVM obtenido mediante entrenamiento del algoritmo.

Este sistema puede ser implementado completamente en el PS, sin embargo la implementación de algunos de estos tres bloques en la FPGA puede llegar a traer alguna ganancia de rendimiento mediante la paralelización de cálculos, principalmente a medida que se aumenta la cantidad de señales recibidas, sin embargo al realizar esto se debe tener cuidado con las interfaces que se agregan entre el PS y la PL, ya que pueden llegar a ser un cuello de botella en el proceso, como así también hay

que tener en cuenta la cantidad de recursos que quedan libres en la FPGA luego de que se instale el sistema de adquisición.

6.3. Muestreador aleatorio

El subsistema muestreador aleatorio recibe la matriz de muestras complejas \mathbf{X} de tamaño $M \times N$ desde el sistema de adquisición y entrega una matriz de muestras \mathbf{X}_{RS} de tamaño $M \times N_{RS}$ donde $N_{RS} = p \cdot N$. En este caso p puede considerarse un factor de decimación, ya que es el número que define cómo se reduce la cantidad de muestras a la salida a partir de una cierta cantidad de muestras en la entrada. La matriz de salida se forma mediante elecciones aleatorias de las columnas de \mathbf{X} , utilizando el método que se explica en el Capítulo 4.

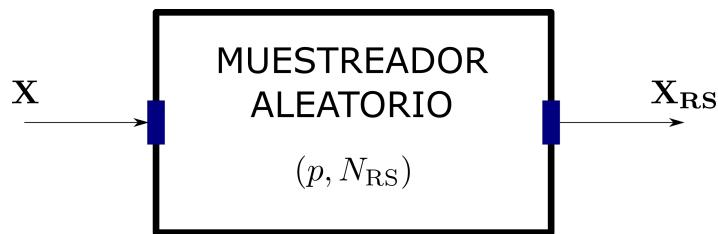


Figura 6.2: Representación en bloque del muestreador aleatorio con sus interfaces.

Otra manera de implementarlo es haciendo que este bloque reciba solo vectores de tamaño $M \times 1$ y que estos vectores se muestren en la salida con probabilidad p . Esta implementación evita tener que operar con matrices de tamaño $M \times N$.

6.3.1. Implementación en FPGA

Debido a que este módulo trabaja directamente con las muestras que entrega el sistema de adquisición, el contar con las muestras de cada elemento almacenadas en registros en la PL [20] permite una implementación rápida de este bloque que puede significar un ahorro de recursos en el PS, ya que, ubicando el muestreador aleatorio fuera del PS, se podría pensar en una implementación en el que este esté trabajando siempre con matrices de tamaño $M \times N_{RS}$ y no $M \times N$, reduciendo el consumo de memoria y los tiempos de carga de datos. A partir de esto, en la Figura 6.3 se muestra el diseño de bloques de una propuesta de implementación de este módulo en FPGA.

Este diseño cuenta con 16 FIFOs², una por cada elemento del ARU, capaces de almacenar una cantidad N_{RS} de datos de 32 bits, suponiendo (sin pérdida de generalidad) que estos datos almacenan la parte real y la parte imaginaria de una muestra en porciones de 16 bits para cada una. Cada FIFO tiene una señal FULL que sirve para indicar al subsistema estimador de DOA el momento en el que las muestras se encuentran listas para ser procesadas, instante en el cual este subsistema puede levantar la señal de lectura para comenzar con la carga de los datos. La escritura de las FIFOs se realiza simultáneamente a partir de una señal externa que indica cuándo existen datos válidos en las entradas, y una señal interna generada a partir de la comparación de dos números de 16 bits: el número p fijado por el usuario y otro número generado aleatoriamente cada tiempo de reloj por un bloque generador de números aleatorios. De esta manera, las FIFOs se cargan en tiempos aleatorios pero de manera simultánea entre ellas, entregando en sus salidas en el momento de lectura los vectores de la matriz \mathbf{X}_{RS} que se muestra en la Figura 6.2. En este caso no se utiliza la señal de EMPTY de las FIFOs ni se evita la escritura en las mismas cuando la señal FULL se encuentra en alto debido a que

²First in, first out: bloque que permite el almacenamiento y la lectura de datos de manera tal que el primer dato en ser almacenado sea el primero en ser leído.

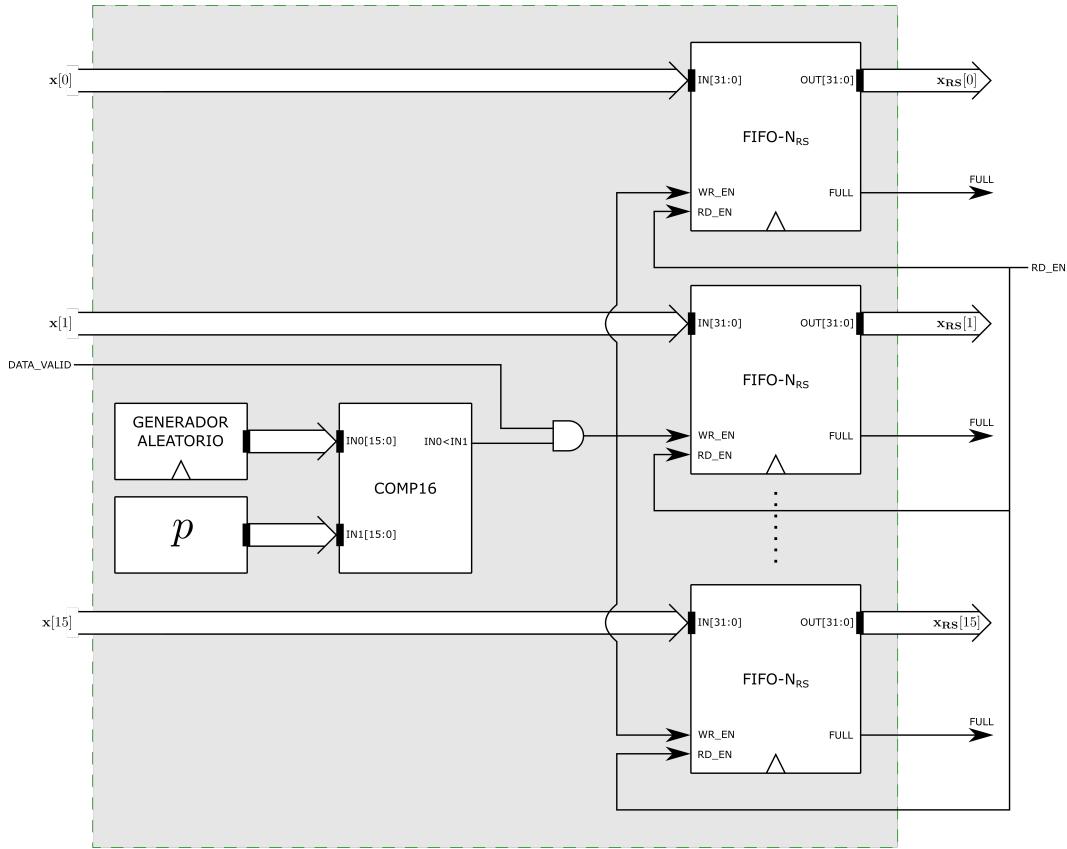


Figura 6.3: Diseño de bloques de la propuesta de implementación del muestreador aleatorio en la FPGA.

se busca que las FIFOs se encuentren actualizadas en todo momento con las muestras más recientes, ya que en esas muestras se encontrará la información de la DOA, la cual se desea mantener siempre actualizada. El único momento en el cual se debe impedir la escritura de datos es el momento en el que el subsistema estimador de DOA se encuentra haciendo la lectura de los datos que se encuentran dentro de las FIFOs.

6.4. Estimador de dirección de arribo

El subsistema estimador de dirección de arribo, cuya representación como bloque se muestra en la Figura 6.4a, es el subsistema más complejo de los tres debido a las operaciones que realiza. Como se muestra en la Figura 6.4b, está conformado por 3 bloques:

- **SVD:** bloque encargado de realizar la descomposición en valores singulares de la matriz de entrada \mathbf{X}_{RS} , obteniendo la matriz de valores singulares Σ y los vectores singulares izquierdos \mathbf{E} .
- **SVM:** bloque encargado de realizar la estimación de la cantidad de señales recibidas D realizando una clasificación de los valores singulares contenidos en la matriz Σ utilizando el algoritmo de Máquina de Vectores de Soporte que se analizó en la Sección 5.3.2. Utilizando un kernel gaussiano, el vector $\bar{\Theta}$ debe incluir, además, los parámetros adicionales C y γ .
- **ESPRIT:** bloque encargado de realizar la estimación de DOA de las D señales recibidas a partir de los vectores singulares izquierdos de la matriz \mathbf{E} . En su salida entrega los vectores $\bar{\theta}$ y $\bar{\varphi}$ de tamaño $D \times 1$.

Debido a la dinámica en el tamaño de los datos con los que opera este subsistema, ya que un cambio en la cantidad de señales recibidas implica un cambio en la cantidad de columnas con las que debe operar el bloque ESPRIT, y a la dificultad de implementar los algoritmos algebraicos de los bloques SVD y SVM en FPGA, comparada con la escasa dificultad de implementarlos en software, este sistema debe ubicarse en el PS.

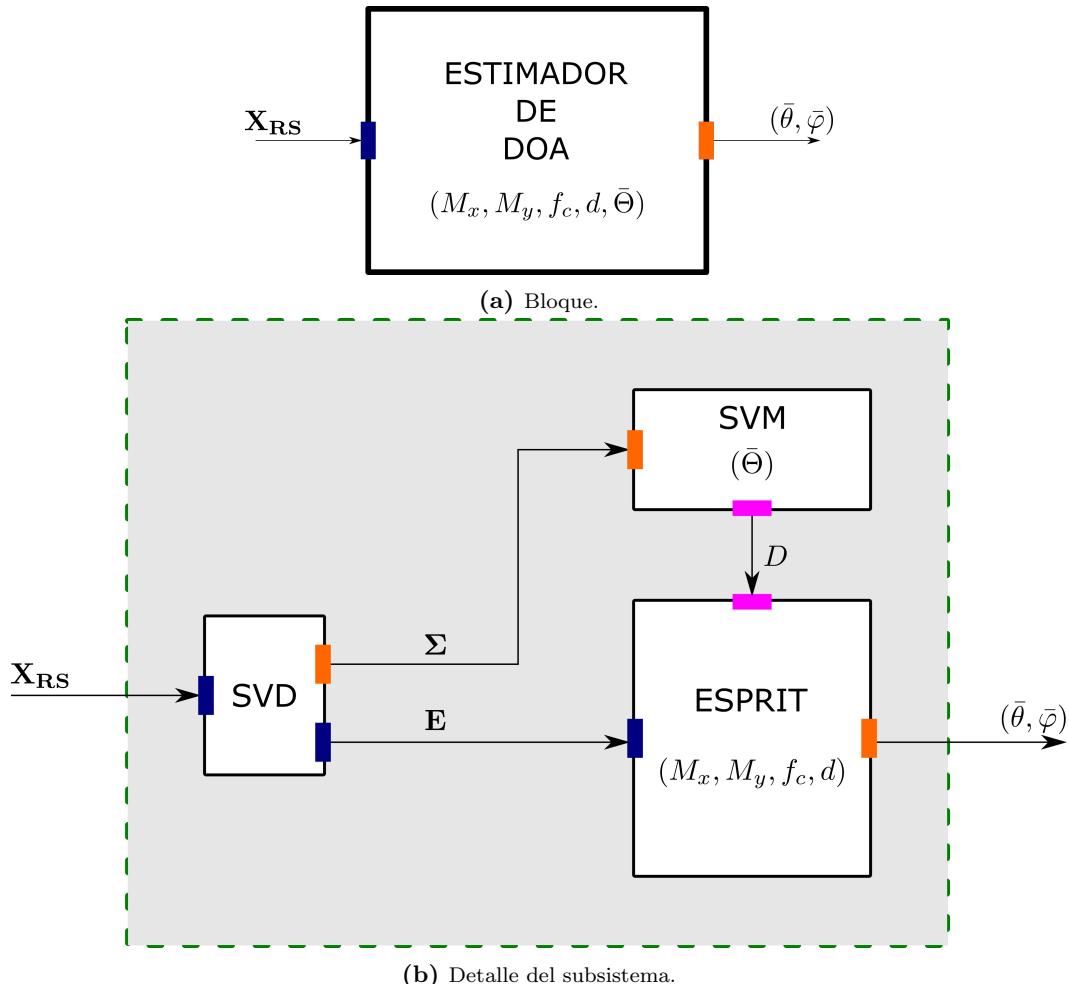


Figura 6.4: Representación como bloque del subsistema estimador de DOA con sus correspondientes interfaces.

6.5. Conformador de haz

El último subsistema a implementar es el conformador de haz, cuya representación en bloque se muestra en la Figura 6.5. Con la información obtenida de los otros dos subsistemas, el conformador de haz se encarga de procesar las muestras de cada elemento del ARU restándole la diferencia de fase con respecto al elemento de referencia y promediando todas las muestras de manera tal de obtener las D señales a la salidas. Además, si se conoce el patrón de radiación de cada elemento del ARU puede afectarse a cada muestra por la ganancia del arreglo en la DOA para normalizar las señales recibidas. Por cada señal detectada la operación que realiza este subsistema es la siguiente:

$$\begin{aligned}
\hat{s}_d[n] &= \frac{1}{M \left(g(\hat{\theta}_d, \hat{\varphi}_d) \right)^2} \cdot \bar{a}_{\text{ARU}}^H(\hat{\theta}_d, \hat{\varphi}_d) \cdot \bar{x} \\
&= \frac{1}{M \cdot g(\hat{\theta}_d, \hat{\varphi}_d)} \cdot \begin{bmatrix} 1 & \dots & e^{jkd[(M_x-1)\cos\hat{\theta}_d\cos\hat{\varphi}_d + (M_y-1)\cos\hat{\theta}_d\sin\hat{\varphi}_d]} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{M-1} \end{bmatrix} \\
&= \frac{1}{M \cdot g(\hat{\theta}_d, \hat{\varphi}_d)} \sum_{m=0}^{M-1} a_m^*(\hat{\theta}_d, \hat{\varphi}_d) \cdot x_m,
\end{aligned} \tag{6.1}$$

con $d = 0, 1, \dots, D - 1$ y donde \bar{a}_{ARU} es el vector de apuntamiento definido en la Ecuación 2.13

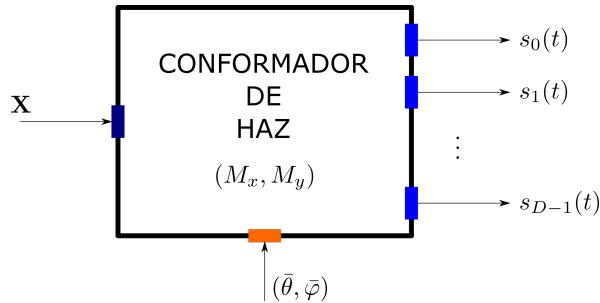


Figura 6.5: Representación como bloque del subsistema conformador de haz con sus correspondientes interfaces.

6.5.1. Implementación en FPGA

Este algoritmo puede ser implementado tanto en el PS como en la FPGA, debido a que no requiere realizar cálculos de mayor complejidad. Una ventaja de implementarlo en la PL junto al muestreador aleatorio es, como ya se mencionó, evitar que el PS tenga que manipular matrices de tamaño $M \times N$ y en su lugar trabajar con matrices $M \times N_{\text{RS}}$ donde N_{RS} puede ser incluso más de dos órdenes de magnitud menor que N . En la Figura 6.6 se muestra un diagrama de bloques de una propuesta de implementación de este subsistema en FPGA.

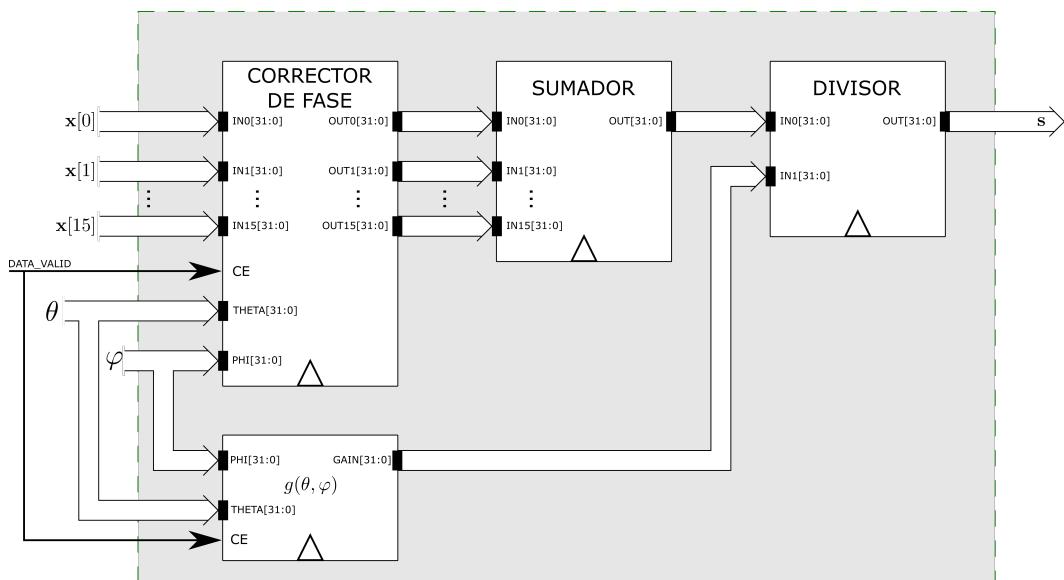


Figura 6.6: Propuesta de implementación del subsistema conformador de haz en FPGA.

En este diseño el sistema recibe como entradas las 16 muestras de 32 bits provenientes del sistema de adquisición, los ángulos de elevación y azimut provenientes del subsistema estimador de DOA (el cual se encuentra en el PS), y una señal de validación proveniente del PS que indica el instante en el que existen ángulos de arribo válidos en las correspondientes entradas. Dentro del subsistema existen cuatro bloques: un corrector de fase, el cual se encarga de restar la fase de cada muestra dependiendo de con qué elemento fue tomada, un bloque sumador que suma las 16 salidas del bloque corrector de fase para realizar el promediado de la señal recibida, un bloque que a partir de una DOA entrega la ganancia del ARU en esa dirección, y un divisor que utiliza esta ganancia para normalizar la señal recibida. Siendo que en este caso la cantidad de elementos del ARU es 16, el promediado puede hacerse quitando los 4 bits menos significativos de esta salida. Hay que notar que este diseño solo sirve para realizar la conformación de haz de una única señal, en caso de aumentar las salidas del sistema debe repetirse este bloque y alimentarlo con las direcciones de arribo de las señales restantes. Esta es una desventaja de la implementación en la PL, ya que el diseño no puede ser dinámico según la información recibida y debe ser dimensionado para el peor caso.

Capítulo 7

Integración en GNURadio

“The radio is playing all the usual, and what’s a wonder wall anyway?”
— Fran Healy

7.1. Introducción

Hasta principios de los 2000s, implementar un sistema de RF era una tarea muy costosa debido a la especificidad de los equipos de hardware requeridos y a la necesidad de contar con costosas licencias para el software utilizado en la operación de los mismos. Esto cambió con la aparición de un nuevo paradigma para la construcción de sistemas de RF llamado *Radio Definida por Software* (*SDR*, por sus siglas en inglés). SDR es un sistema de RF donde la mayor cantidad de elementos están implementados en software. Este paradigma tuvo un gran crecimiento durante la última década con la aparición de interfaces de RF para computadoras con costos inferiores a los cientos de dólares, como es el caso del dispositivo RTL-SDR, con un costo de alrededor de U\$D 25 [21], como así también debido al crecimiento de la capacidad de procesamiento de los sistemas informáticos y al crecimiento de la comunidad de software libre dedicada al desarrollo de herramientas de SDR, de la cual salieron aplicaciones de gran utilidad que se utilizan tanto en ambientes educativos y aficionados, como así también en la industria [22]. Una de las principales ventajas que tiene SDR sobre las tradicionales implementaciones en hardware dedicado es que permite la actualización y el soporte de múltiples estándares de comunicaciones, siempre y cuando la arquitectura sobre la cual esté instalado este software sea lo suficientemente potente y programable. Esto no es posible en los sistemas tradicionales ya que una actualización o un cambio en una característica requiere el diseño y la implementación de muchos componentes de hardware, tarea costosa y que demanda mucho tiempo [23]. Otra gran ventaja de los sistemas SDR es que reducen enormemente el tiempo de prototipado, ya que no requiere de la fabricación de hardware específico para probar un diseño.

En la Figura 7.1 se muestra un esquema de un sistema SDR, conformado por una etapa de RF encargada de llevar la señal recibida a banda base o de elevar la señal transmitida a una frecuencia de portadora, una etapa de conversión analógica-digital encargada de obtener las muestras digitales de la señal recibida y de hacer el efecto recíproco en la transmisión, y finalmente una computadora que conforma la arquitectura donde correrá el sistema SDR. En un SDR ideal la etapa de RF se implementa dentro del software, sin embargo esto no es simple de realizar debido a que para lograrlo se debe muestrear la señal de entrada a una frecuencia por encima del doble de la portadora, la cual puede estar por encima de los cientos de megahertz o incluso el gigahertz, solo para obtener información de una señal cuyo ancho de banda es, en general, al menos un orden menor que esta

frecuencia. Una alternativa para **sobrepasar** este problema podría ser emplear muestreo pasabanda, pero en general esta solución no suele ser económicamente viable.

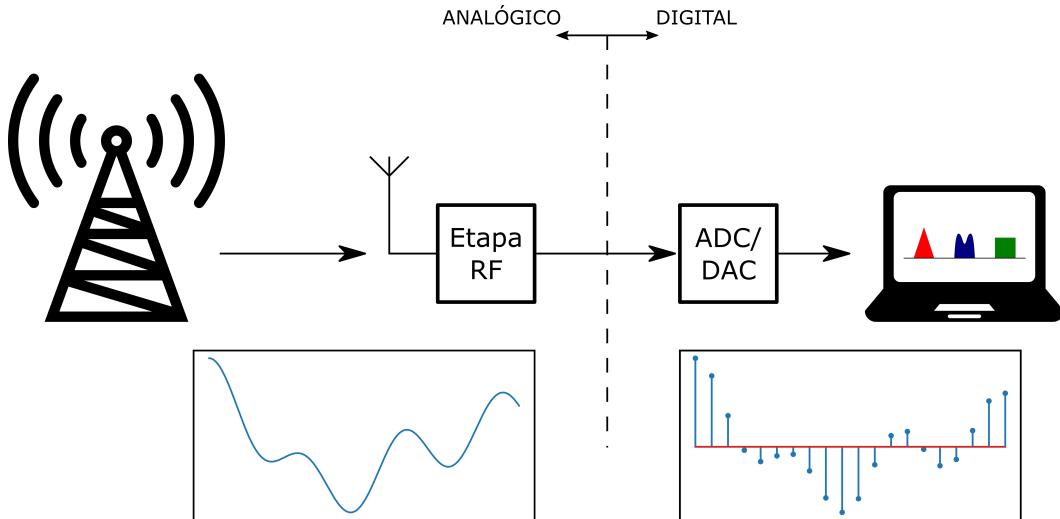


Figura 7.1: Esquema de un sistema de radio definida por software.

Dentro del marco de aplicaciones de SDR, *GNURadio* es el entorno de desarrollo de procesamiento más popular. Esta herramienta provee de bloques de procesamiento de señal que permiten realizar la implementación de sistemas SDR mediante un entorno gráfico llamado *GNURadio Companion*, el cual es semejante al que proveen herramientas privativas como *Simulink®* de la empresa *MathWorks®* o *LabVIEW™* de *National Instruments™*. Aparte de permitir la utilización de las librerías de bloques incluidas en la instalación, *GNURadio* provee de herramientas e instrucciones para poder generar bloques personalizados mediante programación utilizando los lenguajes *C++* o *Python*, como así también permite compartir las librerías creadas entre usuarios. *GNURadio* está disponible en los sistemas operativos Microsoft Windows y Linux, sin embargo la versión de Linux es la única que cuenta con soporte oficial. De todas maneras la versión oficial puede ser ejecutada en Windows utilizando el *Windows Subsystem for Linux*, el cual permite ejecutar una instancia de Linux dentro de Windows de una manera más eficiente que ejecutando una máquina virtual.

GNURadio toma un rol vital en este proyecto, proveyendo no solo la capacidad de realizar la implementación de todo el sistema conformador de haz en software mediante la programación de bloques, sino que, además, entrega un entorno de simulación y prueba de diseños que permite verificar el funcionamiento del sistema en tiempo real.

7.2. El modelo equivalente banda base

Debido a que, como se dijo, al trabajar en SDR las señales deben ser llevadas a banda base para disminuir la frecuencia de muestreo requerida en el software, las muestras con las que se trabaja son de tipo complejo, algo que en principio puede resultar poco intuitivo para el usuario. Para demostrar la relación entre estas muestras complejas y las señales originales se analiza, primero, el espectro de señales genéricas de RF montadas sobre una portadora f_c . Si se realiza la transmisión de tres señales reales las cuales se montan sobre una portadora f_c , con un ancho de banda total $BW < f_c/2$, la magnitud de su espectro es simétrica, como se muestra en la Figura 7.2a, y su espectro de fase es antisimétrico [24].

Si a estas tres señales se las lleva de f_c a banda base se obtiene el espectro de la Figura 7.2b, cuya magnitud no es simétrica, y que, por ende, representa a una señal compleja. El espectro de esta señal

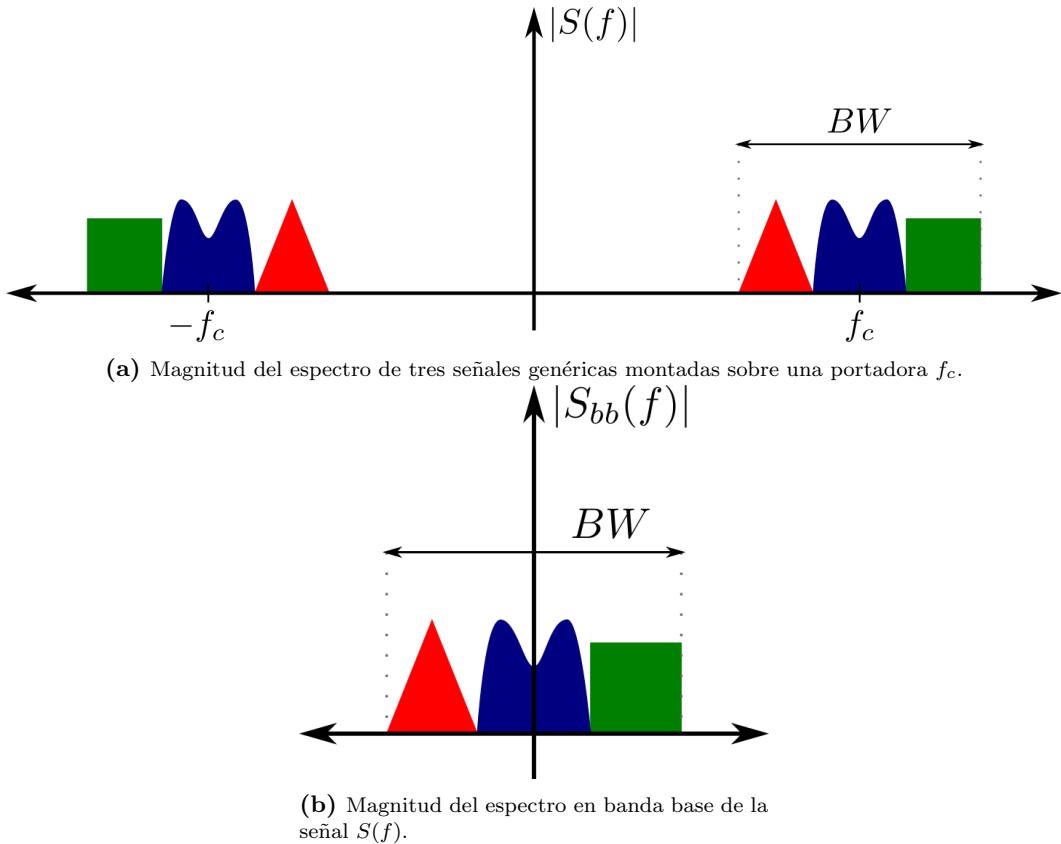


Figura 7.2

en banda base puede escribirse como un corrimiento del espectro de la señal original $S(f)$ haciendo:

$$S_{bb}(f) = \begin{cases} \sqrt{2}S(f + f_c) & f + f_c > 0 \\ 0 & \text{c.o.c.} \end{cases}, \quad (7.1)$$

entonces la señal transmitida $S(f)$ puede escribirse a partir de su representación en banda base como [22]:

$$S(f) = \frac{1}{\sqrt{2}}S_{bb}(f - f_c) + \frac{1}{\sqrt{2}}S_{bb}^*(f + f_c) \quad (7.2)$$

La representación de $S(f)$ en el dominio temporal puede obtenerse aplicando la antitransformada de Fourier a la Ecuación 7.2, obteniendo la siguiente expresión:

$$\begin{aligned} s(t) &= \frac{1}{\sqrt{2}}s_{bb}(t)e^{j2\pi f_c t} + \frac{1}{\sqrt{2}}s_{bb}^*(t)e^{-j2\pi f_c t} \\ &= \frac{1}{\sqrt{2}} \cdot 2 \cdot \operatorname{Re}\{s_{bb}(t)e^{j2\pi f_c t}\} \\ &= \sqrt{2}\operatorname{Re}\{s_{bb}(t)\} \cos(2\pi f_c t) - \sqrt{2}\operatorname{Im}\{s_{bb}(t)\} \sin(2\pi f_c t) \end{aligned} \quad (7.3)$$

siendo $s_{bb}(t)$ la representación en el dominio temporal de $S_{bb}(f)$. La expresión de la Ecuación 7.3 muestra la forma de onda en el receptor antes de realizar la conversión a banda base. Si se desea recuperar la señal $s_{bb}(t)$, es decir, la porción del espectro de la señal $s(t)$ que contiene la información

de interés, puede multiplicarse la Ecuación 7.3 por $\sqrt{2} \cos(2\pi f_c t)$ y por $\sqrt{2} \sin(2\pi f_c t)$ obteniendo:

$$s(t) \cdot \sqrt{2} \cos(2\pi f_c t) = \text{Re}\{s_{bb}(t)\} \underbrace{2 \cos^2(2\pi f_c t)}_{1+\cos(2\pi \cdot 2f_c t)} - \text{Im}\{s_{bb}(t)\} \underbrace{2 \sin(2\pi f_c t) \cos(2\pi f_c t)}_{\sin(2\pi \cdot 2f_c t)} \quad (7.4)$$

$$s(t) \cdot \sqrt{2} \sin(2\pi f_c t) = \text{Re}\{s_{bb}(t)\} \underbrace{2 \sin(2\pi f_c t) \cos(2\pi f_c t)}_{\sin(2\pi \cdot 2f_c t)} - \text{Im}\{s_{bb}(t)\} \underbrace{2 \sin^2(2\pi f_c t)}_{1-\cos(2\pi \cdot 2f_c t)} \quad (7.5)$$

Estas expresiones contienen una componente en banda base de valor $\text{Re}\{s_{bb}(t)\}$ y $\text{Im}\{s_{bb}(t)\}$ sumados a términos centrados espectralmente en $2f_c$, los cuales pueden ser filtrados con un filtro pasa-bajos. A partir de lo obtenido en estas ecuaciones se puede realizar el receptor que se muestra en el diagrama de la Figura 7.3 para así obtener las muestras complejas con las cuales se trabaja en SDR.

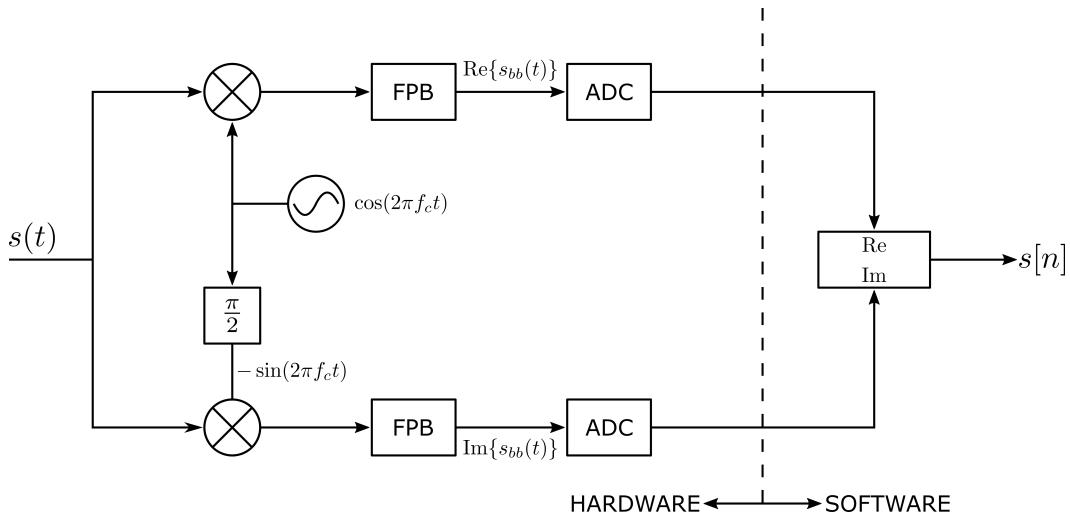


Figura 7.3: Diagrama de bloques de un receptor de un sistema SDR.

7.3. Implementación de módulos

A partir de lo diseñado en el Capítulo 6 se decidió realizar la implementación de todos los módulos utilizando GNURadio para realizar simulaciones en tiempo real del sistema completo y, también, para dejar implementados los subsistemas que posteriormente se instalarán en el PS de la placa de desarrollo. Como ya se dijo, GNURadio permite la creación de bloques utilizando los lenguajes de programación Python y C++. La implementación en Python ofrece las ventajas de ser un lenguaje de alto nivel con gran variedad de librerías de uso libre que permiten la implementación de bloques de una manera muy rápida y sencilla comparada con las implementaciones en C++. Sin embargo, al ser un lenguaje intérprete, corre con una gran desventaja en el apartado de rendimiento, ya que las instrucciones de los programas no son ejecutadas directamente por la máquina sino que existe un agente externo, el *intérprete*, que se encarga de leer el código y ejecutarlo. En cambio C++, al ser un lenguaje compilado, los programas son convertidos directamente en instrucciones que el procesador puede interpretar, evitando contar con un intermediario y aumentando la eficiencia en la ejecución [25]. Es por esto que en GNURadio solo se recomienda implementar bloques en Python en aquellos casos en donde las funciones que realizan no son críticas en el desempeño del sistema, o en aquellos casos en los que se requiere hacer prototipado de bloques para pruebas de funcionamiento rápidas [22]. En esta sección se mencionan algunas características de GNURadio que se utilizaron para la implementación de los bloques necesarios.

7.3.1. Tipos de datos en GNURadio

Las interfaces de los bloques en GNURadio deben tener asignados un tipo de datos definido, los cuales cada uno tiene asignado un color específico. Los tipos de datos permitidos en GNURadio juntos con su código de color se muestran en la Figura 7.4.

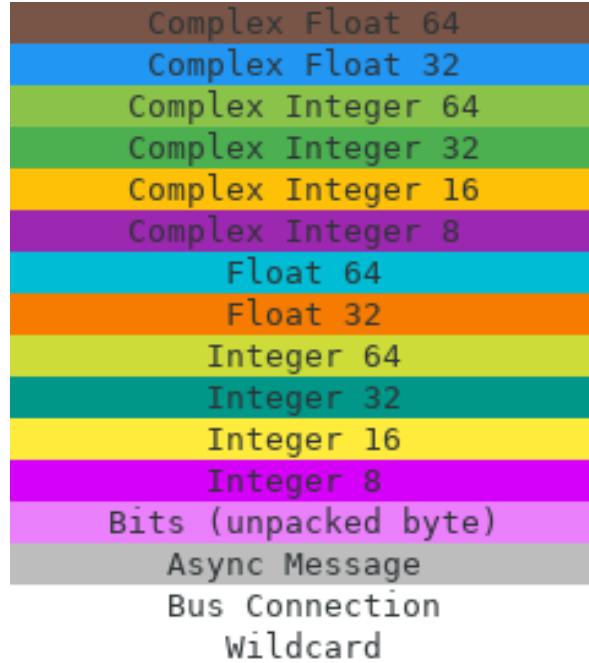


Figura 7.4: Tipos de datos disponibles en GNURadio con su correspondiente identificación de color.

En GNURadio Companion solo pueden conectarse interfaces de bloques que sean del mismo tipo de datos. Para facilitar esta tarea y evitar errores, las interfaces siguen el código de color definido en la Figura 7.4. En la Figura 7.5 se muestra un ejemplo de interconexión correcta e incorrecta entre bloques.

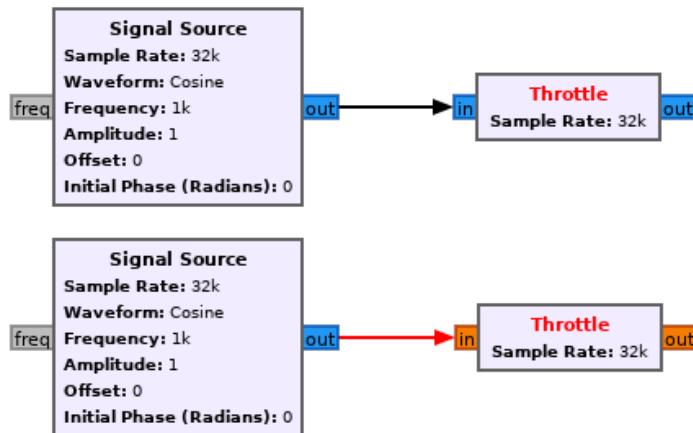


Figura 7.5: Conexiones entre bloques. GNURadio Companion indica con color rojo aquellas conexiones no permitidas.

GNURadio permite, además, el uso de tipo de datos *polimórficos*, los cuales son tipos de datos no explícitos que permiten sobreponer la exigencia del tipado estricto de C++ permitiendo declarar un tipo de dato genérico que se define según el contexto de la aplicación. En este proyecto, este tipo de dato

es de gran utilidad a la hora de realizar el envío de los ángulos estimados por el subsistema estimador de DOA al subsistema conformador de haz, ya que, a pesar de que estos datos serán siempre reales, el tamaño de los vectores enviados cambiarán según cuántas sean las señales detectadas.

7.3.2. Callbacks

En las simulaciones que se muestran en la Sección 7.4 es oportuno contar con la posibilidad de cambiar algunos parámetros en tiempo de ejecución, como por ejemplo variar la DOA de una señal simulada para emular la recepción de un satélite LEO, o variar el piso de ruido de manera tal de poder evaluar en el momento cómo una degradación en la señal afecta la estimación de la DOA. El método que permite realizar esto en GNURadio se llama *callback*. Los callbacks son funciones que se declaran en cada bloque para cada variable que se desea poder variar en tiempo de ejecución, y son llamadas por GNURadio Companion en el momento en el que algunas de estas variables se modifica, de manera tal de poder actualizarla en el momento y ver cómo se propaga ese cambio en la simulación.

7.3.3. La librería VOLK

Dentro del proyecto GNURadio existe un subproyecto llamado *Vector-Optimized Library of Kernels* (*VOLK*), el cual consiste en una librería que contiene kernels de código SIMD¹ para la optimización de operaciones matemáticas orientadas a vectores [26]. Estas librerías se encargan de analizar y elegir cuáles son las instrucciones del procesador que optimizan la ejecución de las distintas operaciones matemáticas según en qué arquitectura de hardware se está ejecutando el programa. Utilizando estas librerías se puede alcanzar mejoras de rendimiento de hasta un 40 % [22].

7.3.4. El emulador de ARU.

Debido a que en el momento de realización de este proyecto no se cuenta con el arreglo de antenas ni con el sistema de adquisición implementados, la manera de realizar las pruebas de los sistemas desarrollados es emulando el comportamiento de un ARU. Para esto se implementa el bloque que se muestra en la Figura 7.6.

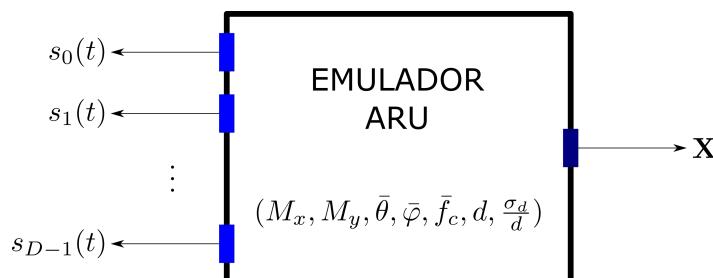


Figura 7.6: Representación como bloque del emulador de ARU para realizar las simulaciones del sistema conformador de haz.

Este bloque recibe D señales en sus entradas y entrega la matriz de salida \mathbf{X} de tamaño $M \times N$, siendo M la cantidad de elementos del arreglo y N la cantidad de muestras temporales. Para hacer esto se basa en el modelo de datos de la Ecuación 3.4 y en la definición del vector de apuntamiento de un ARU definido en la Ecuación 2.13. Este bloque debe ser configurado con la cantidad de elementos del arreglo en ambas direcciones, identificados por M_x y M_y , las direcciones de arriba de las D señales emuladas, identificadas con los símbolos $\bar{\theta}$ y $\bar{\varphi}$, las frecuencias de portadoras de las señales recibidas \bar{f}_c , la distancia de separación entre elementos del arreglo d y el error en esta distancia $\frac{\sigma_d}{d}$.

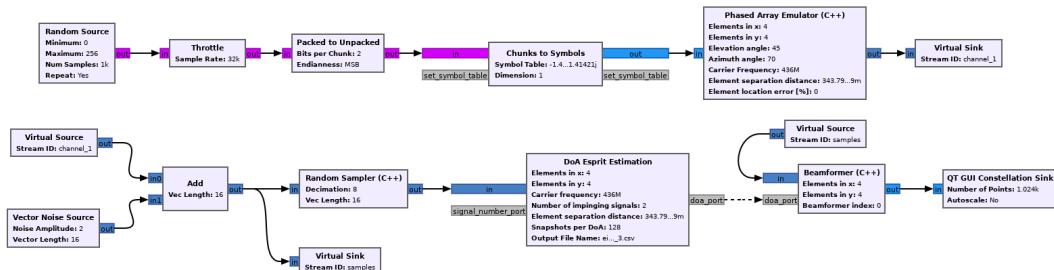
¹Single Instruction, Multiple Data: técnica de software que permite alcanzar paralelismo a nivel de datos.

7.4. Simulaciones

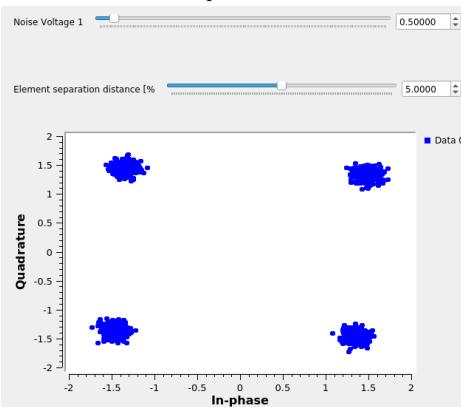
A partir de lo analizado se implementaron los cuatro bloques necesarios para realizar la simulación del sistema completo, estos son:

- **Phased Array Emulator:** emulador de un ARU programado en C++ utilizando VOLK que recibe muestras de señales complejas y entrega el vector de muestras complejas \bar{x} emulando las salidas de cada elemento de un ARU.
- **Random Sampler:** muestreador aleatorio como se mostró en la Sección 6.3, programado en C++, el cual entrega a la salida los vectores de entrada muestreados aleatoriamente.
- **DOA Esprit Estimation:** bloque programado en Python que ejecuta el algoritmo de estimación de DOA Esprit, recibiendo vectores de muestras complejas provenientes del muestreador aleatorio y entregando un mensaje polimórfico que contiene vectores reales correspondientes cuyos elementos corresponden a los ángulos de elevación y azimut de las señales detectadas.
- **Beamformer:** bloque programado en C++ usando VOLK que recibe vectores de muestras complejas y los ángulos de arriba detectados por el bloque DOA Esprit Estimation en forma de mensaje polimórfico y entrega a su salida las muestras de las señales conformadas.

En la Figura 7.7a se muestra una captura del diagrama de bloques de la simulación de una transmisión QPSK utilizando el sistema completo en GNURadio Companion. En la Figura 7.7b se muestra la constelación de la señal entregada por el conformador de haz, la cual corresponde a la señal QPSK transmitida.



(a) Diagrama de bloques del sistema implementado en GNURadio. La zona superior del diagrama corresponde a la transmisión y la parte inferior a la recepción.



(b) Constelación de la salida del sistema conformador de haz al recibir una señal QPSK con una cierta DOA.

Figura 7.7: Simulación de transmisión QPSK.

Luego de comprobar el correcto funcionamiento del sistema se procede a realizar las pruebas

de rendimiento, haciendo hincapié en mediciones de tasa de error de bit (BER) frente a distintos escenarios, y en el procesamiento requerido por cada uno de los subsistemas.

7.4.1. El módulo gr-satellites

El módulo **gr-satellites** desarrollado por Daniel Estévez [27] es uno de los módulos más populares en GNURadio. Este módulo contiene los bloques necesarios para realizar la recepción y decodificación de señales de una gran parte de los satélites de aficionados existentes. Además cuenta con una gran variedad de ejemplos de simulaciones de transmisiones satelitales que pueden ser utilizadas en este proyecto para brindar un grado más de realismo a las pruebas. En particular en este proyecto se utiliza el ejemplo de medición de tasa de error simulating la recepción de información proveniente del satélite LilacSat-1 [28]. Esta simulación realiza esta transmisión iterando para distintos valores de $\frac{E_b}{N_0}$, midiendo la cantidad de errores a la salida y obteniendo una curva de tasa de error de bit. Modificando este ejemplo se agregó el sistema conformador de haz obteniendo el diagrama de bloques que se muestra en la Figura 7.8.

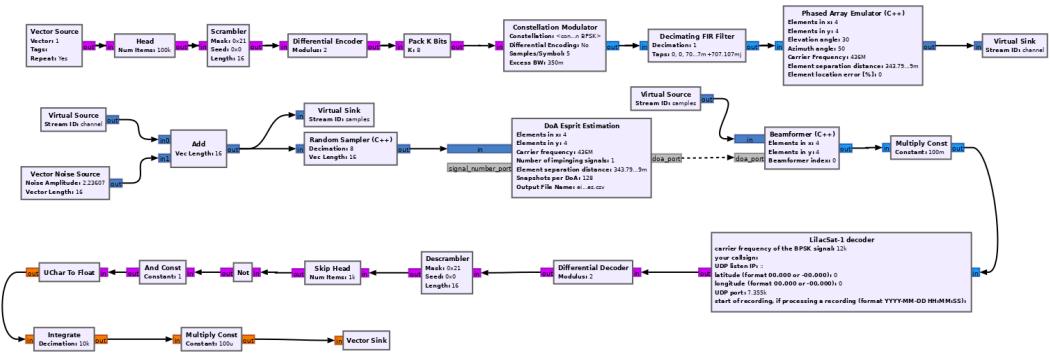


Figura 7.8: Diagrama de bloques utilizado para la medición de tasa de error mediante la simulación de un enlace de comunicación con el satélite LilacSat-1.

7.4.2. Resultados obtenidos

Utilizando el diagrama de la Figura 7.8 se evaluó el desempeño del sistema obteniendo las curvas de tasa de error de bit en función de $\frac{E_b}{N_0}$ para distintos valores de σ_d , obteniendo las curvas que se muestran en la gráfica de la Figura 7.9.

Como puede verse, para un error en la ubicación de elementos del 30 % el desempeño cae 1 dB para este tipo de transmisión.

7.4.3. Requerimientos de procesamiento

Finalmente, se midió el uso del procesador de cada bloque del sistema conformador de haz durante el tiempo de ejecución utilizando la función `top -H` incluida en Linux, la cual permite ver en tiempo real el consumo de cada proceso que está corriendo en el sistema, obteniéndose los resultados de la Tabla 7.1².

Como puede observarse, el bloque estimador de DOA con ESPRIT es el bloque con peor rendimiento en el sistema, principalmente debido a que es el único implementado en Python, por ende, si se desea incrementar el rendimiento para la implementación en la placa de desarrollo, deberá rehacerse en lenguaje C++, utilizando la librería Eigen [29] para la implementación de la SVD.

²El hecho de que la suma de los porcentajes sea superior al 100 % se debe a que la simulación está corriendo en una computadora con un procesador de 8 núcleos, y GNURadio asigna un hilo de ejecución separado para cada bloque para poder ubicar a cada uno en el procesador menos ocupado.

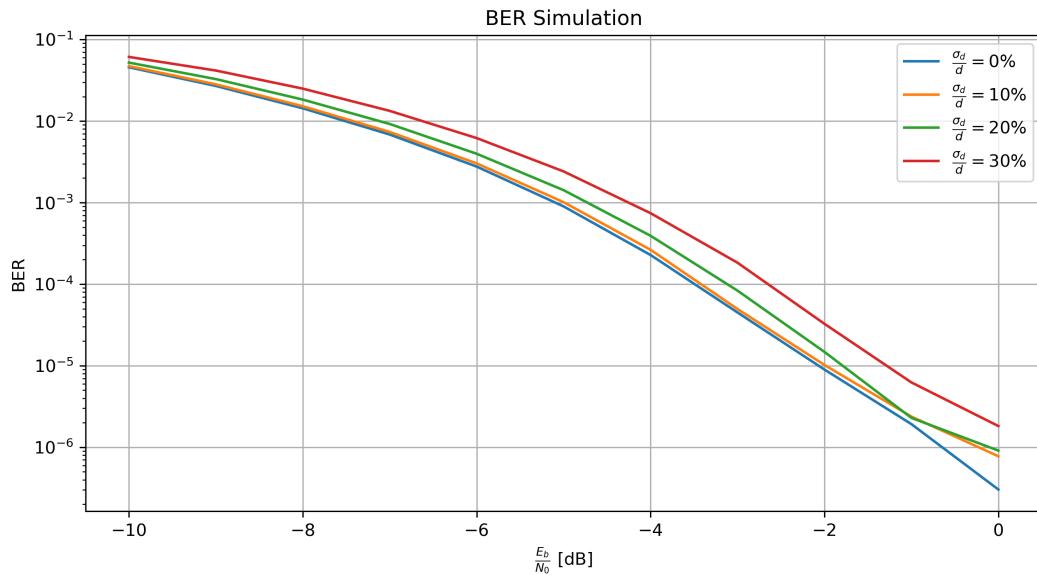


Figura 7.9: Curvas de BER en función de $\frac{E_b}{N_0}$ para distintos valores de errores en la separación de elementos del ARU.

Bloque	CPU [%]
DOA Esprit Estimation	99,3
Beamformer	3,7
Random Sampler	2,3
Phased Array Emulator	2

Tabla 7.1: Utilización del procesador correspondiente a cada bloque del sistema conformador de haz.

Capítulo 8

Trabajo a futuro

“To my future descendants: I have lent to Miss Bloop’s private museum, the Medallion and the Ancestral Tunic which rightfully belong to you.”

— Twinsen

En este capítulo se hace una breve mención del trabajo que resta por hacer para completar la implementación del sistema conformador de haz, así como también posibles alternativas de implementación que no fueron estudiadas a lo largo de este proyecto y que pueden mejorar el producto final.

8.1. Interfaz e integración con el sistema de adquisición

Como se mencionó en el Capítulo 6, el fin de este sistema es ser implementado en una placa de desarrollo CIAA-ACC, sobre la cual va a estar instalado, también, el sistema de adquisición de muestras del ARU. Estos dos sistemas deben estar comunicados, de manera tal que el sistema de adquisición pueda enviar las muestras al sistema conformador de haz. Debido a que el conformador de haz tendrá parte de su implementación en el PS y parte de su implementación en la PL, es necesario definir las interfaces y el protocolo que utilizarán para la comunicación. En el chip Xilinx Zynq-7030, el PS puede comunicarse con la PL a través de un bus AXI-Lite, en el cual el PS toma el rol de maestro y la PL de esclavo. Por esto, es intuitivo pensar que dentro de la FPGA el sistema adquisidor y los bloques muestrador aleatorio y conformador de haz pueden tomar el rol de esclavos AXI y ser comandados por software a través del PS para redirigir las muestras al bloque estimador de DOA. Por otro lado, se puede evaluar el uso de una interfaz directa entre el sistema adquisidor y los bloques conformador de haz y muestrador aleatorio, ya que estos bloques deberán poder tener acceso simultáneamente a los registros del sistema adquisidor donde se almacenan las muestras de los elementos. Asegurarse una conexión punto a punto entre estos bloques es un criterio de diseño primordial para asegurar el mayor rendimiento del sistema. Finalmente, la interfaz entre el maestro AXI y el bloque estimador de DOA, dentro del PS, puede utilizar una **interfaz UDP**, debido a su simplicidad de implementación y a que en esta interfaz no se requiere verificar la integridad de los datos transmitidos. Además, si se decide utilizar GNURadio para la ejecución de este bloque, este cuenta con una interfaz UDP que puede ser instanciada fácilmente con un bloque. En este caso deberá definirse, además, un protocolo de empaquetamiento de los datos transmitidos. En la Figura 8.1 se muestra un esquema de esta propuesta.

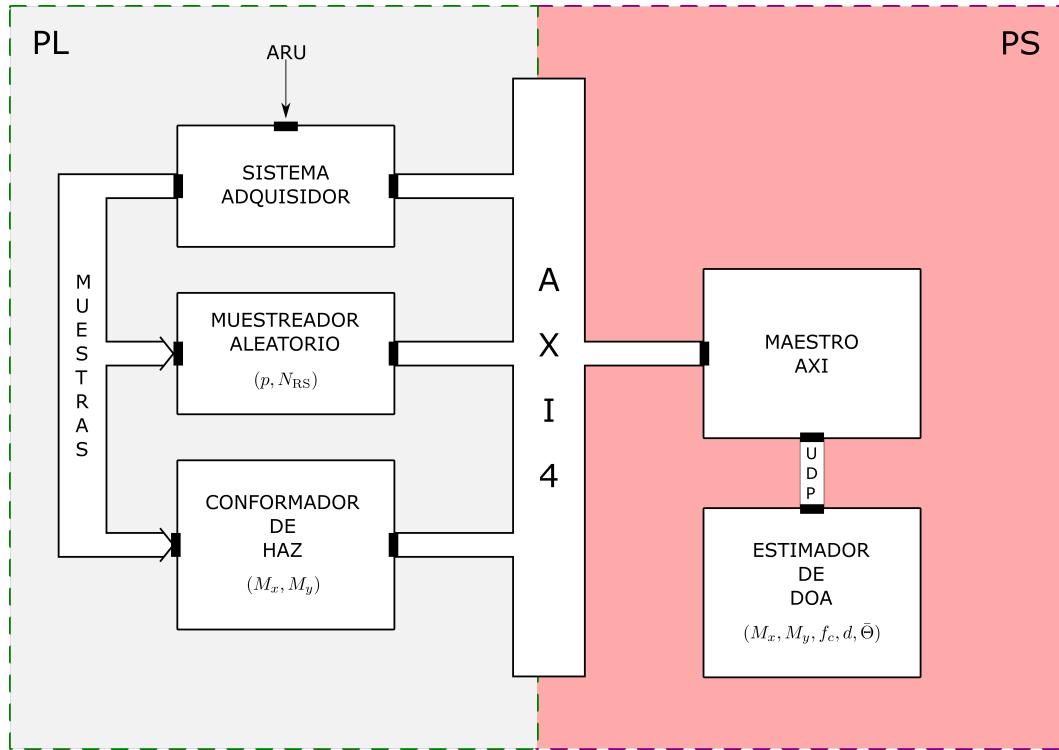


Figura 8.1: Propuesta de interfaz para la integración del sistema conformador de haz con el sistema de adquisición.

8.2. Carga del patrón de radiación del arreglo

Una vez construido el arreglo de antenas con el cual se equipará el sistema, y habiendo realizado una medición de su patrón de radiación, debe cargarse esta información dentro del bloque conformador de haz para poder realizar la normalización de las muestras recibidas, como se mostró en la Sección 6.5.

8.3. Interferencias destructivas

Como se nombró en el Capítulo 2, la técnica de conformación de haz no solo permite realizar la recepción o transmisión direccional de señales sino que también permite generar mínimos en el patrón de radiación de manera tal de poder suprimir interferencias direccionales. La manera de generar estos mínimos es aplicando distintos pesos a las muestras provenientes de los elementos del arreglo. Una vez conocida la dirección de la interferencia, los pesos a aplicar pueden obtenerse mediante la resolución de un sistema de ecuaciones.

8.4. Smart Beamforming

Más allá que el algoritmo que se desarrolló en este proyecto puede considerarse de alguna manera “inteligente” debido a que es capaz de deducir por su cuenta la dirección de arribo de señales, el término *Smart Beamforming* hace referencia a técnicas de conformación de haz que utilizan algoritmos de inteligencia artificial para la estimación de DOA y la conformación de las señales arribantes. Actualmente ya existen estudios que demuestran la factibilidad técnica y la mejora en rendimiento al emplear algoritmos de estimación de DOA basados en inteligencia artificial y, particularmente, en redes neuronales [30], lo cual motiva a tener en cuenta este tipo de análisis para futuras implementaciones.

Capítulo 9

Conclusiones

“He llegado hasta el fin, con los brazos cansados...”

— Gustavo Cerati

En este proyecto se realizó el estudio de distintas técnicas de implementación de un conformador de haz digital adaptativo, partiendo de una introducción teórica sobre los conceptos en los que se basa esta técnica, identificando los distintos tipos de conformadores de haz, definiendo el concepto de arreglo de antenas en fase y caracterizando sus distintos tipos, para luego definir el problema a resolver. A partir de aquí se realizó el estudio de las distintas técnicas de estimación de DOA, componente primordial para el funcionamiento del sistema a implementar. En este estudio se definió el modelo de muestras que se utilizó a lo largo de todo el proyecto y se realizó una introducción teórica sobre los conceptos algebraicos que permiten la descomposición del subespacio de muestras en subespacios de señal y ruido. La comprensión de esta técnica permitió iniciar con el análisis de dos algoritmos populares en lo que respecta a la estimación de parámetros de señales recibidas en arreglos de sensores: MUSIC y ESPRIT. Durante este estudio se desarrolló la teoría en la que se basa el funcionamiento de ambos algoritmos para finalmente realizar implementaciones de ambos que permitieron su comparación. En este estudio, MUSIC demostró ser un excelente algoritmo para introducirse en el análisis de técnicas de estimación paramétrica de señales debido a su intuitivo enfoque gráfico. Sin embargo, al momento de la implementación ESPRIT demostró ser superior en lo que respecta a tiempos de ejecución, alcanzando niveles de error prácticamente idénticos. A partir de esto se decidió continuar con el algoritmo ESPRIT para el resto de la implementación.

Durante la realización de las simulaciones de los algoritmos de estimación de DOA implementados se observó que en situaciones particulares en el que las señales recibidas se encontraban muy correlacionadas se requería operar con una cantidad de muestras que volvía inviable la implementación de un sistema en tiempo real. A partir de aquí se logró dar con una técnica de muestreo aleatorio que dio excelentes resultados al momento de reducir la cantidad de muestras requeridas para realizar la estimación de DOA, reduciendo dicho número por encima de dos órdenes de magnitud. Luego de comprobar su correcto funcionamiento se logró dar con una explicación teórica de su eficacia, la cual se indica en el Capítulo 4.

Para resolver el problema de estimación de cantidad de señales recibidas se observó que este podría ser resuelto utilizando técnicas de clasificación mediante aprendizaje automático. A partir de esto, y con los conocimientos adquiridos luego de realizar el curso “Aprendizaje Automático” dictado por el Profesor Andrew Ng de la Universidad de Stanford [16], se pudo implementar un algoritmo que permitió realizar la clasificación de valores singulares, necesaria para realizar la estimación de cantidad de señales recibidas, con una precisión superior al 95 %.

Una vez definidos los subsistemas que conforman al conformador de haz se realizó un diseño de bloques, definiendo la función y las interfaces de cada uno de ellos, realizando un análisis cualitativo de las ventajas y desventajas que existen al implementar ciertas funciones en FPGA o en el PS. Finalmente se esboza una propuesta de diseño de bloques para FPGA para una futura implementación.

Por último, luego de los conocimientos adquiridos en el “Mini-taller de desarrollo en GNURadio” dictado por el Dr. Federico La Rocca [22] durante el mes de julio de 2020, se pudo realizar una implementación en GNURadio del sistema conformador de haz completo, validando su funcionamiento mediante simulaciones. Diseñando las interfaces necesarias, este software puede ser instalado en el PS de la placa de desarrollo para realizar pruebas iniciales cuando se construyan el resto de los sistemas que complementan al conformador de haz (sistema de adquisición y arreglo de antenas).

Como cierre de este trabajo se esbozaron algunas propuestas de estudio e implementación a futuro que motiven la continuación de este proyecto analizando alternativas que pueden llegar a conseguir resultados aún mejores a los obtenidos.

Apéndice A

Obtención de ángulos de arribo en ESPRIT

Considerando una elección de subarreglos como el de la Figura 3.3 y el sistema de coordenadas definido en la Figura 2.8, los desfasajes φ_x y φ_y obtenidos mediante ESPRIT valen:

$$\begin{aligned}\varphi_x &= e^{-jkd \cos \theta \cos \varphi} \\ \varphi_y &= e^{-jkd \cos \theta \sin \varphi}\end{aligned}\tag{A.1}$$

Quedándonos con la fase de estas expresiones tenemos:

$$\begin{aligned}\angle \varphi_x &= kd \cos \theta \cos \varphi \\ \angle \varphi_y &= kd \cos \theta \sin \varphi\end{aligned}\tag{A.2}$$

Dividiendo $\angle \varphi_y$ con $\angle \varphi_x$ se tiene:

$$\frac{\angle \varphi_y}{\angle \varphi_x} = \frac{\sin \varphi}{\cos \varphi} = \tan \varphi\tag{A.3}$$

por ende de aquí puede obtenerse φ haciendo:

$$\varphi = \arctan \left(\frac{\angle \varphi_y}{\angle \varphi_x} \right)\tag{A.4}$$

Para obtener θ se procede haciendo:

$$\begin{aligned}(\angle \varphi_x)^2 + (\angle \varphi_y)^2 &= k^2 d^2 \cos^2 \theta \underbrace{(\cos^2 \varphi + \sin^2 \varphi)}_1 \\ (\angle \varphi_x)^2 + (\angle \varphi_y)^2 &= k^2 d^2 \cos^2 \theta \\ \cos^2 \theta &= \frac{(\angle \varphi_x)^2 + (\angle \varphi_y)^2}{k^2 d^2} \\ \cos \theta &= \sqrt{\frac{(\angle \varphi_x)^2 + (\angle \varphi_y)^2}{k^2 d^2}} \\ \theta &= \arccos \left(\sqrt{\frac{(\angle \varphi_x)^2 + (\angle \varphi_y)^2}{k^2 d^2}} \right)\end{aligned}\tag{A.5}$$

Bibliografía

- [1] Kulichevsky, J. Antenas de la CONAE en la Estacion Terrena del Centro Espacial Teófilo Tabanera. https://es.wikipedia.org/wiki/Estaci%C3%B3n_Terrena_C%C3%B3rdoba, Noviembre 2010. ix, 6
- [2] Shieber, J. Amazon joins SpaceX, OneWeb and Facebook in the race to create space-based internet services. <https://techcrunch.com/2019/04/04/amazon-joins-spacex-oneweb-and-facebook-in-the-race-to-create-space-based-internet-services/>, Abril 2019. 1
- [3] Balanis, C. A. Arrays: Linear, planar, and circular. En: Antenna theory: analysis and design, págs. 285–368. John Wiley & Sons, 2016. 6
- [4] Maral, G., Bousquet, M. Orbitas and related issues. En: Satellite communications systems: systems, techniques and technology, págs. 19–97. John Wiley & Sons, 2009. 6
- [5] Krim, H., Viberg, M. Two decades of array signal processing research: the parametric approach. *IEEE Signal Processing Magazine*, **13** (4), 67–94, 1996. 8, 12, 17
- [6] Steyskal, H. Digital beamforming antennas, an introduction. *Microwave Journal*, págs. 107–124, 1987. 9
- [7] Weiβ, M. Digital Antennas. *NATO Research and Technology Organisation*, 2009. 9
- [8] Mailloux, R. J. Phased arrays in radar and communication systems. En: Phased Array Antenna Handbook, págs. 1–61. Artech House, 2005. 11
- [9] Ioannides, P., Balanis, C. A. Uniform circular arrays for smart antennas. *IEEE Antennas and Propagation Magazine*, **47** (4), 192–206, 2005. 12
- [10] Roy, R., Kailath, T. Esprit-estimation of signal parameters via rotational invariance techniques. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37** (7), 984–995, 1989. 18, 19, 22, 23, 24, 25, 27
- [11] Schmidt, R. O. Multiple emitter location and signal parameter estimation. *IEEE Transactions on Antennas and Propagation*, **34** (3), 276–280, 1986. 18, 22
- [12] Ingle, V., Kogon, S., Manolakis, D. Harmonic models and frequency estimation techniques. En: Statistical and Adaptive Signal Processing, págs. 478–493. Artech, 2005. 20, 21
- [13] Estévez, D. Repositorio de capturas de beacons de satélites. <https://github.com/daniestevez/satellite-recordings>, Septiembre 2016. 28
- [14] The SciPy community. numpy.linalg.svd. <https://numpy.org/doc/stable/reference/generated/numpy.linalg.svd.html>. 33

- [15] Nyquist, H. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, **47** (2), 617–644, 1928. 36, 39
- [16] Ng, A. Machine Learning. <https://www.coursera.org/learn/machine-learning>, 2011. 46, 48, 52, 77
- [17] scikit-learn developers. Support Vector Machines. <https://scikit-learn.org/stable/modules/svm.html#svm-kernels>, 2020. 51
- [18] Chang, C.-C., Lin, C.-J. LIBSVM – A Library for Support Vector Machines. <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>, 2019. 51
- [19] Xilinx. Zynq-7000 SoC Data Sheet: Overview. https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf, Julio 2018. 57
- [20] Quinteros del Castillo, J. I. Sistema para emulación, adquisición y procesamiento de datos de matrices de sensores. *Instituto Balseiro*, 2020. 59
- [21] RTL-SDR. Roundup of software defined radios. <https://www.rtl-sdr.com/roundup-software-defined-radios/>, Agosto 2014. 65
- [22] La Rocca, F. Mini-taller de desarrollo en GNURadio. <https://iie.fing.edu.uy/personal/flarroca/teaching/taller-de-desarrollo-en-gnu-radio/>, Julio 2020. 65, 67, 68, 70, 78
- [23] Vachhani, K., Mallari, R. A. Experimental study on wide band FM receiver using GNURadio and RTL-SDR. págs. 1810–1814, 2015. 65
- [24] Haykin, S., Van Veen, B. Signals and systems. John Wiley & Sons, 2007. 66
- [25] freeCodeCamp. Interpreted vs compiled programming languages: What's the difference? <https://www.freecodecamp.org/news/compiled-versus-interpreted-languages/>, Enero 2020. 68
- [26] GNU Radio project. Vector-Optimized Library of Kernels. <https://www.libvolk.org/>, 2020. 70
- [27] Estévez, D. gr-satellites. <https://github.com/daniestevez/gr-satellites>, 2020. 72
- [28] Estévez, D. BER simulation in GNU Radio. <https://destevez.net/2017/07/ber-simulation-in-gnu-radio/>, 2020. 72
- [29] Jacob, B., Guennebaud, G. EIGEN. <http://eigen.tuxfamily.org/>, 2020. 72
- [30] Fernández, S., Padilla, Y., Guzmán Obregón, O. A., Arbella, Y. Optimización de la estimación de doa en sistemas de antenas inteligentes usando criterios de redes neuronales. *Ingeniería Electrónica, Automática y Comunicaciones*, **34**, 70–86, 04 2013. 76

Agradecimientos

Esta sección se realizará para la versión final del documento.

